THE FIFTEENTH INTERNATIONAL
INTERNET & SOFTWARE
QUALITY WEEK

San Francisco • September 3 - 6, 2002

Produced by
Software Research Institute

The Wired World

It's the driving factor in today's technology growth.

QW2002 aims to face internet and software quality issues directly.

Keynoters and speakers with real-world experience that you can apply immediately.

Fifteen years of serving the SQA community with cutting-edge experience and state-of-the-art technology.

In cooperation with

American Society for Quality ASQ Software Division

WOW World Organization of Webmasters *Community, Education, Certification*

Media Sponsors

Software Business — The Magazine of the Software Industry

SD Times — Software Development — The Industry Newspaper for Software Development Managers

Industry Sponsors

eValid — Your e-Business Partner

LogiGear — Expert Testing...Real World Solutions

www.qualityweek.com

QualityWeek 2002
The Wired World

San Francisco, California, USA | 3 - 6 September 2002

QW2002 Program

NOTE: The program selections, speaker biographies, and presentation abstracts may be incomplete in some instances; this material is being updated constantly. All material presented is based on the best information available and may be subject to change. Updated 16 August 2002.

QUICK ACCESS TO THE FOUR DAY PROGRAM AT QW2002	
Pre-Conference Tutorials	Tuesday 3 September 2002
Conference Day 1	Wednesday 4 September 2002
Conference Day 2	Thursday 5 September 2002
Conference Day 3	Friday 6 September 2002
Post Conference Workshops	Friday 6 September 2002
Vendor Demonstration Sessions	Wednesday 4 September 2002 Thursday 5 September 2002

T	Tuesday 3 September 2002 PRE-CONFERENCE TUTORIALS						
8:30 - 12:00	**Tutorial A1** *Test Automation: A Context-Driven Approach* Mr. Douglas Hoffman Software Quality Methods, LLC. & Mr. Bret Pettichord	**Tutorial B1** *Web Site Robustness and Scalability Testing* Mr. Ross Collard Consultant	**Tutorial C1** *Web Testing: A Practical Approach* Dr. Edward Miller eValid, Inc.	**Tutorial D1** *Writing Good Requirements (PART 1)* Mr. Erik Simmons Intel Corporation	**Tutorial E1** *Practical Web/e-Commerce Stress Testing* *(PART 1)* Mr. Robert Sabourin Amibug.com	**Tutorial F1** *Test Management And Organization* Mr. Rex Black RBCS, Inc.	**Tutorial G1** *Better Data: Better Testing* Mr. James Lyndsay Workroom Productions Ltd.
12:00 - 1:30	TUTORIAL DAY LUNCH AND NETWORKING						
1:30 - 5:00	**Tutorial A2** *Test Automation: Survey of Architectures* Mr. Bret Pettichord Pettichord Consulting LLC & Mr. Douglas Hoffman	**Tutorial B2** *Security Testing for Web Sites, Web Applications and Software* Mr. Hung Nguyen LogiGear Corporation	**Tutorial C2** *Software Projects -- Effective Estimating & Planning Techniques* Mr. Robert L. Galen RGalen Consulting Group, LLC	**Tutorial D2** *Writing Good Requirements (PART II)* Mr. Erik Simmons Intel Corporation	**Tutorial E2** *Practical Web/e-Commerce Stress Testing* *(PART II)* Mr. Robert Sabourin Amibug.com	**Tutorial F2** *Managing Test Estimation* Mr. Rex Black RBCS, Inc.	**Tutorial G2** *Paradigms Of Black Box Software Testing* Dr. Cem Kaner Florida Institute of Technology
5:00 - 7:00	EXPO Open: Welcome Reception In The Exhibit Hall						

Wednesday 4 September 2002
CONFERENCE DAY #1

QW2002 Exhibition: 12:00 PM to 6:30 PM

1 8:30 - 10:00	**PLENARY SESSION** **Plenary Session Introduction:** Dr. Edward Miller, Chairman (Software Research, Inc.) **Keynote 1G1:** *Internet Reliability under Stress* Mr. Fred Baker Cisco Systems **Keynote 1G2:** *The Human Side of Risk* Mr. Erik Simmons Intel Corporation					
10:00 - 10:30	REFRESHMENTS IN EXHIBIT HALL					
	Vendor Technical Track	**Technology Track**	**Applications Track**	**Internet Track**	**Process Track**	**QuickStart Track**
10:30 **2** 11:15	**Session 2V1** *Carrier Compliance Testing of Mobile Handsets* Mr. Mitch Krause TestQuest, Inc.	**Paper 2T1** *Experience in Testing Pocket PC Applications* Mr. Ibrahim El-Far (Florida Institute of Technology), Ms. Florence Mottay (J.D. Edwards), & Dr. Herbert Thompson (System Integrity)	**Paper 2A1** *Testing and Managing Risky Content in the Online World* Mr. Michael Weider Watchfire Corporation	**Paper 2I1** *Web Site Performance and Qos Monitoring* Dr. Ludmila Cherkasova Hewlett Packard Labs	**Paper 2P1** *Outsourcing in QA and Testing* Mr. Vijay Sikka & Mr. Anurag Khemka Nirixa, Inc.	**Session 2Q** *Effective Performance Test Planning and Implementation* Mr. Hung Nguyen LogiGear Corporation
	Session 2V2 *Agile Software Process Improvement* Mr. Jessee Ring Software Quality First	**Paper 2T2** *Quality Assurance Aspects in Developing a Component-based System* Dr. Josef Withalm Siemens Ag	**Paper 2A2** *Testing A Large Scale Application* Mr. Peter Lafleur Public Works & Government Services Canada	**Paper 2I2** *Web Services QoS Indicators* Dr. Selim Aissi Intel	**Paper 2P2** *Creating Quality From Scratch: How To Build a Test Organization* Mr. Eric Patel RapidSQA	
12:00 - 1:30	CONFERENCE LUNCH AND NETWORKING IN EXHIBIT HALL					
	Vendor Technical Track	**Technology Track**	**Applications Track**	**Internet Track**	**Process Track**	**QuickStart Track**
	Session 3V1 *Change Impact Testing* Mr. Joe Ponczak McCabe & Associates	**Paper 3T1** *Interface-driven Model-based Test Generation of Java Test Drivers* Dr. Mark Blackburn, Robert Busser & Aaron Nauman	**Paper 3A1** *Software Defect Arrival Modeling With the Weibull Distribution* Mr. Erik	**Paper 3I1** *Measuring and Improving Web Site Quality: A Consumer Focused System* Dr. John M. Ryan	**Paper 3P1** *Applied Six Sigma to Software Process Improvement* Dr. Huey-Der	**Session 3Q** *A Short Course In Metrics & Measurement Dysfunction* Dr. Cem Kaner Florida Institute

		T-VEC Technologies/SPC & Dr. Ramaswamy Chandramouli (National Institute of Standards and Technology)	Simmons Intel Corporation	Safe Shopping Network	Chu National Defense Management College	of Technology
1:30 **3** 2:15	**Session 3V2** *DMS: Software Quality Enhancement via Automated software Analysis, modification and Generation* Dr. Ira Baxter Semantic Designs, Inc.	**Paper 3T2** *Tool Support for Model Based Statistical Testing* Dr. Stacy Prowell & Mr. William Thomas Swain The University of Tennessee	**Paper 3A2** *Using Defect Tracking Integrations to Improve your QA Process* Dr. Kelly A. Shaw TeamShare, Inc	**Paper 3I2** *InBrowser WebSite Testing: The Client-Side Approach* Dr. Edward Miller eValid, Inc., USA	**Paper 3P2** *Effort Estimation for QA Projects Statistical Approaches & Challenges* Mr. Raja Mohapatra & Mr. Bibhash Saha Infosys Technologies Ltd	
3:00 - 3:30	REFRESHMENTS IN EXHIBIT HALL					
	Vendor Technical Track	**Technology Track**	**Applications Track**	**Internet Track**	**Process Track**	**Panel Discussion**
3:30 **4** 4:15	**Session 4V1** To be announced.	**Paper 4T1** *Quality Process for XML Interfaces* Mr. Glenn Breslin Independent	**Paper 4A1** *The mysteries of Unit Testing Explained for Testers* Mr. Michael Hackett LogiGear Corporation	**Paper 4I1** *Bugs in Your Shopping Cart: A Taxonomy* Mr. Giri Vijayaraghavan & Dr. Cem Kaner Florida Tech Dept of Computer Sciences	**Paper 4P1** *Big And Complex Projects: Beyond Extreme Programming* Mr. Joan Bosch Sole NTE, s a.	**Session 4Q** *The Value of Agile Testing* Ross Collard Consultant & Mr. Bret Pettichord Pettichord Consulting
	Session 4V2 LogiGear Corporation	**Paper 4T2** *Innovative WebSite Mapping Tool* Dr. Edward Miller eValid, Inc.	**Paper 4A2** *Reduce Overall Project Development Costs with Structured Testing* Ms. Jan Grinnell GovConnect	**Paper 4I2** *Deployment of Globalised Wireless Internet Applications* Mr. Ed Adams VeriTest	**Paper 4P2** *'Excel'erating Test Status Reporting* Mr. Jim Hazen SysTest Labs	
5:00 - 6:30	EXPO RECEPTION Drinks and hors d'oeuvres are served in the Expo Hall.					

Thursday 5 September 2002
CONFERENCE DAY #2

QW2002 Exhibition: 10:00 AM to 3:30 PM

	PLENARY SESSION
	Plenary Session Introduction: Dr. Edward Miller (Software Research, Inc.)
5	**Keynote 5G1:** *Science, Computer "Sciences", Mathematics, and Software Development*

8:30 - 10:00	Prof. Dick Hamlet Portland State University					
	Keynote 5G2: *Achieving Very High Reliability for Ubiquitous Information Technology* Mr. Robert V. Binder Mobile Systems Verification					
10:00 - 10:30	REFRESHMENTS IN EXHIBIT HALL					
	Vendor Technical Track	**Technology Track**	**Applications Track**	**Internet Track**	**Process Track**	**QuickStart Track**
10:30 **6** 11:15	**Session 6V1** *Test Automation, Build to Last Action Based Testing Methods and Tools at Work!* Mr. Hans Buwalda Logigear Corporation	**Paper 6T1** *Revisiting Comparisons between Manual and Automated testing* Mr. Anurag Khemka & Mr. Vijay Sikka Nirixa, Inc.	**Paper 6A1** *Reducing Test Automation Maintenance Costs* Mr. Michael Waller & Timothy Nelson STAMP Technologies	**Paper 6I1** *Performance Testing Applications In Internet Time* Ms. Nancy Landau Alltel Technology Services	**Paper 6P1** *Optimizing Test Productivity to Maximize Product Quality* Dr. Nancy Eickelmann Motorola	**Session 6Q** *Testimonial - A Fully Illustrated Overview of Software Testing* Mr. Robert Sabourin Amibug.com
	Session 6V2 To be announced.	**Paper 6T2** *A Custom Automation Framework and Test Case Management Solution* Mr. Darin Magoffin PowerQuest	**Paper 6A2** *Risk Based Regression Testing* Mr. Gregory Pope University of California LLNL	**Paper 6I2** *Comparative Strategies for Measuring the Internet: The Whole is More than the Sum of the P* Dr. Chris Overton Keynote Systems, Inc.	**Paper 6P2** *Beta Testing -- Boot Camp Basics and Beyond* Mr. Ralph Dalebout IBM Corporation	
12:00 - 1:30	CONFERENCE LUNCH AND NETWORKING IN EXHIBIT HALL					
	Vendor Technical Track	**Technology Track**	**Applications Track**	**Internet Track**	**Process Track**	**QuickStart Track**
1:30 **7** 2:15	**Session 7V1** To be announced.	**Paper 7T1** *Streaming Media Quality: The Orphan Child of "Old Media" and Internet Flakiness* Dr. Chris Overton Genimedia & Dr. Simon Robins Genimedia	**Paper 7A1** *Adventures in Session-based Testing* Mr. James Lyndsay, Workroom Productions Ltd. & Niel vanEeden, Jobpartners Ltd.	**Paper 7I1** *Get Real! The Importance of Realism in Web Site Capacity Assessment* Mr. Philip Joung Caw Networks	**Paper 7P1** *Managing Test Products, The Next Challenge* Mr. Hans Buwalda LogiGear Corporation	**Session 7Q** *Overview of ASQ's 2002 CSQE Body of Knowledge* Mr. Douglas Hoffman Software Quality Methods, LLC.
	Session 7V2 To be announced.	**Paper 7T2** *SPARTA: Architecture for Automated Testing of Network Protocol Stacks* Mr. Srivatsa Srinivasan & Mr. Sandeep Prabhu Microsoft	**Paper 7A2** *Experiences with Action Based Testing, a Case Study* Mr. Flavio Y Kubagawa, QAD Inc. & Mr. Hans Buwalda, LogiGear Corp.	**Paper 7I2** *Don't Get Trampled by the Crowd: Realistic Load Testing of Web Sites Across the Internet* Mr. Eric D. Siegel Keynote Systems, Inc.	**Paper 7P2** *A Climber's View of Software Quality* Dr. John Dalbey California Polytechnic State University	
3:00 -						

3:30	REFRESHMENTS IN EXHIBIT HALL					
	Vendor Technical Track	**Technology Track**	**Applications Track**	**Internet Track**	**Process Track**	**Panel Session**
3:30 **8** 4:15	**Session 8V1** To be announced.	**Paper 8T1** *Branch Coverage Tools For Arbitrary Languages made Easy!* Dr. Ira Baxter Semantic Designs, Inc.	**Paper 8A1** *Using SW Process Assessment to Manage Quality of Suppliers: An Experience in Automotive [suppliers]* Dr. Fabrizio Fabbrini & Mr. Mario Fusani IEI-CNR	**Paper 8I1** *Vulnerabilities and Developing for the Net* Mr. Robert A. Martin The MITRE Corporation	**Paper 8P1** *Common Problems in Tool Adoption* Karen S. King King Consulting	**Session 8Q** *Web Services QoS, Reliability, and Security* Dr. Selim Aissi Intel
	Session 8V2 To be announced.	**Paper 8T2** *Automating Testing on ASP.NET Applications* Mr. Thomas R. Arnold Xtend Development, Inc.	**Paper 8A2** *Training Testing Professionals: Making the transition to Web Based Application Testing* Mr. Robert Sabourin AmiBug.Com, Inc.	**Paper 8I2** *Implementing a Web Based Testing Framework* Mr. Steve Whitchurch Network Associates, Inc.	**Paper 8P2** *The Making of an Open Source Stress Test Tool* Mr. Danny R. Faught Tejas Software Consulting	

Friday 6 September 2002 CONFERENCE DAY #3					
	Technology Track	**Applications Track**	**Internet Track**	**Process Track**	**QuickStart Track**
9 8:30 - 10:00	**Paper 9T1** *Virtual Test Management: Rapid Testing Over Multiple Time Zone* Mr. Ed Adams VeriTest	**Paper 9A1** *Performance and Scalability Bottlenecks in J2EE Applications* Mr. Ed Lycklama Sitraka	**Paper 9I1** *Assessing risks in wireless solution implementation* Mr. Yiftach Resheff Antenna Software	**Paper 9P1** *Testing Efficiency: Taking Advantage of Test Overlap* Ms. Lauri MacKinnon PhaseForward Inc.	**Session 9Q** *A Survey of Freeware Test Tools* Mr. Danny Faught Tejas Software Consulting
	Paper 9T2 *Cross Platform Framework For Integration Testing* Dr. Nagesh Vempaty Aalayance Incorporated	**Paper 9A2** *Legacy Data Conversion: Making Coffee* Mr. Joshua Kitchen IBeta	**Paper 9I2** *A Framework for Testing Wireless Applications* Mr. Ibrahim K. El-Far, Roussi Roussev, and Nattawut Sridranup Florida Institute of Technology	**Paper 9P2** *Measuring the Effectiveness of Software Testers* Dr. Cem Kaner Florida Institute of Technology	
10:00 - 10:30	REFRESHMENTS				
10:30	**PLENARY SESSION** **Plenary Session Introduction:** Dr. Edward Miller (Software Research, Inc.) **Keynote 10G1:**				

10	*Competitiveness Versus Security* Mr. Don O'Neill Center for National Software Studies
11:15	**Keynote 10G2:** *You Want It When?* Mr. Gregory Pope University of California LLNL
	AWARDS PRESENTATION
12:30 - 1:30	LUNCH FOR WORKSHOP ATTENDEES

Friday 6 September 2002 POST-CONFERENCE WORKSHOPS				
W 1:30 - 5:00	**Technology Workshop W1** *Introduction to Performance on the Internet and Web* Mr. Eric D. Siegel Keynote Systems, Inc.	**Applications Workshop W2** *Jump Starting Your Test Automation* Mr. Hung Nguyen & Hans Buwalda LogiGear Corporation	**Internet Workshop W3** *Performance Testing & Issues* Dr. Subraya BM Infosys Technologies Ltd.	**Management Workshop W4** *Bug Advocacy: Effective Bug Reporting* Mr. Cem Kaner Florida Institute of Technology

SB	Standby Presentations
	Process Track
	The Essentials of Testing and Test Automation Mr. Hans Buwalda LogiGear Corporation

QW2002 Tutorial A1

Mr. Douglas Hoffman
(Software Quality Methods, LLC.) & Mr. Bret Pettichord

Test Automation: A Context-Driven Approach

Key Points

- Analyze your requirements for test automation
- Understand how product architecture affects test automation
- Learn the key elements to test automation architecture
- Learn five approaches for verifying test results
- Realize the limits of regression testing
- Avoid common blind alleys
- Determine whether your organization is ready to automate

Presentation Abstract

Doug Hoffman and Bret Pettichord bring together their extensive experience to provide key information on test automation. Successful testing groups have used a variety of test automation architectures. Pettichord and Hoffman's combined perspectives and insights will help you select and customize an effective architecture for your context.

Most software test automation efforts fail to achieve their goals, some rather spectacularly. Effective software test automation starts with understanding and good planning. This morning tutorial describes how to take a context-driven approach to test automation. Instead of using a generic test automation approach, successful companies tailor their automation architecture to their specific context. Avoid common blind alleys that many test teams have wasted time and energy on. Learn how your staff profile, product architecture and test mission affect your test automation architecture.

About the Author

Douglas Hoffman is an independent consultant with Software Quality Methods, LLC. He has been in the software engineering and quality assurance fields for nearly 30 years and now is a management consultant in strategic and tactical planning for software quality. Throughout his career he has automated testing using a wide variety of hardware and software techniques. His experience includes consulting, teaching, managing, and engineering across the computer systems and software industries. His recent experience has focused on creating and transforming software quality and development groups, and architecting software test automation environments.

Douglas is a founder and past Chairman of the Santa Clara Valley Software Quality Association (SSQA) and the Silicon Valley Section of the American Society for Quality (ASQ). He has been a presenter and participant at dozens of software quality conferences and has been Program Chairman for several international conferences on software quality. He is a member of the ACM and IEEE and is active in the ASQ as a Senior Member, participating in the Software Division, the Santa Clara Valley Section, and the Software Quality Task Group. He has earned a BA in Computer Science, an MS in Electrical Engineering, an MBA, a Certificate from ASQ in Software Quality Engineering, and has been a registered ISO 9000 Lead Auditor.

Bret Pettichord is an independent consultant with experience using a wide variety of test tools and approaches. He's developed automated tests for software in various domains, including publishing, accounting, sales, systems management, education and benefits administration. The tutorial builds on Lessons Learned in Software Testing: A Context- Driven Approach, the best-selling book he coauthored with Cem Kaner and James Bach. Many of his writings are collected at www.pettichord.com.

Specializing in software testing and test automation, Bret has 14 years experience in software development, including positions as a staff consultant at Segue Software and a test automation lead at BMC and then IBM/Tivoli before becoming an independent consultant. Recent clients have included Texas Instruments, Rational Software, Texas Department of Human Services, Deliotte Consulting, Whisperwire and Netpliance. He learned about software testing as a tax programmer at Arthur Andersen and a software tester and automator at Interleaf.

QW2002 Tutorial B1

Mr. Ross Collard
(Collard & Company)

Web Site Robustness and Scalability Testing

Key Points

- Testing availability of Web sites
- Testing scalability
- Robustness and recoverability of Web sites

Presentation Abstract

First on reliability -- nature of Web site reliability, modes of failure, testing for recoverability, load & stress test project outlines,load & stress test plan templates, and common test chalenges.

Second session on scalability -- reasons Web sites don't scale, how to test for scalalabity, predictig scalability, and what test equipment and tools are needed.

About the Author

Ross Collard is president of Collard & Company, a consulting firm which is headquartered in Manhattan, New York City. His consulting and training clients have included: ADP, Alcatel, American Express, Anheuser-Busch, Apple, AT&T, Banamex, Bank of America, Bechtel, Blue Cross/Blue Shield, Boeing, British Airways, the CIA, Ciba Geigy, Cisco, Citibank, Computer Associates, Dayton Hudson, Dell, EDS, Exxon, General Electric, Goldman Sachs, Federal Reserve, Ford, Hewlett-Packard, Hughes Aircraft, IBM, Intel, Johnson & Johnson, JP Morgan, McGraw Hill, MCI, Merck, Microsoft, Motorola, NASA, Nortel, Novell, Procter & Gamble, Prudential, Sears Roebuck, Swiss Bank, U.S. Air Force, Verizon and Worldcom.

He has conducted seminars on business and information technology topics for businesses, governments and universities, including George Washington, Harvard and New York Universities, MIT, Stanford and U.C. Berkkeley.He has a BE in Electrical Engineering from the University of New Zealand (where he grew up), an MS in Computer Science from the California Institute of Technology and an MBA from Stanford University. His set of books on software testing is due to be published next year.

QW2002 Tutorial C1

Mr. Edward Miller
(eValid, Inc.)

Web Testing: A Practical Approach

Key Points

- WebSite testing imposes new constraints on testing and analysis technology.
- Very complext WebSites have extremely high "negative payoff" for failure.
- WebSite quality requires both static and dynamic testing, and dynamic testing requires 100% maintenance of session context.
- Continual monitoring of WebSite quality appears to be a requirement for a complex WebSite.

Presentation Outline

WebSite Properties, Concerns
Static Testing
Dynanmic Testing
  Functional/Verification
  Timing/Tuning
  Server Loading
  Mapping
WebSite Monitoring
Case Study
Source Material

About the Author

Dr. Edward Miller is Chairman and President of Software Research, Inc., San Francisco, California, and Chief Technical Architect for software test tools development and software engineering quality questions. Dr. Miller has worked in the software quality management field for 25 years in a variety of capacities, and has been involved in the development of families of automated software, analysis and WebSite quality analysis tools.

He was chairman of the 1985 1st International Conference on Computer Workstations, and has participated in IEEE conference organizing activities for many years. He has been Chairman of the Quality Week Conferences since 1988 and of the Quality Week Europe Conferences since 1996. He is a Bember of the Board of the Center for National Software Studies, a not-for-profit organization devoted to furtherance of national software concerns, and a member of the External Advisory Board of the EE Department at Iowa State University. He is the author of Software Testing and Validation Techniques, an IEEE Computer Society Press tutorial text. Dr. Miller received his Ph.D. (Electrical Engineering) degree from the University of Maryland, an M.S. (Applied Mathematics) degree from the University of Colorado, and a BSEE from Iowa State University.

QW2002 Tutorial D1 & D2

Mr. Erik Simmons
(Intel Corporation)

Writing Good Requirements

Key Points

- Discuss and choose from different techniques to specify requirements
- Improve written requirements and tell good requirements from bad ones
- Write non-functional requirements so they are verifiable

Presentation Abstract

Back by popular demand, in full-day format. This tutorial has earned consistent praise from attendees at Quality Week 2001, Quality Week Europe 2002 (average 4.9 out of 5 rating), and PNSQC 2000 (44 of 45 evaluations rated "valuable" or "very valuable"). Attendees enjoy the fast pace and depth of information presented. This material has been taught to nearly 4,000 students at Intel sites around the world. Poorly written requirements result in lost productivity, increased re-work, dissatisfied customers, poor end product quality, and even project cancellations. So, why are good requirements so hard to write? Many people do not know the key attributes of a "Good Requirement", and have not been exposed to the various effective ways to specify requirements.

This 1-day workshop focuses on and applies the best-known methods behind improved requirements writing. Based closely on a popular course taught at Intel, the course covers the different types of requirements and what activities are important when specifying requirements. The emphasis is on practical solutions to common problems, and contains valuable real examples from Intel documents in both original and improved formats. Students will gain an understanding of the attributes of a good requirement, and learn ways to identify whether the requirement is unambiguous, concise, necessary, correct, and traceable. Many useful "take it home and use it tomorrow" techniques for writing both functional and non-functional requirements are presented. Several exercises are included to reinforce the techniques. Attendees are invited to bring their existing requirements documents for use in the final exercise if desired.

About the Author

Erik Simmons has 15 years experience in multiple aspects of software and quality engineering. Erik currently works as Platform Quality Engineer in the Platform Quality Methods group, part of the Corporate Quality Network at Intel Corporation. He is responsible for Requirements Engineering practices at Intel, and lends support to several other corporate software and product quality initiatives. Erik is a member of the Pacific Northwest Software Quality Conference Board of Directors and the Steering Committee of the Rose City SPIN. He holds a Masters degree in mathematical modeling and a Bachelors degree in applied mathematics from Humboldt State University in California, and was appointed to the Clinical Faculty of Oregon Health Sciences University in 1991.

QW2002 Tutorial E1 & E2

Robert Sabourin
(Amibug.com)

Practical Web/e-Commerce Stress Testing

Key Points

- Stress testing for Web applications
- Performance testing for Web applications
- Load testing for Web applications
- Test automation for Load Testing of Web applications
- How stress can impact all quality factors
- Stress testing tools and techniques
- How to simulate large system loads
- How to implement site monitoring
- What is stress testing?
- When to apply stress testing
- How to assess software reliability
- How to assess software stability

Presentation Abstract

You know you should be doing stress testing, you're just not sure when. This workshop explores this question, which is particularly important to developers in Internet multi-tier development projects. You will review practical methods of assessing software reliability, and learn why it is important to "stress early and stress often."

This workshop focuses on stress testing Web and e-Commerce applications. You will learn how to find the weakest part of an application by testing to failure and then working with developers to determine the root cause of the defect. You will learn about the tools to support stress testing, including several available for free!

This workshop introduces you to using tools to automate some important parts of testing Web/e-Commerce applications. You will study how to organize and run stress testing, and review several sample automations.

Several examples based on commercial Web/e-Commerce applications will be described.

Benefits

After completing this workshop, you will understand many approaches to stress testing applications at all phases of development. After completing this workshop, you will understand how to plan and automate stress testing for Web/e-Commerce applications. After completing this workshop, you will know how and when to implement stress testing as part of the software development process.

About the Author

Robert Sabourin, P. Eng. has 20+ years management experience leading teams of software development professionals to consistently deliver projects on-time, on-quality and on-budget. He is a well-respected member of the software engineering community who has managed, trained, mentored and coached hundreds of top professionals in the field. He frequently writes and speaks to conferences around the world on software engineering, SQA, testing, management and internationalization.

QW2002 Tutorial F1

Mr. Rex Black
(RBCS, Inc.)

Test Management And Organization

Key Points

- Explain the organizational context of an independent testing team.
- Create and explain a cost-of-quality based business case for testing.
- Figure out how test efforts should fit into various system lifecycles.

Presentation Abstract

Organizations invest in quality in general, and testing specifically, because they expect the time, money, and care spent prior to releasing a system to pay dividends after the release. These dividends may come in the many forms, including but not limited to financial rewards. Testing is, therefore, a risk management exercise that helps the organization reduce the likelihood of having to bear certain unanticipated future costs. With such high stakes involved, testing is an endeavor that deserves careful planning, crisp execution, a talented team, and significant resources. This tutorial will examine the project and organizational context of testing projects, the need for alignment between testing and quality, and techniques that a test manager can use to ensure crisp test project management.

About the Author

Rex Black (Rex_Black@RexBlackConsulting.com) is the President and Principal Consultant of Rex Black Consulting Services, Inc. *http://www.RexBlackConsulting.com*, an international software and hardware testing and quality assurance consultancy. He and his consulting associates help clients such as Bank One, Cisco, Compaq, Dell, Schlumberger, Williams Communications, and others with implementation, consulting, training, and staffing for testing, test automation, and quality assurance projects. His first book, Managing the Testing Process, published in June 1999, sold thousands of copies in North and South America, Europe, and Asia. In July 2002, Wiley published the updated and improved second edition. Following on the heels of these successful books, Addison-Wesley will publish Critical Testing Processes, Volume I and Volume II in late 2002 and early 2003, respectively.

QW2002 Tutorial G1

Mr. James Lyndsay
(Workroom Productions Ltd.)

Better Data: Better Testing

Key Points

- Data-driven functional test techniques
- The construction of "Good Data", and how it can help avoid comon problems
- Data and communication

Presentation Abstract

Data is a crucial part of most functional testing - but its importance can be missed in test planning, and is often only appreciated after things have started to go wrong. This course will help delegates improve their testing by improving their test data.

About the Author

James Lyndsay is an independent consultant with more than ten years experience. After working in analysis, coding and testing at IBM and in the City, he formed Workroom Productions in 1994 (http://www.workroomproductions. com/).
As a Test Strategist, he has spent the last seven years working with multinational corporations, long projects, and even the occasional web start-up. His business experience includes banking, telecoms, utility billing, logistics, electronic publishing and retail, and he pays keen attention to the way that his clients focus is shifting away from functional testing.

QW2002 Tutorial A2

Mr. Bret Pettichord
Pettichord Consulting LLC & Mr. Douglas Hoffman

Test Automation: Survey of Architectures

Key Points

- Learn the benefits and risks of developing user interface abstraction libraries.
- Understand when to use data-driven, keyword-based or model-based approaches.
- Learn about options for automating non-user interfaces.
- Learn how focusing on test evaluation can enable high-volume test creation.

Presentation Abstract

Bret Pettichord and Doug Hoffman bring together their extensive experience to provide key information on test automation. Successful testing groups have used a variety of test automation architectures. Pettichord and Hoffman's combined perspectives and insights will help you select and customize an effective architecture for the needs and challenges of your context.

This afternoon tutorial surveys and explores eleven architectural patterns for automating software testing. Each architecture is described in terms of the contextual factors that favor its adoption. These patterns include:

- Scripting Frameworks
- Data-Driven Scripts
- Action Keywords
- Test-First Programming
- API Tests
- Thin GUI
- Consult an Oracle
- Automated Monkey
- Assertions and Diagnostics
- Quick and Dirty

About the Author

Bret Pettichord is an independent consultant with experience using a wide variety of test tools and approaches. He's developed automated tests for software in various domains, including publishing, accounting, sales, systems management, education and benefits administration. The tutorial builds on Lessons Learned in Software Testing: A Context- Driven Approach, the best-selling book he coauthored with Cem Kaner and James Bach. Many of his writings are collected at www.pettichord.com.

Specializing in software testing and test automation, Bret has 14 years experience in software development, including positions as a staff consultant at Segue Software and a test automation lead at BMC and then IBM/Tivoli before becoming an independent consultant. Recent clients have included Texas Instruments, Rational Software, Texas Department of Human Services, Deliotte Consulting, Whisperwire and Netpliance. He learned about software testing as a tax programmer at Arthur Andersen and a software tester and automator at Interleaf.

Douglas Hoffman is an independent consultant with Software Quality Methods, LLC. He has been in the software engineering and quality assurance fields for nearly 30 years and now is a management consultant in strategic and tactical planning for software quality. Throughout his career he has automated testing using a wide variety of hardware and software techniques. His experience includes consulting, teaching, managing, and engineering across the computer systems and software industries. His recent experience has focused on creating and transforming software quality and development groups, and architecting software test automation environments.

Douglas is a founder and past Chairman of the Santa Clara Valley Software Quality Association (SSQA) and the Silicon Valley Section of the American Society for Quality (ASQ). He has been a presenter and participant at dozens of software quality conferences and has been Program Chairman for several international conferences on software quality. He is a member of the ACM and IEEE and is active in the ASQ as a Senior Member, participating in the Software Division, the Santa Clara Valley Section, and the Software Quality Task Group. He has earned a BA in Computer Science, an MS in Electrical Engineering, an MBA, a Certificate from ASQ in Software Quality Engineering, and has been a registered ISO 9000 Lead Auditor.

QW2002 Tutorial B2

Mr. Hung Q. Nguyen
(LogiGear Corporation)

Security Testing for Web Sites, Web Applications and Software

Key Points

- To introduce software security and security testing concepts
- To share strategies and tactics for software security test planning and test design
- To discuss common software security vulnerabilities and how to test for them

Presentation Abstract

Security issues are one of the highest concerns to many organizations. Despite this fact, security testing is often the least understood and least defined task. Security testing is a broad effort that requires a domain of expertise beyond traditional software testing. This one-day course teaches you security issues and outlines how software-testing roles fit in the big picture. The course focuses on security-related testing as it is applied to testing Web sites, Web applications, and software applications. This course is ideal for software testing professionals who are facing the new challenges and responsibilities of determining the scope of security testing to be done at the application level. QA professionals wanting to understand common vulnerabilities and hacking techniques used against Web sites and applications will especially benefit from this course. The course layout walks you through an overview of fundamental security issues including the hacking process and the strategies and technologies used in defending a system. Vulnerabilities in Web sites, Web applications and software applications as well as how the test for them will be discussed. Participants will come away with a clear understanding of testing for software system security, and many testing techniques, tools and resources that can be immediately applied to any project.

About the Author

Hung Q. Nguyen is founder, president, and CEO of LogiGear® Corporation. He’s held leadership roles in business management, product development, business development, engineering, quality assurance, testing, and information technology. Hung is an international speaker and contributor to industry publications. He authors and teaches software testing curriculums for LogiGear University and the University of California. He is the original architect of TRACKGEAR™, a Web-based defect management system, and the author of Testing Applications on the Web (Wiley). He also wrote (with Kaner and Falk) the best-selling book Testing Computer Software (Wiley), which is also published in Japanese. He holds a B.Sc. in Quality Assurance from Cogswell Polytechnical College, is an ASQ-Certified Quality Engineer, and a member of the Advisory Council for the Department of Applied Computing and Information Systems at UC Berkeley Extension.

QW2002 Tutorial C2

Mr. Robert L. Galen
(RGalen Consulting Group, LLC)

Software Projects -- Effective Estimating & Planning Techniques

Key Points

- An in-depth overview, with practice, of the presented techniques
- Skills for how to use collaborative, team based techniques to improve your work estimation
- Not only estimating, but how to use collaboration, team based techniques for planning
- Finally, skills in applying the right parts of the techniques to a variety of project situations

Presentation Abstract

The project and cultural dynamics of planning and estimating are one of the greater challenges facing technologists today. There is ever increasing pressure to “get things done”, so there is little “time” for estimating or planning. All too often, business derived dates or unrealistic dates drive projects. In conjunction with this, teams lack solid estimating and planning skills, usually resorting to “quick best guesses”, then compensating for poor estimates and plans with extremely hard work with a low probability of success. This workshop is intended to provide a review of three collaborative estimating and planning techniques that will enhance and improve your abilities to effectively estimate and plan your projects. Almost half the workshop is spent in exercises learning to apply the techniques.

About the Author

Robert Galen is employed at EMC Corporation in Research Triangle Park, NC as a Sr. QA & Test Manager. He has also recently started a consulting firm, RGalen Consulting Group, L.L.C., where he is Principal Consultant. Bob has held director, manager and contributor level positions in both software development and quality assurance organizations. He has over 20 years of experience working in the following domains:
- Computer systems
- Financial trading systems
- Mail processing equipment
- Medical diagnostics systems
- Telecommunications & network analysis equipment

He has a broad background that spans real-time embedded to client/server systems architecture and development in a wide variety of languages and technologies. Since the early 1990’s, he has been involved in successfully leading software development and process improvement initiatives. Areas of particular interest include:
- Requirement analysis and management
- Software project management
- Software quality, testing and metrics
- Software leadership and development team dynamics

Bob is an active member of ACM, ASQ, IEEE/CS, PMI and active (Program & Publicity Chair) in the local RTP-SPIN group - www.rtpspin.org. He is passionate about and committed to the profession of software engineering and product development.

QW2002 Tutorial D1 & D2

Mr. Erik Simmons
(Intel Corporation)

Writing Good Requirements

Key Points

- Discuss and choose from different techniques to specify requirements
- Improve written requirements and tell good requirements from bad ones
- Write non-functional requirements so they are verifiable

Presentation Abstract

Back by popular demand, in full-day format. This tutorial has earned consistent praise from attendees at Quality Week 2001, Quality Week Europe 2002 (average 4.9 out of 5 rating), and PNSQC 2000 (44 of 45 evaluations rated "valuable" or "very valuable"). Attendees enjoy the fast pace and depth of information presented. This material has been taught to nearly 4,000 students at Intel sites around the world. Poorly written requirements result in lost productivity, increased re-work, dissatisfied customers, poor end product quality, and even project cancellations. So, why are good requirements so hard to write? Many people do not know the key attributes of a "Good Requirement", and have not been exposed to the various effective ways to specify requirements.

This 1-day workshop focuses on and applies the best-known methods behind improved requirements writing. Based closely on a popular course taught at Intel, the course covers the different types of requirements and what activities are important when specifying requirements. The emphasis is on practical solutions to common problems, and contains valuable real examples from Intel documents in both original and improved formats. Students will gain an understanding of the attributes of a good requirement, and learn ways to identify whether the requirement is unambiguous, concise, necessary, correct, and traceable. Many useful "take it home and use it tomorrow" techniques for writing both functional and non-functional requirements are presented. Several exercises are included to reinforce the techniques. Attendees are invited to bring their existing requirements documents for use in the final exercise if desired.

About the Author

Erik Simmons has 15 years experience in multiple aspects of software and quality engineering. Erik currently works as Platform Quality Engineer in the Platform Quality Methods group, part of the Corporate Quality Network at Intel Corporation. He is responsible for Requirements Engineering practices at Intel, and lends support to several other corporate software and product quality initiatives. Erik is a member of the Pacific Northwest Software Quality Conference Board of Directors and the Steering Committee of the Rose City SPIN. He holds a Masters degree in mathematical modeling and a Bachelors degree in applied mathematics from Humboldt State University in California, and was appointed to the Clinical Faculty of Oregon Health Sciences University in 1991.

QW2002 Tutorial E1 & E2

Robert Sabourin
(Amibug.com)

Practical Web/e-Commerce Stress Testing

Key Points

- Stress testing for Web applications
- Performance testing for Web applications
- Load testing for Web applications
- Test automation for Load Testing of Web applications
- How stress can impact all quality factors
- Stress testing tools and techniques
- How to simulate large system loads
- How to implement site monitoring
- What is stress testing?
- When to apply stress testing
- How to assess software reliability
- How to assess software stability

Presentation Abstract

You know you should be doing stress testing, you're just not sure when. This workshop explores this question, which is particularly important to developers in Internet multi-tier development projects. You will review practical methods of assessing software reliability, and learn why it is important to "stress early and stress often."

This workshop focuses on stress testing Web and e-Commerce applications. You will learn how to find the weakest part of an application by testing to failure and then working with developers to determine the root cause of the defect. You will learn about the tools to support stress testing, including several available for free!

This workshop introduces you to using tools to automate some important parts of testing Web/e-Commerce applications. You will study how to organize and run stress testing, and review several sample automations.

Several examples based on commercial Web/e-Commerce applications will be described.

Benefits

After completing this workshop, you will understand many approaches to stress testing applications at all phases of development. After completing this workshop, you will understand how to plan and automate stress testing for Web/e-Commerce applications. After completing this workshop, you will know how and when to implement stress testing as part of the software development process.

About the Author

Robert Sabourin, P. Eng. has 20+ years management experience leading teams of software development professionals to consistently deliver projects on-time, on-quality and on-budget. He is a well-respected member of the software engineering community who has managed, trained, mentored and coached hundreds of top professionals in the field. He frequently writes and speaks to conferences around the world on software engineering, SQA, testing, management and internationalization.

QW2002 Tutorial F2

Mr. Rex Black
(RBCS, Inc.)

Managing The Testing Process: Test Estimation

Key Points

- Analyze risks to system quality to determine what should be tested.
- Create an actionable, realistic estimate of the tasks, dependencies, resources and time required.
- Adjust the estimated budget and schedule to fit project constraints.

Presentation Abstract

Based on the upcoming Second Edition of Rex's popular book, Managing the Testing Process, and nearly two decades of software, hardware, and systems experience, Rex Black will guide course attendees through a risk-based approach to creating realistic, actionable estimates of the testing tasks required for a project. Using techniques ranging from the informal to ISO 9126 to Failure Mode and Effect Analysis, Rex will demonstrate how to determine what you should test--and how extensively--by developing a prioritized list of risks to system quality. Rex will then use a hypothetical case study to translate that list of quality risks into an estimated schedule and budget via techniques like the Delphic Oracle, Three-Point, and Wideband team task sizing methods, plus the use of historical data and rules of thumb for additional accuracy. Finally, Rex will examine options for testers when they're asked to shrink their testing efforts into pre-existing schedule or budget targets, including risk-driven test effort descoping. Throughout, attendees will have a chance to solidify their understanding of the techniques through in-class discussion, Q&A, and three exercises.

About the Author

Rex Black (Rex_Black@RexBlackConsulting.com) is the President and Principal Consultant of Rex Black Consulting Services, Inc. *http://www.RexBlackConsulting.com*, an international software and hardware testing and quality assurance consultancy. He and his consulting associates help clients such as Bank One, Cisco, Compaq, Dell, Schlumberger, Williams Communications, and others with implementation, consulting, training, and staffing for testing, test automation, and quality assurance projects. His first book, Managing the Testing Process, published in June 1999, sold thousands of copies in North and South America, Europe, and Asia. In July 2002, Wiley published the updated and improved second edition. Following on the heels of these successful books, Addison-Wesley will publish Critical Testing Processes, Volume I and Volume II in late 2002 and early 2003, respectively.

QW2002 Tutorial G2

Cem Kaner
(Florida Institute of Technology)

Paradigms Of Black Box Software Testing

Key Points

- There are several different, effective styles of black box testing
- You can probably improve testing in your company by introducing one new style

Presentation Abstract

There are several substantially different approaches to black box testing. We'll look at nine of these approaches. In my experience as a consultant and teacher, moving through many different companies, many test groups rely heavily on only one or two of them and are virtually unaware of some of the alternatives.

The tutorial will consider each of the approaches in turn. For some of the approaches (the limiting factor will be time), we'll look at examples of the types of tests that someone would design when working within a given paradigm (or approach), the types of problems most easily found, and some types of problems that are more likely missed.

About the Author

Cem Kaner is Professor of Computer Sciences at the Florida Institute of Technology. Prior to joining Florida Tech, Kaner worked in Silicon Valley for 17 years, doing and managing programming, user interface design, testing, and user documentation. He is the senior author (with Jack Falk and Hung Quoc Nguyen) of TESTING COMPUTER SOFTWARE (2nd Edition) and (with David Pels) of BAD SOFTWARE: WHAT TO DO WHEN SOFTWARE FAILS.

Through his consulting firm, KANER.COM, he teaches courses on black box software testing and consults to software publishers on software testing, documentation, and development management. Kaner is also the co-founder and co-host of the Los Altos Workshop on Software Testing, the Software Test Managers' RoundTable, the Workshop on Heuristic & Exploratory Techniques, and the Florida Workshops on Model-Based Testing.

Kaner is also attorney whose practice is focused on the law of software quality. He is active (as an advocate for customers, authors, and small development shops) in several legislative drafting efforts involving software licensing, software quality regulation, and electronic commerce. Kaner holds a B.A. in Arts & Sciences (Math, Philosophy), a Ph.D. in Experimental Psychology (Human Perception & Performance: Psychophysics), and a J.D. (law degree). He is Certified in Quality Engineering by the American Society for Quality.

QW2002 Paper 1G1

Mr. Fred Baker
(Cisco Systems)

Internet Reliability under Stress

Presentation Abstract

The internet, which was designed for reliability through redundancy, has been criticized regarding its reliability on specific paths. Contributing factors have included the reliability of end system and infrastructure software, database design, provisioning of circuits with bandwidth, and control of traffic using policy and routing, to name but a few. The reliability of the network has come under test recently on 11 September, in the Code Red and Nimda virus attacks, and with the current Klez worm. Fred Baker, who contributes technically to Internet QoS standards, chairs the Internet Emergency Preparedness Working Group, and sits on the Internet Architecture Board, will address these issues and put them in context.

About the Author

Fred Baker has worked in the telecommunications industry since 1978, building statistical multiplexors, terminal servers, bridges, and routers.

At Cisco Systems, his primary interest areas include the improvement of Quality of Service for best effort and real time traffic, the development of routing and addressing, and issues in law enforcement and emergency use of the Internet. In addition to product development, as a Cisco Fellow, he advises senior management of industry directions and appropriate corporate strategies.

In addition, he is the chair of the Internet Society's Board of Trustees.

His principal standards contributions have been to the IETF, for which he served as IETF Chair in from 1996 to 2001. In that forum, he has contributed to Network Management, OSPF and Manet Routing, PPP and Frame Relay, the Integrated and Differentiated Services QoS architectures, and RSVP. He now serves on the IETF's Internet Architecture Board and chairs the Internet Emergency Preparedness Working Group, as well as directly contributing technically.

QW2002 Paper 1G2

Mr. Erik Simmons
(Intel Corporation)

The Human Side of Risk

Key Points

- We must better understand our tendencies and limitations as humans when it comes to assessing risks and estimating their severity and probability.
- We must better understand how the way we express a risk influences someone's response to it.
- We must better understand how culture, values, corporate history, and similar factors influence risk perception and management.

Presentation Abstract

A quick search using the terms Risk, Management, and Assessment on an Amazingly large book Website turns up 249 books on the topic, not to mention the chapters and sections devoted to various aspects of risk in texts on project management. Add hundreds of magazines, articles, conference proceedings, presentations, and courses at public and private institutions and you come up with a very large body of knowledge indeed.

But, the vast majority of that material concerns the analytical side of risk assessment and management. It's the other side of risk - the human side - where we need to increase our knowledge. If we don't, even the best analytical methods will be ineffective.

About the Author

Erik Simmons has 15 years experience in multiple aspects of software and quality engineering. Erik currently works as Platform Quality Engineer in the Platform Quality Methods group, part of the Corporate Quality Network at Intel Corporation. He is responsible for Requirements Engineering practices at Intel, and lends support to several other corporate software and product quality initiatives.

Erik is a member of the Pacific Northwest Software Quality Conference Board of Directors and the Steering Committee of the Rose City SPIN. He holds a Masters degree in mathematical modeling and a Bachelors degree in applied mathematics from Humboldt State University in California, and was appointed to the Clinical Faculty of Oregon Health Sciences University in 1991.

The Human Side of Risk

Erik Simmons, Intel Corporation

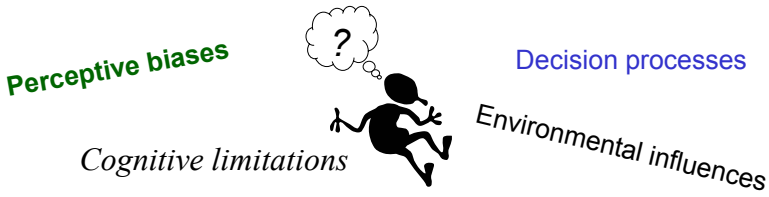
Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

The Study of Risk

Risk is a highly studied subject:

- 249 books found on an Amazingly Large Bookstore's Website
- Hundreds of papers, conference proceedings, tutorials, classes, and book chapters

But, few of us understand the human side of risk



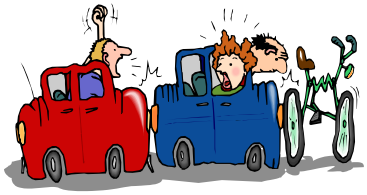
Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Are We Rational or Irrational?

What's the greater risk of death?

1:659,779

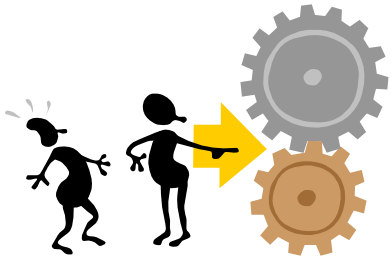
1:100



1:6,585

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Risk Perception
Influences and Factors



Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

The Two Sides of Risk

Analytical	Behavioral
• Logical • Data Driven • Scientific	• Emotional • Experiential • Irrational
Failure Modes & Effects Analysis (FMEA) Risk Management	Subjective assessment or ignorance of risk Gambling

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Some Risk Perception Influencers

- *Dread* associated with the risk
- Perceived **controllability**
- *Imposed* versus **voluntary** risk
- *Immediate* versus **delayed** consequences
- *Low* versus **high** knowledge about the consequences
- **Chronic** versus *acute* consequences
- *Severe* versus **non-severe** consequences
- *Signaling* ability for other similar or more severe events
- **Familiarity** (experience) with the risk
- *Availability* of examples

Heightens Perception	**Diminishes Perception**

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Software Examples

- *Dread:* Security flaws, product recall, loss of life
- *Imposed* versus **voluntary** risk: Schedule reductions by management versus the development team
- *Immediate* versus **delayed** consequences: Loss of fixed bid contract versus renegotiation later on
- *Low* versus **high** knowledge about the consequences: Commitment to support a unreleased OS version
- *Severe* versus **non-severe** consequences: Complete build failure versus a few hours lost to an SCM issue
- *Signaling* ability for other similar or more severe events: Performance issues in the first of many products on a new architecture

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Cultural & Environmental Influences

Culture and environment influence the way risk is dealt with, and when. For example, the US culture is marked by*:

- Insistence for choice
- Pursuit of dreams
- Big is better
- Impatience
- Tolerance of mistakes
- Urge to improvise
- Fixation on what's "new"

Various cultures might view risk awareness as pessimism, realism, or exemplary behavior

How does your company's culture and environment affect your perception of risk?

*Adapted from *The Stuff Americans are Made of,* J. Hammond and J. Morrison

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

The Message Matters

Suppose you had a rare medical condition for which surgery was the only cure

If there were two procedures available, would you chose:

1. A procedure where nearly 1 in 5 patients die in surgery
2. A procedure with 85% probability of survival

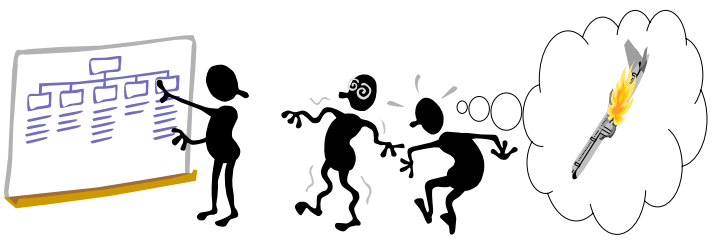
Hmmm… Do the options really feel the same?

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Unintended Messages

A fear of flying program had an expert explain to participants how unlikely it was for planes to crash

- The expert listed all the catastrophic failure modes and stated the probability of failure for each
- Result: The participants came away *more* afraid, because no one knew until then that there were so many ways a plane could crash!



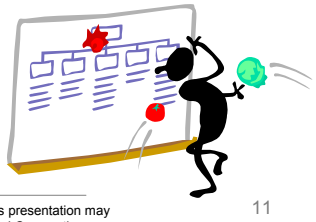
Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Too Much Information

Software risk spreadsheets can send the wrong message to other team members and management

Rigorous risk analysis and complete spreadsheets can cause:

- Added scrutiny from upper management (risk = problem)
- Individuals who state risks to be branded as “negative”, “not a team player”, etc.
- Avoidance of risks because “it’s too much to deal with”



Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

What Could We Do Instead?

Monitor all the risks, but consider using a format other than the entire spreadsheet to report on them. For example:

- Report only the Top 10 risks
- Report rolled-up metrics, such as total exposure (probability of occurrence X impact in time or dollars)
- Rotate the category of risk reported each meeting
- Educate people on risk management

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Cognitive & Behavioral Limitations

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Test Your Knowledge

How many people died as a result of motor vehicle crashes in the United States in 2000?

- Provide a point estimate and the smallest range that you are certain contains the true answer.
- Write your answer below

Point estimate:

Range:

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Questions:

Did you come close?

Was the real answer included in you range estimate?

How did you arrive at your estimate?

What influenced your estimation and decision process?

- Incomplete information
- Personal history
- Time pressure
- Cost of being wrong
- Familiarity with the risk
- And more…

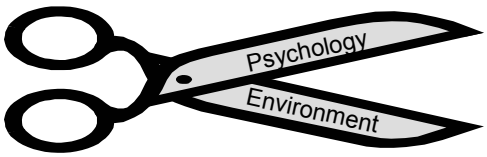
Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Bounded Rationality

A completely rational decision is practically impossible:

- We cannot know everything we would like before making decisions
- Even with incomplete information, the data complexity is often overwhelming
- Emotion also plays a significant and often unpredictable role in decisions

We are forced to create and use simplified heuristics and models to aid decision making



Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Decisions, Decisions…

Three basic building blocks for a decision heuristic:

- Search rule: How to locate alternatives
- Stopping rule: When to stop looking
- Decision rule: How to decide between alternatives

How much time should we spend deciding how much time to spend?

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Cognitive “Limitations”?

Fast and Frugal Heuristics for decision making can outperform more complex decision structures like regression and neural nets in certain environments

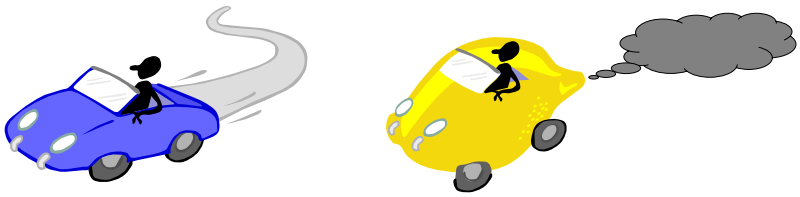
We must consider such heuristics in risk management

There are times when more information is not better

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Which Would You Choose?

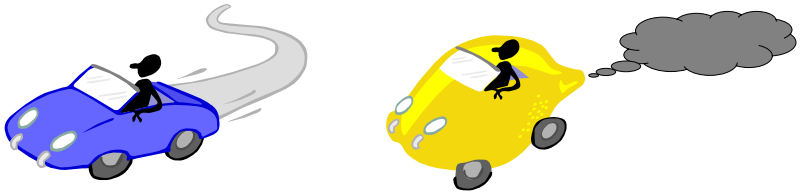
Car Name	Cost	Quality	Performance
Aardvark	Best	Best	Average
Belchfire Mark II	Worst	Average	Best
Conestoga Coupe	Average	Worst	Worst



Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

How About Now?

Car Name	Cost	Quality	Performance
Aardvark	Best	Worst	Average
Belchfire Mark II	Worst	Average	Best
Conestoga Coupe	Average	Best	Worst



Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Some Fast & Frugal Heuristics

- Take the first solution that meets minimum requirements in the most important dimension
- Take the best solution in the most important dimension
- Take the equally weighted best solution in all dimensions
- Imitate the most successful in cases where no clear best solution exists
- Imitate the majority in volatile or complex environments
- Take the recognized solution over unrecognized solutions

Schedule Security Performance Ease of Use Cost

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

On The Other Hand…

Experts sometimes run things intuitively rather than analytically

In those cases, new twists, unseen conditions, or novel combinations of events can be disastrous

Chernobyl happened to a very experienced team

Many reasons behind it, including

- Overconfidence from repeated safety rule violations without negative consequences
- Time pressure
- Perceived controllability

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

It Happens in Software, Too

An experienced team wants to skip inspecting a requirements document in order to save time. Why?

- *Time pressure & short term thinking:* over-insuring small, likely losses while under-insuring for catastrophic loss
- *Perceived controllability:* Exposures to this risk without catastrophic consequences in the past
- *Over-weighting local experience:* Generalization of local experience to infer a lower risk than really exists

What can you do about it?

- Express the risk over time rather than single incident
- Provide data that describes risk rates and consequences based on large samples and studies
- Advocate a balance between short and long term thinking

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Emotion, Risk, and Decision Making

Past success creates pride and self worth

- In this setting, people tend to chose risk minimizing strategies

Failures create stigma, shame and a lack of self worth

- When this happens, people tend to chose reward maximizing strategies and ignore risk

Dread can focus inordinate attention on certain risks and influence decision making

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Affect and Risk Perception

Affect is the feelings or emotions elicited by an external stimulus

Affect can strongly influence risk perception

What feelings and emotions do the following terms create?

- Ice cream
- Nuclear waste
- Tropical vacation
- Chemicals
- Public speaking

- Inspections
- Metrics program
- Agile methods
- Process improvement
- Object Orientation

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

What's the Greater Risk of Death?

1:659,779

1:2

1:373,787

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Summary

- We sometimes hold very inaccurate and irrational views about risks
- Risk means different things to different people, and risk perception is influenced in many ways
- Risk definition is an exercise in power. If you control how risk is defined, you can greatly influence which strategy is best in response
- More information is not necessarily better - it may confuse or alarm rather than inform
- Presentation matters!

intel

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Summary

- We must better understand our tendencies and limitations as humans when it comes to assessing risks and estimating their severity and probability
- We must better understand how the way we express a risk influences someone's response to it
- We must better understand how culture, values, corporate and personal history, and other similar factors influence risk perception and management

Otherwise, the best analytical tools and methods for risk will have little effect

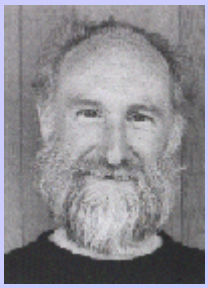
intel

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

For More Information

- *The Perception of Risk,* Paul Slovic, Earthscan Publications Ltd, 2000
- *The Logic of Failure,* Dietrich Dörner, Metropolitan Books, 1996
- *Bounded Rationality: The Adaptive Toolbox,* Gerd Gigerenzer and Reinhard Selten, MIT, 2001
- *Elements of Reason: Cognition, Choice, and the Bounds of Rationality,* Arthur Lupia et al, Cambridge, 2000

Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.



QW2002 Paper 5G1

Prof. Dick Hamlet
(Portland State University)

Science, Computer "Science", Mathematics, and Software Development

Key Points

- What is science?
- The traditional relationship among science, mathematics, and engineering.
- Why is software engineering different?
- So what is software, really?
- What's to be done?

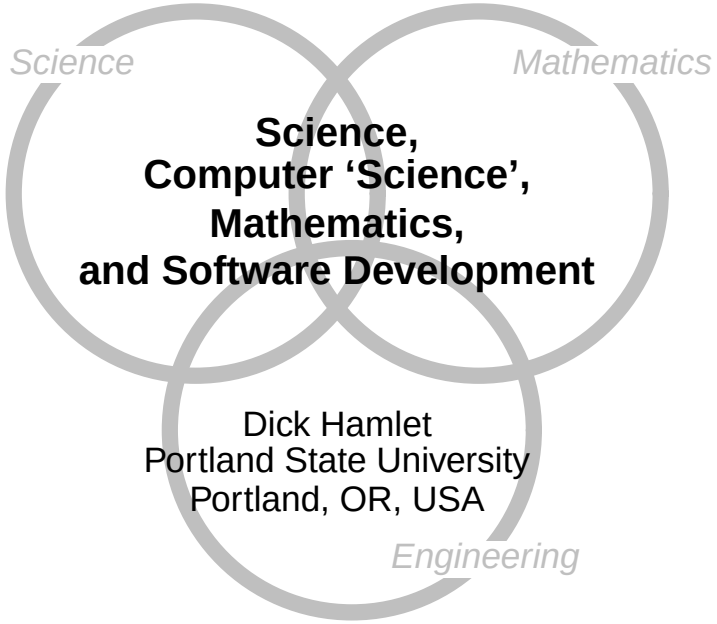
Presentation Abstract

Science is the basis for every field of engineering -- except software engineering. Software obeys no physical laws, not because our discipline is immature, but as an intrinsic property. Software is therefore given the hard problems to solve, because other technologies are limited; and, the software designer finds it hard to tell a good design from a bad one. What software engineers need is a way to say `no' to impossible requirements.

About the Author

Dick Hamlet is Professor of Computer Science at Portland State University. He has been active in software development and research for more than 30 years, as a programmer, manager, teacher, and researcher. He was a member of the software engineering research group at the University of Maryland for 12 years, a visiting lecturer at University of Melbourne in 1982, and a Fulbright scholar at National University of Ireland, Galway, in 1998-99. He is the author of three textbooks and more than 50 refereed conference and journal publications. He has implemented major software systems for two programming languages, the first mutation testing system, a transportable image-processing system, and a prototyping system for testing tools.

He holds a BS (electrical engineering) from the University of Wisconsin, MS (engineering physics) from Cornell, and PhD (computer science) from the University of Washington. Currently he is investigating the theoretical foundations of testing, and is principal investigator on a National Science Foundation grant to study the reliability of systems built from software components.



Science, Computer 'Science', Mathematics, and Software Development

Dick Hamlet
Portland State University
Portland, OR, USA

Outline of the Talk

I. Philosophy of Software Engineering

II. Mathematics, Science, Engineering

III. And for Software Engineering ... ?

IV. What's To Be Done About It?

Fundamental Questions

- What is software engineering?

"Whatever it is I do all day..."

... is not much good

There are good answers to similar questions:

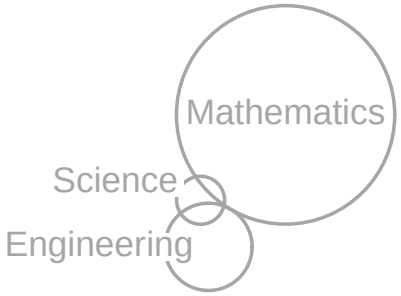
- What is physics?
- What is civil engineering?

(Some of those answers along the way)

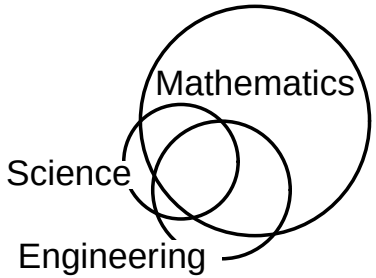
But What Use is It?

- Impress people at parties...
- Ask for a raise...
- Write a book and go on talk shows...
- Feel good (or bad) about your profession ...
- But knowledge may interfere with your work
  - ▷ Newton was seeking God's truth
  - ▷ (And everyone thought he'd found it)

Science, Math, and Engineering



In the beginning



100 years later

What Are The Disciplines?

Questions answered by:

Mathematics:
How to describe complicated things?

Science:
How does the world really work?
What are its natural laws?

Engineering:
How can I make it happen as I want?

Science & Mathematics	*Technology*
Understanding of reality	Human **control**

Formal Definitions (c. 2000)

(Mathematical) theory: A body of definitions, axioms, and theorems.

(Scientific) theory: A collection of assertions about reality that may be falsified by observation.
Ideally, applied mathematics.

Engineering design rules (Theory?): Systematic procedures for making artifacts, drawn from science, practice, invention, and experiment.

Applied Mathematics

- Start with a rich mathematical theory
- Identify the theory's objects with physical entities
- Check that the theory's axioms are true for those entities
- Exploit the theorems of the theory

☞ Creates a precise scientific theory ☜

Testing a Scientific Theory

Suppose a theorem of the mathematical theory (in the scientific theory) is observed not to hold.

We say, "The theory is wrong," meaning the scientific one.

- The axioms did *not* hold.

  OR

- Mathematical logic is wrong.

We much prefer this one

'Normal' Engineering Design

- Design rules tried and true, used before
- No new 'engineering science' allowed
- 'Safety factors' cover design errors
- Very likely to succeed in use
- Rare failures publicly analyzed

In 'pre-normal' times engineers can't work properly

Too many failures force design-rule change – ending a 'normal' period

(See Addis, inspired by Kuhn)

Summary: Traditional Paradigm

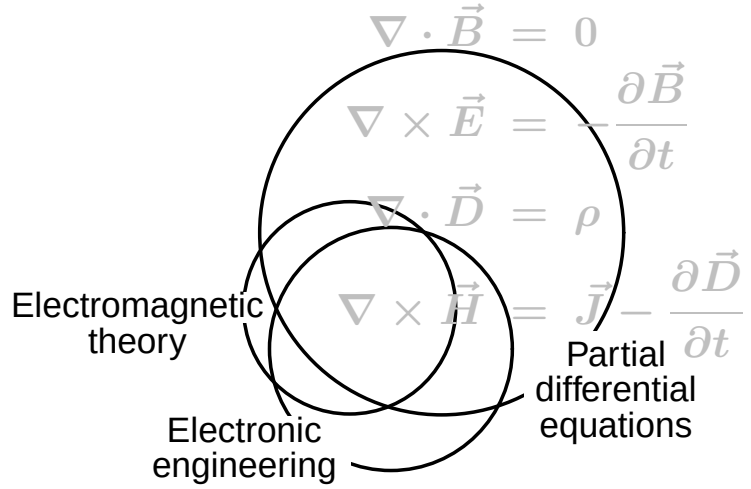
Science seeks to accurately describe the world's laws

Engineering design must conform to scientific laws – normal design removes some of the uncertainty (See Vincenti)

Mathematics is the handmaiden of science, the tool of engineering

☞ Spectacular success in mechanical, civil, aeronautical, and electrical engineering ☜

Traditional Engineering (Electrical)



☞ Maxwell equations: applied mathematics of electromagnetic theory ☜

Today's Philosophical Truths

- **Mathematics** isn't true or false.
  Mathematical objects are merely "the things that satisfy the axioms" (if any)
  We hope they also satisfy the theorems
- **Science** isn't objective, but 'theory saturated'.
  Science starts with a problem to be explained, then comes a theory, and finally observations testing theory (Karl Popper)
- **Engineering** design rules must be usable – they don't have to be scientific.
  Safety factors compensate for incorrect theories in the rules

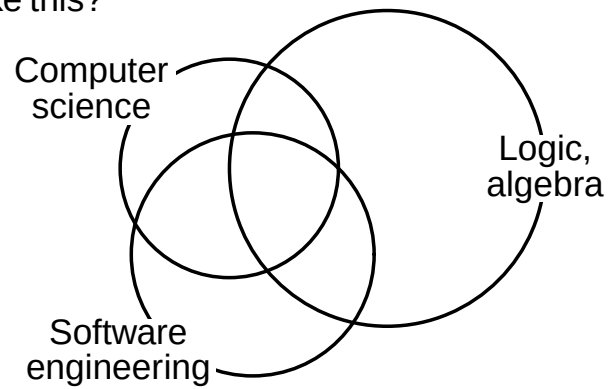
Quotations Supporting the Truths

- "Mathematics is the subject in which we never know what we are talking about, nor whether what we are saying is true." –Bertrand Russell
- "It is also a good rule not to put overmuch confidence in the observational results that are put forward until they are confirmed by theory."
  –Sir Arthur Eddington
- "The Doric design procedures ... were elegantly simple... They required the selection of a single fundamental 'module' equal to one half the diameter of a column, all parts of the work adjusted by means of calculations based upon it."
  –William Addis

Software *Engineering*?

- It *is* technology – software controls the world today

Is it like this?



Computer *Science*?

- Donald Knuth's volumes are entitled *The Art of Computer Programming*
- C.A.R. Hoare (q.v.) looks (forward!) to 'laws' and 'science' for programming. But (on Ada):

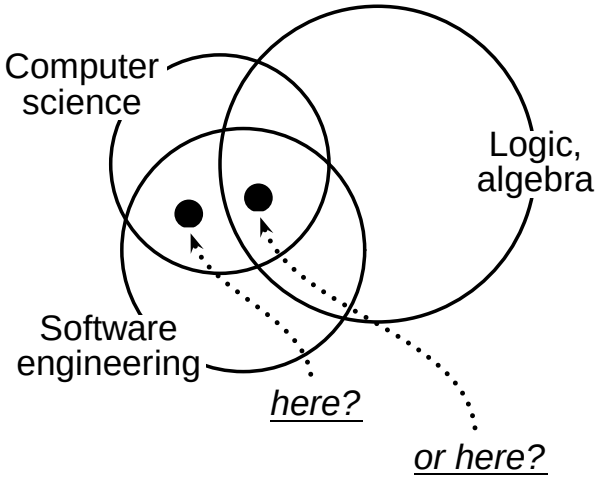
  "Almost anything in software can be implemented, and even used, given enough determination."

  ▷ Not even the DoD can buy physical laws
- In an accredited CS curriculum physics and calculus are required but not used
- "If a discipline has 'science' in its name, it isn't."

What is Programming?

- Programming is at the heart of CS
- There is no science of programming
  - ▷ Programming skill is taught by example
  - ▷ Particular programs are studied as artifacts
- What are the laws of programming?
  - ▷ You must use C++?
  - ▷ You must choose identifiers to company standard?
  - ▷ You must not write: `if X = Y` for `if (X == Y)` ?

Where Does Programming Fit?



Computer ‘Science’ Isn’t

- There are no falsifying experiments
  - ▷ ‘Experiment’ in CS means to implement an idea until it works
- Programming languages are invented
  - ▷ They can be changed at will (past time!)
  - ▷ Their properties can be *proved* (and if the proof fails, change something!)

☞ Scientists can’t change reality to fit theory ☜

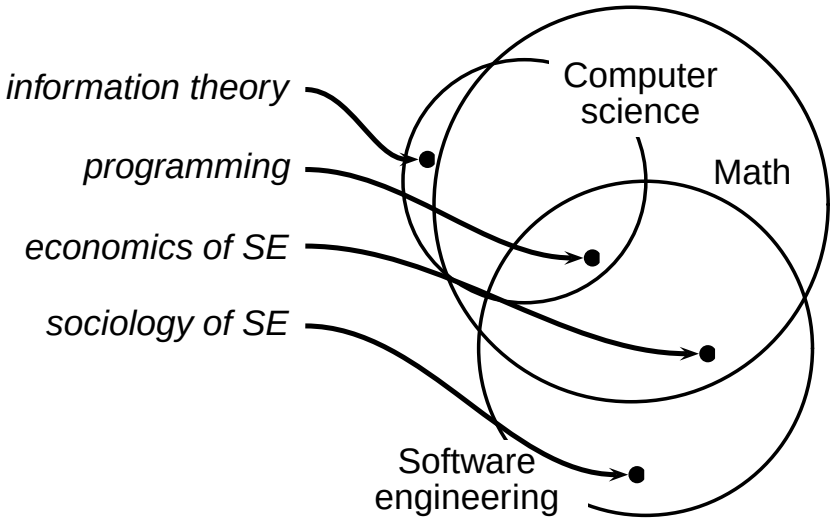
Digression: There’s *Some* Science...

- Information theory and undecidability (complexity) theory are something like thermodynamic laws
- ‘Science’ overlaps with ‘rational discussion’

  “I may be wrong and you may be right, and by an effort we may get nearer to the truth.”

  –Karl Popper
  - ▷ Mathematics and CS: consistency, depth, and elegance replace experiments
- The sociology and economics of software engineering are human laws, but they have immense inertia

Computer *Mathematics*?



Evaluating Engineering Designs

- Solving problems – an analogy:
  - Civil engineering: Design a bridge consistent with physical reality (may be impossible)
  - Software engineering: Design a program consistent with customer requirements (??)
- Einstein said that in making the laws of physics, God was “subtle, but not damn mean”
  - Sometimes customers *are* damn mean...

☞ How is a good design different from a poor one? ☜

Examples of Models in Engineering

- Aeronautical engineering: predict wing behavior
  - May be proved wrong by experiment
  - Failures covered by a safety factor
- Software engineering: predict efficacy of a testing strategy
  - Can be verified by mathematical proof
  - Failures may be arbitrarily bad

☞ In software, a model is mathematics to explain some other mathematics ☜

The Woes of the Craft

- Software gets to do the hard parts
  - The others, limited by natural law, can’t!
- Crackpot Requirements
- No professional tools
  - A builder needs a better saw than a hobbyist – but Windows Pro isn’t
- Ugly theory (bad mathematics)
- 60-hour weeks
  - “We delivered the compiler on time, but none of the marriages survived.”

All Problems Solved by Philosophy?

- Fundamental understanding should help us deal with our difficulties

☞ Software is not subject to natural law ☜

- Recommendations:
  - ▷ Take responsibility
  - ▷ Use good mathematics
  - ▷ Keep things straight

Take Responsibility

☞ Without natural law we have no one to blame but ourselves ☜

- Civil engineer's "No" backed by physical law
- Software engineer's "No" based only on sociology and economics
  - ▷ Crazy requirements and schedules
  - ▷ Releasing untested or failed software
  - ▷ But: no politics in "No"
- Better tools and working conditions

Use Good Mathematics

☞ Bad mathematics is a matter of choice ☜

- Safe programming languages 40 years old
  - ▷ Java isn't bad but for the wrong reasons
  - ▷ Giving a software engineer a Turing complete programming language is like giving a child an AK-47
- Redundant design – software 'safety factors'
- What about 'Formal Methods'?
  - ▷ Pro: Mathematical theory of a single program – capture it in deep theorems
  - ▷ Con: Mathematics may be all right for expressing God's high-quality laws...

Keep Things Straight

☞ Don't confuse sociology/economics with mathematics ☜

- Taylorism hides in software process

  "Management is not a skill or a craft or a profession but a command relationship; a sort of bad habit inherited from the army or the church."
  –A Lucas Aerospace worker

- eXtreme Programming fixes many woes
- Mathematics doesn't have to be 'validated'
  - ▷ To model bean counting it isn't necessary to amass thousands of beans and check 1 + 0 = 1, ..., 2 + 2 = 4, ...

Annotated Bibliography

William Addis, *Structural Engineering: The Nature of Theory and Design*, Ellis Horwood, 1991.

- A marvelous book by one of the few philosophers of engineering. Addis speaks as a civil engineer who has studied his discipline historically, and he is not daunted by the immense difficulty of really understanding the past. His purpose is to define engineering design, which he rightly believes is a better name for what engineers do than the overused ‘practice’.

More Bibliography

Thomas S. Kuhn, *The Structure of Scientific Revolutions*, 3rd ed., University of Chicago Press, 1996.

- Kuhn’s thesis is that science has ‘normal’ periods in which a dominant theoretical paradigm enables scientists to work productively, and ‘revolutions’ in which the theoretical paradigm is forced to change. This view of the field is arguably the most influential today. In particular, it has inspired engineers like Addis and Vincenti.

More Bibliography

Karl R. Popper, *Conjectures and Refutations: The Growth of Scientific Knowledge*, 5th ed., Routledge, 1992.

- Although Popper and Kuhn do not agree about intrinsic ‘truth’ in science (Kuhn thinks there is none, while Popper still hopes for it), they do agree that the usual descriptions of the so-called scientific method are nonsense. Popper’s view is that theory directs most scientific work, and that science is defined by theories that can be tested and may prove false. Popper believes strongly in the process of rational dispute, and he therefore calls mathematics a science.

More Bibliography

C.A.R. Hoare, “Programming: Science or Sorcery?,” in *Essays in Computing Science*, Prentice-Hall, 1989.

- Hoare presents his vision of a software profession. I would change his ‘science’ to ‘mathematics,’ and his ‘law’ to ‘theorem.’

Walter G. Vincenti, *What Engineers Know and How they Know it*, Johns Hopkins Press, 1990.

- Vincenti does not have Addis’s philosophical turn of mind, but he knows aeronautical engineering and has made a taxonomy of engineering knowledge, with good examples (especially on parameter variation).

QW2002 Paper 5G2

Mr. Robert V. Binder
(Mobile Systems Verification)

Achieving Very High Reliability for Ubiquitous Information Technology

Key Points

- Ubiquitous IT presents new challenges
- Highly automated testing can meet these challenges
- Achieving high reliablity is not an exotic luxury

Presentation Abstract

The human race is rapidly moving to ubiquitous information technology -- unthethered broadband to the hand: anyone, anything, anytime, anywhere. This adds several degrees of freedom to the already daunting combinatorics of testing. "Five nines" was once arcane telco jargon, but it now headlines press releases from Microsoft. In sharp contrast to hardware advances, present-day software technology is no better at achieving high reliability than it was ten years ago. Software development thus faces a significant challenge. This talk presents a strategy for achieving very high reliability (at least five nines) for ubiquitous IT: automated, mobile-aware, high-fidelity, profile-based, end-to-end testing. The strategy is explained through a brief experience report about its use to achieve very high reliability in large, high-volume, distributed application. The talk concludes with a sketch of how this approach is currently evolving for mobile technology.

Outline

Overview
The Vision
The New IT Reality
The Unchanged IT Reality: Software
Unique Wireless Test Problems
Reliability Arithmetic
Some Reliability Data Points
The New IT Reality: Reliability Sells!
The Engineering Challenge
The Strategy
Testing Process
Testing by Poking Around
Automated Test Script
Automated Generation/Agent
Full Test Automation
Test Effectiveness
Full Automation Case Study
Lessons Learned
Current Work - Wireless Testing System
Conclusion

About the Author

Robert V. Binder is internationally recognized as an expert on testing and author of the definitive *Testing Object-oriented Systems: Models, Patterns, and Tools*. As president of RBSC Corporation for fifteen years, he's lead many projects to design and build advanced automated testing systems.

QW2002 Paper 10G1

Mr. Don O'Neill
(Center for National Software Studies)

Competitiveness Versus Security

Key Points

- Participants will be able to reason about the practices and factors driving the competitiveness versus security trade off and assess the leading indicators underlying these factors for their own organization.
- While both are essential, it is clear that competitiveness and security travel on separate paths that do crisscross and overlap at certain points. The competitiveness versus security trade off may be tilted towards competitiveness, thereby, exposing the nation's critical software infrastructure to predictable security threats.

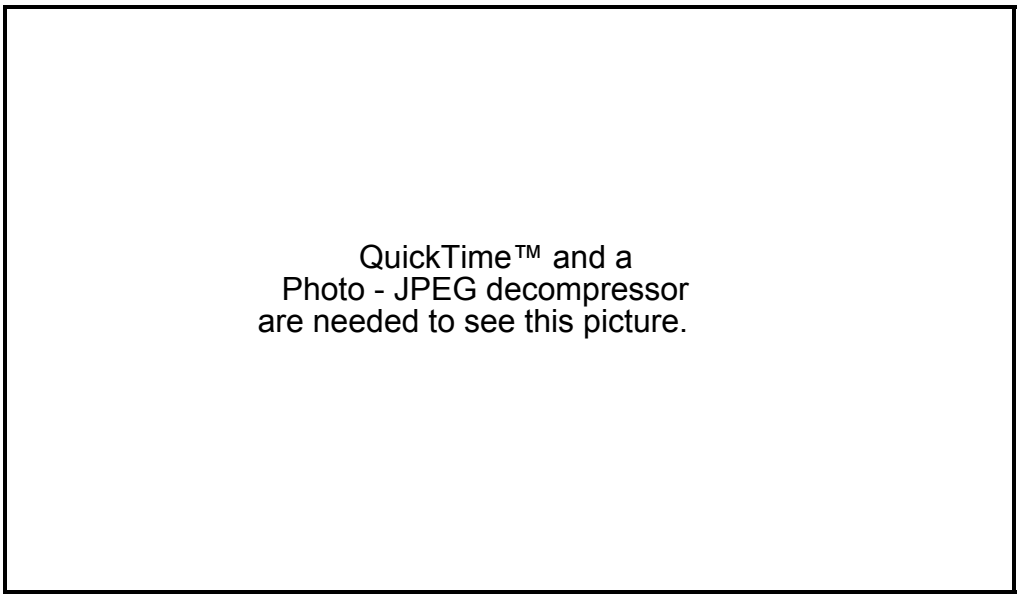
Presentation Abstract

There is an important national debate on CyberSecurity. It centers on who pays the bill, the private or public sector. On the one hand, the public sector argues that security and competitiveness move together, therefore, the private sector should pay the cost to be competitive. On the other hand, the private sector argues that security costs too much, and the probability of occurrence is too low to force the investment especially during the period of economic recovery.

The knowledge required in this trade off revolves around the practices and factors that embrace both competitiveness and security and those that embrace one at the expense of the other. A web-based scoring and analysis tool is used to assess the impact of the three types of practices and factors used to frame the issue including trustworthiness, cost effectiveness, and survivability. Leading indicators are identified for each practice and factor.

About the Author

Mr. Don O'Neill: Following his twenty-seven year career with IBM's Federal Systems Division, Mr. O'Neill completed a three year residency at Carnegie Mellon University's Software Engineering Institute (SEI) under IBM's Technical Academic Career Program. An independent consultant, he focuses on Software Inspections training, directing the National Software Quality Experiment, and conducting Global Software Competitiveness Assessments. He is a founding member of the Washington DC Software Process Improvement Network (SPIN) and the National Software Council (NSC) and serves as the Executive Vice President of the Center for National Software Studies (CNSS) http://www.CNsoftware.org. He is a collaborator with the Center for Empirically-based Software Engineering (CeBASE).



Competitiveness Versus Security

Considerations in Ensuring Future US Competitiveness In an Era of Increased Security Needs and The Role of Public and Private Collaboration

Don O'Neill
Executive Vice President
Center for National Software Studies

September 2002

Featuring

Abstract
The Many Dimensions of Security
Competitiveness
Who Pays the Bill?
The Trade Off Factors
Outlook
Conclusion

Competitiveness Versus Security

Considerations in
Ensuring Future US Competitiveness
In an Era of Increased Security Needs and
The Role of Public and Private Collaboration

Abstract

There is an important national debate on CyberSecurity. It centers on who pays the bill, the private or public sector. On the one hand, the public sector argues that security and competitiveness move together, therefore, the private sector should pay the cost to be competitive. On the other hand, the private sector argues that security costs too much, and the probability of occurrence is too low to force the investment especially during the period of economic recovery.

As Deming taught us, there is no substitute for superior knowledge. The knowledge required in this trade off revolves around the practices and factors that embrace both competitiveness and security and those that embrace one at the expense of the other. Three types of practices and factors are used to frame the issue including trustworthiness, cost effectiveness, and survivability. Leading indicators are identified for each practice.

A web-based scoring and analysis tool is used to analyze the impact of trustworthiness, cost effectiveness, and survivability practices and factors on competitiveness and security. A set of notional quick look scores are postulated for commercial, DOD industry, and government. Participants are asked what scores they would assign each practice and factor and are invited to exercise the tool to complete the analysis. An initial set of findings is suggested.

While both are essential, it is clear that competitiveness and security travel on separate paths that do crisscross and overlap at certain points. The competitiveness versus security trade off may be tilted towards competitiveness, thereby, exposing the nation's critical software infrastructure to predictable security threats.

The Many Dimensions of Security

We are experiencing the fall out from the lunge towards a paper less society without a technology infrastructure. As McNamara said during Vietnam, "If you don't watch the periphery, it will soon become the center." Security has become the center but a center that spans many dimensions.

CyberSecurity has many dimensions, and currently players are free to choose the dimension that best suits their background, experience, interest, or business objective. The challenge facing the country is to frame the issue realistically, to distill those factors that impact on the national interest, and to do so with intellectual honesty and not self interest. In large measure we are engaged in operation barn door.

What are the dimensions of security?

It spans threats, vulnerabilities, and readiness.
It spans the industry's underlying software architecture and environment and its inability to field trustworthy software systems.
It spans industry best practices and certification of processes, people, and products.

It spans the private and public sector and the tensions between them.
It spans legislative directions with their unintended consequences that impact security.
It spans business with its lack of essential driving incentive to promote security.

Security spans threats, vulnerabilities, and readiness. The primary software security focus needs to shift from threats and vulnerability to readiness and survivability. Threats are not well understood. Even as we struggle to determine the profile of future incidents, the analysis of past incidents yields only an incomplete and sometimes contradictory profile.

60% of security threats are random; 40% are targeted, but the degree of persistence is unknown.
100% of enterprises are attacked, but only 30% admit to being attacked.
70% of attacks are carried out by insiders.
17% of attacks attributed to industrial espionage and competitive intelligence.
Vulnerabilities are predominately in implementation not design. These vulnerabilities are examples of neglect and stem from unanticipated input, incorrect usage of protocols and connectivity, and accepting default settings. Understanding these vulnerabilities involves chasing down execution paths and their uncountably large number of possibilities.
The industry dependence on Microsoft products with its large pool of users and its common and numerous vulnerabilities greatly facilitates security intrusion into the nation's critical infrastructure accounting for 90% of all vulnerabilities.
Security cannot be bolted on; it must be designed in.
Some of the industry approaches to readiness are wrong. A few say that security depends on those doing the protecting. Some say security is a journey, not a destination. Many are approaching security as a process improvement activity.

Security spans the industry's underlying software architecture and environment and its inability to field trustworthy software systems. Industry must make the technical sacrifices and accommodations needed to achieve enterprise security and national security.

Security may require sacrificing certain preferred attributes of trustworthy software systems, such as, openness, interoperability, and modifiability.
Security may require sacrificing certain architectural styles in favor of those that facilitate ease of deterministic recovery and reconstitution following a security intrusion.

Security spans industry best practices and certification of processes, people, and products. The primary software security focus on industry practices and certification must shift from process and people to product.

Industry software configuration management practice is poor, and patches are made without adequate testing.
Beyond that, the industry practice is to procrastinate on implementing security patches because upgrades lead to problems, and personnel to test and retest are in short supply.

Security spans the private and public sector and the tensions between them. It is necessary to trade knowledge for power as the coin of the realm in seeking common ground in the public-private collaboration.

There is a public and private consensus that industry must lead in addressing security.
If the private sector doesn't come up with market driven security standards, then government will step up its regulatory pace.
However, the government itself has earned failing grades in its report card on security

readiness.

In addition the private sector is reluctant to report security intrusions to the government due to the Freedom of Information Act.

Security spans legislative directions with their unintended consequences that impact security. It is necessary to revise the legislative actions whose consequences are impacting national security.

Unintended consequences have accompanied UCITA, H1B High Tech Immigration Visa Program, Clinger-Cohen Act, and Freedom of Information Act.

Software companies often operate as services and not subject to product liability.

The availability of security liability insurance might diminish the incentive to improve the software security infrastructure. The insurers lack actuarial data on software security, and may demand compliance with good security practice as a prerequisite to underwriting insurance.

Furthermore contractors may be reluctant to support government security initiatives without indemnification made possible under Public Law 85-804 and related executive orders.

Security spans business with its lack of essential driving incentive to promote security. It is necessary to provide effective mechanisms that tilt the essential business calculation from cost effectiveness and competitiveness to trustworthiness, survivability, and security.

Enterprise management is driven by "quicker, better, cheaper" and cost effective software practices that enhance competitiveness while increasing security risk.

Quality concerns register with enterprise management ten times higher than security concerns.

The high cost of security readiness and the perceived low probability of impact due to security intrusion conspire to promote inaction.

In 2001, $13B in impact was attributed to security intrusion.

The enterprise must analyze what is to be protected and how important it is to be protected.

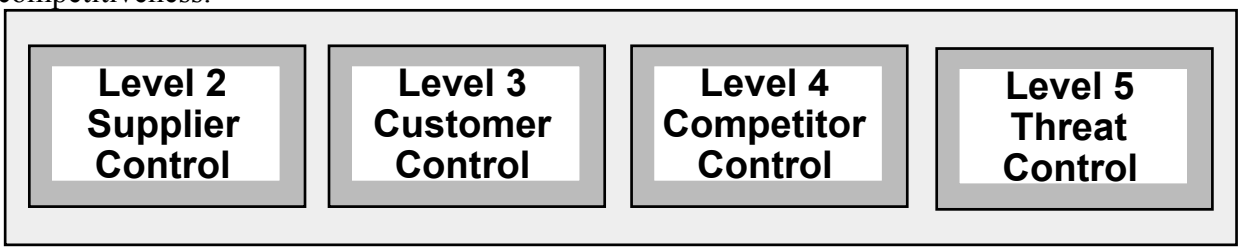
The scope of topics under the security tent is broad and deep; consequently there are no experts. Stovepipe knowledge is increasing with respect to past and current threats and vulnerabilties, but understanding and practicing readiness are lagging. Security threats come from unexpected places. This makes risk management difficult. The attempt to get a balanced security risk management program leads to nuanced approaches that look good under the uncritical light of management review but buckle under the intense glare of the factory floor and operating center. A collection of 90% approaches does not yield a 100% solution. The antidote for security threats is survivability. For enterprises with software operations at the center of the nation's critical infrastructure, nothing else will do.

Competitiveness

The government is responsible for prosperity, and industry is responsible for competitiveness. The leading indicators of prosperity span competitiveness, security, and infrastructure because without security and infrastructure, competitiveness cannot be achieved.

The Council on Competitiveness in Washington, D.C. defines competitiveness as the capacity of a nation's goods and services to meet the test of international markets while maintaining or boosting the real income of its citizens.

In software, competitiveness is achieved by providing fuel, setting direction, and controlling the environment including personnel resources, customer satisfaction and value add, competitors and new entrants, and event threats and change.There are five levels of global software competitiveness.



Level 1 is the absence of expectation, achievement, and engagement in the conversation on global software competitiveness.
Level 2 is associated with the availability of personnel skills and resources and their deployment.
Level 3 is associated with value to the customer derived through vigorous competition for current market niche with mature products that deliver value and earn customer satisfaction.
Level 4 is associated with competing for the future by setting the industry standard and practicing reuse and domain architecture technology to meet it.
Level 5 is associated with managing change and controlling event threats through strategic software management that raises the ability to improve to a core competence.

The elements of competitive software behavior revolve around commitment and how agreements are made and met, perfection and how to hold a participant's feet to the fire and when to cut slack, and people and dealing with the stresses of commitment and perfection. Supplier driven behavior is common in traditional and old style development with fewer suppliers competing. Market driven behavior is common in new and emerging style development with high customer bargaining power. Competition driven behavior is common in entrepreneurial style development with many suppliers competing.

Who Pays the Bill?

The government has bought-in on the security problem, but industry has not yet been sold. Industry appears to treat security as either a business challenge or a business opportunity, but it has not made a commitment to the essential investment in infrastructure. There is a public and private consensus that industry must lead in addressing security; however, with industry slow to take the lead, the government can be heard rattling its regulatory sword in the form of standards.

While the government approach to industry continues to favor market driven mechanisms, the government regulatory infrastructure is being readied.

National Security Telecommunications and Information Systems Security Policy No. 11 requires that all commercial off the shelf products must be certified by one of several agencies. These are software products that process, store, display, or transmit national security information. It became effective in July 2002.
Presidential Decision Directive 63 is intended to promote cooperation among industry and government. The interconnection of the various sectors of the nation's critical infrastructure introduces the risk of cascading consequences following a terrorist attack whether physical attack or CyberAttack. To counter this threat, Information Sharing and Analysis Centers (ISAC's) have been created to gather, analyze, and disseminate information and to promote

public-private cooperation. However, the Freedom of Information Act is throttling the willingness of industry to participate fully and share openly. Compliance with PDD 63 is achieved through vulnerability assessments using the InfoSec Assessment Training and Rating System administered by several organizations.

The availability of cyber insurance may tweak the business calculation, perhaps diminishing the incentive to invest in a secure technology infrastructure. To counter this, insurers may insist on the adoption of industry best security practices and compliance with security standards before issuing insurance. Currently actuarial data on security threats is incomplete.

There is an important national debate on CyberSecurity. It centers on who pays the bill, the private or public sector. On the one hand, the public sector argues that security and competitiveness move together, therefore, the private sector should pay the cost to be competitive. On the other hand, the private sector argues that security costs too much, and the probability of occurrence is too low to force the investment especially during the period of economic recovery.

The Trade Off Factors

As Deming taught us, there is no substitute for superior knowledge. The knowledge required in this trade off revolves around the practices and factors that enhance both competitiveness and security and those that enhance one at the expense of the other.

Three types of practices and factors are used to frame the issue including trustworthiness, cost effectiveness, and survivability.

Trustworthiness revolves around an engineering practice that tolerates change and yields dependability of results.

Well engineered software products are complete, correct, consistent, conforming, traceable, simple not complex, scalable, predictable, and usable.

Dependable software products are available, reliable, predictable, tested, defect free, fault free, failure free, stable, private, and safe.

Well engineered software products are change tolerant and are adaptable, extensible, interoperable, modifiable, and open.

Cost effective production is driven by a variety of factors involving personnel resources and skills and development environment and its process, methods, and tools. Specifically there has been a heavy dependence on several approaches:

The use of foreign nationals and offshore outsourcing.

The incorporation of commercial off the shelf products.

The deepening of community relations through collaborative research.

The management of personnel factors in particular personnel retention.

Survivability spans the resistance to CyberAttack, the recognition of a CyberAttack, and the reconstitution of enterprise software operations following a CyberThreat or CyberAttack. Survivability is achieved through the right blend of function, form, and fit.

Function includes user authorization, access control, encryption, firewalls, proxy servers, normal operation monitoring, backup and shadow operations, data and program restoration, and disaster recovery.

Form includes dispersion of data, diversification of systems, rules of construction, state data isolation, disciplined data, intrusion usage patterns, virus scans, internal integrity, secure state data monitor, exception handlers, full system state architecture, minimum essential function, and isolation of damage.
Fit includes adherence to loading limits, predictable response, no memory leaks, rate monotonic scheduling, time line or event driven scheduling, monitor memory management, time line predictability, watch-dog timer, full system predictability.

	Competitiveness	Security
Engineering Practice	+	+
Dependable Product	+	+
Change Tolerance	+	- [Ease of Change]
Cost Effectiveness	+	- [Foreign Nationals, COTS]
Deep Comm unity Rel.	+	- [Collaborative Research]
Personnel Management	- [Personnel Retention]	- [Personnel Retention]
Survivability	- [Resist, Recognize, Reconstitute]	+

Leading indicators are identified for each practice and form the basis for the trade off which is structured along the following lines:

Engineering practices and dependable product factors enhance both competitiveness and security.

While change tolerance and ease of change benefit competitiveness, they also provide easy access for those with malevolent intent.

While cost effectiveness benefits competitiveness, some of the means for achieving it present security exposures.

While foreign nationals are skilled and cheap, they possess the means in the form of superior knowledge and access to intrude on the nation's critical infrastructure and they lack allegiance to the United States.

While commercial off the shelf products provide quick and cheap solutions, they are produced with unknown workforces using unknown practices that yield unknown trustworthiness, a security exposure.

While collaborative research with appropriate intellectual controls is necessary to achieve high maturity in competitiveness, this same knowledge could be used to launch a highly intelligent security intrusion.

Personnel turnover impacts both competitiveness and security; deep domain knowledge must kept be intramural.

Survivability practices essential for security impact competitiveness through added cost, product inconvenience, and increased complexity.

The leading indicators selected to characterize the practices and factors of competitiveness and security are drawn from the attributes of Trustworthy Software Systems, Global Software Competitiveness, and CyberSecurity Survivability.

Leading Indicators of Competitiveness and Security

Engineering Practice
- Complete
- Correct
- Consistent
- Conforming
- Traceable
- Low complexity
- Scalable
- Predictable
- Usable

Dependable Product
- Available
- Reliable
- Predictable
- Tested
- Defect free
- Failure free
- Fault free
- Stable
- Private
- Safe

Change Tolerant
- Adaptable
- Extensible
- Interoperable
- Modifiable
- Open

Foreign Nationals and Outsourcing
- Immigration Policy
- Domestic Outsource
- Offshore Outsource

Commercial Off the Shelf
- Reuse Technology Practice
- Product Line Practice
- Domain Architecture

Deep Community Relationships
- Collaborative Research
- Government Research
- University Research

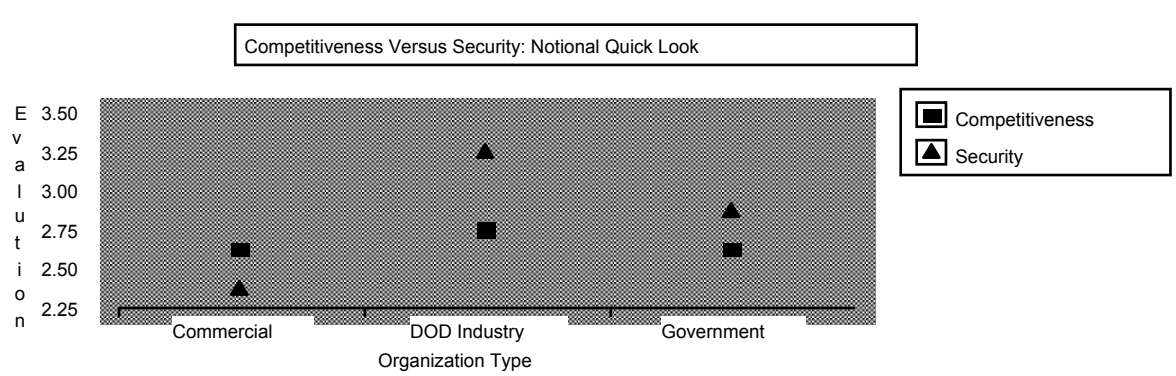
Personnel Management
- Open Requisitions
- Personnel Turnover
- Staff Churn

Survivability
- Resistance
- Recognition
- Reconstitution

A web-based scoring and analysis tool is being used to assess the impact of trustworthiness, cost effectiveness, and survivability practices and factors on competitiveness and security. A set of notional quick look scores are postulated for commercial, DOD industry, and government. Participants are asked what scores they would assign each practice and factor and are invited to exercise the tool to complete the analysis [http://members.aol.com/ONeillDon2/comp-sec_frames.html]. Using this tool the factor impact analysis was conducted to analyze the behavior of trustworthiness, cost effectiveness, and survivability.

Each practice and factor is rated from low to high on a 1 to 5 scale. The expressions used to evaluate competitiveness and security are:

competitiveness=(engineering+dependable+change+foreign+cots+research+(4-personnel)+(4-survivability))/8
security=(engineering+dependable+(4-change)+(4-foreign)+(4-cots)+(4-research)+(4-personnel)+survivability)/8



Practice	Commercial	DOD Industry	Government
Engineering Practice	**1**	**3**	**2**
Dependable Product	**2**	**3**	**1**
Ease of Change	**2**	**3**	**1**
Foreign Nationals	**4**	**2**	**3**
Commercial Products	**4**	**2**	**2**
Collaborative Research	**2**	**4**	**3**
Personnel Management	**4**	**3**	**2**
Survivability	**2**	**4**	**1**

Outlook

While both are essential, it is clear that competitiveness and security travel on separate paths but paths that do crisscross and overlap at certain points. This competitiveness versus security trade off may be tilted towards competitiveness thereby exposing the nation's critical infrastructure to predictable security threats.

Chief among these exposures is the need to rebalance certain cost effectiveness practices used in the production of the nation's software systems.

The emphasis on "better, quicker, cheaper" is leading to an increased use of ad hoc programming practice which yields a high rate of defects and failures, necessitates a high frequency of release, and results in highly complex software systems, all security exposures.

The increased dependence on foreign national sources has produced a massive population of high tech workers who possess superior knowledge and access to the nation's software systems but who lack allegiance to the United States, a security exposure.

The emphasis on commercial off the shelf software is resulting in widespread usage of software produced by an unknown workforce using unknown practices and yielding unknown trustworthiness, a security exposure.

In addition, the industry software capacity and capability needs to be improved.

Industry capability to produce trustworthy software systems depends on the software product engineering methodology practiced. The ad hoc programming method currently practiced must be replaced with structured and disciplined software engineering.

The industry must possess the capability to counter the CyberSecurity threat by resisting and recognizing CyberAttacks and reconstituting software operations following an attack. The steps to reconstitute critical software operations center around the technology for ensuring

continuous operations, backing up and switching over, and restarting critical operations.

Also legislative directions must be revisited. When it comes to software and security, our nation's leaders are ignorant. As a result, legislation enacted for some purpose may possess side effects that lead to unintended consequences. Several legislative factors impacting security include:

UCITA excuses defects, faults, and failures providing an incentive for producing less dependable software products, a security exposure. Software vendors are inoculated from lawsuits brought by the users of these products.

H1B High Tech Immigration Visa Program facilitates the placement of foreign nationals as key employees in the software operations of nation's software systems, a security exposure.

Clinger-Cohen Act calls for commercial off the shelf usage, a security exposure.

Freedom of Information Act is a barrier to the public-private partnership and trust essential to sharing CyberSecurity incident data, a security exposure.

Finally the government needs to weigh-in on the side of security to rebalance the competitiveness versus security trade off. The government favors a market driven approach to solving the software security problem. While there is a public-private consensus that industry must take the lead in addressing security, industry has been slow to do so.

Accordingly, the industry can look to government to adopt a software security tax credit policy. This tax credit will favor those enterprises in the nation's critical infrastructure whose behaviors and practices effectively address security and who share security information. The recipients of the software security tax credit will be those who post highly pro-active responses to the following questions:

Does the enterprise maintain a software security policy on what software operations are critical and why they are critical?

Is the enterprise and its software operation a part of the nation's critical infrastructure?

If it ceased to continue software operations, are there cascading consequences that would impact the critical infrastructure and are these well defined including mitigation strategies?

Does the software security improvement for which the tax credit is being sought produce a measured benefit in the resistance and recognition of software security threats and the reconstitution of software operations following an alert or attack?

Does the enterprise compute a software security index of suspicion using current and credible threat information and up to date readiness information and use the result in connection with its software security policy?

Does the enterprise share software security incident attack and impact information with the government agency responsible for its collection, analysis, and dissemination?

Conclusion

When it comes to security, knowledge must replace both power and money as the coin of the realm. Both government and industry have responsibilities to reconcile the conflicting factors encountered in seeking both competitiveness and security.

While the government cannot make us safe from CyberAttack, the government can tilt the business calculation towards security through tax credits and insurance mechanisms designed to incentivize readiness.

Since the industry's software products make us vulnerable to CyberAttack, industry must

make the sacrifices needed to achieve security by rebalancing its cost effectiveness tactics and ensuring the readiness and survivability of software products.

3602 words

Bibliography: Security

[ACM 99] Lipson, Howard F. and David A. Fisher, "Survivability- A New Technical and Business Perspectives on Security", ACM Proceedings of the 1999 New Security Paradigms Workshop, 22-24 September 1999, Ontario, Canada

[Alford 01] Alford, Lt.Col. Lionel D. Alford.Jr., "Cyber Warfare: A New Doctrine and Taxonomy", CrossTalk: The Journal of Defense Software Engineering, Vol. 14 No. 4, April 2001

[ATAM] Architecture Tradeoff Analysis Method (ATAM), Software Engineering Institute, Pittsburgh, Pa.

[CeBase 01] Basili, Victor R. and Barry Boehm, "COTS-Based Systems Top 10 List", Computer, Mat 2001, pp. 91-93

[Computerworld 5/9/02] "At Senate Hearing, CyberTerrorism Fears on the Rise", Computerworld, Dan Verton, 9 May 2002

[Computerworld 4/8/02-1] "Outflanking the CyberTerrorist Threat", Computerworld , Dan Verton, 8 April 2002

[Computerworld 4/8/02-2] "Terrorism 101 With Eric Shaw", Computerworld , 8 April 2002

[CNSS 01] "Center for National Software Studies: Prospectus & Strategic Plan", Center for National Software Studies, 2 January 2001
http://www.CNsoftware.org

[CURE] COTS Usage Risk Evaluation (CURE), Software Engineering Institute, Pittsburgh, Pa.

[NIST 02] Tassey, Gregory. "The Economic Impacts of Inadequate Infrastructure for Software Testing", Final Report, National Institute of Standards and Technology, RTI Project Numer 7007.011, May 2002

[OCTAVE] Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE), Software Engineering Institute, Pittsburgh, Pa.

[O'Neill 02] O'Neill, Don, "Homeland Security Infrastructure for Critical Software Operations", The Competitor Vol 5 No 4, March 2002
http://members.aol.com/ONeillDon2/competitor5-4.html

[O'Neill 01] O'Neill, Don, "An Introduction to Global Software Competitiveness", submitted to CrossTalk, The Journal of Defense Software Engineering, submitted August 2001

[O'Neill 01] O'Neill, Don, "Country Report on the US Software Industry", IEEE Software Magazine, November/December 2001, submitted by request August 2001
http://members.aol.com/oneilldon2/competitor5-3.html

[O'Neill 99] O'Neill, Don, "Set Direction, Provide Fuel, and Control Environment... Be Globally Competitive", e-GOV Journal, The Journal for Electronic Commerce, Volume 2 Issue 1, December/January 1999
http://members.aol.com/oneilldon2/new_vol1 no3.html

[O'Neill 98] O'Neill, Don, "Threats to the Nation's IT Resources and Potential Management Responses", Electronic Government Journal, March/April 1998, Page 6
http://members.aol.com/oneilldon2/competitor2-1.html

[O'Neill 97] O'Neill, Don, "Software Value Add Study", ACM Software Engineering Notes, Vol 22 No 4, July 1997
http://members.aol.com/oneilldon2/new_competitor_initial.html

[Post 01] "Government Gets an 'F' in Computer Security", Washington Post, page A4, 10 November 2001

[SEI 01] Linger, Richard C. and Andrew P. Moore, "Foundations for Survivable System Development: Service Traces, Intrusion Traces, and Evaluation Models", CMU/SEI-2001-TR-029, October 2001

[SEI 98] Ellison, Robert J. et al, "Survivability: Protecting Your Critical System", IEEE Proceedings of the International Conference on Requirements Engineering, 6-10 March 1998, Colorado Springs, Colorado

[Schneier 01] Schneier, Bruce, "Managed Security Monitoring: Network Security for the 21st Century", Counterpane Internet Security, 2001

[Standish] "Chaos Report", Standish Group,
http://www.scs.carleton.ca/~beau/PM/Standish-Report.html

[Vatis 01] Vatis, Michael A., "Cyber Attacks During The War on Terrorism: A Predictive Analysis", Institute for Security Technology Studies at Dartmouth College, 2001

Bibliography- Competitiveness

[Allaire 96] Allaire, Paul et al, "Endless Frontier, Limited Resources: US R&D Policy for Competitiveness", Council on Competitiveness, 1996

[Byrne 00] Byrne, John A, "Management by Web", Business Week, 28 August 2000

[Carmel 99] Carmel, Erran, "Global Software Teams", Prentice Hall, 1999, 269 pages

[CIO 00] "A Passage to India", CIO magazine, December 2000

[COC 96] Council on Competitiveness, "U.S. Competitiveness: A Ten Year Strategic Assessment", Washington DC, October 1996

[COC 97] Council on Competitiveness, “Global R&D Choices”, Washington DC, 1997

[Competitor 98] O’Neill, Don, “Microsoft Antitrust Lawsuit and Its Impact on Global Software Competitiveness”, The Competitor Vol. 1 No. 6, July 1998 http://members.aol.com/oneilldon2/competitor1-6.html

[Fed 00] Cox, Michael, “16 Stats on the New Economy”, Federal Reserve Bank of Dallas, 2000

[Hamel 96] Hamel, Gary and C.K. Prahalad, “Competing for the Future”, Harvard Business School Press, 1996, 357 pages

[Humphrey 89] Humphrey, Watts S., "Managing the Software Process", Addison-Wesley Publishing Company, Inc., 1989, 494 pages

[Kelly 98] Kelly, Kevin, “New Rules for the New Economy”, Penguin Group, 1998, 179 pages

[Moitra 01] Moitre, Deependra, “Country Report on India’s Software Industry”, IEEE Software Magazine, January 2001

[Moore 96] Moore, James F., “The Death of Competition”, Harper Business, 1996, 297 pages

[O’Neill 97.1] O’Neill, Don, “Software Value Add Study”, ACM Software Engineering Notes, Vol 22 No 4, pp. 22-24, July 1997

[O’Neill 97.2] O'Neill, Don, "Global Software Competitiveness Assessment Program", Quality Week Europe Conference, Brussels, 1997

[O’Neill 97.3] O’Neill, Don, “Software Maintenance and Global Competition”, Journal of Software Maintenance: Research and Practice, John Wiley and Sons, Ltd., Vol. 9, pp.379-399, November/December 1997

[O’Neill 98] O’Neill, Don, “Threats to the Nation’s IT Resources and Potential Management Responses”, Electronic Government Journal, p. 6, March/April 1998

[O’Neill 99] O’Neill, Don, “Set Direction, Provide Fuel, and Control Environment... Be Globally Competitive”, e-GOV Journal, The Journal for Electronic Commerce, Volume 2 Issue 1, p. 19, December/January 1999

[O’Neill 95,96,00] O'Neill, Don, "National Software Quality Experiment: A Lesson in Measurement”, Software Technology Conference, Salt Lake City, 1995, 1996, and 2000

[Paulk 95] Paulk, Mark C., “The Capability Maturity Model: Guidelines for Improving the Software Process”, Addison-Wesley Publishing Company, 1995

[Peterson 95] Peterson, Ivars, "Fatal Defect", Random House, Inc., New York, 1995, 260 pages

[Pyster 96] Pyster, Arthur et al, "Record of the National Software Summit", National Software Council, January 1996

[SEI 96] Ford, G. and N.E. Gibbs, "Mature Profession of Software Engineering", CMU/SEI-96-TR-004, Software Engineering Institute, Carnegie Mellon University, 1996

[SEI 01] "Process Maturity Profile of the Software Community 2000 Year End Update, Software Engineering Institute, Carnegie Mellon University, 2001

[SPC 98] "The Frameworks Quagmire", Software Productivity Consortium, http://www.software.org/quagmire

Don O'Neill

Don O'Neill is a seasoned software engineering manager and technologist currently serving as an independent consultant. Following his twenty-seven year career with IBM's Federal Systems Division, Mr. O'Neill completed a three year residency at Carnegie Mellon University's Software Engineering Institute (SEI) under IBM's Technical Academic Career Program. There he developed a blueprint for charting software engineering evolution in the organization including the training architecture and change management strategy needed to transition skills into practice.

As an independent consultant, Mr. O'Neill conducts defined programs for managing strategic software improvement. These include implementing an organizational Software Inspections Process, directing the National Software Quality Experiment, implementing Software Risk Management on the project, conducting the Project Suite Key Process Area Defined Program, and conducting Global Software Competitiveness Assessments. Each of these programs includes the necessary practitioner and management training. As an expert witness, he provides testimony on the state of the practice in developing and fielding large scale industrial software and the complex factors that govern their outcome.

In his IBM career, Mr. O'Neill completed assignments in management, technical performance, and marketing in a broad range of applications including space systems, submarine systems, military command and control systems, communications systems, and management decision support systems. He was awarded IBM's Outstanding Contribution Award three times:

Software Development Manager for the Global Positioning (GPS) Ground Segment (500,000 source lines of code) and a team of 70 software engineers within a $150M fixed price program.

Manager of the FSD Software Engineering Department responsible for the origination of division software engineering strategies, the preparation of software management and engineering practices, and the coordination of these practices throughout the division's software practitioners and managers.

Manager of Data Processing for the Trident Submarine Command and Control System Engineering and Integration Project responsible for architecture selections and software development planning (1.2M source lines of code).

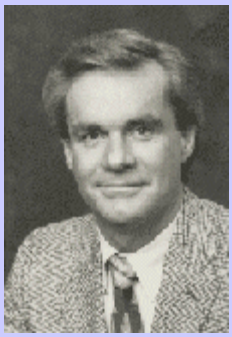
Mr. O'Neill served on the Executive Board of the IEEE Software Engineering Technical Committee and as a Distinguished Visitor of the IEEE. He is a founding member of the Washington DC Software Process Improvement Network (SPIN) and the National Software Council (NSC) and serves as the Executive Vice President of the Center for National Software Studies (CNSS). He is a collaborator with the Center for Empirically-based Software Engineering (CeBASE). He is an active speaker on software engineering topics and has numerous publications to his credit. Mr. O'Neill has a Bachelor of Science degree in mathematics from Dickinson College in Carlisle, Pennsylvania.

Mission of CNSS

The Center for National Software Studies (CNSS) is a public policy research organization established as a non-profit 501(c)(3) status. The CNSS is a private corporation governed by a board of directors and accepts funding through contributions and grants. With a mission to elevate software to the national agenda, the CNSS is set up to provide objective expertise, studies, and recommendations on national software issues. The software issues of national importance identified by the CNSS include:

- Software Value to US Economic Competitiveness
- Software System Trustworthiness
- Research and Development Funding
- Software Workforce Issues
- Maintaining Security and Privacy in Electronic Commerce
- Protecting Intellectual Property and Preventing Piracy

Currently in Phase I, the CNSS startup operation is a web-based eCenter intended to prove its viability and value as a national resource. Background information is available in the CNSS Prospectus & Strategic Plan and the CNSS web page at http://www.CNsoftware.org.



QW2002 Paper 10G2

Mr. Gregory Pope
(University of California LLNL)

You Want It When?

Key Points

- We seldom get all the test time we need
- Most of us don't know how to negotiate
- Specific proven tactics will get you more time

Presentation Abstract

This presentation includes proven tactics for negotiating software testng schedules.

About the Author

Gregory M. Pope is a Computer Scientist at the University of California Lawrence Livermore Laboratory. Over the past quarter century, Mr. Pope has worked in a variety of capacities at virtually all levels of software development and testing. He began his career developing software used to test jet engines and helicopters. He later worked in the defense industry, testing missioncritical software for military and space applications. Working in the private sector, he has developed and patented techniques for computer-aided testing. Among his inventions is the Ferret, a highly regarded software-testing tool manufactured and marketed by his company.

As a consultant and teacher, Mr. Pope has conducted hundreds of seminars for software development professionals throughout the United States, Asia, Canada, Mexico, South America, and Europe. In his thriving consulting, training, and testing practice, he has worked with many Fortune 500 companies, including Microsoft, IBM, Apple Computer, Sun Microsystems, AT&T, Eastman Kodak, DHL Airways, and Knight- Ridder, as well as NASA, the Pentagon, U.S. defense contractors, the Internal Revenue Service, and numerous foreign companies.

Mr. Pope is sought out to write articles on the subject of software testing for a number of industry publications, including CIO Magazine, Computer Design, Industry Week, Computer World, Signal, Electronic Defense News, San Jose Mercury, and Software Maintenance News. He holds a BS degree from Connecticut State University, an MBA from University of Phoenix, and is a member of IEEE.

You Want it When?

Estimating and Negotiating Test Schedules

By:
Gregory M. Pope
University of California
Lawrence Livermore National Laboratory

Quality Week
San Francisco, CA
September 6, 2002

Slide 1

Heard This Before ?

- It is January 1st and marketing has an idea
- "Must get the new product out by June 1st"
- "Development says it will take 5 months"
- "That leaves a month to test, you can do it"
- "We will use the FAD methodology"

Slide 2

FAD Methodology
(Fantasy Application Development)

- Step 1 - Announce product and release date
- Step 2 - Design logo and make T-Shirts
- Step 3 - Determine what the product is
- Step 4 - Estimate development time
- Step 5 - Write the code and web pages
- Step 6 - Write the spec (optional)
- Step 7 - Beta release (ready or not)
- Step 8 - Give incomplete version to Test
- Step 9 - Announce upgrade program and patches

Slide 3

What If Other Industries Used FAD ?

- Pharmaceuticals - Here is a new drug, works great on frogs and pigs, you can try it at no charge. (Beta Program)
- Automotive - It is a great car, and in 3 months we are updating it (a small fee for you) to add the trunk and reverse gear. (Upgrades)
- Aviation - Welcome aboard, today we are going to be the first aircraft to try the new GE engines with the old Bendix flight controls. (Platforms)
- Airlines - Sorry for the delay, but the latest version of our cockpit software no longer supports landing at our original destination, so we are going to Pittsburgh instead. (Version Compatibility)

Slide 4

Informal Survey - Test Schedule (1993-2002 - 6,000 Developers/Testers)

- Testers having too much time to test - 0
- Testers having the right amount of time to test - 250
- Testers having too little time to test - 5,750

Slide 5

Why Incorrect Estimates Lengthen Project Time

- Time is up before requirements are understood
- Time is up before design is complete
- Coding begins with fuzzy requirements, incomplete design
- Developers may make requirement and design decisions in their own best interest
- Unit test may become debugging
- Test schedule shortened to hold end date
- Product-complete milestone slips to delivery date
- Acceptance of the overrun

For More Information: “Software Engineering Economics”, Barry Boehm

Slide 6

Universal Estimating Laws

- Law 1 - Software takes as long to develop as it takes
- Law 2 - Software development time is not influenced by what we would like it to take
- Law 3 - Estimating a shorter time than it will actually take only defines the length of the overrun

Slide 7

Countermeasure - Negotiating Skills

- Requires acceptance of problem
- Not taught in colleges
- Only having one negotiating style is limiting ("You will ship it over my dead body.")
- We negotiate all the time anyway in various places
- Leads to more Win-Win solutions

Slide 8

Negotiation - Definitions

- Exploration to formulate viewpoints.
- Delineate areas of agreement or contention.
- Working out practical arrangements.

For More Information:
Gerald I. Nierenberg- *Fundamentals of Negotiating*
Nightingale-Conant - *The Art of Negotiating*

Slide 9

Negotiation Examples

- Negotiating a test and development schedule with the program manager
- Ordering a meal at a restaurant
- Setting up a meeting time
- Obtaining the release of hostages

Slide 10

Negotiate What?

Product	Testing Service
Price	Schedule
Delivery	Budget
Features	Staff
Shipping	Tools
Terms	Release Dates
Follow-on	Acceptance
Warranty	Environment

Slide 11

Success Likelihood

- The issue is negotiable (buying a car is, selling your child is not)
- The negotiator's interest in giving and taking value, and compromise (toll taker)
- Negotiating parties trust each other to some extent
- Trust - Words and Actions match

Slide 12

Negotiation Essentials

- Knowledge of human behavior
- Preparation, know all the related facts on both sides of the issue
- Understanding of techniques, strategies, and tactics
- Understanding the needs of both sides, both direct and indirect.

Slide 13

Example Preparation

- Collect previous actuals on:
  - Defects per thousands of lines of code
  - Hours to find defects
  - Time to run tests
  - Pass and fail percentage
  - Defects found before and after delivery
  - Classification of defects (requirement, coding, system, testing, documentation)
  - Accuracy of prior estimates
  - Customer satisfaction survey results
- Know style of the person negotiating with

Slide 14

Directness and Openness

- Direct - Do it and do it now per these instructions.....
- Indirect - You should have known what I was thinking.....
- Open - I feel angry and disappointed that we have not been given time to test the new build.....
- Self Contained - Everything is fine.

For More Information "Relationship Strategies" by Alexandra and Cathcart

Slide 15

Relationship Strategies

	Indirect	Direct
Open	Relater	Socializer
Self Contained	Thinker	Director

Slide 16

Director Style

- Self-contained, direct
- Most Important - The bottom line on the issue, strategies
- Wants things to be measured, likes competition, work first
- Assertive, Responsible, Straightforward, Practical, Self Motivated
- Needs to be in charge of others, imposes their standards on others
- Usually does not have a "product," juggles many things at once
- Usually migrates to manager positions
- Black and White approach, cutting edge,
- Jack Lord in Hawaii Five 0, Barbara Walters, Managers

Slide 17

Thinker Style

- Indirect, self-contained
- Most Important - The logic behind the issue, the details, the rules
- Detail oriented, methodical, predictable, dependable, precise
- Likes to engage in intellectual debates, organized, loyal, orderly
- Usually introverted, has "show me" attitude, focused
- Likes to work alone, has the right tools, loves to gather data
- Perfectionist, less interested in outcome, problem solver
- Jack Webb in Dragnet, Joyce Brothers, Data on Star Trek, Developers, Testers, Accountants

Slide 18

Socializer Style

- Direct - Open
- Most Important - Interactions between people
- Flexibility, goes with the flow, avoids formal plans
- Goes for the gusto, jumps in and takes charge, ready, fire, aim
- Avoids bureaucracy, likes challenges, very optimistic
- Hates to work alone, likes to be where the action is
- Likes recognition, Egotistical, Impatient, Impulsive
- Mickey Rooney as Andy Hardy, Dom Delouise, Sales and Marketing persons

Slide 19

Relater Style

- Open, indirect
- Most Important - Knowing how everyone feels about the issue
- Good listener, empathetic, emotional, helpful
- Flexible, goes with the flow, team player
- Not the job that counts, it is the people
- Likes the personal approach, not formal
- Warm, reliable, patient, relaxed
- Mr. Rogers, Marcus Welby, Mary Tyler Moore, Therapists, Nurses, Physicians

Slide 20

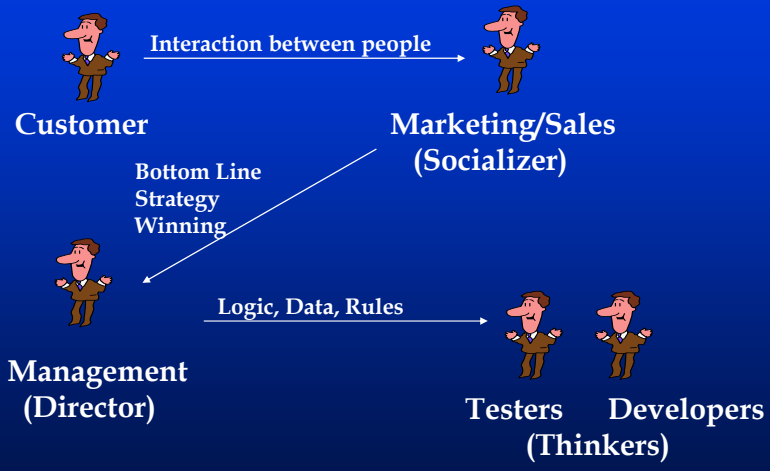
Dealing With Humans

	Indirect	Direct
Open	Relater	Socializer
Self Contained	Thinker	Director

Areas of Friction

Slide 21

Software Development Communicating and Styles



Slide 22

Dealing With Style

- Be aware of your own style
- Be aware of the style of the people you must interact with
- Be aware of the company's style
- Present information in the same style of the person you are negotiating with to reduce friction

Slide 23

Selling Test Plan to Marketing

- The Test Plan is flexible
- Gives awards and recognizes achievements
- Eliminates bureaucracy and paperwork
- Allows frequent participation in meetings
- Participation in major milestones
- Options for using contractors, outsourcing, customer service as testing resources
- Show how test plan focuses testing on frequent 800 line complaints

Slide 24

Selling Test Plan to Developers

- Test Plan is logical, rule based
- Test Plan details have been thought out
- Test Plan contains checklists and templates that simplify work
- Test Plan has been demonstrated to work on a pilot program
- Test Plan provides the right tools for the job
- Test Plan provides data to track progress

Slide 25

Selling Test Plan to Management

- Test Plan follows a strategy (use case based, risk based, top down, etc.)
- An attractive return on investment to the bottom line ($4,000 per defect)
- Shortest possible test time with reasonable risk
- It is a better Test Plan than the competition's
- Assumptions and dependencies clearly stated
- Measurable and highly visible milestones
- Early detection and correction of estimates

Slide 26

Favorite Tactics For Negotiating Test Schedule

- Columbo - play dumb, ask questions, reflect back
- Trump - bracketing estimates
- Reversal - opposite of what is expected
- Horsetrading - trade time for tools, outsource, temps
- Gasp (and Counter Gasp) – that was the good news
- Switch Hitter - use the best people
- Limited Authority - don't shoot from the hip
- Participation - group decides
- Salami - get what you need *poco a poco*
- Pinocchio - the last resort

Slide 27

Summary

- The biggest obstacle in the software industry is lack of enough time to do our jobs well
- We apply our education, experience, and tools, but the problem persists
- We may not think of negotiating as an option because it is not part of our education
- Negotiating can buy valuable time, help management avoid blunders
- Negotiating can make the job less stressful and more creative
- We will never get things our way completely, and good thing, else nothing would get shipped.

Slide 28

QW2002 Paper 2Q

Mr. Hung Nguyen
(LogiGear Corporation)

Effective Performance Test Planning and Implementation

Key Points

- Create useful load testing plans in the dark
- Set and meet performance testing expectations
- Design and construct useful tests through the understanding of user-based and server-based workload modeling, and other critical and often misunderstood variables

Presentation Abstract

This QuickStart discusses how to plan and implement a successful, cost effective performance testing program. It's ideal for professionals taking on the new responsibilities of determining the scope of performance testing, selecting the right strategy to fit the need and budget; acquiring the resources to implement the program, designing useful tests to reveal findings that support business and engineering decisions, and educating the rest of the organization about performance testing.

About the Author

Hung Q. Nguyen is founder, president, and CEO of LogiGear® Corporation. He’s held leadership roles in business management, product development, business development, engineering, quality assurance, testing, and information technology. Hung is an international speaker and contributor to industry publications. He authors and teaches software testing curriculums for LogiGear University and the University of California. He is the original architect of TRACKGEAR™, a Web-based defect management system, and the author of Testing Applications on the Web (Wiley). He also wrote (with Kaner and Falk) the best-selling book Testing Computer Software (Wiley), which is also published in Japanese. He holds a B.Sc. in Quality Assurance from Cogswell Polytechnical College, is an ASQ-Certified Quality Engineer, and a member of the Advisory Council for the Department of Applied Computing and Information Systems at UC Berkeley Extension.

Effective Performance Test Planning and Implementation

© 2002 LogiGear Corporation. All Rights Reserved.

Contact Information

***Logi*Gear Corporation**
551 Pilgrim Drive, Suite A-1
Foster City, CA 94404
Tel.650.572.1400
Fax.650.572.2822
lobby@logigear.com
http://www.logigear.com

© 2002 LogiGear Corporation. All Rights Reserved.

Objectives

- Create useful load testing plans in the dark
- Set and meet performance testing expectations
- Design and construct useful tests through the understanding of user-based and server-based workload modeling, and other critical and often misunderstood variables

© 2002 LogiGear Corporation. All Rights Reserved.

What is Performance Testing?

Performance testing is a capacity analysis and planning process that's designed to predict when future load levels will exhaust the Web system. It helps in developing effective enhancement strategies that maintain *acceptable user experience*.

© 2002 LogiGear Corporation. All Rights Reserved.

Performance Test the System Resources

- In performance and performance-related testing, simulated workload is used to exhaust system resources, including:
  - Memory: Physical, virtual, and storage; heap and stack space
  - CPU time
  - TCP/IP addresses (in a DHCP pool)
  - File handles
  - Hardware interrupts and I/O processes
  - Memory run-time errors such as leakage, overwrite, and pointer errors; database file handles and deadlocks; and multithreading related problems.
  - Network bandwidth

© 2002 LogiGear Corporation. All Rights Reserved.

The Performance Testing Process

- Set expectations and define deliverables
- Gather requirements
- Define workload
- Define performance goals and identify metrics to collect
- Identify tests to run, and when to run them
- Decide on a tool option
- Write a test plan
- Design user-scenarios and create test scripts
- Getting ready
- Run tests
- Analyze results
- Change the system to optimize performance
- Run new tests, as well as old tests

© 2002 LogiGear Corporation. All Rights Reserved.

The Three Phases

- The Planning Stage
  - Set expectations and define deliverables
  - Gather requirements
  - Define workload
  - Define performance goals and identify metrics to collect
  - Identify tests to run, and when to run them
  - Decide on a tool option
  - Write a test plan
  - Design user-scenarios and create test scripts
  - Getting ready
- The Testing Stage
  - Run tests
- The Postmortem
  - Analyze results
  - Change the system to optimize performance
  - Run new tests, as well as old tests

© 2002 LogiGear Corporation. All Rights Reserved.

We Need Answers

- Can the system handle the expected load while maintaining acceptable response time?
- As demand grows, will the system be able to handle increased load while maintaining acceptable response time?
- If not, at what point does system performance begin to deteriorate; which components cause the degradation?
- Is the current system scaleable enough to accommodate future growth?
- When performance fails to meet acceptable customer-experience levels, what will be the effect on company sales and technical support costs?

© 2002 LogiGear Corporation. All Rights Reserved.

Performance Test Objectives

- It takes time, effort, and commitment to plan for and execute performance testing. Performance testing involves more people in an organization than just testers. Usually, a well-planned program is a joint effort by members of a product team, including upper management, marketing, development, information technology, and software testing.
- Let's examine performance issues from the management and development perspectives (departments that should normally be part performance testing planning and execution).

© 2002 LogiGear Corporation. All Rights Reserved.

Performance Test Objectives

- Management's objectives: To avoid financial losses in sales and technical support and to avoid customer dissatisfaction.
  - Will the Web application be capable of supporting the projected number of users while preserving acceptable performance—and at what cost?
  - At what point will the Web application's load handling-capability begin to degrade?
  - How will the degradation affect the business financially?
  - What can be done to increase the Web application load-handling capability? What are the associated costs?
  - After deployment, how can we know when to take appropriate actions to prevent the Web application from reaching its saturation point (monitoring after deployment)?

© 2002 LogiGear Corporation. All Rights Reserved.

Performance Test Objectives

- Development and testing objectives
  - What is the definition of “projected number of users?”
    - Users are not created equal (user activities vary).
    - User access and activity frequency can vary during a specific time cycle.
    - How do we represent “projected number of users” in a workload model?
    - How do we simulate real-world users? What is the correlation between real users and virtual users?
  - What is the definition of “acceptable performance?”
    - How do we measure “performance” and at what cost?
    - Which metrics should we use?
    - Which factors affect “performance?”
    - Which tools should we use and how should we evaluate them?
  - Data analysis and corrective-action planning
    - How can performance degradation be resolved?
      - Added system resources
      - Improved network system architecture
      - Improved programming
  - How can workload demand from users be monitored on an on-going basis so that appropriate actions can be taken to avoid reaching the saturation point?

© 2002 LogiGear Corporation. All Rights Reserved.

Setting Expectations

- Preparing the organization and managing people’s expectations is essential for success
- Understand the test requirements and scope of testing
- Define, or at least ask:
  - What are the objectives of the performance tests?
  - Who cares? Why measure?
- Communicate:
  - Your service capabilities and limitations
  - Where you need help and how you can get help
- Upon completion of your test plan, seek review and buy-in from the stakeholders

© 2002 LogiGear Corporation. All Rights Reserved.

Define Your Deliverables

- Test plan
  - Performance testing goals
  - Workload definitions
  - User scenario designs
  - Performance test designs
  - Test procedures
  - System-under-test configurations
  - Metrics to collect
- Tool evaluation and selection reports (first time, or as needed)
- Test scripts/suites
- Test run results
- Analysis reports against the collected data
- Performance related error reports (e.g., failed transactions)
- Functional bug reports (e.g., data integrity problems)
- Periodic status reports
- Final report

© 2002 LogiGear Corporation. All Rights Reserved.

Test Requirements

- Environment and Resources
- Workload
- Service Response Time

© 2002 LogiGear Corporation. All Rights Reserved.

Environment and Resources

- Network access variables
- Demographic variables
- ISP infrastructure variables
- Client configurations
  - Computer variables
  - Browser variables
- Server configurations
  - Specifying mixes of system hardware, software, memory, network protocol, bandwidth, etc.

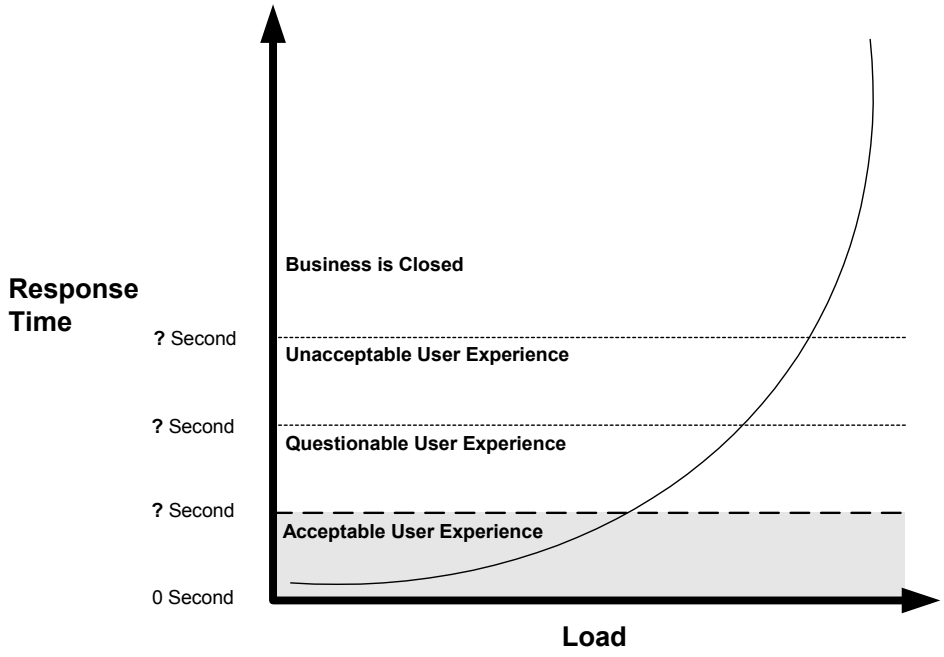
© 2002 LogiGear Corporation. All Rights Reserved.

The Workload

- Number of users
- Number of hits
- Number of page views
- Average page size
- Percentage of peak traffic over peak period
- Average user-session time
- Users: their activities and behaviors
  - How many groups of users will be involved in the load test over a certain time interval?
  - How frequently will each user in each group access the Web application?
  - Does average session time vary?
  - What are the typical activities, and at what frequency are they performed by each group of users (refer to the earlier example)?

© 2002 LogiGear Corporation. All Rights Reserved.

Response Time: What's Your Number?



© 2002 LogiGear Corporation. All Rights Reserved.

Response Time and Capacity

- Percentage of requested static pages that must meet the acceptable response time
- Percentage of requested scripts that must meet the acceptable response time
- The baseline multiplier (2x, 4x, ...) that the system must be capable of handling
- The peak ratio that the system must be capable of handling
- The spike (overload caused by expected and/or unexpected events) ratio that the system must be capable of handling

© 2002 LogiGear Corporation. All Rights Reserved.

What If Requirements Don't Exist?

- Tell people what you are going to do
- Get feedback
- Perform the testing
- Communicate your progress and results as you go
- Ask specific questions
- Fill in the blanks

© 2002 LogiGear Corporation. All Rights Reserved.

Questions to Consider

- What is the definition of "workload"?
- What is the definition of "system"?
- How do we size the workload?
- What is the expected workload?
- What is response time?
- What is acceptable response time?
- Which metrics should we collect?
- What is the definition of "increased load"?
- What is the correlation between demand and increased load?

© 2002 LogiGear Corporation. All Rights Reserved.

Questions to Consider

- How do we determine which components are problematic?
- What is the definition of scalability? How can it be tested?
- What is the definition of reliability and availability? How can it be tested?
- What is future growth? Can it be quantified?
- How do we correlate financial implications?
- How many other users are using the same resources on the system under test (SUT)?
- Is the SUT inside or outside the firewall?
- Is the load coming from the inside or outside of the firewall?
- Are you testing the SUT in its complete, real-world environment (with load balances, replicated database, etc.)?
- What's the mix ratio of static pages vs. code?

© 2002 LogiGear Corporation. All Rights Reserved.

The Notion of Workload

- Workload can be described using a combination of three terms
  - User - Number of users and their common activities.
  - Application - Workload generated at the application or server level (such as HTTP requests) to carry out user activities.
  - Resource - The resource requirements for handling the workload.

© 2002 LogiGear Corporation. All Rights Reserved.

Workload Concept

- Users
  - Total number of users in all unique classes.
  - Percentage of users in each class.
  - Number of sessions per user in each class over a period of time.
  - Length of user sessions in each class.
  - User activities - Number of each kind of unique activity carried out by users during the sessions.
  - User behaviors such as patience, speed, expertise, etc.
  - ***Maximum number of concurrent users in all unique classes that the system has to support over a period of time.***

    Read Savoia's "Trade Secrets from a Web Testing Expert"
  - Percentage of concurrent users in each class.
- Application
  - Activities are expressed in terms of service rates such as Transactions per Second (TPS) or throughput, Kilobytes per second.
- Resources
  - Resources required to handle the workload while preserving acceptable performance—usually expressed in response time.

© 2002 LogiGear Corporation. All Rights Reserved.

Sizing the Workload

- Before deployment
  - Performance requirement document (If you are lucky!)
  - Hypothesize the number of concurrent users, classify users into groups, estimate percentage of each class of user, length of their session, their activities and the frequency of each activity, user behaviors, etc. You will eventually get a value—you may not get an accurate estimate. However you will have opportunity to calibrate your estimates after several runs of tests.
  - Assemble a small group of users that will represent a sample of various unique real world users. Have them use the product in the ways that you expect the product will be used. Configure your server to log user activity. Use the collected data to estimate the baseline workload.
- After deployment
  - Use Web or Proxy server logs to collect workload data.

© 2002 LogiGear Corporation. All Rights Reserved.

Sizing the Workload

- **Server-based profiling using Web server log data**
- **User-based profiling using user scenario modeling**

© 2002 LogiGear Corporation. All Rights Reserved.

Web Server Log Metrics

- General metrics
  - Metrics that convey the overall load and performance of the system.
- Application-specific metrics
  - Metrics that convey the load and performance of the system in the context of specific user activities or requests, such as requests of certain pages.

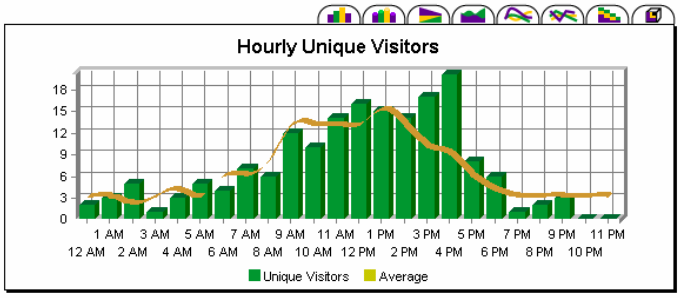
© 2002 LogiGear Corporation. All Rights Reserved.

General Log Metrics

- **Number of users over a period of time**
- **Estimated Peak Ratios**

By User:
Max UV /Ave UV = 13 / 5.8 = 2.8

By Page Views:
Max PV / Ave PV = 75 / 26.8 = 2.2



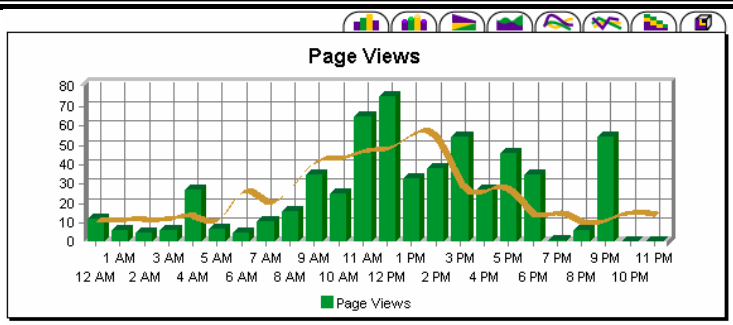
Friday, March 16, 2001 (1 Day)

General Statistics					
Total Page Views					589
Daily Unique Visitors					127
Total Visits					169
First Time Visitors					96
Returning Visitors					31
Averages					
Average Page Views per Visitor					4.6
Average Visits per Visitor					1.3
Average Visit Length					00:03:56
Average Page Views per Hour of this Day					26.8
Average Unique Visitors per Hour of this Day					5.8
Average Visits per Hour of this Day					7.7
Most Active Period			**Least Active Period**		
Page Views	12:00:00 PM	75	Page Views	7:00:00 PM	1
Unique Visitors	4:00:00 PM	13	Unique Visitors	7:00:00 PM	0

© 2002 LogiGear Corporation. All Rights Reserved.

General Log Metrics

Page view distribution can be translated into the workload imposed on the system



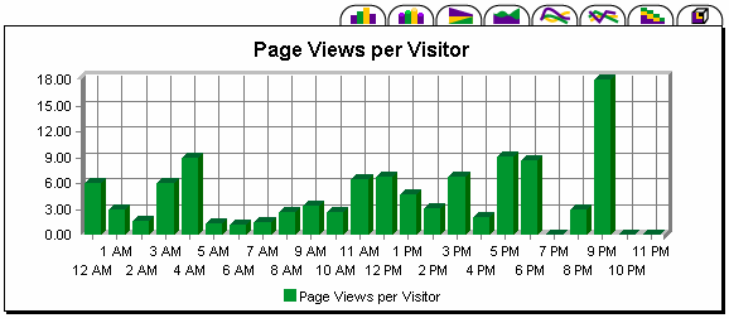
Friday, March 16, 2001 (1 Day)

Hour	Average	Page Views	Hour	Average	Page Views
12 AM	10	12	12 PM	47	75
1 AM	10	6	1 PM	54	33
2 AM	10	5	2 PM	52	38
3 AM	11	6	3 PM	28	54
4 AM	12	27	4 PM	25	27
5 AM	10	7	5 PM	26	46
6 AM	25	5	6 PM	13	35
7 AM	19	11	7 PM	14	1
8 AM	27	16	8 PM	9	6
9 AM	40	35	9 PM	10	54
10 AM	42	25	10 PM	14	0
11 AM	46	65	11 PM	13	0
Total					**589**

© 2002 LogiGear Corporation. All Rights Reserved.

General Log Metrics

Page-views-per-visitor distribution can be translated into user activity



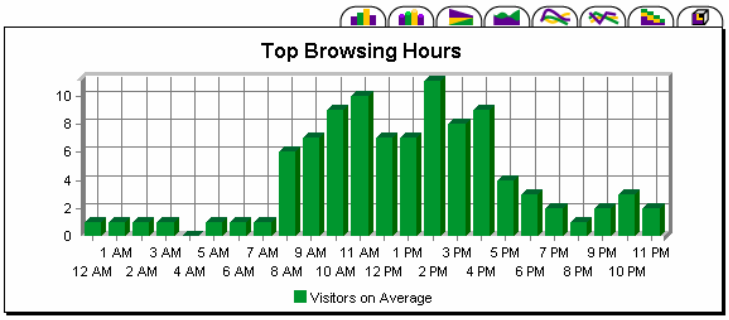
Friday, March 16, 2001 (1 Day)

Hour	Page Views per Visitor	Hour	Page Views per Visitor
12 AM	6.00	12 PM	6.82
1 AM	3.00	1 PM	4.71
2 AM	1.67	2 PM	3.17
3 AM	6.00	3 PM	6.75
4 AM	9.00	4 PM	2.08
5 AM	1.40	5 PM	9.20
6 AM	1.25	6 PM	8.75
7 AM	1.57	7 PM	0.00
8 AM	2.67	8 PM	3.00
9 AM	3.50	9 PM	18.00
10 AM	2.78	10 PM	0.00
11 AM	6.50	11 PM	0.00
Overall Average			**4.64**

© 2002 LogiGear Corporation. All Rights Reserved.

General Log Metrics

Weekly average of peak traffic is from: 8:00AM to 5:00PM (server's local time)



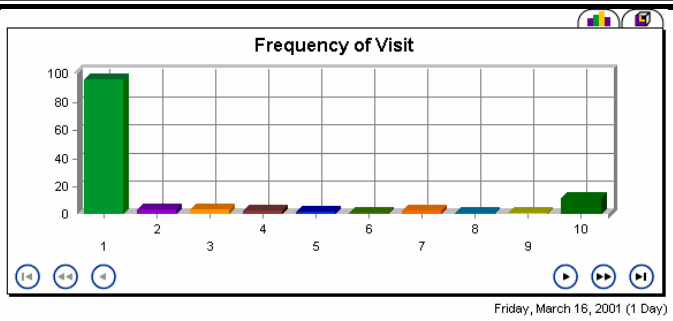
Sunday, March 11, 2001 to Saturday, March 17, 2001 (1 Week)

Hour	Visitors on Average	Hour	Visitors on Average
12 AM	1	12 PM	7
1 AM	1	1 PM	7
2 AM	1	2 PM	11
3 AM	1	3 PM	8
4 AM	0	4 PM	9
5 AM	1	5 PM	4
6 AM	1	6 PM	3
7 AM	1	7 PM	2
8 AM	6	8 PM	1
9 AM	7	9 PM	2
10 AM	9	10 PM	3
11 AM	10	11 PM	2

© 2002 LogiGear Corporation. All Rights Reserved.

General Log Metrics

Ratios of New users to returning-users. Based on their familiarity with the site, returning user may navigate the site faster.



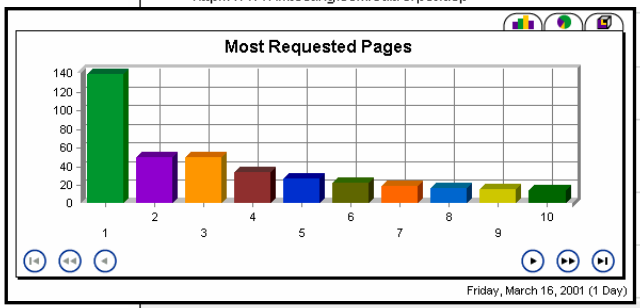
Number of Visits	Visitors	%
1 Visit	96	75.59%
2 Visits	4	3.15%
3 Visits	4	3.15%
4 Visits	3	2.36%
5 Visits	2	1.57%
6 Visits	1	0.79%
7 Visits	3	2.36%
8 Visits	1	0.79%
9 Visits	1	0.79%
10 or More Visits	12	9.45%
Total for Unknown	**0**	**0.00%**
Total	**127**	

© 2002 LogiGear Corporation. All Rights Reserved.

Application-Specific Log Metrics

Analyze user activities by studying the ratios of specific requests received by the system.

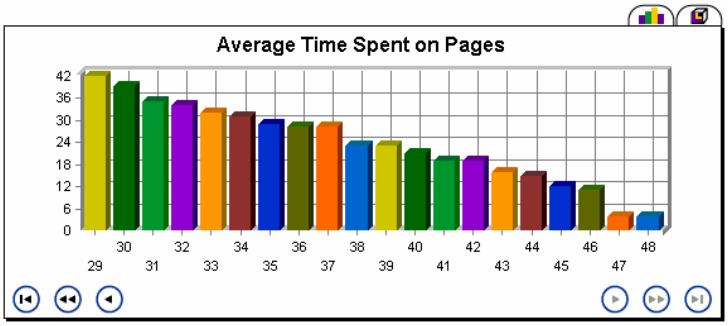
Page Name	Page Views	%
1. **Login Page** http://www.mtesting.com/login.asp	139	23.60%
2. **Search Page** http://www.mtesting.com/search.asp	50	8.49%
3. **Distribution Chart** http://www.mtesting.com/distribution.asp	50	8.49%
4. **New Report** http://www.mtesting.com/newreport.asp	34	5.77%
5. **Edit Report** http://www.mtesting.com/editrerpot.asp	27	4.58%
	22	3.74%
	19	3.23%
	17	2.89%
	15	2.55%
	14	2.38%
Total of all Pages	**589**	



© 2002 LogiGear Corporation. All Rights Reserved.

Application Specific Log Metrics

The read/think/data-input time of various pages



© 2002 LogiGear Corporation. All Rights Reserved.

Server Log Metrics

- A few key metrics to analyze
  - ***Number of users and/or user sessions***
  - ***Average session time***
  - Number of page views
  - ***Average page views per session***
  - ***The type of pages that were requested during the session***
  - Peak period (e.g., 75% of traffic is from 11:00 AM-4:00 PM)
  - Number of hits
  - Average page size

© 2002 LogiGear Corporation. All Rights Reserved.

Server Log Metrics

- Other interesting metrics to analyze
  - New users vs. returning users
  - Frequency of visits (e.g., 75% of users made one visit)
  - Demographics
  - Client information such as browser, browser version, Java script support, Java script enable/disable, and so on.
  - User read/think/data-input time.
    - One way of computing read/think/data-input time is

$$\frac{\textit{Average Session Time}}{\textit{Average Page Views per Session}} = \textit{Read/Think/Data-Input Time}$$

© 2002 LogiGear Corporation. All Rights Reserved.

Things to Consider

- Average vs. Maximum
  - When examining server logs, pay attention to both the average numbers as well as the maximum numbers
  - Use averages with care
    - Be careful with arithmetic averages
    - Consider using range, percentile, median, etc.

      Read Savoia's "Trade Secrets from a Web Testing Expert"
- An increase in sessions or users does not necessarily correlate linearly with page views per hour.
  - This behavior is due to the fact that:
    - User session time is shorter. Therefore, average page views per user is lower.
    - Both number of users and session duration affect load.

© 2002 LogiGear Corporation. All Rights Reserved.

Things to Consider

- Analyze the log carefully so you'll have a thorough understanding of how the values are calculated and what they represent.
- Pick one log monitoring tool and use it consistently.
- A few Web log tools
  - Web Trends (http://www.webtrends.com)
  - Analog (http://www.statslab.cam.ac.uk/~sret1/analog/)
  - NetTracker (http://www.sane.com/products/NetTracker/)

© 2002 LogiGear Corporation. All Rights Reserved.

Things to Consider

- Collect
  - **General activities by time of day**
  - **General activities by day of week**
  - **Application-specific activities by time of day**
  - **Application-specific activities by day of week**

© 2002 LogiGear Corporation. All Rights Reserved.

User-Based Workload Profiling

- **Workload can be derived from**
  - **Number of concurrent users**
  - **Ratios of various user variables**
  - **Effects of each user variable**

© 2002 LogiGear Corporation. All Rights Reserved.

User Behaviors

- **Different Functional groups will:**
  - **Request different sets of features**
  - **Use features at different frequencies**
  - **Incur different session-elapse times**
- **Human speed affects the user's read/think/data-input speed**
- **Human patience affects how quickly one cancels a request**
- **Groups with different levels of expertise will use the product or site at different read/think/data-input speeds**
- **Regular or returning users make requests at a faster rate than new users**

© 2002 LogiGear Corporation. All Rights Reserved.

User Ratios

- Percentage by functional group
- Percentage by human speed
- Percentage by human patience
- Percentage by domain expertise
- Percentage by familiarity
- etc.

© 2002 LogiGear Corporation. All Rights Reserved.

Creating Realistic Conditions

- Having a good mix
  - Simulate various user groups to create a good mix of specific requests
  - Simulate user cancellations, to cover cancellations caused by response time

    Read Savoia's "Trade Secrets from a Web Testing Expert"
  - Simulate both novice users and advanced users, to cover speed variances
  - Simulate user from different locations

© 2002 LogiGear Corporation. All Rights Reserved.

Calculating Transactions per Second

$$\frac{\text{User Type \%} \times \text{Number of Users} \times \text{Frequency}}{(\text{Session Length-Minutes} \times 60 \text{ Seconds})}$$

Note:
This formula is based on an assumption that the transactions are distributed evenly over the duration of a session. Therefore, the resulting number only represents the minimum number or concurrent transactions per second.

© 2002 LogiGear Corporation. All Rights Reserved.

Example of Estimating User Cancellation Rate

- User's tolerance varies depending upon several factors:
  - Access rate
  - Activity (e.g., downloading a 500K file as suppose to requesting a 10K page)
  - Human behavior

Patience	% Canceling Users	Response Time Greater Than
Low	20%	6 sec
Medium	50%	12 sec
High	30%	18 sec

© 2002 LogiGear Corporation. All Rights Reserved.

An Example of Estimating User Speed

Domain Expertise	% of Users	Estimated Read/Think/Data-Input Time	Weighted Factor	Scripted Estimated Read/Think/Data-Input Time
Novice Users	75%	45 sec	1.5	67.5 sec
Experienced Users	20%	45 sec	1.0	45.0 sec
Power Users	5%	45 sec	0.5	22.5 sec

© 2002 LogiGear Corporation. All Rights Reserved.

Example of Estimating Arrival Rate

Demographics	% of Users	Arrival Rate Weighted Factor
USA East Coast	30%	1.5
USA Central	15%	1.0
USA West Coast	45%	0.5
Europe	10%	2.0

© 2002 LogiGear Corporation. All Rights Reserved.

Example of Common System Metrics

- Transactions Per Second (TPS) - The number of transactions handled by the server per second. In the previous example, the aggregate number of simple query transactions is 7 TPS.
- Hits per Second - Typically, the number of hits per second the Web server receives from users. Note that a transaction requesting a single HTML page can trigger multiple hits to the server. As the number of transactions rises and the number of hits-per-second reaches saturation point, transaction-round-trip-time (latency) becomes longer.
- Concurrent connections - The number of concurrent open connections.
- Throughput that the server processes (measured in KB/Sec) .

© 2002 LogiGear Corporation. All Rights Reserved.

A Throughput Calculation Example

- The Objective - Determine bandwidth requirement for handling the load.
- A Simplified Scenario
  - A Web log shows that 10,000 concurrent users request a document from a pool of 10 different HTML documents every 3.5 minutes, with an average page size of 2 KBytes each.
- Calculating bandwidth requirement for throughput handling:

  Throughput = 10,000 * (2 * 1024 * 8) / (3.5 * 60) = 780,190 bps

  To handle this throughput load, the network connection should be at least a T1 line (1,544,000 bps).

© 2002 LogiGear Corporation. All Rights Reserved.

When Can We Start Testing?

- Testing should be performed as early as possible, and should be repeated as many times as possible. It is easier and less costly to correct errors early in the development process.
  - The earlier that testing is begun, the more times it can be repeated.
  - The more often that tests are performed, the more likely they are to uncover errors.
  - Tests can be part of the regression-testing suite to be performed with each build. Regression testing can determine if an error was added in the latest release.
  - Early detection is particularly important if the system does not meet the desired performance requirement. This affords the developer more time to adequately address problems.

© 2002 LogiGear Corporation. All Rights Reserved.

When Can We Start Testing?

- Certain requirements must be met before testing can begin.
  - Hardware must be installed and operational.
  - Network should be fully operational.
  - Functionality of the application under test must be complete.
  - Tests themselves, tools and/or scripts must be fully developed or incorporated into current test scripts (Previously developed test scripts can often be used “as is”).

© 2002 LogiGear Corporation. All Rights Reserved.

Types of Tests

- Acceptance test (10-100 users with simple scripts)
- Baseline test
- 2B1 load test
- Goal-reaching test
- Peak test
- Stress test
- Scalability tests
- Availability and reliability tests

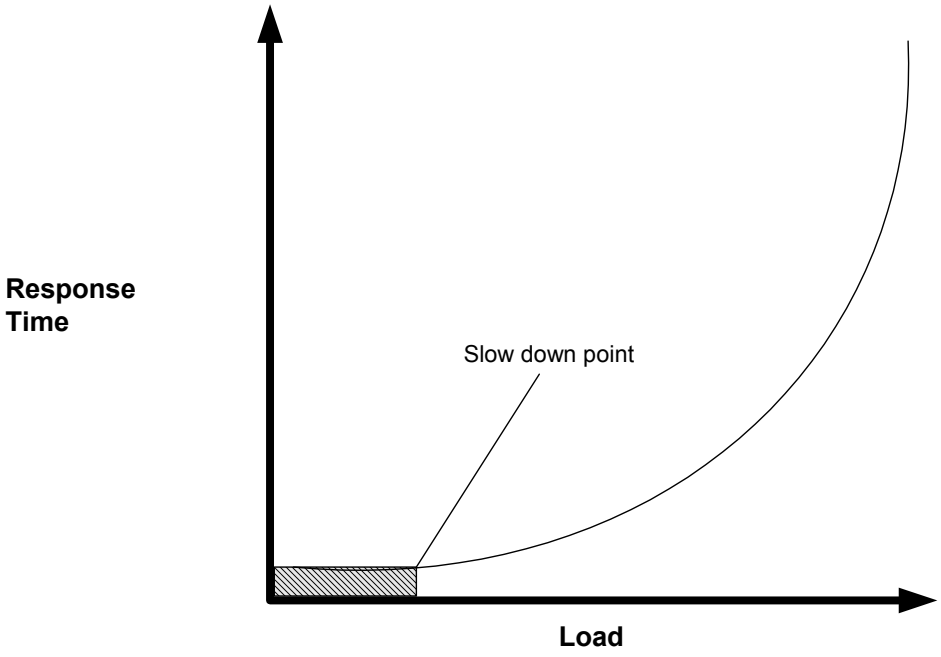
© 2002 LogiGear Corporation. All Rights Reserved.

Other Tests to Consider

- Regression and benchmark tests
  - Application architecture changes
  - Code changes
  - Client/Server architecture changes
  - Server configuration changes
  - Middleware and other component changes
  - User volume and behavior changes
- Data-empty vs. full tests

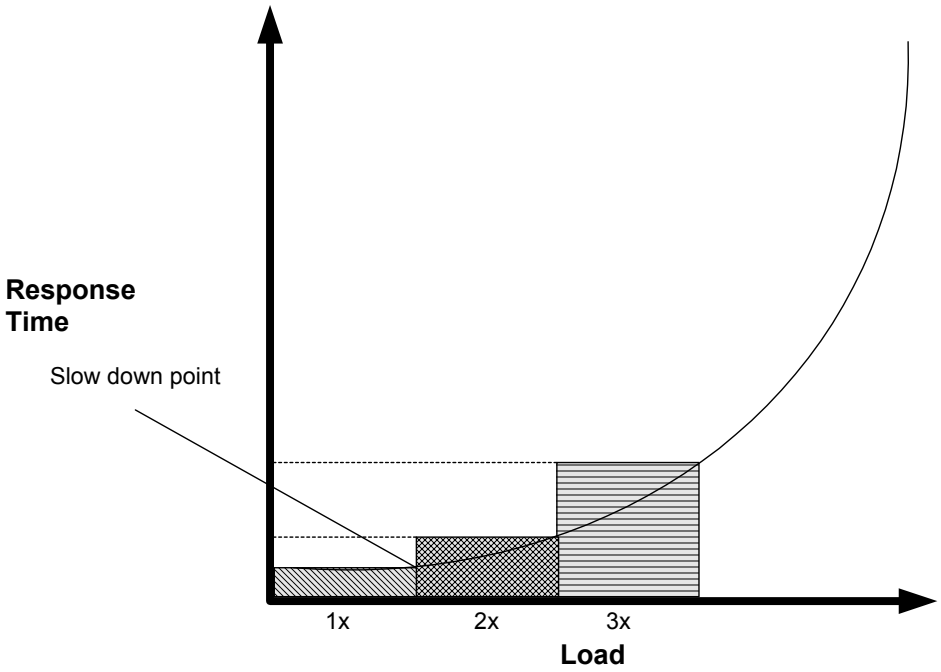
© 2002 LogiGear Corporation. All Rights Reserved.

Formulating the Load Baseline



© 2002 LogiGear Corporation. All Rights Reserved.

2B1 or 3B1 Load Test



© 2002 LogiGear Corporation. All Rights Reserved.

Load Ratios to Consider

- **Tolerance ratio: Imposed load ± 25 %?**
- **Safety ratio: Imposed load x 2?**
- **Peak ratio: Imposed load x 4?**
- **Spike ratio: Imposed load x 5?**

© 2002 LogiGear Corporation. All Rights Reserved.

Estimating Load Capacity

- **Estimate the load baseline**
- **Increase the load by multiplying the load baseline by 1x, 2x, 3x, 4x, Nx gradually until unacceptable response time is reached**

Note:

Perceived user response is only as accurate as how closely the virtual load matches the real world load.

© 2002 LogiGear Corporation. All Rights Reserved.

Response Time: How Slow is Too Slow?

- Obtain market data to determine the threshold of response time that will turn users away
- Judge it yourself through experimentation

© 2002 LogiGear Corporation. All Rights Reserved.

Be Specific

- Specify what tests you will run
- Estimate how many cycles of each test you will run
- Schedule your tests ahead of time
- Specify by what criteria you will consider the SUT to be ready to test
- Forward thinking: Determine and communicate the planned tests and how the tests are scheduled

© 2002 LogiGear Corporation. All Rights Reserved.

Document Your Plan

- Test Plan Objectives
  - Documents test objectives, test requirements, test designs, test procedures, and other project management information.
  - Solicits feedback and builds consensus.
  - Defines development and testing deliverables.
  - Secures commitment and resources for the test effort.
  - Take advantage of the test plan walkthrough process.

© 2002 LogiGear Corporation. All Rights Reserved.

Things to Consider

- Consensus and approval is achieved through test plan reviews and communication.
- The test plan walkthrough is a powerful technique for collecting requirements, soliciting feedback, and encouraging team involvement.
- Obtain budget and resource allocation upon or after completion and approval of the plan.
- Human resource options:
  - Train and use your own staff?
    - The technology
    - The test objectives
    - The test environments
    - The test methodology
    - The business implications
  - Rely on your developers to do it?
  - Hire an outside consultant?

© 2002 LogiGear Corporation. All Rights Reserved.

The Postmortem

- Analyze test results
  - The three-step process
    - Characterize: Look at what it does
    - Analyze: Look for bottlenecks
    - Optimize: Change the system to optimize performance
- Restore software and hardware to the base state condition
  - Restore database
  - Clear server cache
  - Empty temporary and log files
  - Reconfigure local network access
- Write a Performance Test Report
- Prepare regression and other tests

© 2002 LogiGear Corporation. All Rights Reserved.

Web Resource: www.QACity.com

© 2002 LogiGear Corporation. All Rights Reserved.

References

Anderson, M. D., "13 Mistakes in Load Testing Applications", Software Testing and Quality Engineering: September/October 1999.

Asbock, S., *Load Testing for eConfidence*, Lexington, MA, Segue Software: 2000

Kolish, T. & Doyle, T., *Gain eConfidence: The E-Business Reliability Survival Guide*, Lexington, MA, Segue Software: 1999

Menasce, D. A., Almeida, V. A. F., *Capacity Planning For Web Performance*, Upper Saddle River, NJ: 1998.

Menasce, D. A., Almeida, V. A. F., *Scaling for E-Business*, Upper Saddle River, NJ: 2000.

Nguyen, N. Q., "Testing Web Applications", LogiGear Corporation Training Handbook: 2000.

Nguyen, N. Q., *Testing Applications on the Web*, New York, Wiley Computer Publishers, John Wiley & Sons: 2001

Radview Software Inc., "The Web Load User's Guide", Lexington, MA: 1998.

Savoia, A., "The Science and Art of Web Site Load Testing", STQE STAREAST: 2000.

Schelstrate, M., " Stress Testing Data Access Components in Windows DNA Applications", MSDN News, March/April: 2000. **http://msdn.microsoft.com/voices/news**

© 2002 LogiGear Corporation. All Rights Reserved.

About Hung Q. Nguyen

Hung Q. Nguyen is founder, president, and CEO of LogiGear® Corporation. He's held leadership roles in business management, product development, business development, engineering, quality assurance, testing, and information technology. Hung is an international speaker and contributor to industry publications. He authors and teaches software testing curriculums for LogiGear University and the University of California. He is the original architect of TRACKGEAR™, a Web-based defect management system, and the author of Testing Applications on the Web (Wiley). He also wrote (with Kaner and Falk) the best-selling book Testing Computer Software (Wiley), which is also published in Japanese. He holds a B.Sc. in Quality Assurance from Cogswell Polytechnical College, is an ASQ-Certified Quality Engineer, and a member of the Advisory Council for the Department of Applied Computing and Information Systems at UC Berkeley Extension.

© 2002 LogiGear Corporation. All Rights Reserved.

About *Logi*Gear Corporation

LogiGear Corporation is the first Silicon Valley-based software testing company to offer a full range of solutions to advance individual and organizational excellence in software testing. LogiGear offerings include in-depth technical and management expertise in software quality engineering, comprehensive advanced test engineering such as Action Based Testing™, a structured approach to testing and testing automation, and outsource testing solutions, skill-based training curriculum for software testing professionals through LogiGear University, and world-class testing support products including TRACKGEAR, a Web-based defect management solution.

www.LogiGear.com

© 2002 LogiGear Corporation. All Rights Reserved.

QW2002 Paper 3Q

Cem Kaner
(Florida Institute of Technology)

A short course in metrics & measurement dysfunction

Key Points

- Measurement programs often fail for good reasons
- Many measurement programs are dysfunctional
- We often have to take some measurements, here are a few ideas on what to do

Presentation Abstract

Relatively few software publishers (or in-house IT groups) have metrics programs in place. Many have tried, few have succeeded. I don't think that the widespread failure of metrics programs in the field is due to a lack of professionalism, laziness, lack of interest in quality or lack of sophistication of the staff. Rather, I think the failures stem from mistrust of the measurements and the people who take them, lack of respect for the models that relate the measurements taken to the constructs they supposedly measure, and examples of manipulation or misues of the data collected. These are serious problems. We should take them seriously.

About the Author

Cem Kaner is Professor of Computer Sciences at the Florida Institute of Technology. Prior to joining Florida Tech, Kaner worked in Silicon Valley for 17 years, doing and managing programming, user interface design, testing, and user documentation. He is the senior author (with Jack Falk and Hung Quoc Nguyen) of TESTING COMPUTER SOFTWARE (2nd Edition) and (with David Pels) of BAD SOFTWARE: WHAT TO DO WHEN SOFTWARE FAILS.

Through his consulting firm, KANER.COM, he teaches courses on black box software testing and consults to software publishers on software testing, documentation, and development management. Kaner is also the co-founder and co-host of the Los Altos Workshop on Software Testing, the Software Test Managers' RoundTable, the Workshop on Heuristic & Exploratory Techniques, and the Florida Workshops on Model-Based Testing.

Kaner is also attorney whose practice is focused on the law of software quality. He is active (as an advocate for customers, authors, and small development shops) in several legislative drafting efforts involving software licensing, software quality regulation, and electronic commerce. Kaner holds a B.A. in Arts & Sciences (Math, Philosophy), a Ph.D. in Experimental Psychology (Human Perception & Performance: Psychophysics), and a J.D. (law degree). He is Certified in Quality Engineering by the American Society for Quality.

A Short Course in Metrics and Measurement Dysfunction

Cem Kaner

International Software Quality Week

September, 2002

Much material is from participants of *Software Test Managers Roundtable* (STMR) and the *Los Altos Workshop on Software Testing* (LAWST).

- STMR 1 (Oct-Nov 1999) focused on the question, ***How to deal with too many projects and not enough staff?*** Participants: Jim Bampos, Sue Bartlett, Jennifer Brock, David Gelperin, Payson Hall, George Hamblen, Mark Harding, Elisabeth Hendrickson, Kathy Iberle, Herb Isenberg, Jim Kandler, Cem Kaner, Brian Lawrence, Fran McKain, Steve Tolman and Jim Williams.
- STMR 2 (April-May 2000) focused on the topic, ***Measuring the extent of testing***. Participants: James Bach, Jim Bampos, Bernie Berger, Jennifer Brock, Dorothy Graham, George Hamblen, Kathy Iberle, Jim Kandler, Cem Kaner, Brian Lawrence, Fran McKain, and Steve Tolman.
- STMR 5 (Oct. 2001) focused on ***Measuring the effectiveness of test groups.*** Lisa Anderson, Laura Anneker, James Bach, Sue Bartlett, Harold Crawford, David Gelperin, Mark Harding, Doug Hoffman, Cem Kaner, Brian Lawrence, Hung Quoc Nguyen, Alberto Savoia, Jennifer Smith-Brock, Steve Tolman, Jo Webb, Jim Williams, Garrin Wong,
- STMR 6 (May 2002) focused on ***Measuring the effectiveness of software testers.*** Laura Anneker, James Bach, Sue Bartlett, Rex Black, Jennifer Smith-Brock, Doug Hoffman, Kathy Iberle, Cem Kaner, Brian Lawrence, Bret Pettichord, Sid Snook, Steve Tolman, and Jo Webb.
- LAWST 8 (December 4-5, 1999) focused on ***Measurement***. Participants: Chris Agruss, James Bach, Jaya Carl, Rochelle Grober, Payson Hall, Elisabeth Hendrickson, Doug Hoffman, III, Bob Johnson, Mark Johnson, Cem Kaner, Brian Lawrence, Brian Marick, Hung Nguyen, Bret Pettichord, Melora Svoboda, and Scott Vernon.

What is measurement?

- Is measurement really "the assignment of numbers to objects or events according to a clear cut rule"?
  - No, it can't be. If it was, then many inappropriate rules would do.
- **Measurement is the assignment of numbers to objects or events (attributes) according to a rule derived from a model or theory.**
- A software metric is a standard way of measuring some attribute or result of the software process. Examples of these attributes are size, costs, defects, communications, difficulty and environment.

Copyright © Cem Kaner, 2000-2002.

Types of attributes often measured

- **Resource**
  - Amount of resource available and/or used
- **Process**
  - Attributes of the development artifacts (other than the product), such as specifications, test materials
  - Attributes of the methods and practices employed
- **Product**
  - Attributes of the product under development, such as size, reliability, usability.
- **Impact**
  - The effect of the product, such as support costs, changed user productivity, change in user safety.

Copyright © Cem Kaner, 2000-2002.

Why measure? (Some examples—add your own)

- Track project progress
- Gain control of processes
- Demonstrate the productivity of your staff
- Demonstrate the quality of your work
- Compare different engineering practices
- Increase your credibility with your management
- Identify where improvements are needed
- Determine (relative) complexity or other attributes of the software
- Help us understand whether we have achieved a certain quality level (value on some desirable attribute, such as reliability, performance, usability, accessibility, etc.)
- Gain control of characteristics of the products you make
- Gain the respect of your customers
- Demonstrate the effectiveness of the product
- Learn more about software engineering
- Evaluate models, provide a basis for scientific development of better ways to produce better products.

Copyright © Cem Kaner, 2000-2002.

Models

- It's an abstraction—some details are omitted or simplified
  - Try to measure distances on a subway map
    - What is the scale of the subway map?
    - Is it useful?
- Abstractions allow us to focus on a few variables and their relationships.
- Abstractions allow us to use mathematics to study relationships.

Copyright © Cem Kaner, 2000-2002.

Simple measurement

- You have a room full of tables that appear to be the same length. You want to measure their lengths.
- You have a one-foot ruler.
- You use the ruler to measure the lengths of a few tables. You get:
  - 6.01 feet
  - 5.99 feet
  - 6.05 feet
- You conclude that the tables are “6 feet” long.

Copyright © Cem Kaner, 2000-2002.

Simple measurement (2)

- Note the variation
  - Measurement errors using the ruler
  - Manufacturing variation in the tables
- Note the rule:
  - We are relying on a direct matching operation and on some basic axioms of mathematics
    - The sum of 6 one-foot ruler-lengths is 6.
    - A table that is 6 ruler-lengths long is twice as long as one that is 3 ruler-lengths long.
- These rules don’t always apply. What do we do when we have something hard to measure?

Copyright © Cem Kaner, 2000-2002.

A Framework for Measurement

- A measurement involves at least 10 factors:
  - **Attribute to be measured**
    - appropriate scale for the attribute
    - variation of the attribute
  - **Instrument that measures the attribute**
    - scale of the instrument
    - variation of measurements made with this instrument
  - **Relationship between the attribute and the instrument**
  - **Likely side effects of using this instrument to measure this attribute**
  - **Purpose**
  - **Scope**

Copyright © Cem Kaner, 2000-2002.

Framework for Measurement

Attribute ➢ Extent of testing – *What does that mean?*

Instrument ➢ What should we count? *Lines? Bugs? Test cases? Hours? Temper tantrums?*

Mechanism • How will increasing "extent of testing" affect the reading (the measure) on the instrument?

Side Effect ➢ If we do something that makes the measured result look better, will that mean that we've actually increased the extent of testing?

Purpose ➢ Why are we measuring this? What will we do with the number?

Scope ➢ Are we measuring the work of one tester? One team on one project? Is this a cross-project metrics effort? Cross-departmental research?

Copyright © Cem Kaner, 2000-2002.

Attributes and Instruments

Length	Ruler
Duration	Stopwatch
Speed	Ruler / Stopwatch
Sound energy	Sound level meter
Loudness	Sound level comparisons by humans
Tester goodness	??? Bug count ???
Code complexity	??? Branches ???
Extent of testing???	???
----Product coverage	??? Count statements / branches tested ???
----Proportion of bugs that we've found	??? Count bug reports or graph bug curves???

Copyright © Cem Kaner, 2000-2002.

Defining the Attribute

- Imagine being on the job. Your local PBH (pointy-haired boss) drops in and asks

  "So, tell me.

  How much testing have you gotten done?"

Copyright © Cem Kaner, 2000-2002.

The Question is Remarkably Ambiguous

Common answers are based on the:

Product ➢ We've tested 80% of the lines of code.

Plan ➢ We've run 80% of the test cases.

Results ➢ We've discovered 593 bugs.

Effort ➢ We've worked 80 hours a week on this for 4 months. We've run 7,243 tests.

Copyright © Cem Kaner, 2000-2002.

The Question is Remarkably Ambiguous

Common answers are based on the:

Obstacles ➢ We've been plugging away but we can't be efficient until X, Y, and Z are dealt with.

Risks ➢ We're getting a lot of complaints from beta testers and we have 400 bugs open. The product *can't be* ready to ship in three days.

Quality of Testing ➢ Beta testers have found 30 bugs that we missed. Our regression tests seem ineffective.

History across projects ➢ At this milestone on previous projects, we had fewer than 12.3712% of the bugs found still open. We should be at that percentage on this product too.

Copyright © Cem Kaner, 2000-2002.

What Are We Measuring?

- Before we can measure something, we need some sense of what we're measuring. It's easy to come up with "measurements" but we have to understand the relationship between the thing we want to measure and the statistic that we calculate to "measure" it.
- **If we want to measure the "extent of testing", we have to start by understanding what we mean by "extent of testing."**

Surrogate measures

- "Many of the attributes we wish to study do not have generally agreed methods of measurement. To overcome the lack of a measure for an attribute, some factor which can be measured is used instead. This alternate measure is presumed to be related to the actual attribute with which the study is concerned. These alternate measures are called surrogate measures."
  - Mark Johnson's MA Thesis
- "Surrogates" provide unambiguous assignments of numbers according to rules, but they don't provide an underlying theory or model that relates the measure to the attribute allegedly being measured.

Consider bug counts

- Do bug counts measure testers?
- Do bug counts measure thoroughness of testing?
- Do bug counts measure the effectiveness of an automation effort?
- Do bug counts measure how near we are to being ready to ship the product?

How would we know?

Copyright © Cem Kaner, 2000-2002.

Bug counts and testers

To evaluate an instrument that is supposed to measure an attribute, we have to ask two key questions:

- What ***underlying mechanism***, or fundamental relationship, justifies the use of the reading we take from this instrument as a measure of the attribute? ***If the attribute increases by 20%, what will happen to the reading?***
- What can we know from the instrument reading? How tightly is the reading traceable to the underlying attribute? ***If the reading increases by 20%, does this mean that the attribute has increased 20%.*** If the linkage is not tight, we risk serious ***side effects*** as people push the reading (the "measurement") up and down without improving the underlying attribute.

Copyright © Cem Kaner, 2000-2002.

Bug counts and testers: mechanism?

Suppose we could improve testing by 20%.

This might mean that:

- We find more subtle bugs that are important but that require more thorough investigation and analysis
- We create bug reports that are more thorough, better researched, more descriptive of the problem and therefore more likely to yield fixes.
- We do superb testing of a critical area that turns out to be relatively stable.

The bug counts might even go down, even though tester goodness has gone up.

Bug counts & testers: Side effects

What if you could increase the count of reported bugs by 20%?

If you reward testers for higher bug counts, won't you make changes like these more likely?

- Testers report easier to find, more superficial bugs
- Testers report multiple instances of the same bug
- Programmers dismiss design bugs as non-bugs that testers put in the system to raise their bug counts
- No one will work on the bug tracking system or other group infrastructure.
- Testers become less willing to spend time coaching other testers.

Example: Bug Counts

Attribute ➢ Not sure. Maybe we're thinking of percentage found of the total population of bugs in this product.

Instrument ➢ Bugs found. (Variations: bugs found this week, etc., various numbers based on bug count.)

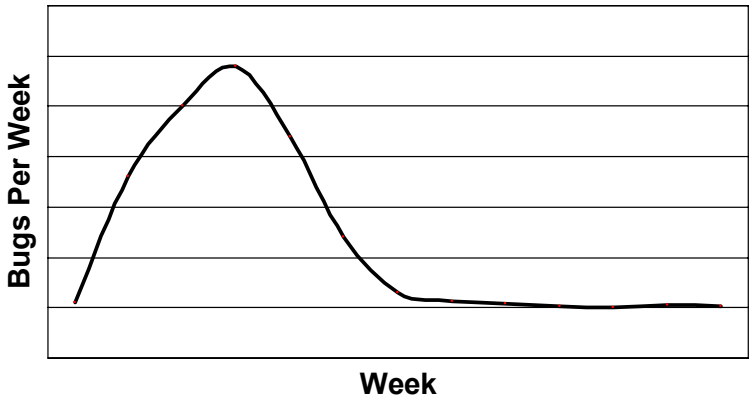
Mechanism ➢ If we increase the extent of testing, does that result in more bug reports? *Not necessarily.*

Side Effect ➢ If we change testing to maximize the bug count, does that mean we've achieved more of the testing? *Maybe in a trivial sense, but what if we're finding lots of simple bugs at the expense of testing for a smaller number of harder-to-find serious bugs.*

Copyright © Cem Kaner, 2000-2002.

The Bug Curve

What Is This Curve?



Copyright © Cem Kaner, 2000-2002.

Example: Bug Curves

Attribute ➢ We have a model of the rate at which new bugs will be found over the life of the project.

Instrument ➢ Bugs per week. A key thing that we look at is the agreement between the predictive curve and the actual bug counts.

Mechanism ➢ As we increase the extent of testing, will our bug numbers conform to the curve? *Not necessarily. It depends on the bugs that are left in the product.*

Side Effect ➢ If we do something that makes the measured result look better, will that mean that we've actually increased the extent of testing? *No, no, no. See side effect discussion.*

Copyright © Cem Kaner, 2000-2002.

Side Effects of Bug Curves

Earlier in testing: (Pressure is to increase bug counts)

- Run tests of features known to be broken or incomplete.
- Run multiple related tests to find multiple related bugs.
- Look for easy bugs in high quantities rather than hard bugs.
- Less emphasis on infrastructure, automation architecture, tools and more emphasis of bug finding. (Short term payoff but long term inefficiency.)

Copyright © Cem Kaner, 2000-2002.

Some Side Effects of Bug Curves

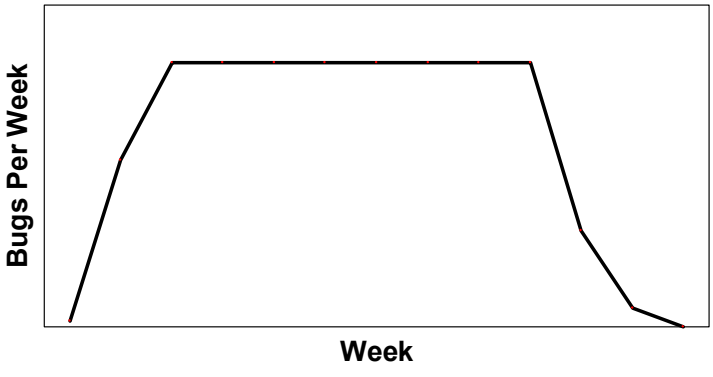
Later in testing: (Pressure is to decrease new bug rate)

- Run lots of already-run regression tests
- Don't look as hard for new bugs.
- Shift focus to appraisal, status reporting.
- Classify unrelated bugs as duplicates
- Class related bugs as duplicates (and closed), hiding key data about the symptoms / causes of the problem.
- Postpone bug reporting until after the measurement checkpoint (milestone). (Some bugs are lost.)
- Report bugs informally, keeping them out of the tracking system
- Testers get sent to the movies before measurement checkpoints.
- Programmers ignore bugs they find until testers report them.
- Bugs are taken personally.
- More bugs are rejected.

Copyright © Cem Kaner, 2000-2002.

Bug Curve Counterproductive?

Shouldn't We Strive For <u>This</u> ?



Copyright © Cem Kaner, 2000-2002.

Austin on Measurement Dysfunction

- Schwab & U.S. Steel
  - Counting ingots
  - How might these people have improved *measured* productivity?

Robert Austin, Measuring and Managing Performance in Organizations.

Copyright © Cem Kaner, 2000-2002.

Measurement Dysfunction

- In an organizational context, dysfunction is defined as the consequences of organizational actions that interfere with the attainment of the spirit of stated intentions of the organization. (Austin, p. 10)
- Dysfunction involves fulfilling the letter of stated intentions but violating the spirit.

Copyright © Cem Kaner, 2000-2002.

Measurement Dysfunction

- Examples from law enforcement
  - Quotas
  - Percentage successful prosecutions
  - Ratio of arrests to prosecutions
    - Measured from the perspective of the police
    - Measured from the perspective of the prosecutor
    - Measured from the perspective of the crime lab

Copyright © Cem Kaner, 2000-2002.

Austin on the 2-Party Model

- Principal
  - E.g. the employer, the person who wants the result and who directly profits from the result.
  - In Austin's model, we assume that the employer is motivated by maximum return on investment
- Agent
  - E.g. the employee.
  - In Austin's model, the employee wants to do the least work for the most money

Copyright © Cem Kaner, 2000-2002.

Austin: 2-Party Model – Supervisory Issues

- No supervision
  - No work
- Partial supervision
  - Work only on what is measured
- Full supervision
  - Work according to the production guidelines laid out by the employer

Copyright © Cem Kaner, 2000-2002.

Austin's 3-Party Model

- Principal
  - With respect to the agent, same as before: least pay for the most work.
  - With respect to the customer, wants to increase customer satisfaction
- Agent
  - With respect to principal, same as before: least work for the most pay
  - With respect to the customer, motivated by customer satisfaction
- Customer
  - Wants the most benefit for the lowest price

Copyright © Cem Kaner, 2000-2002.

Austin's 3-Party Model Supervisory Model

- No supervision
  - Agent works to the extent that increasing customer satisfaction provides more "benefit" to the agent (worker) than it costs the agent to provide the work
- Full supervision
  - Agent does exactly what should be done to increase customer satisfaction

Austin's 3-Party Model Supervisory Model

- Partial supervision
  - Agent is motivated by increased customer satisfaction and by rewards for performing along measured dimensions.
  - To the extent that the agent works in ways that don't maximize customer satisfaction at a given level of effort, we have distortion.
  - To the extent that the agent works in ways that reduce customer satisfaction below the level that would be achieved without supervision, we have dysfunction

Austin's 3-Party Model Supervisory Model

- Back to full supervision
  - What benefits are associated with full supervision?
  - What costs are associated with full supervision?
  - Imagine you were supervising a programmer who had a 6 week (best guess) programming task. What would you have to know / measure in order to achieve full supervision?
  - In general, what are the obstacles to achieving full supervision of knowledge workers?
  - Is it reasonable to try for full supervision or are we stuck with partial (or no) supervision?

Austin's 3-Party Model Supervisory Model

- A key aspect of this model is that it builds in the notion of internal motivation.
- Under ***full supervision*** with forcing contracts, perhaps internal motivation is unnecessary. (I disagree, but perhaps we can pretend that it is unnecessary.)
- Under ***partial supervision*** and no supervision, internal motivation plays an important role in achieving customer satisfaction and in eliciting effort and results from the agent.
- This comes into play in Austin's vision of delegatory management.

Tie This Back to Our Measurement Model

- Full supervision:
  - There is an instrument for every significant attribute and every instrument directly measures the attribute. ***PROBABLY IMPOSSIBLE.***
- Partial supervision:
  - Some attributes are unmeasured or are measured using proxies (or surrogates), measures that are loosely tied to the attribute under study.
- Distortion:
  - Side effects
- Dysfunction
  - Really bad side effects
- Informational vs Motivational Measures
  - Purpose of use
  - Scope of use

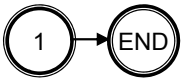
What protects data gathered for informational purposes from being used for motivational purposes?

Flowgraphs: Basic definitions

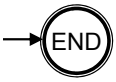
- Flowgraph is a directed graph
  - Nodes
    - Start
    - Terminal
    - Predicate / Decision
    - Procedural
  - Edges - connect two nodes
  - Arcs - directed edges

Nodes

A single statement



The GOTO statement



- In our course (and in our text), nodes are "statement nodes". They normally correspond to a single statement. Other computer scientists often represent states with nodes. The action that transforms the program from one state to another (such as execution of a statement) is shown on an arc.
- A GOTO statement does not appear on a node. It is a pure vector, pointing to the place to transfer control.
- The terminal node has a single function—it is the end, such as an endif. It is a logical connection point, not a source of action.

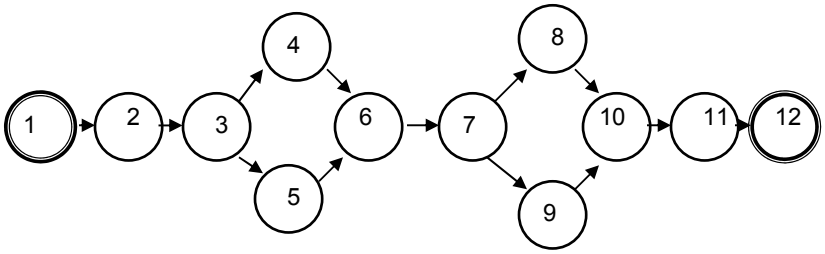
Copyright © Cem Kaner, 2000-2002.

Degree of a Node

- In-degree = # arcs to node
  - In-degree of start node is often but not necessarily 0
- Out-degree = # arcs from the node
  - Out-degree of terminal node = 0
- Procedure Node = out-degree = 1
- Predicate Node = out-degree > 1

Copyright © Cem Kaner, 2000-2002.

Zuse example Figure 2.3



Identify the following types of nodes: Start, Terminal, Predicate, Procedure
Identify the in-degrees of the nodes. Can you find nodes with in-degree of 0? 1? 2?
Identify the out-degrees of the nodes. Can you find nodes with out-degree of 0? 1? 2? 3?

Copyright © Cem Kaner, 2000-2002.

"Proper" Flowgraph

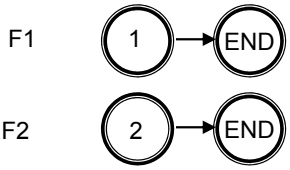
- Execution starts at the start node, S and ends at the terminal node, T
- For each node, N,
  - There is a path from start node, S, to N
  - There is a path from N to terminal node, T
  - N could be replaced with a proper flowgraph and the resulting flowgraph will still be a proper flowgraph

Copyright © Cem Kaner, 2000-2002.

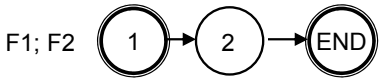
Sequencing and Nesting of Flowgraphs

- If F1 and F2 are two flowgraphs
  - We make a **sequence** of F1 and F2 by replacing the terminal node of F1 with the start node of F2.
  - Notation:
    - F1; F2
    - Seq (F1, F2)
    - P2 (F1, F2)
    - F1 o F2

Sequencing these:



Yields this

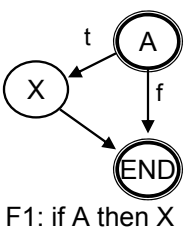


Copyright © Cem Kaner, 2000-2002.

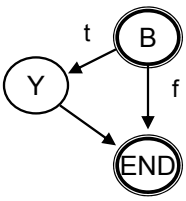
Sequencing and Nesting of Flowgraphs

- If F1 and F2 are two flowgraphs and X is a procedure node.
  - F2 is **nested in F1 at X** if we replace the arc from X with the flowgraph F2 (F2's start node is X)
  - Notation:
    - F1(F2 on X)
    - F1(F2) is OK if there is no ambiguity

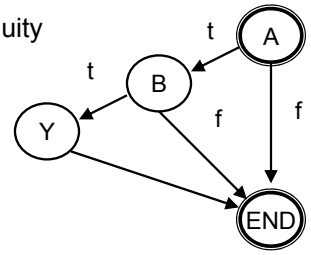
F1(F2):
if A then if B then Y



F1: if A then X



F2: if B then Y



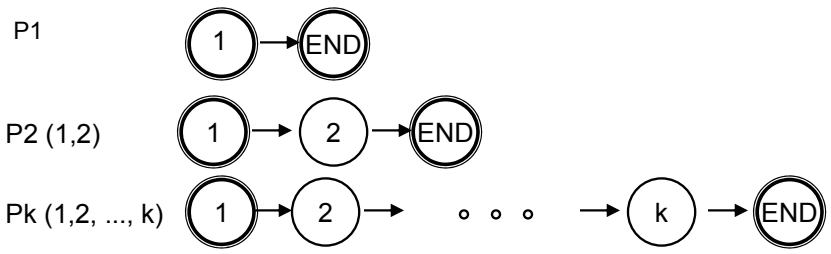
Copyright © Cem Kaner, 2000-2002.

Prime Flowgraph

- Cannot be non-trivially decomposed by sequencing or nesting
- Different languages have different prime flowgraphs.
- Common ones:
  - Pk = Sequence of length K
  - D0 = If ... Then
  - D1 = If ... Then ... Else
  - D2 = While ... Do
  - D3 = Repeat ... Until
  - Ck = Case statement with K cases

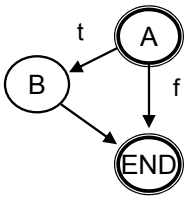
Copyright © Cem Kaner, 2000-2002.

Prime Flowgraphs Pk – A simple series of statements



Copyright © Cem Kaner, 2000-2002.

Prime Flowgraphs D0 If A then B

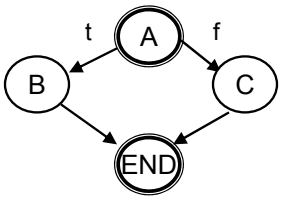


D0 (A,B)

Copyright © Cem Kaner, 2000-2002.

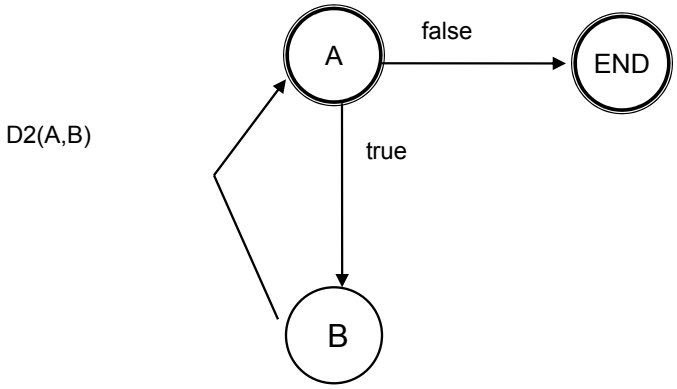
Prime Flowgraphs D1 If A then B else C

D1 (A,B,C)



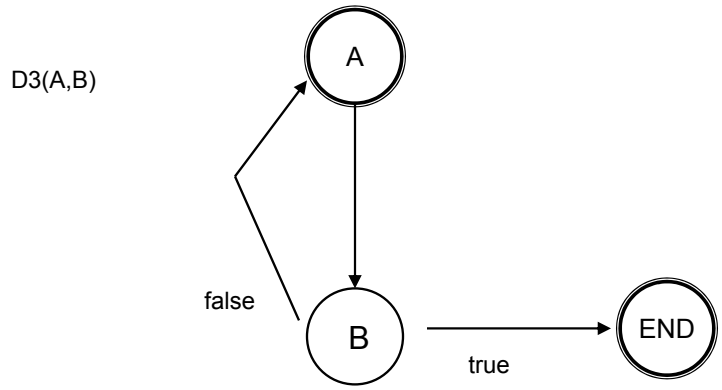
Copyright © Cem Kaner, 2000-2002.

Prime Flowgraphs D2 while A do B



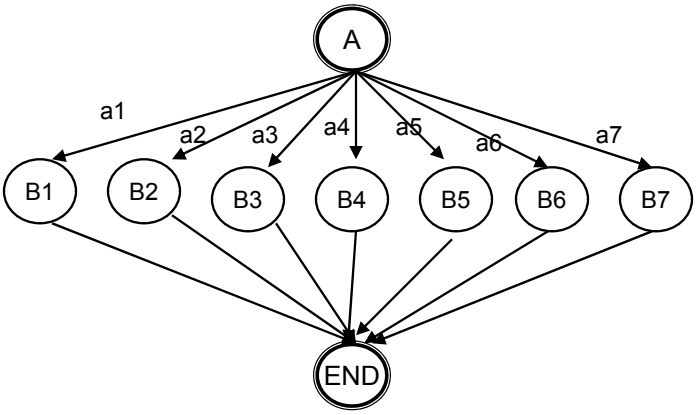
Copyright © Cem Kaner, 2000-2002.

Prime Flowgraphs D3 repeat A until B



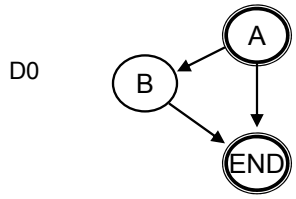
Copyright © Cem Kaner, 2000-2002.

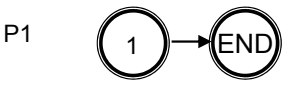
Prime Flowgraphs Ck Case statement

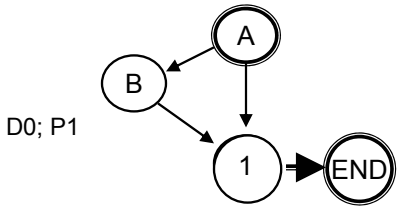


Copyright © Cem Kaner, 2000-2002.

Sequence Practice



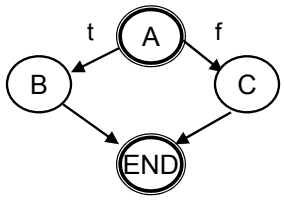




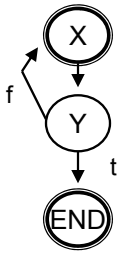
Copyright © Cem Kaner, 2000-2002.

Sequence Practice

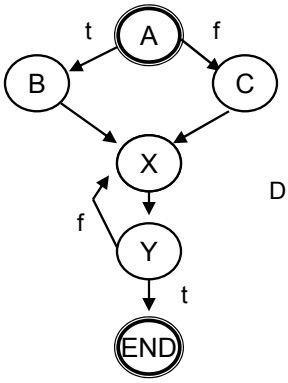
D1



D3



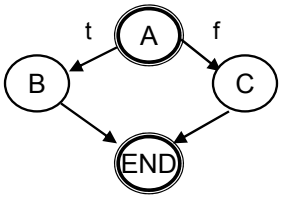
D1; D3



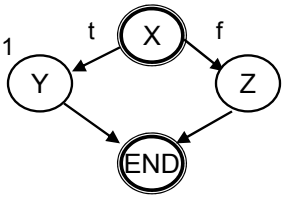
Copyright © Cem Kaner, 2000-2002.

Sequence Practice

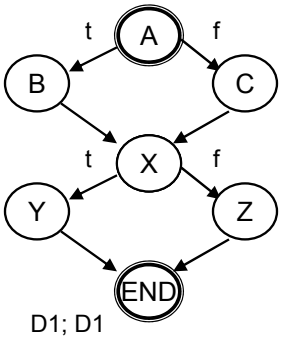
D1



D1

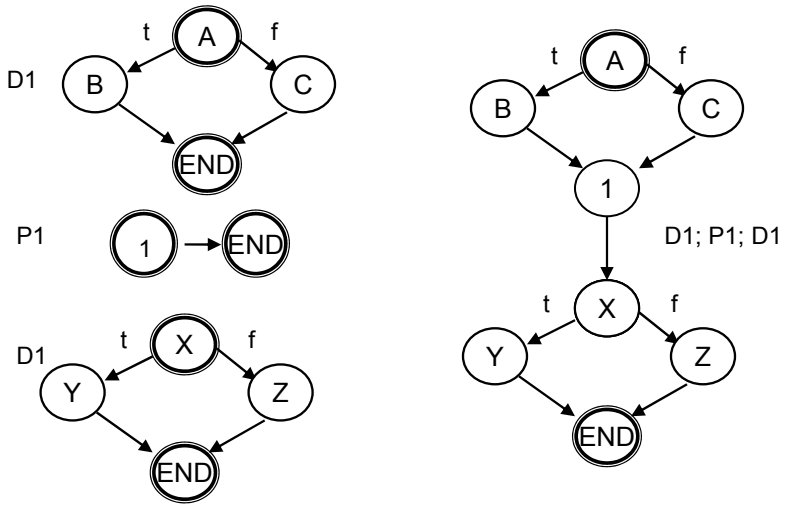


D1; D1

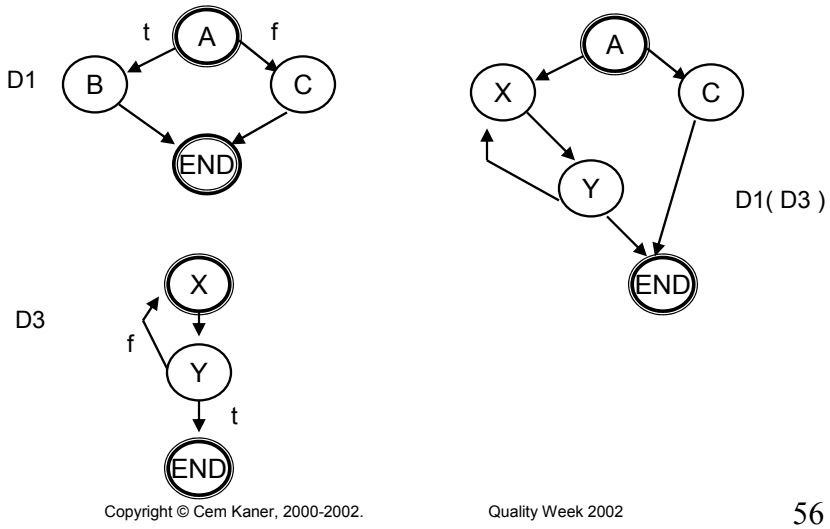


Copyright © Cem Kaner, 2000-2002.

Sequence Practice



Nesting Practice

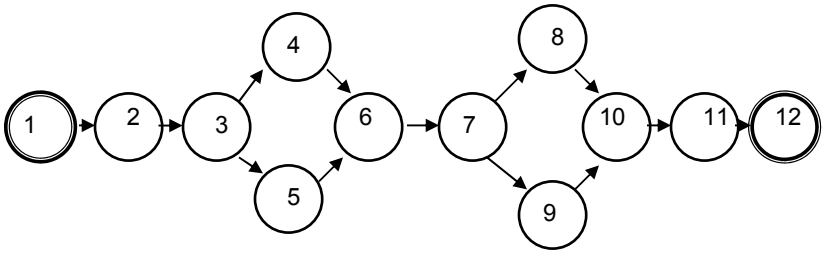


Composition of flowgraphs

- Functions can be modeled as directed graphs, and built up by composition of the basic flowgraphs
- Every flowgraph has a unique decomposition into a hierarchy of primes

Copyright © Cem Kaner, 2000-2002.

Decompose this

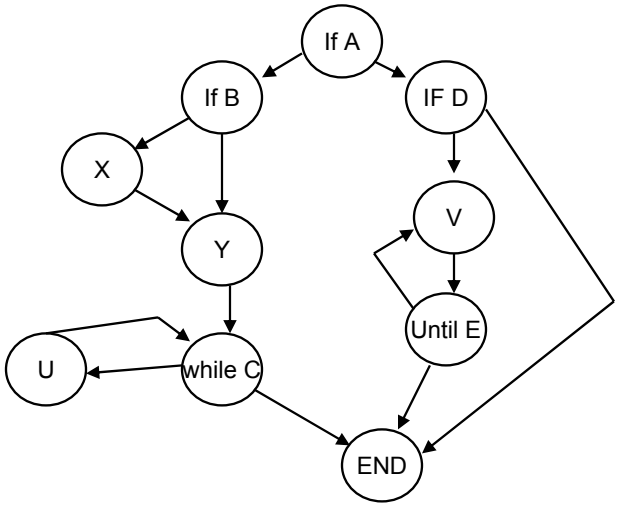


Copyright © Cem Kaner, 2000-2002.

Graph this (From Fenton / Pfleeger)

```
If A
    then
        begin
                If B then do X;
                Y;
                while C do U
        end
    else
        if D
                then do
                        repeat V until E
```


Graph this (continued)



Now, decompose it into a hierarchy of primes

Structuredness

- For a family, S, of prime flowgraphs
- A family of graphs is S-structured (or are S-graphs) if it satisfies the following recursive rules:
  - Each member of S is S structured
  - If F1 and F2 are S S structured graphs then so are
    - F1; F2
    - F1 (F2) wherever nesting of F2 onto F1 is defined
  - No flowgraph is an S S structured graph unless it can be shown to be generated by a finite number of applications of the above steps

NOTE: If SD = {P1, D0, D2}, the set of SD-graphs is the class of "D-structured" or "structured" graphs. Every algorithm can be encoded as an SD-graph (Bohm & Jacopini)

Hierarchical measures – in general

- A measure M is a hierarchical measure if it can be defined on the set of S-graphs by specifying
  - M(F) for each F in S (rule M1)
  - The sequencing function
    M(F1;F2;...;Fk) (rule M2)
  - The nesting functions for
    each F in S (rule M3)
- We can compute a hierarchical measure for a program once we know the rules, M1, M2 and M3 and the decomposition tree.
- (These slides are closely based on Fenton / Pfleeger)

Hierarchical measures: Nesting

- Primes (define as follows)
  - Depth of nesting of P1 is 0
  - Depth of nesting of any other prime is 1
    - D (P1) = 0
    - D (F prime but <> P1) = 1
- Sequence
  - Depth of nesting of sequence F1, F2, ..., Fk is maximum of the depth of nesting of the Fi's.
    - D (F1; ... ; Fk) = Max (D(F1), ..., D(Fk))
- Nesting
  - Depth of nesting of flowgraph F(F1, ..., Fk) is max of the depth of nesting of the F1 plus 1 b/c of the extra nesting level in F). So
    - D (F(F1; ... ; Fk)) = 1+ Max (D(F1), ..., D(Fk))

Hierarchical measures: Length

- M1:
  - V(P1) = 1
  - V(F) = N+1, where N is number of procedure nodes in F
- M2:
  - V(F1;F2;...;Fk) = Sum (V(Fi))
- M3:
  - V(F(F1,...,Fk))= 1+Sum(V(Fi) for each prime Fi <> P1)
- Example: compute v(D1((D0;P1;D2),D0(D3)))

Hierarchical measures – number of nodes

- M1:
  - n(F) = number of nodes in F, for each prime F
- M2:
  - n(F1; ... ; Fk) = Σ n(Fi) – k +1
- M3:
  - N(F(F1, ... , Fk)) = n(F) + Σ n(Fi)- 2Kfor each prime F

- Try it for
  - P2 (and decompose it to P1;P1)
  - D1 (D3, D1)

Copyright © Cem Kaner, 2000-2002.

Hierarchical measures – number of edges

- M1:
  - e(F) = number of edges in F, for each prime F
- M2:
  - e(F1; ... ; Fk) = Σ e(Fi)
- M3:
  - e(F(F1, ... , Fk)) = e(F) + Σ e(Fi)- kfor each prime F

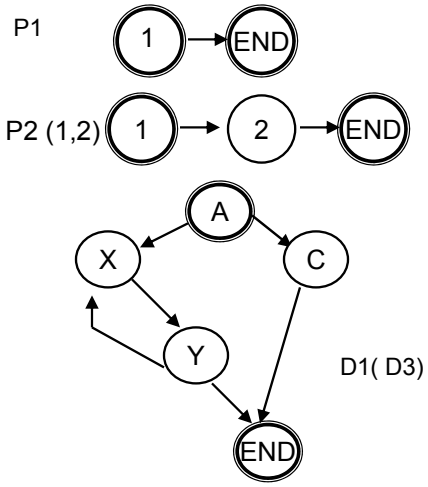
- Try it for
  - P2 (and decompose it to P1;P1)
  - D1 (D3, D1)

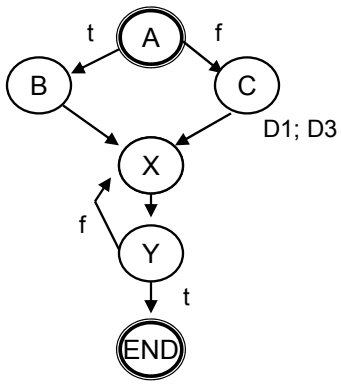
Copyright © Cem Kaner, 2000-2002.

Hierarchical measures – cyclomatic complexity

- Cyclomatic complexity (F) = e(F) – n(F) + 2
  - e(F) = number of arcs in F
  - n(F) = number of nodes in F
- This is the number of linearly independent paths through F
- M1:
  - c(F) = 1+d where d is number of predicates in F, prime F
- M2:
  - c(F1; . . .; Fk) = Σ c(Fi) – k + 1, for each prime Fi
- M3:
  - c(F(F1, ... , Fk)) = c(F) + Σ c(Fi)- kfor each prime F

Cyclomatic complexity, simple examples





Test coverage metrics – basis path coverage

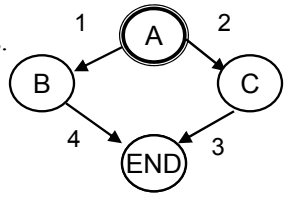
- McCabe's metric counts the number of basis paths through the program, essentially the number of linearly independent paths through the program. If you design your tests to hit every basis path, you will cover every statement and every branch in the program.

Graph theory underlying basis paths (definitions)

- **Strongly connected graph**: for any node, x, there is a path from x to y and a path from y to x.
- A **circuit** is a path that begins and ends at the same node.
- A **cycle** is a circuit with no node (other than the starting node) included more than once.
- A path, P, is a **linear combination of paths**, P1, ..., Pn if there are integers, Ai such that $P = \Sigma Ai * Pi$.
- A set of paths is **linearly independent** if no path in the set is a linear combination of the others.

Basis paths

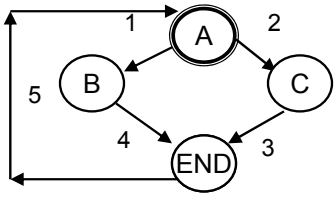
- Notation:
  - Rather than labeling nodes, lets label edges.



  - We can describe a path as an n-tuple, such as <2, 3> or <1,4>
  - We can create a path vector that shows the number of times each path is traversed
    - <2,3> = [0 1 1 0].
    - <1,4> = [1 0 0 1].
    - <1,2,3,4> = [1 1 1 1] but this is an **infeasible** path.
    - (1 1 1 1) = [1 0 0 1] + [0 1 1 0] (this is basic matrix algebra)
  - Cyclomatic complexity = e(F) – n(F) + 2 = 4 – 4 + 2 = size of basis set

Basis paths

- To create the basis set of cycles, we can turn a directed graph into a strongly connected graph by drawing an arc from the end to the start.
- [1 0 0 1 1] is a circuit
- [0 1 1 0 1] is a circuit
- [1 1 1 1 2] is a circuit *(what is the path?)*
- [1 0 0 1 1] and [0 1 1 0 1] are cycles but [1 1 1 1 2] is not.
- [1 1 1 1 2] is a linear combination of [1 0 0 1 1] and [0 1 1 01]
- Any other path that you could actually take through the graph is a linear combination of [1 0 0 1 1] and [0 1 1 01]
- This is a **basis set** of cycles

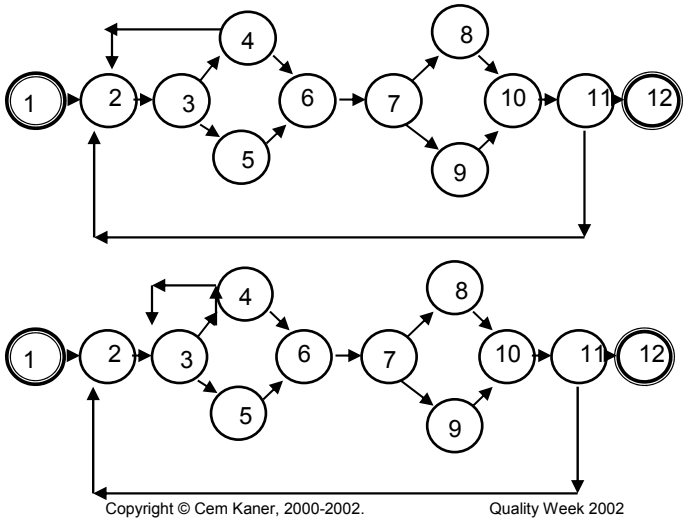


Basis paths

- Once we know the basis set of cycles, we eliminate the fictitious branch (from stop to start), reducing the vectors by a column:
  - [1 0 0 1] and [0 1 1 0] is a basis set of linearly independent paths.
  - Basis sets (of cycles or flowgraph paths) are not unique
  - Question: Aren't [1 1 0 0] and [0 0 1 1] linearly independent of the basis paths? Why aren't they usable?

Copyright © Cem Kaner, 2000-2002.

Reality Check: Which is more complex?



Two programs with the same McCabe complexity number can have very different complexity.

Copyright © Cem Kaner, 2000-2002.

Code coverage

- Coverage measures the amount of testing done of a certain type. Since testing is done to find bugs, coverage is a measure of your effort to detect a certain class of potential errors:
  - ***100% line coverage*** means that you tested for every bug that can be revealed by simple execution of a line of code.
  - ***100% branch coverage*** means you will find every error that can be revealed by testing each branch.
  - ***100% coverage*** should mean that you tested for every possible error. This is obviously impossible.

Copyright © Cem Kaner, 2000-2002.

Benefits of coverage

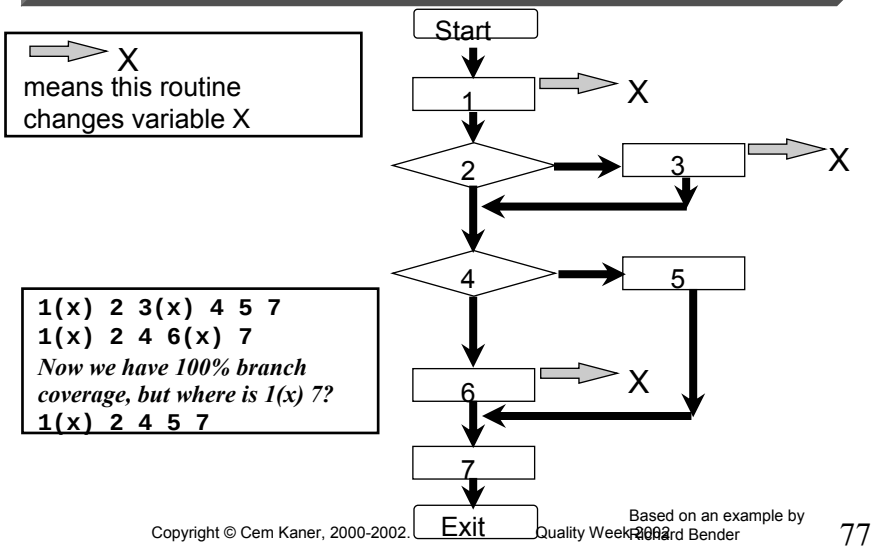
Before I attack coverage measures, let me acknowledge that they are often useful.

- Many projects achieve low statement coverage, as little as 2% at one well-known publisher that had done (as measured by tester-hours) extensive testing and test automation. The results from checking statement coverage caused a complete rethinking of the company's automation strategy.
- Coverage tools can point to unreachable code or to code that is active but persistently untested.

Coverage tools can provide powerful diagnostic information about the testing strategy, even if they are terrible measures of the extent of testing.

Copyright © Cem Kaner, 2000-2002.

Statement / Branch Coverage and Data Flows



Example: Statement/Branch Coverage

Attribute ➢ Extent of testing – ***How much of the product have we tested?***

Instrument ➢ Count statements and branches tested

Mechanism ➢ If we do more testing and find more bugs, does that mean that our line count will increase? ***Not necessarily. Example—configuration tests.***

Side Effect ➢ If we design our tests to make sure we hit more lines, does that mean we'll have done more extensive testing? ***Maybe in a trivial sense, but we can achieve this with weaker tests that find fewer bugs.***

Purpose ➢ Not specified

Scope ➢ Not specified

Statement / Branch Coverage Just Test the Flowchart

You're not testing:

- » data flow
- » tables that determine control flow in table-driven code
- » side effects of interrupts, or interaction with background tasks
- » special values, such as boundary cases. These might or might not be tested.
- » unexpected values (e.g. divide by zero)
- » user interface errors
- » timing-related bugs
- » compliance with contracts, regulations, or other requirements
- » configuration/compatibility failures
- » volume, load, hardware faults

Copyright © Cem Kaner, 2000-2002.

If we use "coverage"?

- If we improve testing by 20%, does this result in a 20% increase in "coverage"? Does it necessarily result in ANY increase in "coverage"?
- If we increase "coverage" by 20%, does this mean that there was a 20% improvement in the testing?
- If we achieve 100% "coverage", do we really think we've found all the bugs?

Copyright © Cem Kaner, 2000-2002.

Side effects and "coverage"

- Without a mechanism that ties changes in the attribute being measured to changes in the reading we get from the instrument, we have a "measure" that is ripe for abuse.
- People will optimize what is tracked. If you track "coverage", the coverage number will go up, but (as Marick has often pointed out) the quality of testing might well go down.

Copyright © Cem Kaner, 2000-2002.

Statement / Branch Coverage and Data Flows

- How many basis paths are there in this program?

1	2	3	4	5	6	7	Exit
1	1	1	1	1	0	1	1
1	1	0	1	0	1	1	1

- Can we achieve complete basis path coverage without ever hitting the critical data flow?

Copyright © Cem Kaner, 2000-2002.

Basis paths

- By the way, which of these columns is variable?

1	2	3	4	5	6	7	Exit
1	1	1	1	1	0	1	1
1	1	0	1	0	1	1	1
1	1	1	1	0	1	1	1

Basis paths

- So we can reduce the big chart to the simpler small chart. It is much easier to prove that the vectors form a basis path in the simplified matrix.

1	2	3	4	5	6	7	Exit
X	X	1	X	1	0	X	X
X	X	0	X	0	1	X	X
X	X	1	X	0	1	X	X

3	5	6
1	1	0
0	0	1
1	0	1

Data Flows

- A set-use pair is a dataflow
- A set-use pair with no intervening set is a first-order dataflow.
  - In the example, set the value of X in lines 1, 3, and 6.
  - Path 1(set x) 2 3(set x) 4 5 7 (print x) has
    - a first-order dataflow from line 3 to line 7 and
    - a second-order dataflow from line 1 (through line 3 where x is reset) to line 7
- 100% dataflow coverage (in testing) usually means covering all the first-order dataflows.

Copyright © Cem Kaner, 2000-2002.

Goal-Question-Metrics Approach

- To decide what to measure, we should first know why we care about the answer. Given a goal for the measurement, we can work forward to collect information that can help us meet that goal.
- Basic approach
  - Set the goal
  - Identify questions that would give you information that you need in order to meet the goal
  - Determine whether there are (or whether you can create) metrics that can help you answer those questions.

Copyright © Cem Kaner, 2000-2002.

GQM Template for Defining a Goal

- Questions usually look for information like:
  - **Purpose:** TO (characterize, evaluate, predict, motivate, etc.) THE (process, product, model, metric, etc.) IN ORDER TO (understand, assess, manage, engineer, learn, improve, test, etc.)
  - **Perspective:** EXAMINE THE (cost, effectiveness, correctness, defects, changes, product metrics, reliability, etc.) FROM THE POINT OF VIEW OF (the programmer, manager, customer, corporate perspective, etc.)
  - **Environment:** The environment consists of the following: process factors, people factors, problem factors, methods, tools, constraints, etc.

Adapted from Basili

Copyright © Cem Kaner, 2000-2002.

Example from Rosenberg

- To PREDICT the SCHEDULE in order to MANAGE it.
  - What are some relevant questions?
  - Which ones might be answerable with metrics?
  - What assumptions or preconditions or challenges are associated with those questions or metrics?

Copyright © Cem Kaner, 2000-2002.

Another example

- To EVALUATE the COSTS AND BENEFITS of CODE INSPECTIONS in order to DETERMINE WHETHER TO CONTINUE THIS PROCESS.
- To analyze this, we have to break it down. What question(s) would we ask about each of these?
  - Evaluate
  - Costs
  - Benefits
  - Code inspections
  - Determine
  - Continue
  - This process

GQM Done Poorly or Well

- The goal has to be one that can be achieved via measurement.
- **Evaluate the questions.** If you could answer them, would you achieve your goal? If not, what other information would you need?
- **Evaluate the metrics.** If you collected them, would they provide you all of the information you need to answer your questions?
- ***GQM in practice is often a rationalization to collect the same old metrics.***
- **Evaluate the metrics.** Apply the 10-factor analysis (or some other careful analysis of validity) to them.

Multi-Dimensional Measurement

- The idea of multi-dimensional measurement is to put together a pattern of information that, collectively, gives a more accurate picture.
- COCOMO is a leading example of this approach. See http://www.jsc.nasa.gov/bu2/COCOMO.html and http://sunset.usc.edu/research/COCOMOII/Docs/stc.pdf
- Balanced scorecards are a general scheme of this type. (Kaplan & Norton, *The Balanced Scorecard: Translating Strategy into Action*). Rather than reporting a single not-very-representative measure, use:
  - a small number (maybe 5 - 10) of different measures,
  - all of them meaningful to you,
  - none of them perfect,
  - all of them substantially different from each other,
  - selected in a way that distortion caused by attempting to optimize on a single measure will be reflected as a negative in at least one other measure.

Copyright © Cem Kaner, 2000-2002.

For example: Treat "Extent" as a Multidimensional Problem

- We developed the 8 aspects (or dimensions) of "extent of testing" by looking at the types of measures of extent of testing that we were reporting.
- Consider using a combination measure that looks at the 8 dimensions
  - product coverage — plan / agreement
  - effort — results
  - Obstacles — risks
  - quality of testing — project history

Copyright © Cem Kaner, 2000-2002.

Project Report / Component Map

Status report used by Elizabeth Hendrickson

Page 1 --- Issues that need management attention
Page 2 --- Component map
Page 3 --- Bug statistics

Component	Test Type	Tester	Total Tests Planned / Created	Tests Passed / Failed / Blocked	Time Budget	Time Spent	Projected for Next Build	Notes

We see in this report:

- Progress against plan
- Effort
- Obstacles / Risks
- Results

Copyright © Cem Kaner, 2000-2002.

Bach's Dashboard

Testing Dashboard				Updated 11/1/00	Build 32
Area	**Effort**	**Coverage Planned**	**Coverage Achieved**	**Quality**	**Comments**
File/edit	High	High	Low	😐	1345, 1410
View	Low	Med	Med	☺	
Insert	**Blocked**	Med	Low	☹	**1621**

We see coverage of areas, progress against plan, current effort, key results and risks, and obstacles.

Copyright © Cem Kaner, 2000-2002.

One approach: Balanced scorecard

- **For 101 examples of possible coverage measures, that might be suitable for balancing, see "Software Negligence and Testing Coverage" at www.kaner.com. These are merged in a list with over 100 additional indicators of extent of testing in the paper, "Measurement Issues & Software Testing", which is included in the proceedings.**

Suggested Lessons

- Simple charts can carry a lot of useful information and lead you to a lot of useful questions.
- Report multidimensional patterns, rather than single measures or a few measures along the same line.
- Think carefully about the potential side effects of your measures. Robert Austin criticizes the balanced scorecard approach because it can, and often does, still lead to abuse, especially if the measures don't balance each other out.
- Listen critically to reports (case studies) of success with simple metrics. If you can't see the data and don't know how the data were actually collected, you might well be looking at results that were sanitized by working staff (as a side effect of the imposition of the measurement process).

QW2002 Panel Session 4Q

Mr. Ross Collard
(Collard & Company) & Mr. Bret Pettichord
(Pettichord Consulting)

The Value of Agile Testing

Key Points

- Fact: agile testing is a hot new item.
- Claim: agile testing has great potential!
- Counter-claim: it's bunk, or even dangerous!

About the Panel Moderator

Ross Collard is president of Collard & Company, a consulting firm which is headquartered in Manhattan, New York City. His consulting and training clients have included: ADP, Alcatel, American Express, Anheuser-Busch, Apple, AT&T, Banamex, Bank of America, Bechtel, Blue Cross/Blue Shield, Boeing, British Airways, the CIA, Ciba Geigy, Cisco, Citibank, Computer Associates, Dayton Hudson, Dell, EDS, Exxon, General Electric, Goldman Sachs, Federal Reserve, Ford, Hewlett-Packard, Hughes Aircraft, IBM, Intel, Johnson & Johnson, JP Morgan, McGraw Hill, MCI, Merck, Microsoft, Motorola, NASA, Nortel, Novell, Procter & Gamble, Prudential, Sears Roebuck, Swiss Bank, U.S. Air Force, Verizon and Worldcom.

He has conducted seminars on business and information technology topics for businesses, governments and universities, including George Washington, Harvard and New York Universities, MIT, Stanford and U.C. Berkkeley.He has a BE in Electrical Engineering from the University of New Zealand (where he grew up), an MS in Computer Science from the California Institute of Technology and an MBA from Stanford University. His set of books on software testing is due to be published next year.

Bret Pettichord is an independent consultant with experience using a wide variety of test tools and approaches. He's developed automated tests for software in various domains, including publishing, accounting, sales, systems management, education and benefits administration. The tutorial builds on Lessons Learned in Software Testing: A Context- Driven Approach, the best-selling book he coauthored with Cem Kaner and James Bach. Many of his writings are collected at www.pettichord.com.

Specializing in software testing and test automation, Bret has 14 years experience in software development, including positions as a staff consultant at Segue Software and a test automation lead at BMC and then IBM/Tivoli before becoming an independent consultant. Recent clients have included Texas Instruments, Rational Software, Texas Department of Human Services, Deliotte Consulting, Whisperwire and Netpliance. He learned about software testing as a tax programmer at Arthur Andersen and a software tester and automator at Interleaf.

Position Statement on Agile Testing for Quality Week Debate

Testers Should Embrace Agile Programming

Bret Pettichord
Pettichord Consulting LLC
www.pettichord.com

Copyright © 2002 Bret Pettichord, all rights reserved

Agile software development is based on a hierarchy of values. Individuals and interactions are valued over processes and tools. Working software is valued over comprehensive documentation. Customer collaboration is valued over contract negotiation. Responding to change is valued over following a plan [1]. Extreme programming is the most popular agile development practice, but any practice that adheres to these values can properly be called agile.

Agile software development is a rejection of the traditional ideas about software development. These depend on a phased approach, often known as a *waterfall*: first requirements, then analysis, then design, then coding and finally testing and deployment. Each step results in a document or deliverable that is used to define the next. When followed fastidiously, testers receive software with complete design and requirements specifications. More often, testers get software with incomplete documentation. Even so, they often find bugs resulting from errors in design or analysis. These bugs are found relatively late in the process. They often result in unexpected delays.

Two recent revisions to the traditional approach deserve mention. One is the *spiral model*, which runs through each of these phases multiple times in shorter time frames or iterations. Another is the *V-model*, which has testers check the deliverables of each phase so that problems can be fixed before moving on to the next. Both models find ways to involve testers earlier in the process.

Agile methods take a more radical approach. They discourage the detailed documentation of requirements and design, emphasize writing code as soon as possible, and encourage frequent design changes. These ideas scare a lot of people, including my friend and colleague Ross Collard. Many of these people have been warning of the dangers of ill-defined requirements, poor planning and cowboy coding for years. Too often the results have been buggy software, schedule delays and software systems that didn't perform as expected.

These critics have been frustrated by their lack of impact even when they were getting little public resistance to their warnings. Now a vocal group of influential developers are boldly encouraging these "dangerous" practices and giving programmers a license to hack.

That's how I understand their concern and the context behind it. I take a very different view. In my 14 years of working in software development, I've noticed that speed has often been of the essence. Building software fast was not only essential to

meeting market windows, but it was also necessary to maintain project momentum. Without it, projects easily floundered with dueling designs and grandiose plans.

I've also seen projects buried under the weight of their documentation, too vast to read and too troublesome to revise, quickly becoming out of date. Effective communication requires concise documentation. A key insights of agile programming is that it's not the documentation that matters: it's the communication. Documentation is no good if it goes unread.

It's important to distinguish between programmers who are careless and undisciplined and programmers who are honestly trying to follow the new agile methods. They require discipline, although a different kind of discipline than traditional development. For example, agile programmers are expected to practice test-first design all the time. This practice has programmers writing tests before writing code. Testers have been encouraging this for at least a decade. Now the agile programmers are saying it too! Agile programmers report that test-first design speeds coding and improves design (with less coupling and more cohesion). Skeptics, accustomed to programmers who claim to do more testing than they actually do, might wonder whether agile programmers actually follow through. In fact, agile programmers have released a cornucopia of open-source testing frameworks over the past few years. They are serious about testing, and are creating the tools needed to do it well.

They are also interested in automated test suites to support *refactoring*. Refactoring is the practice of changing the structure of code without changing its behavior. Traditional development tries to understand how all the code will work together in advance. This is the design. With agile methods, this difficult process of imagining what code might look like before it is written is avoided. Instead, the code is restructured as needed to maintain a coherent design. Agile methods replace high-level design with frequent redesign (refactoring). Successful refactoring requires a way of checking whether the behavior was inadvertently changed. That's where the tests come in.[2]

Yet another reason is their preference for tests (code) to text (words) for describing system behavior. Tests are more precise than human language and they are also a lot more likely to be updated when the design changes. How many times have you seen design documents that no longer accurately described the current workings of the software? Out-of-date design documents look pretty much like up-to-date documents. Out-of-date tests fail.

Many testers are unhappy with agile development not so much because they think it is a bad practice but simply because it fails to give them the documentation they need for testing. This is an age-old complaint. Some testers claim to be unable to test a system without detailed and authoritative specifications describing how it should work. I take a different view. I often find many sources of expectations and intentions regarding software behavior. There may be gaps, conflicts and ambiguities, but they often provide enough to test from, reporting any surprises. Bug reports may or may not report bona fide defects. Nonetheless testers are most useful when they report discrepancies and let the team sort out right from wrong. Testers need to ask questions, and they need to be open-minded when their bug reports are rejected. Now we have a bunch of programmers who are encouraging this kind of reaction. I'm all for it.

I believe that agile programming is a major step forward. You may disagree. But regardless, agile programming is the wave of the future. The practices will develop and some of the extreme edges may be worn off, but it's only growing in influence and attraction. Some testers may not like it, but those who don't figure out how to live with it are simply going to be left behind.

Some testers are still upset that they don't have the authority to block the release of buggy software. Do they think they now have the authority to block the adoption of these new development methods? They'll need to get on this ship and if they want to try to keep it from the shoals. Stay on the dock if you wish. Bon Voyage!

1. Agile Manifesto. 2001. http://www.agilemanifesto.org/
2. Martin Fowler, *Refactoring: Improving the Design of Existing Code* (Addison-Wesley: 2000). The book uses 17 sound bites to summarize the key ideas for effective refactoring. Of them, 9 address issues relating to tests and testing.

QW2002 Paper 6Q

Mr. Robert Sabourin
(AmiBug.Com, Inc.)

Testimonial - A Fully Illustrated Overview of Software Testing

Key Points

- Entertaining view of over 150 types of software testing
- How different types of testing fit in the software development process
- New practitioners will learn to distinguish many methods and approaches

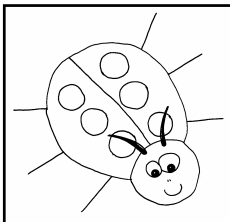
Presentation Abstract

How many different ways do you know to test software? This important workshop outlines over 150 different ways to test your software! Fully illustrated - with plenty of lively examples! You will also learn how testing fits into different software development processes and organizational structures. This workshop provides a powerful "TESTimonial" for testing in software development.

Topics Covered
Where testing fits in the development process
Where testing fits in the org chart
over 150 different ways to test your software!

About the Author

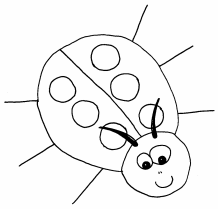
Robert Sabourin, P. Eng. has 20+ years management experience leading teams of software development professionals to consistently deliver projects on-time, on-quality and on-budget. He is a well-respected member of the software engineering community who has managed, trained, mentored and coached hundreds of top professionals in the field. He frequently writes and speaks to conferences around the world on software engineering, SQA, testing, management and internationalization.



Testimonial

Robert Sabourin
President
AmiBug.Com, Inc.
Montreal, Canada
rsabourin@amibug.com
www.amibug.com

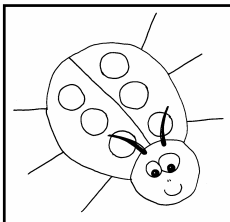
© Robert Sabourin, 2000



Software Testing Methods & Tools

Quick-Start Testimonial

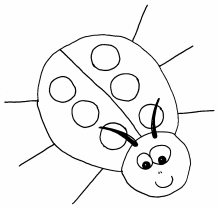
© Robert Sabourin, 2000



Testimonial

- Robert Sabourin , *Software Evangelist*
- President
- AmiBug.Com Inc.
- Montreal, Quebec, Canada
- rsabourin@amibug.com

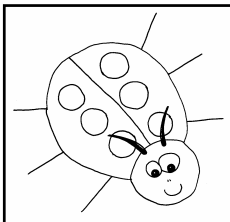
© Robert Sabourin, 2000



AmiBug.Com, Inc.

- Software Development & SQA Consulting
- Services
  - Training, Coaching and Professional Development
  - Light Effective Process
  - Team Building and Organization
  - We help people to get things done!

© Robert Sabourin, 2000



I am a Bug

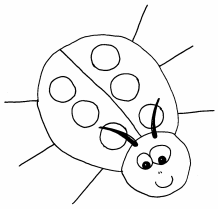
Robert & Catherine Sabourin

ISBN: 0-9685774-0-7

www.amazon.com

In the style of a children's book. Explains elements of software development process in a fun easy to read format.

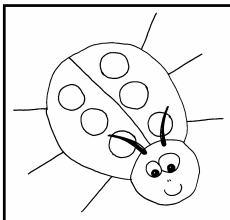
© Robert Sabourin, 2000



Testing in Development Process

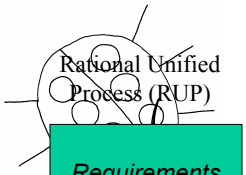
- Testing activities take place in all parts of software development
- From requirement eliciting to final shipment
- Testing is part of the development process
- Testing is part of the company business process

© Robert Sabourin, 2000



Types of testing and definitions

- Validation and Verification
  - Validate
    - correctness or suitability
    - vertical experts to confirm master results
  - Verification
    - confirm software operates as it is required to
    - double check to ensure results match those previously validated and if not then re-validate them



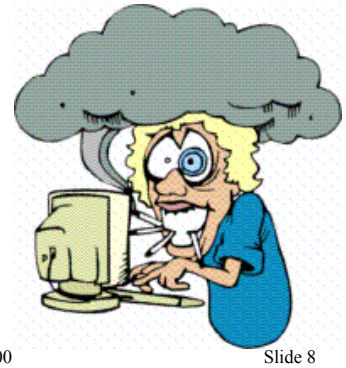
Phase

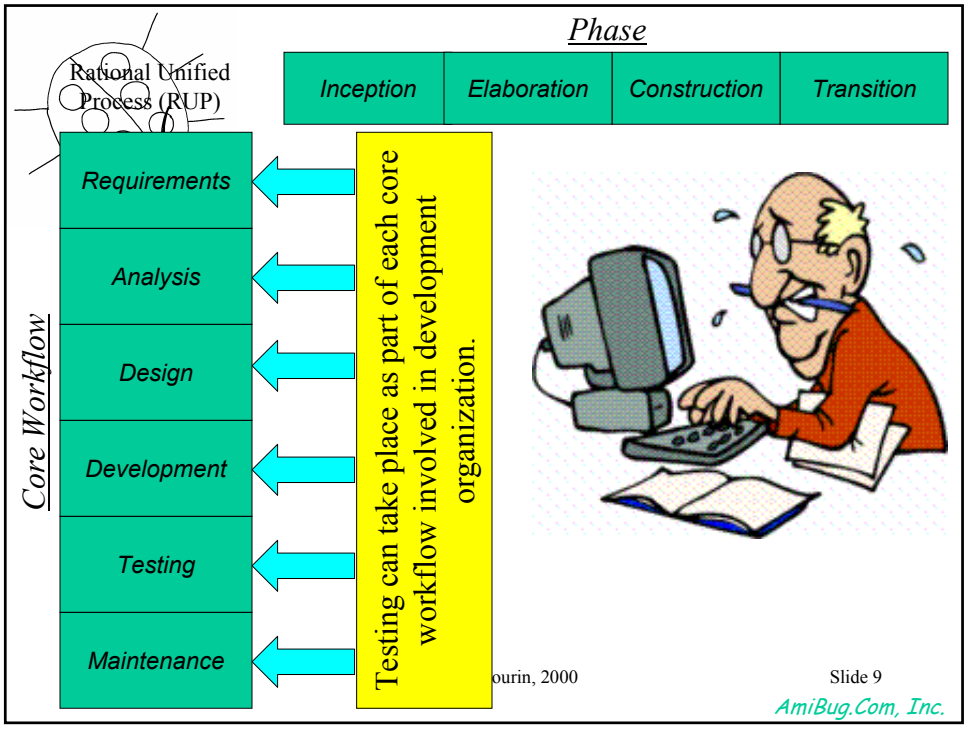
Inception	Elaboration	Construction	Transition

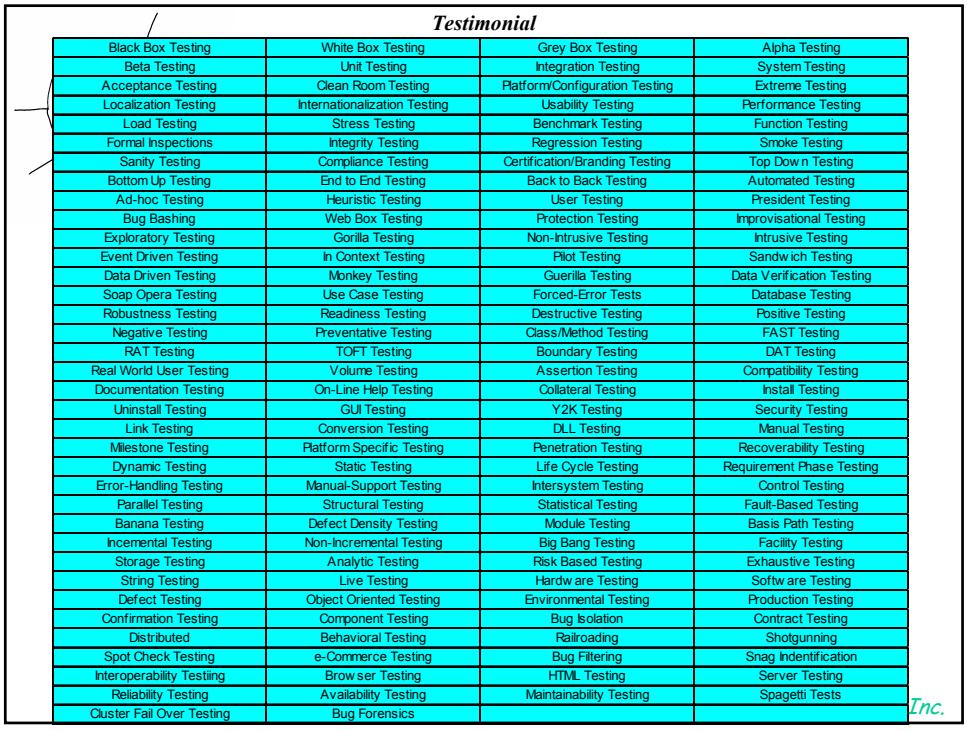
Testing can take place as part of each phase of development .

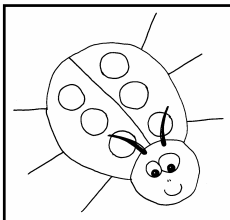
Core Workflow

- Requirements
- Analysis
- Design
- Development
- Testing
- Maintenance



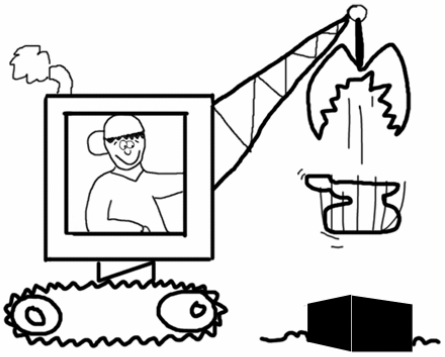

Phase
Rational Unified Process (RUP)
Inception
Elaboration
Construction
Transition
Core Workflow
Requirements
Analysis
Design
Development
Testing
Maintenance
Testing can take place as part of each core workflow involved in development organization.
ourin, 2000
Slide 9
AmiBug.Com, Inc.


Testimonial
Black Box Testing
White Box Testing
Grey Box Testing
Alpha Testing
Beta Testing
Unit Testing
Integration Testing
System Testing
Acceptance Testing
Clean Room Testing
Platform/Configuration Testing
Extreme Testing
Localization Testing
Internationalization Testing
Usability Testing
Performance Testing
Load Testing
Stress Testing
Benchmark Testing
Function Testing
Formal Inspections
Integrity Testing
Regression Testing
Smoke Testing
Sanity Testing
Compliance Testing
Certification/Branding Testing
Top Down Testing
Bottom Up Testing
End to End Testing
Back to Back Testing
Automated Testing
Ad-hoc Testing
Heuristic Testing
User Testing
President Testing
Bug Bashing
Web Box Testing
Protection Testing
Improvisational Testing
Exploratory Testing
Gorilla Testing
Non-Intrusive Testing
Intrusive Testing
Event Driven Testing
In Context Testing
Pilot Testing
Sandwich Testing
Data Driven Testing
Monkey Testing
Guerilla Testing
Data Verification Testing
Soap Opera Testing
Use Case Testing
Forced-Error Tests
Database Testing
Robustness Testing
Readiness Testing
Destructive Testing
Positive Testing
Negative Testing
Preventative Testing
Class/Method Testing
FAST Testing
RAT Testing
TOFT Testing
Boundary Testing
DAT Testing
Real World User Testing
Volume Testing
Assertion Testing
Compatibility Testing
Documentation Testing
On-Line Help Testing
Collateral Testing
Install Testing
Uninstall Testing
GUI Testing
Y2K Testing
Security Testing
Link Testing
Conversion Testing
DLL Testing
Manual Testing
Milestone Testing
Platform Specific Testing
Penetration Testing
Recoverability Testing
Dynamic Testing
Static Testing
Life Cycle Testing
Requirement Phase Testing
Error-Handling Testing
Manual-Support Testing
Intersystem Testing
Control Testing
Parallel Testing
Structural Testing
Statistical Testing
Fault-Based Testing
Banana Testing
Defect Density Testing
Module Testing
Basis Path Testing
Incemental Testing
Non-Incremental Testing
Big Bang Testing
Facility Testing
Storage Testing
Analytic Testing
Risk Based Testing
Exhaustive Testing
String Testing
Live Testing
Hardware Testing
Software Testing
Defect Testing
Object Oriented Testing
Environmental Testing
Production Testing
Confirmation Testing
Component Testing
Bug Isolation
Contract Testing
Distributed
Behavioral Testing
Railroading
Shotgunning
Spot Check Testing
e-Commerce Testing
Bug Filtering
Snag Indentification
Interoperability Testiing
Browser Testing
HTML Testing
Server Testing
Reliability Testing
Availability Testing
Maintainability Testing
Spagetti Tests
Cluster Fail Over Testing
Bug Forensics
Inc.



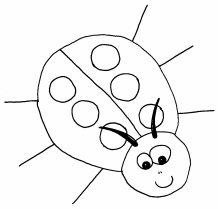
Black Box Testing

- Tester views the program as a black box
- Test is not concerned about the internal behavior and structure of the program
- Test is designed to observe and confirm outcome of program in response to input and system state



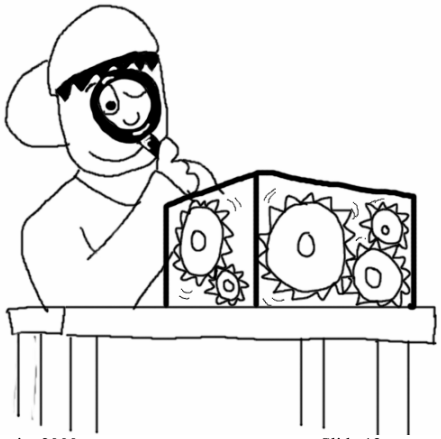
© Robert Sabourin, 2000

AmiBug.Com, Inc.



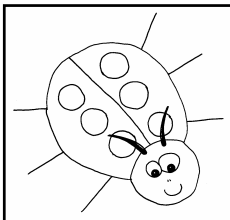
White Box Testing

- Tester reviews the programs behavior, internal structure and data flow
- Test design is based on examination of code
- Used by developers as code is written
- Assumes code required
- Does not confirm requirements are met



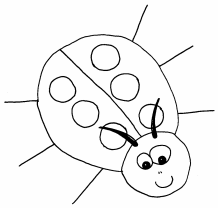
© Robert Sabourin, 2000

AmiBug.Com, Inc.



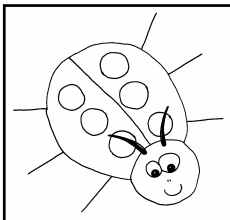
Grey Box Testing

- "Looking Under Hood"
- Internals knowledge is used in test design
  - *Effectiveness* leads to productive new ideas for tests.
  - *Efficiency* allows tester to eliminate redundant tests.



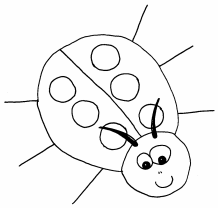
Grey Box Testing

- Examples
  - You might discover that session information is stored on disk, which should make you think about *full disks, lack of permissions, what happens when the session file is deleted, etc*.
  - suppose that two features maintain lists and sort them. If you know that they both use the same sorting library, you don't need to fully test sorting in each feature. You only need to test that each feature uses the sorting library correctly.



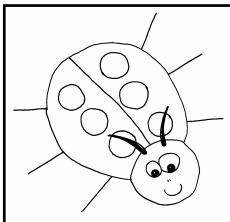
Unit Testing

- Testing of a unit of software as soon as this is available
- Unit is exercised against its detailed design
- Ensuring that developed logic is tested
- Often uses white box methods
- Done by developers at completion of task



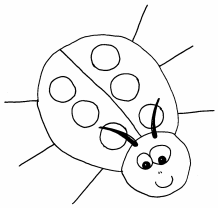
Module Testing

- Myers "The Art of Software Testing" defines Module Testing as:
  - A process of testing individual subprograms, subroutines and procedures in a program
  - Test a program in small blocks as they are built
  - Type of *Unit Testing*



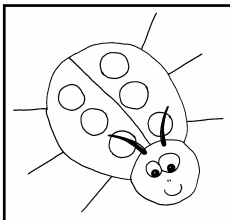
Component Testing

- Organize testing around components or subsystem
  - confirm operation of each component, as they become available, independently of other components
  - test interoperability of different system components



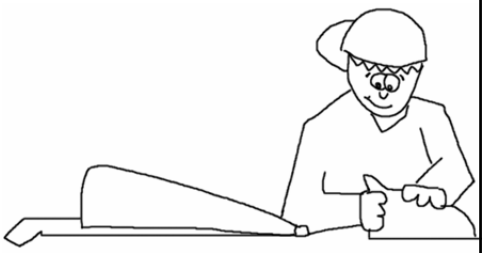
Component Testing

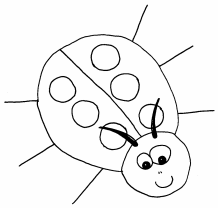
- Organize testing around components or subsystem
  - stubs for missing components
  - test harness in lieu of working application



Integration Testing

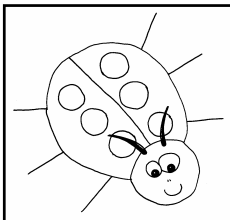
- Testing of combination of two or more units of software
- May involve black or white box methods
- Testing done as soon as integration takes place
- Testing typically done by independent testers working closely with developers





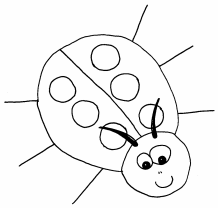
System Testing

- Confirm that the total software system satisfies all of its requirements
- Often mainly black-box methods
- Done when all code and integration is complete
- Simulates target operational environment



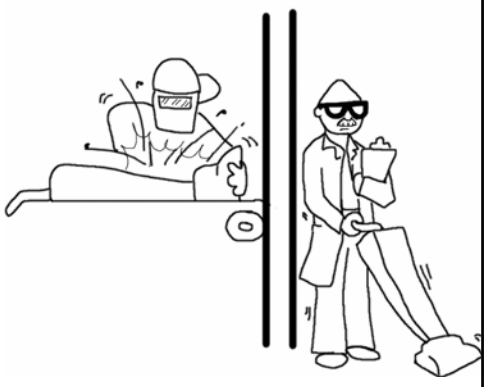
Acceptance Testing

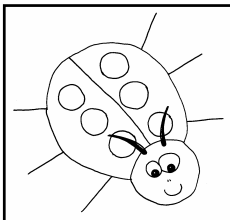
- Testing done by the customer to confirm that the software meets their requirements
- Generally very well defined in a contract
- Generally software must pass this acceptance testing before final payment is made!



Alpha Testing

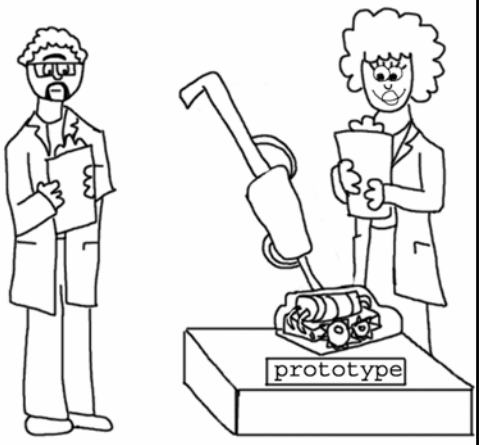
- Testing is performed "in-house"
- After an intermediate project milestone
- A build of the program is delivered to Integration or System Testers
- First testing done by someone other that the software developers





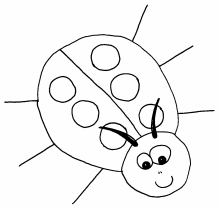
Beta Testing

- Testing is performed by target customers or end users
- All, or a usable subset, of the functionality has been implemented
- Can be done after or in parallel with system testing
- *Danger - Sometimes used for a sales demo*



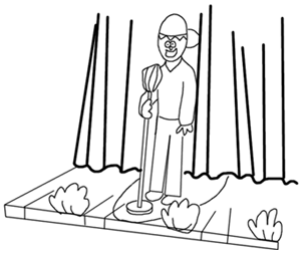
© Robert Sabourin, 2000

AmiBug.Com, Inc.



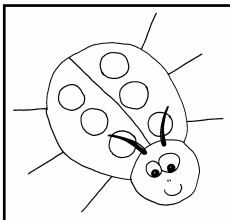
Live Testing

- System being tested is operational
  - used by the customer
  - it has been paid for!
- Testing does not to interfere with the system
- Measure
  - performance
  - resources



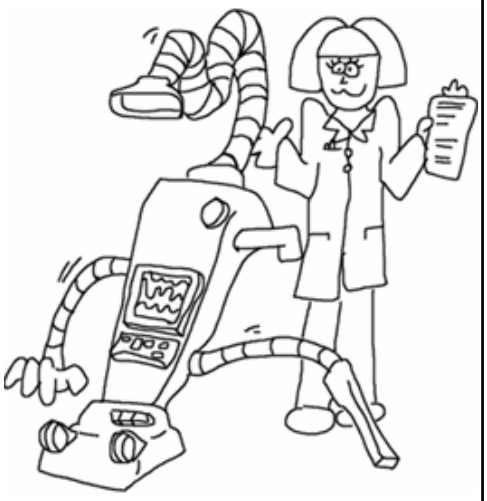
© Robert Sabourin, 2000

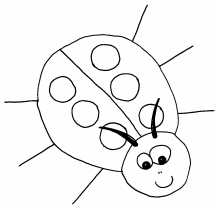
AmiBug.Com, Inc.



Usability Testing

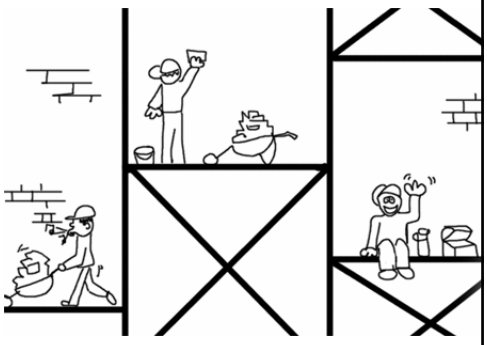
- Testing done to help design effective user interfaces
- Part of software design process
- Generally done by human factors and ergonomic experts
- Checklist approaches used in system testing

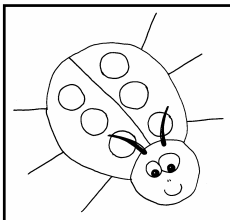




Platform / Configuration Testing

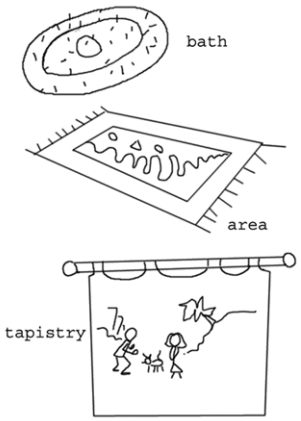
- Ensure functionality operates as required on different hardware and software configurations
- Different versions of operating systems
- Different locales
- Different versions of www browsers, plug-ins
- Different versions of any co-dependent software





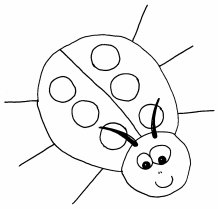
Platform Specific Testing

- Testing included in the test plan related to the specific application target platform!
  - For example tests needed du to using Oracle database to ensure operational functions are compatible with the application, these would be different if SQL server or INTERBASE were being used to implement the DBMS



© Robert Sabourin, 2000

AmiBug.Com, Inc.

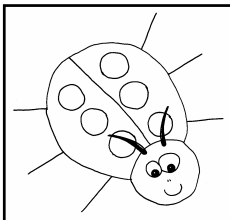


Browser Testing

- Common term in Web Application Development for GUI (Graphical User Interface) testing
  - *Test objects which operate within the browser*
  - *What happens when transactions are interrupted by browser functionality*
    - *backward, forward, refresh, go to URL*

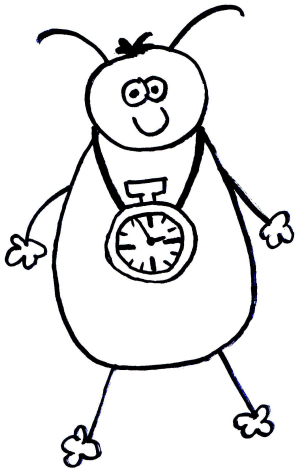
© Robert Sabourin, 2000

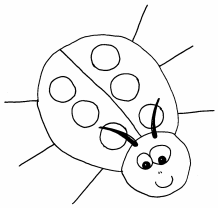
AmiBug.Com, Inc.



Performance Testing

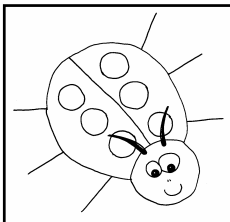
- Measure applications performance, verify
- Response time from input event to outcome
- Throughput or volume (transactions/time)
- Operational characteristics
- Often automated!





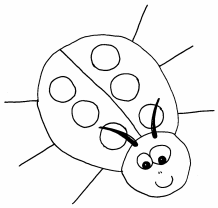
Function Testing

- Verify that application under test functions as intended



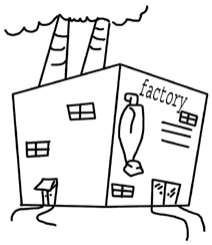
User Testing

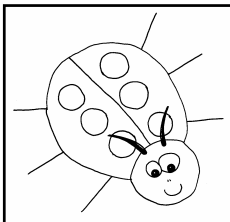
- for each category of user exercise the system with real - or very close to real data using real usage scenarios



Facility Testing

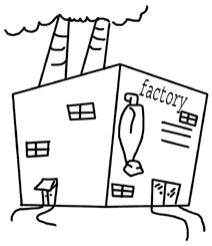
- Often forgotten but very relevant type of testing (especially in Internet Era!)
  - Confirm that each Facility (or function or feature) of the product has been implemented
  - Can be done manually
  - Cross check against objectives of project and requirements (did we skip one?)

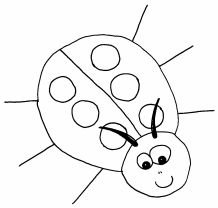




Facility Testing

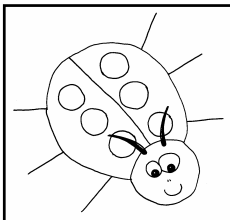
- New versus base system
  - If you find a bug in the web system we check to see if it existed in the original system!





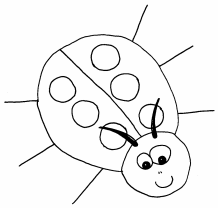
Integrity Testing

- compliance to standards (software, API, operational)
- data integrity



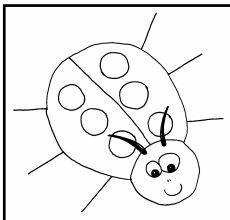
Stress Testing

- Testing operational characteristics of application within a harshly constrained environment
  - Limit processor speed
  - Limit memory
  - Limit disk space
  - Diminish access to shared resources



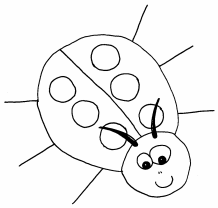
Load Testing

- Vary work loads of the system and study operational characteristics
- How much traffic can the server handle?
- How is performance affected by varying load?
- What about reliability and availability?



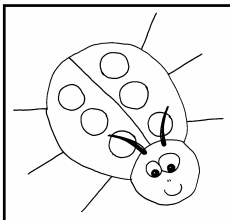
Volume Testing

- Type of load testing in which large amounts of data are processed through the system
- Study behavior of system under test when experiencing extreme processing demands
- Generally automated!



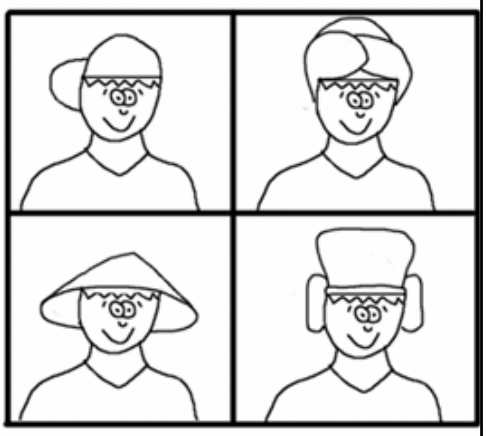
Extreme Testing

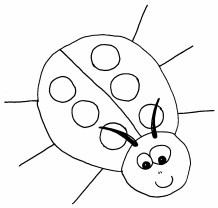
- Testing done as part of an extreme software development process
- Testing is against story board scenarios
- Tightly coupled with development (one iteration at a time)
- Interact with customer of project for acceptance testing



Localization Testing

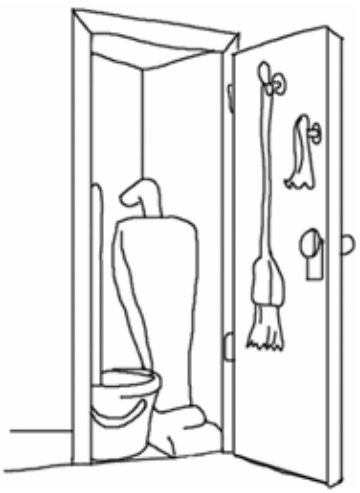
- Testing to ensure localization did not immediate normal operation of the application
- Confirm translations, linguistic locale differences
- Currency, sorting
- Cultural concerns

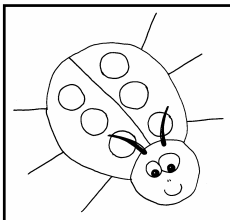




In Context Testing

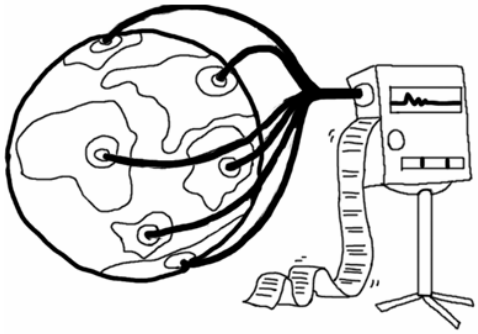
- Part of localization testing
- Testing of translation in real application context
- Make sure localization is correct taken in real context
- Example would be incorrect translation of a word depending on use as verb or noun

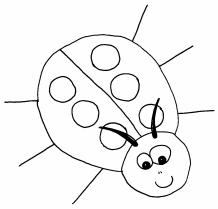




Internationalization Testing

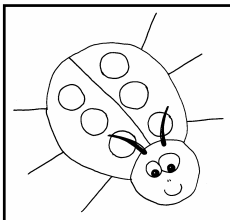
- An internationalized program is able to be localized to operate with many different languages and data representations including those using multi-byte character sets
- Does internationalized software still behave as it did originally?





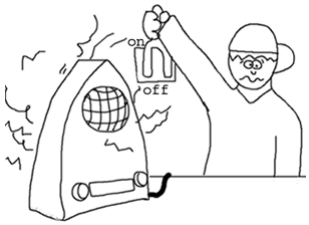
Regression Testing

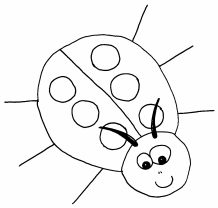
- previously executed tests are re-executed against a new version of the application
  - have code changes broken something that used to work
  - have we introduced new defects
  - typically first part of a testing iteration
  - often automated



Smoke Testing

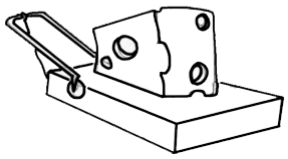
- a smoke test is run on a new build of software to make sure all functions operate well enough to continue testing
- usually run on a build before it is given to testers for integration or system testing
- “turn on a new appliance at the store”

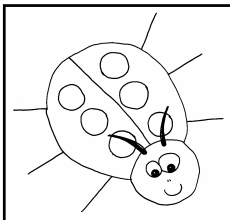




RAT Testing

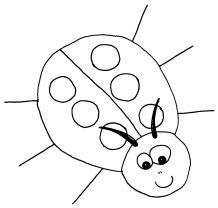
- Release Acceptance Tests
  - Smoke Tests or Build Acceptance Tests
  - Sample of important functions tested with rational data on a sane typical configuration
  - Used to determine if build from development is stable enough to start testing
  - Developers run the test before release





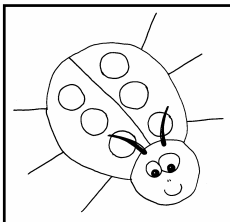
Sanity Testing

- a final test before a release can be shipped
- all normal operations and scenarios are run once with normal valid data
  - install on typical platform
  - run basic data
  - is the release sane!
  - are all components there? do they work?



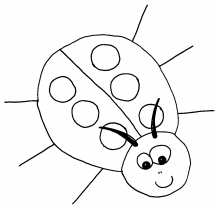
FAST Testing

- Functional Acceptance Simple Tests
  - Wide in breadth, low in depth
  - Exercise every low level function of the application at least once, no combinations with other functions
  - Do all controls exist, are default states correct, tab order, shortcuts, accelerator keys, links, images



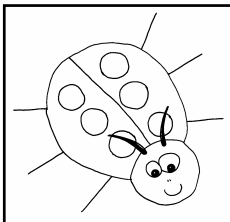
FAST Testing

- Functional Acceptance Simple Tests
  - Do low level functions work well enough on a normal configuration to allow for task oriented functional testing?



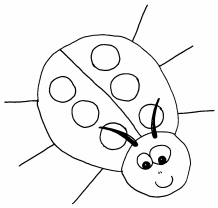
TOFT Testing

- Task Oriented Functional Testing
  - Can the application do useful tasks correctly?
  - Structured around product features
  - Detailed testing against specification and reasonable user expectations
    - at least one test case per feature or function!



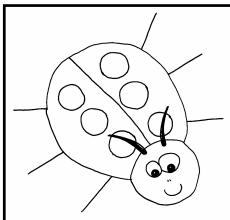
Readiness Testing

- Readiness testing is similar to smoke testing
  - Is the build ready to be processed or used at the next process step?



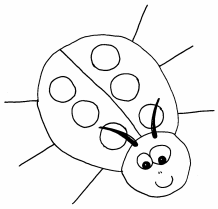
Compliance Testing

- do we conform to an industry, national or international standard?
- do we use a standard API? (Posix compliant, Win32, MFC)



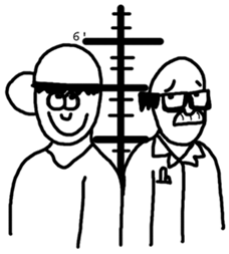
End to End Testing

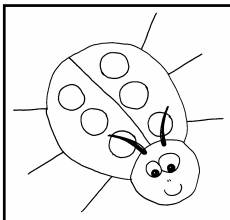
- the entire hardware/software chain involved in the execution of the function is available
- all components, elements, processes are used and a transaction goes through the entire system



Back to Back Testing

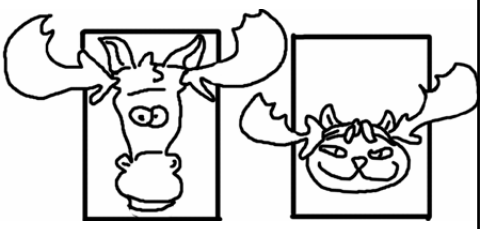
- running the same test on similar implementations or versions and comparing the results.

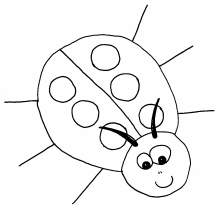




Benchmark Testing

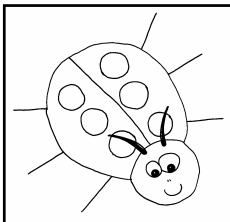
- Compare performance of system to a reference target
- Various indices such as Norton SI
- Compare against reference source for example comparing HP printers with HP compatible printers





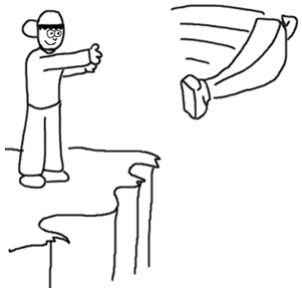
Certification/Branding Testing

- third party testing done to confirm that the application conforms to criteria for certification or branding program
  - Microsoft Windows certification



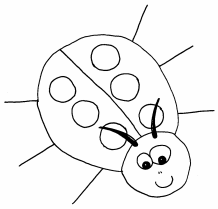
Top Down Testing

- White box method starting with main program and working down through the software.
- Stubs must be created for units not yet completed.



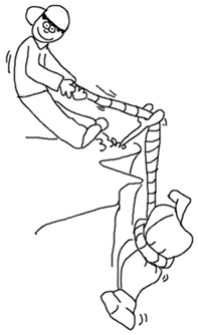
© Robert Sabourin, 2000

AmiBug.Com, Inc.



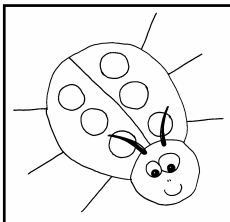
Bottom Up Testing

- White box method starting with lower level units.
- Driver units must be created for units not yet completed, each time a new higher level unit is added to those already tested.



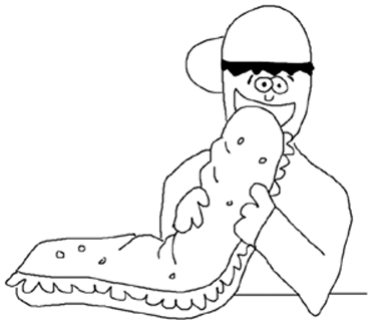
© Robert Sabourin, 2000

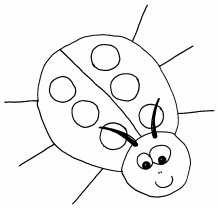
AmiBug.Com, Inc.



Sandwich Testing

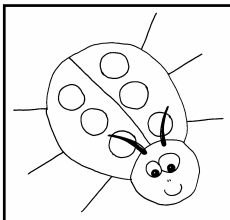
- Combining Bottom Up and Top Down approaches.
- Blend of stubs and drivers depending on part of software being exercised.





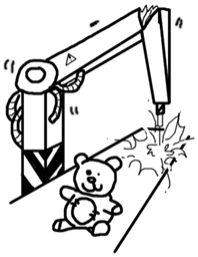
Manual Testing

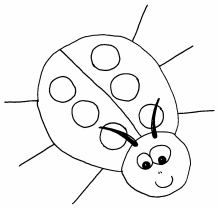
- A test or series of test cases which are executed manually
- These tests involve an operator following a pre-established test procedure and using predetermined test data
  - Tedious
  - Necessary



Automated Testing

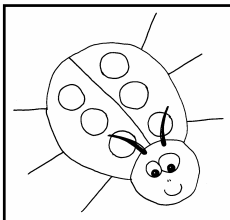
- a test or series of test cases which can be executed automatically
- useful to repeat tests especially complex sequences of instructions
- useful for API testing
- regression with periodic builds





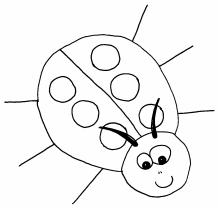
Protection Testing

- Testing to find, or rule out, the presence of faults which could result in corruption, denial of services, unauthorized access or other related side effects.
  - Information protection
  - Protection against attackers
  - Search for back doors
  - Break in



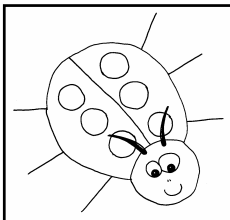
Improvisational Testing

- Type of exploratory testing
- Similar to improv music
  - jazz
- No notes but themes which can be combined to form great music



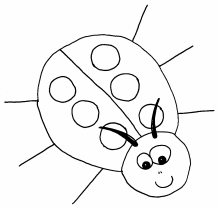
Clean Room Testing

- Testing done as part of a clean room software engineering process
- High reliability
- Statistical analysis to determine test cases by sampling set of possible input data and conditions
- Analytic technique



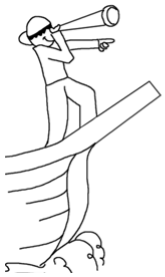
Analytic Testing

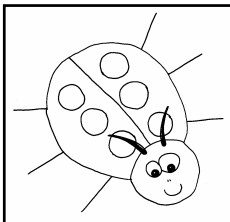
- Test planning based on a detailed analysis of all application functions and operations
- Test cases derived from a detailed analysis of the technical specification
- Can include white box approaches
- *Your testing is based on analysis!*



Exploratory Testing

- Structured way to test
- Concurrent
  - testing
  - test design
  - test planning

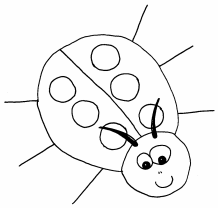




Exploratory Testing

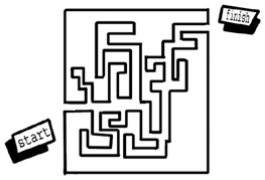
- *"In operational terms, exploratory testing is an interactive process of concurrent product exploration, test design and test execution."*

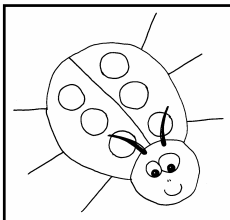
  - James Bach



Ad-hoc Testing

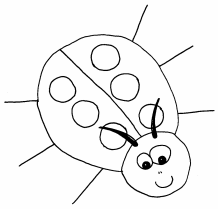
- trying rational test cases in a random but not unreasonable manner!





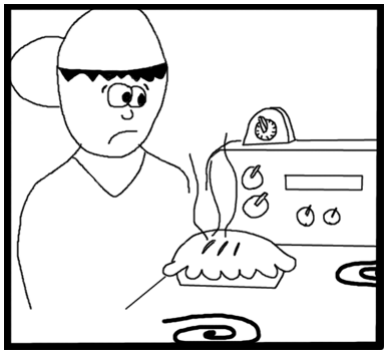
Heuristic Testing

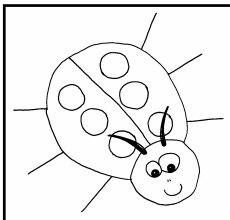
- testing using heuristic reasoning and taking advantage of the technical insight, critical thinking and experience of the tester
- heuristic is defined as:
  - of or relating to exploratory problem solving techniques that use self-education
  - adapt to what we learn!



Event Driven Testing

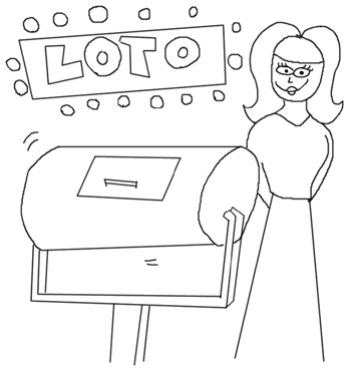
- A type of automated testing
- Test script execution is triggered by an external event or interrupt
- Run test A when event B occurs
  - *Web Site Monitors*
  - *Run consistency check if DBMS is 80% full*





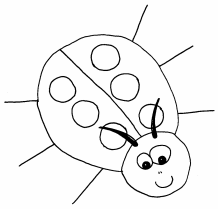
Data Driven Testing

- Test script or procedure in which *test data* is separated from *test operations or actions*
  - parameterized data
  - one set of data per test case
  - run the same test script with different data!
  - data is deliberate, not arbitrary
  - testing is repeatable
  - test can be automated



© Robert Sabourin, 2000

AmiBug.Com, Inc.

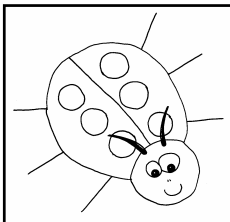


Capture Playback Testing

- *Tool* is used to record all input events of a simulated user
- Test case is automated by replaying all input events
- Difficult to maintain!
- Can be used as a first step in designing a data driven test automation script!

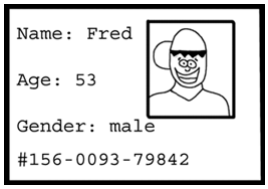
© Robert Sabourin, 2000

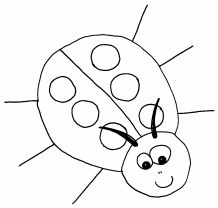
AmiBug.Com, Inc.



Data Verification Testing

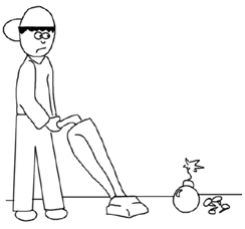
- *Testing data storage and integrity in applications which store and manipulate data.*
  - is data valid or legal and accurate
  - is data of the correct type
  - is data from the correct record
  - self verifying data concepts

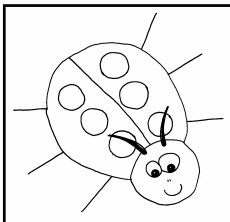




Forced-Error Tests

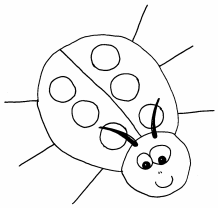
- FETS
- Intentionally drive software into each possible error condition
  - Is error detected?
  - Is error handled?
  - Does system recover gracefully?
  - Is error condition communicated?
  - Any other problems encountered?





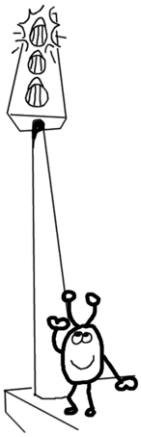
Database Testing

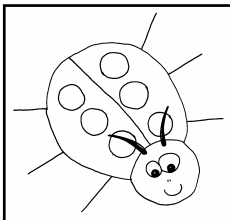
- Identify database related errors
- Errors in:
  - database servers
  - data warehouses
  - data marts
  - find bugs in SQL statements



Positive Testing

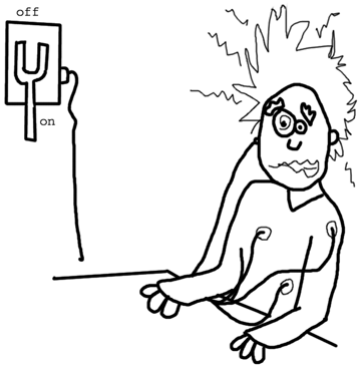
- Does the application perform what it is expected to do given known input and operating state?
  - The test result should be “A”

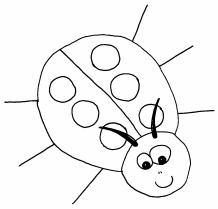




Negative Testing

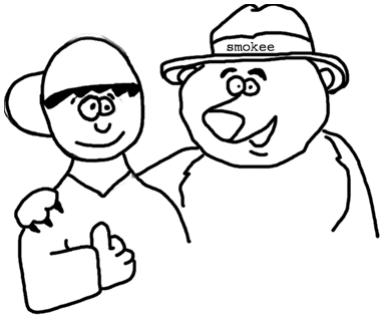
- Does the application not perform what it is not expected to do given known input and operating state?
  - The test result should not be “A”
  - we are testing to ensure it is not “A” we do not care if it is “B” or “C”!

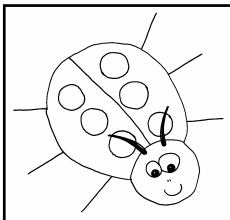




Preventative Testing

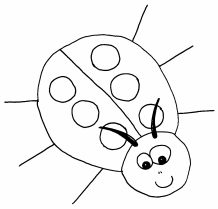
- Testing used to avoid introducing defects in software
- We use preventative testing before we code!
  - Formal Inspections
  - Design Reviews
  - Walkthroughs
  - Peer Code Reviews





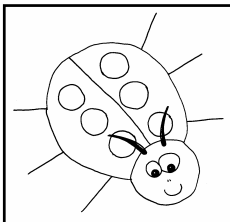
Formal Inspections

- Structured method to efficiently identify defects in any deliverable or artifact of the software development process
- Artifact is review by a team and defects identified are logged
- Tom Gilb - champions technique



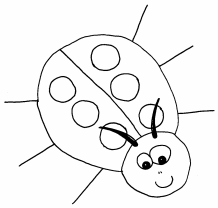
Object Oriented Testing

- Testing Object Orient Software at the object level
- Independent testing of each object
- For each object
  - methods testing
  - class testing
- Combinations of objects interoperating



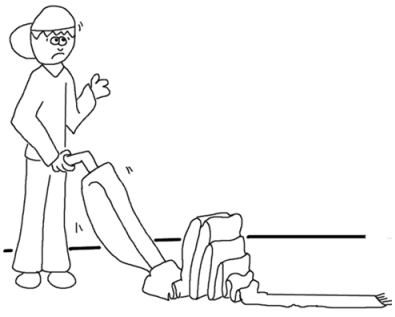
Class/Method Testing

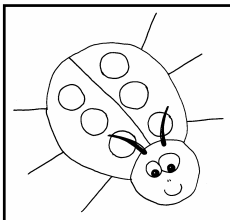
- OOP component testing
  - Testing classes of an object
  - Testing methods of a class
  - Independent of other objects in the system
  - White box method



Boundary Testing

- Test with extreme input values
  - Lower and upper boundaries
  - Any edge conditions
  - Above and below extreme values
- Test to generate extreme output values
  - May or may not require extreme input values
  - Zero divide, overflow
- Record Sizes



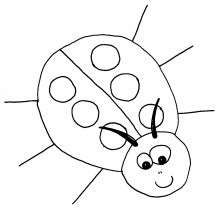


DAT Testing

- Deployment Acceptance Testing
  - Typical testing of web application on fully installed target hardware and software or on a staging site which is equivalent to the actual target or customer site.
  - Functional test suite is run to ensure operation is OK before we go live

© Robert Sabourin, 2000

AmiBug.Com, Inc.

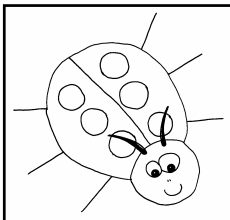


Compatibility Testing

- Similar to platform of configuration testing
  - check that an application functions correctly on various hardware and software environments
  - software
    - OS, Browser, DBMS, Network Software, Concurrent apps
  - hardware
    - CPU, Disk, Video Cards, Memory, Printers

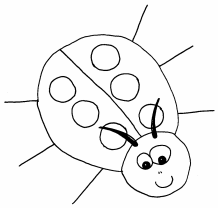
© Robert Sabourin, 2000

AmiBug.Com, Inc.



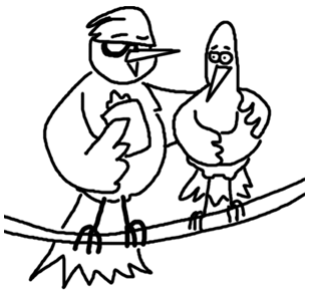
Documentation Testing

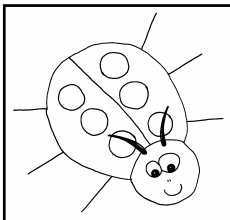
- Test software against user and reference documentation
  - factual and accurate
  - screen images are correct
  - examples work
  - marketing collateral is correct



On-Line Help Testing

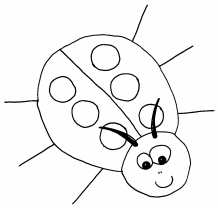
- Test functionality of Help System
  - is help factual and accurate
  - are we launching to correct page
  - does indexing work
  - are links correct
  - are tool tips correct
  - are images correct





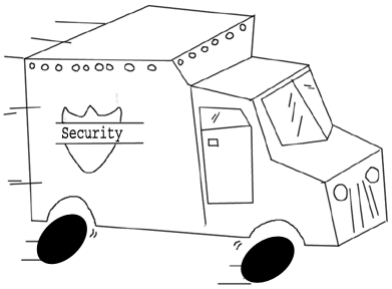
GUI Testing

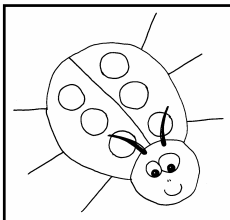
- Graphical User Interface Testing
  - How does GUI operate against specification
    - Navigation
      - Menus, Dialogues, Forms, Tables
    - Images
    - Conformance to Style guide - Look and Feel
    - Consistent to environment
    - Usability checking!



Security Testing

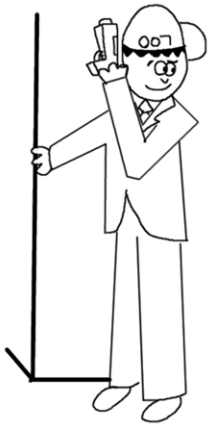
- Similar to protection testing
- Concern about the unauthorized access to all or part of the system, are security policies and requirements implemented
- Protect against internal and external threats
  - Servers, Databases, Clients





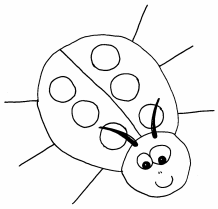
Penetration Testing

- Evaluate effectiveness of network defenses
  - External expertise
  - Conducted before system is live
  - Continue on live system
  - Work from outside perimeter
  - Simulate work of hackers



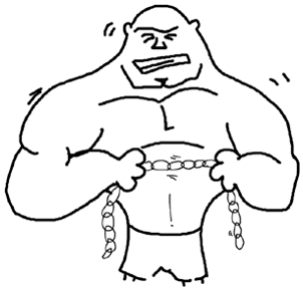
© Robert Sabourin, 2000

AmiBug.Com, Inc.



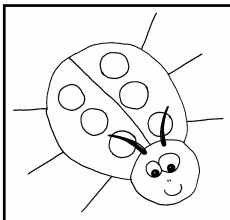
Link Testing

- Test all links between all pages of a web site
  - Point to the correct page?
  - Is page accessible?
  - Are references relative or absolute?
- Several automated link testing tools



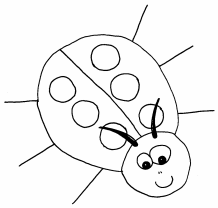
© Robert Sabourin, 2000

AmiBug.Com, Inc.



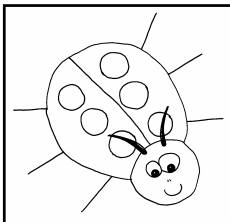
Conversion Testing

- Data conversion occurs when you upgrade from one version of an application to a subsequent version
  - Is all user data correctly converted to the new format?
  - Did we loose or any records? Any fields?
  - Are new fields initialized to correct values?



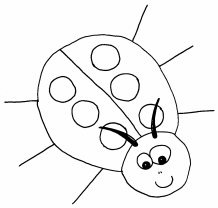
DLL Testing

- Dynamic Link Library Testing
  - Windows applications often use system DLLs.
  - Other applications can install different versions of DLLs for which the test application may be incompatible
  - Microsoft Dependency Walker (free!)
    - provide a list of DLLs



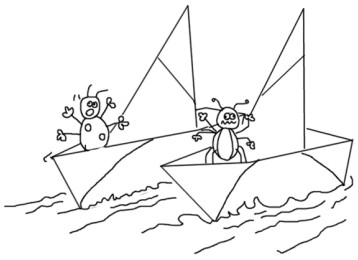
DLL Testing

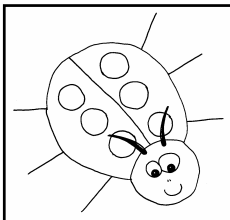
- Dynamic Link Library Testing
  - Are third party DLLs the correct versions?
  - Are OS DLLs the correct versions?
  - What happens if a third party system overwrites a required DLL?



Dynamic Testing

- Testing running software
  - System is operating during testing
    - Code is compiled, linked, build, installed
    - Binary image of code is executing - *running*!
  - Measure operational performance
    - Functionality
    - Impact on environment



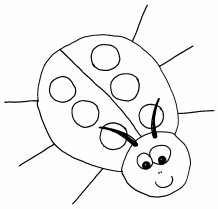


Static Testing

- Testing source code of software
  - Reviews
  - Walkthroughs
  - Inspections
  - Static analysis (automatic or manual)
  - Study code and development artifacts in order to gauge correctness and identify defects

© Robert Sabourin, 2000

AmiBug.Com, Inc.

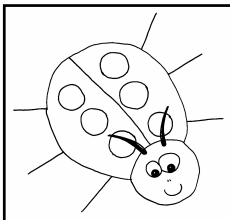


Error-Handling Testing

- Determine the ability of the software being tested to process incorrect transactions
  - Are error conditions recognized by the system?
  - How does program respond to unexpected conditions?
  - What about errors subsequently corrected by end users?

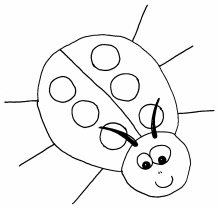
© Robert Sabourin, 2000

AmiBug.Com, Inc.



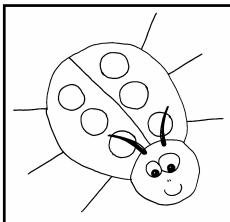
Intersystem Testing

- Test to confirm two systems or applications communicate together correctly
  - What is one is down?
  - Proper parameters passed?
  - Timing and synchronization
  - Error handling



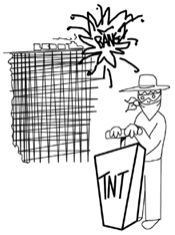
Control Testing

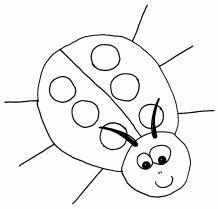
- Part of system testing to ensure that controls on application are correct:
  - accurate and complete data
  - authorized transactions
  - audit trail
  - integrity of processing
- Accounting systems as an example!
  - authentication, electronic authorization



Big Bang Testing

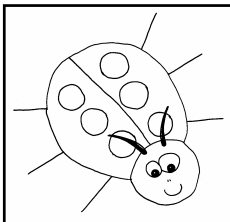
- Synonymous to *Non-incremental Testing*
  - Do not do Integration Testing until the entire system has been build
  - Attack all at once
    - Risky
    - No leverage!





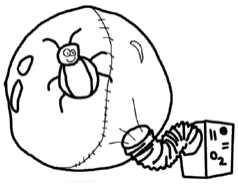
Production Testing

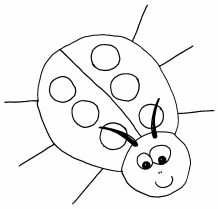
- Testing as part of the production process
  - Was an item manufactured correctly
  - Often called quality control
- *Confirming that a production copy of any shrink-wrap software being sold is an exact image of the gold master!*



Bug Isolation

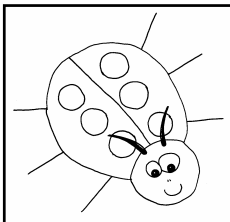
- Critical testing activity
  - Determines minimal/ consistent way to reproduce a bug
  - Facilitates debugging
  - Provides valuable input to developers





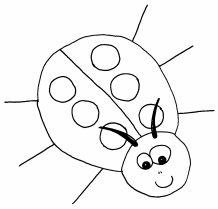
Bug Isolation

- Eliminate ambiguity
  - Isolate sub-system
  - Hardware problem
  - Software problem
  - Internal factors
  - External factors
  - Is it really a bug?



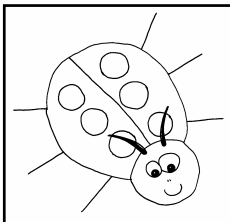
Confirmation Testing

- Typically:
  - Tester finds the bug
  - Product/Development leads prioritize the bug
  - Developer fixes the bug
  - Tester confirms that the fixed bug is really fixed in the appropriate software build



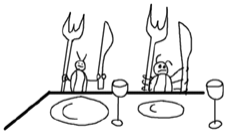
Confirmation Testing

- For each build
  - testers confirm that all bugs which were supposed to have been fixed, ***as indicated by developers***, are actually fixed
  - after the build has passed a smoke test
  - before in depth regression testing



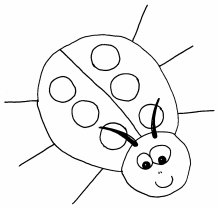
Behavioral Testing

- Testing based on what a system is supposed to do
- Tests based on how a system functions
- Tests based on how a system is supposed to be used
- Black box testing



© Robert Sabourin, 2000

AmiBug.Com, Inc.

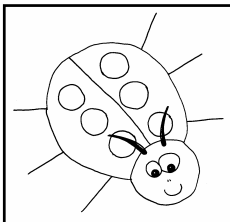


Interoperability Testing

- Ensure that software under test interacts properly with target platforms
  - operating system
  - equipment
  - applications
- Same as platform or configuration testing

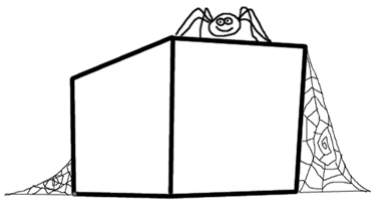
© Robert Sabourin, 2000

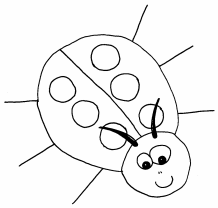
AmiBug.Com, Inc.



Web Box Testing

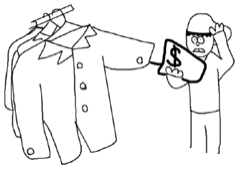
- Commercial play on the word
  - White Box Testing
- Testing using web based automated workflow management

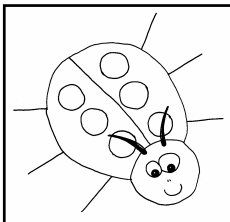




HTML Testing

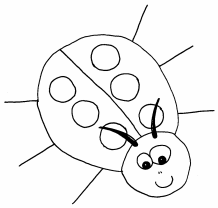
- Common term in Web Application Development for static analysis of Web page HTML source
  - *Is HTML constructed properly?*
  - *Are any tags missing?*
  - *Are links correct?*
  - *Is Syntax OK? Is it standard compliant?*





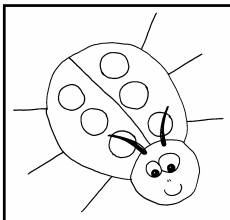
Server Testing

- Common term in Web Application Development for testing software objects which run on the server
  - CGI components
  - Business tier
  - Data tier
  - Active Server Pages



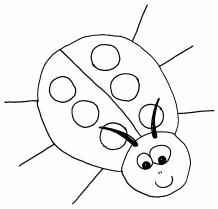
Presidents Testing

- believe it or not, in many companies the boss or some senior executive takes pride in running software on their desk just before ship date to see if they can break it!
- Some SQA teams clone the Presidents PC and know his favorite commands and operations - this is run as a special test!



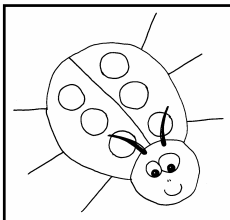
Bug Bashing

- Testing done in a blitz
- A lot of testing in a very short period of time
- Sometimes used to test web sites before going live "everyone on team bashes the system"



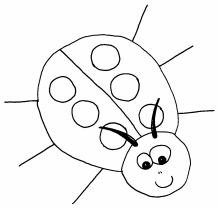
Gorilla Testing

- Unstructured way to test
- Try everything you can to crash or break the application



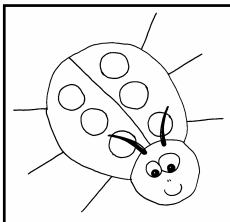
Monkey Testing

- *"… six monkeys pounding on six typewriters at random for a million years will recreate all the works of Isaac Asimov…"* Noel Nyman, Microsoft
  - random data entry
  - automated or manual
  - dumb monkeys, monkeys with savvy



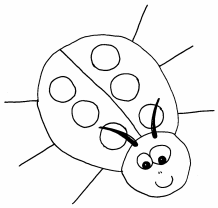
Guerrilla Testing

- Wage a Guerrilla attack on the software being tested, observe behavior
  - Do evil things
  - Damage things
  - Intentionally corrupt data



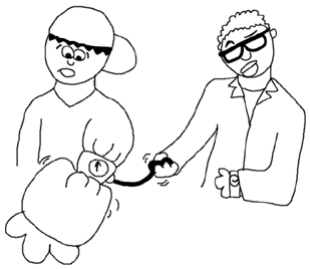
Non-Intrusive Testing

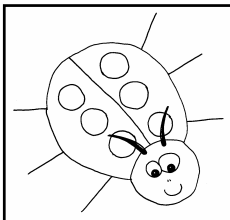
- System testing
- Black Box
- Does not interfere with system under test
- Leaves system in same state after test



Intrusive Testing

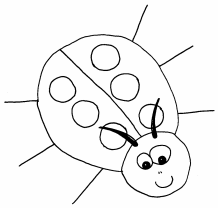
- System testing
- Interferes with system under test to allow for probing for test data
- Interferes with system under test to simulate fault
- Leaves system in different state after test





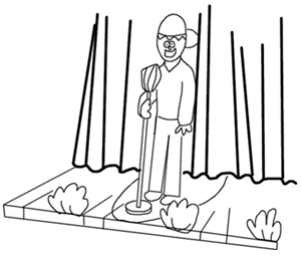
Pilot Testing

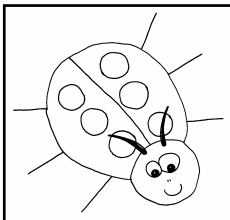
- Similar to Beta testing
- Usually a first project between two companies (Pilot project)
- If Pilot Testing passes then larger scale deployment will take place, otherwise the business deal will be reworked!



Live Testing

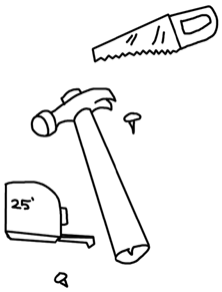
- System being tested is operational
  - used by the customer
  - it has been paid for!
- Testing does not to interfere with the system
- Measure
  - performance
  - resources





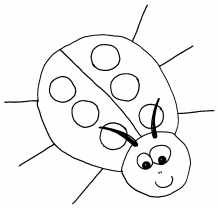
Hardware Testing

- Complete solution is developed
  - confirm hardware delivered meets requirement
  - confirm hardware is compatible with software
- In embedded systems
  - test hardware under development
  - synchronize development of hardware/electronics with embedded firmware



© Robert Sabourin, 2000

AmiBug.Com, Inc.

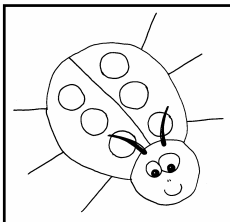


Software Testing

- In systems development
  - Testing software deliverables of a system
- In Software Quality Assurance
  - Work associated with verifying that software conforms to requirements
  - Checking to see if software actual does what someone expects it to do!

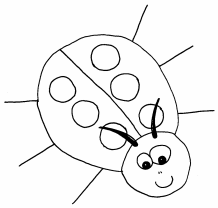
© Robert Sabourin, 2000

AmiBug.Com, Inc.



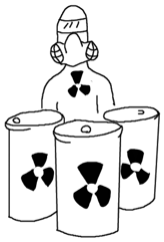
Defect Testing

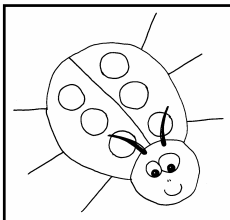
- Testing with the express purpose of identifying defects in the software systems
  - As opposed to testing to confirm that a requirement is met!
  - *Defects are the root cause of bugs!*



Environmental Testing

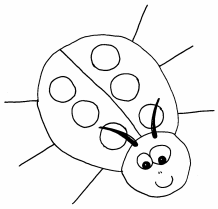
- Testing to ensure system operates in the target environment
  - vary environmental characteristics
    - humidity
    - pressure
    - temperature (oven)
    - wind
    - shock and vibration





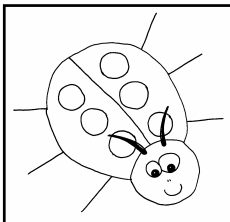
Contract Testing

- Use of third party contract testing organization
  - Must have reasonable specification for subsystem being tested
  - Excellent for *heavy metal* stress testing
  - Excellent for *highly specialized expertise*
  - Communication issues



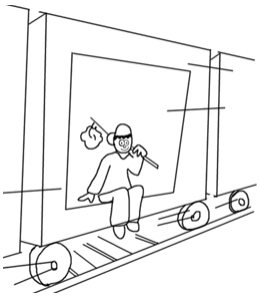
Distributed Testing

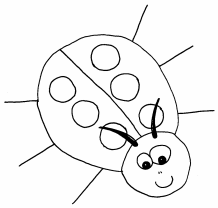
- Testing staff is split into physically distributed teams
  - Center of excellence
  - Localization in-context testing
  - Merger between different companies
  - Pure Black Box
  - Outsourcing
  - Overnight across world parallelism



Railroading

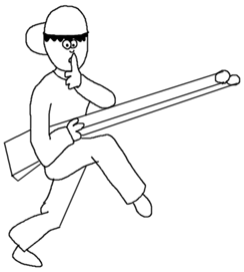
- Testing continues in the original defined sequence of a test suite when a new build arrives
- Testing a new build starts exactly where testing the previous build left off
- Goal is to achieve acceptable levels of test coverage

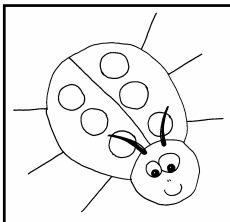




Shotgunning

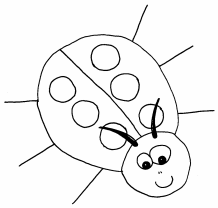
- Distribute test suites randomly across test cycles
- Distribute test configurations randomly across test cycles
- Goal is to achieve acceptable level of coverage





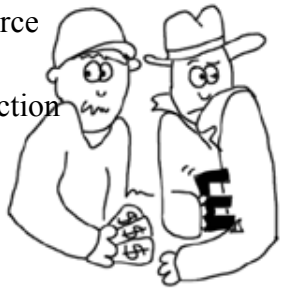
Spot Check Testing

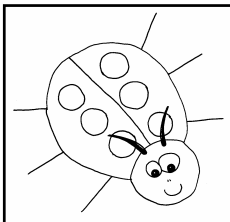
- Spot Checking
  - Test a random sample of functions of an application
  - Sometimes used as a sanity test where random functions as selected and tested with sane data.
  - Sample randomly from a population to make a probabilistic statements about the population.



e-Commerce Testing

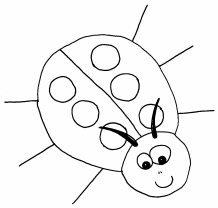
- Buzz word of the decade!
  - Software Testing activities associate with the development of a web based e-Commerce system.
  - Popular title used for end to end transaction testing of an e-Commerce system.





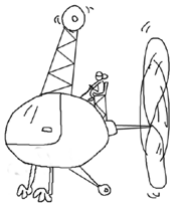
Bug Filtering Testing

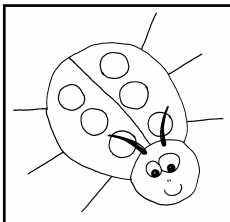
- Testing with a prior knowledge of which Prioritization decision will have been made about several broad classes of bugs
  - For example *spelling mistakes or typos* which do not impact meaning, and are not any user selectable dialogue window or control will be assigned a low priority.



Snag Identification

- A *"snag"* is a type of *"bug"* typically captured by a companies "IT" or "MIS" department
- *CAE* of Montreal, identify snags in IT systems, logged, prioritized and corrected them
- Results of testing an IT system are *"snags"*



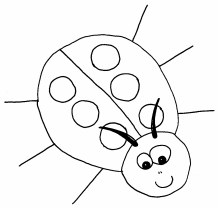


Reliability Testing

- Testing to determine the reliability of software
  - Statistical samples
  - Accelerated life testing
  - Project MTBF
    - Mean Time Between Failures

© Robert Sabourin, 2000

AmiBug.Com, Inc.

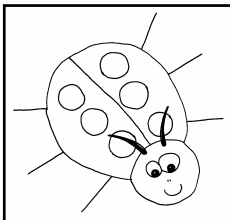


Availability Testing

- Testing to determine the availability of software
  - Can users connect?
  - Does application respond to input?
  - How many failed attempts to load a page occur as the system is loaded?
  - Can the system run 24/7?

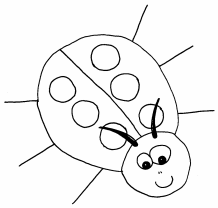
© Robert Sabourin, 2000

AmiBug.Com, Inc.



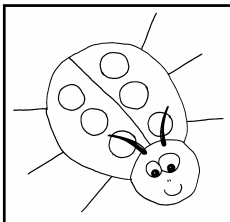
Maintainability Testing

- Testing to determine whether the software can be maintained after commercial deployment
  - Can field installations be upgraded?
  - What it the associated PITA factor?
  - Can code be modified by developers not familiar with code base?
  - Inspections/Reviews may be needed



Spaghetti Tests

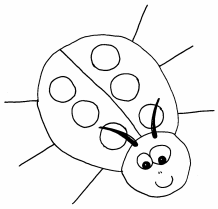
- Term describing a disorganized collection of automated test scripts.
  - Similar to Spaghetti Code
  - Tests are hard to maintain
  - Get more complex when modified
  - Hard to understand
  - Lack design



Cluster Fail Over Testing

- Multiple server systems
  - Ensure that if, for any reason, one node of a system fails that work is distributed to other nodes
  - Common in horizontally scalable Web or e-Commerce applications

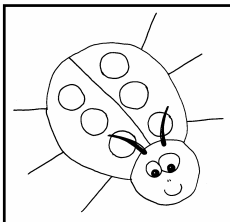
© Robert Sabourin, 2000



Bug Forensics

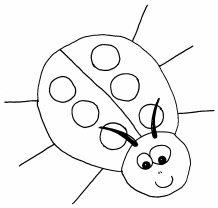
- When a computer system abnormally stops or unexpectedly fails, Bug Forensics are the actions taken to identify the exact time, place and cause of system death
- *Quincy TV metaphor*

© Robert Sabourin, 2000



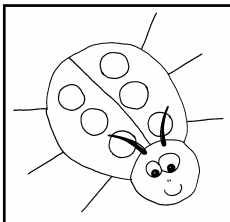
Banana Testing

- Fruit product distribution analogy.
- Software ripens at the customer site!



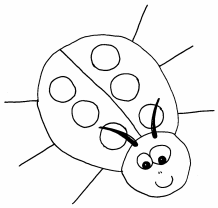
Basis Path Testing

- Method introduced by McCabe
  - Create a flow graph
  - Identify all unique executable paths through the code being tested (called the Basis Set)
  - Derive one test case for each path in the Basis Set
  - *White Box method*



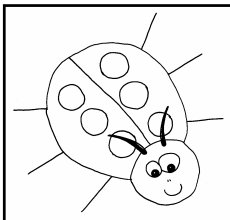
Incremental Testing

- Test modules as they are integrated into a system
  - Form of *Integration Testing*



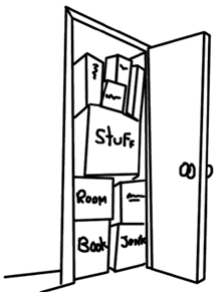
Non-incremental Testing

- Test modules individually and then when they are all integrated together
  - *Unit Testing* is done on each Module
  - *System Testing* is done combining all Modules
  - No *Integration Testing* is done



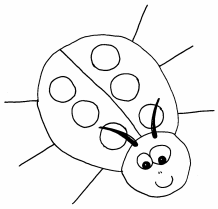
Storage Testing

- Disk and memory resource consumption
- Operation with insufficient resources
- What if available memory diminishes due to consumption by other processes?
- Memory leaks
- Garbage collection?



© Robert Sabourin, 2000

AmiBug.Com, Inc.

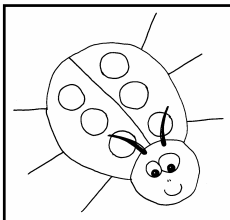


Risk Based Testing

- Testing based on Risk Analysis
  - Spread resources across various testing objectives based on a function of commercial (business or market) and technical risk
  - More effort will be spent on areas of higher risk
  - Order of testing is based on risk
    - test higher risk areas first

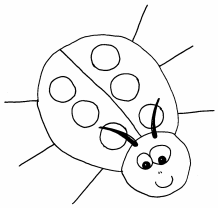
© Robert Sabourin, 2000

AmiBug.Com, Inc.



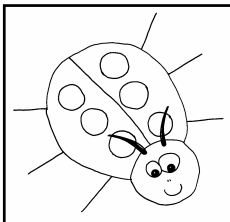
Exhaustive Testing

- Test every possible input set
  - For any function identify, enumerate and list every possible input data set.
  - Try every single case!
- Test every possible outcome
  - For every possible outcome test every possible input data set which is expected to generate it!



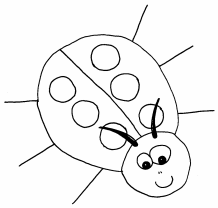
Exhaustive Testing

- Automated testing for large sets
- Examples
  - Keyboard testing
    - what does every keystroke, shift state generate
      - for all keyboards
      - for all languages
    - can each character be generated
    - *Tango over 500,000 test cases*



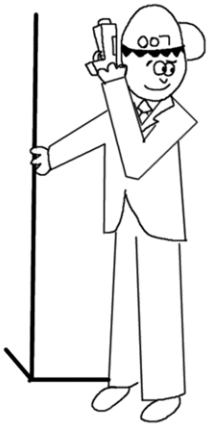
String Testing

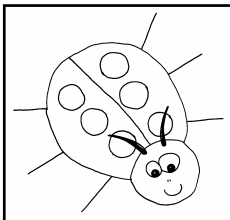
- Focus testing on problems in typical user scenarios
  - Test a "string" of operations
  - Example
    - create, print and save a document
  - Reference
    - *"Managing the Testing Process" Rex Black*



Penetration Testing

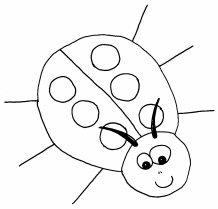
- Evaluate effectiveness of network defenses
  - External expertise
  - Conducted before system is live
  - Continue on live system
  - Work from outside perimeter
  - Simulate work of hackers





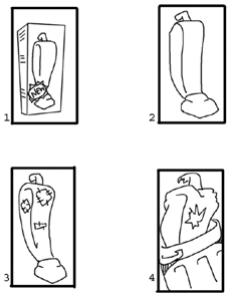
Recoverability Testing

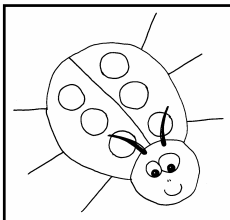
- If the system fails can it recover?
  - Does user loose data from active session at time of failure?
  - Does server restart?
    - JSERVE, HTTP, DBMS
  - Which type of failures do not recover? Should they?



Life Cycle Testing

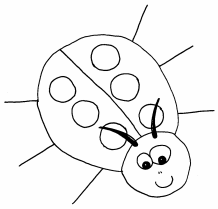
- Continuous testing of the system or software through the entire development process
  - at predetermined milestones results of development process are inspected
  - identify defects early
  - required well defined process





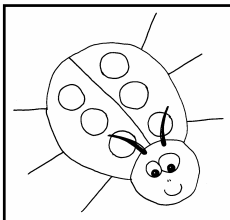
Requirement Phase Testing

- Early in development process, confirm requirements are correctly collected and are accurately articulated in a manner consistent with the customers needs
- Confirm requirements conform to internal standards
- Confirm requirements are testable



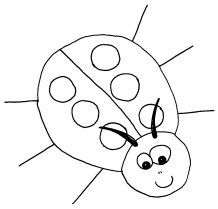
Error-Handling Testing

- Determine the ability of the software being tested to process incorrect transactions
  - Are error conditions recognized by the system?
  - How does program respond to unexpected conditions?
  - What about errors subsequently corrected by end users?



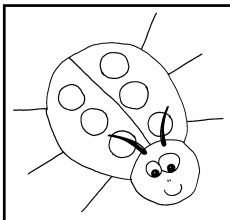
Manual-Support Testing

- Manual operations complement automated operations in a system
  - Are all manual procedures documented correctly? For the right person?
  - Can manual procedures work when software is in an unexpected state?
  - How does software react to manual procedures run at the wrong time?



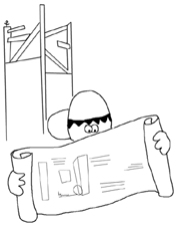
Parallel Testing

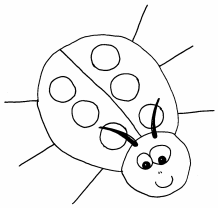
- Similar to "Back to Back" Testing
  - Two versions of application process same data or respond to same input
  - Example - Printer compatibility testing
    - Confirm same results based on same input
  - Can apply to all or part of an application
    - Compatible PCL but not Imaging



Structural Testing

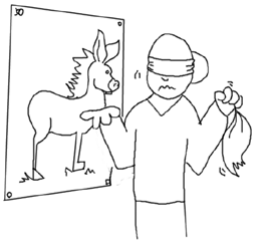
- White Box technique
- Structure of code used to determine test data and testing techniques
  - Complexity
  - Data Flow
  - Execution, Program Flow
    - statement, branch, conditional, expression, path

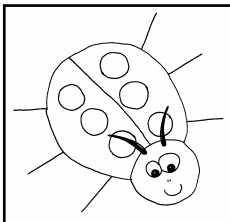




Statistical Testing

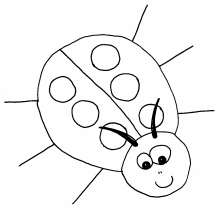
- Determine operational reliability of a system
  - How do faults effect the failure rate?
  - Statistical models are used to generate test data
  - Estimate failure rates





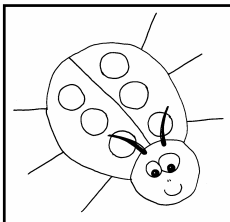
Fault-Based Testing

- Demonstrate that certain types of faults are not in the program.
- Demonstrate that certain types of tests find faults which are injected into a system.



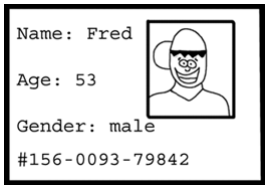
Defect Density Testing

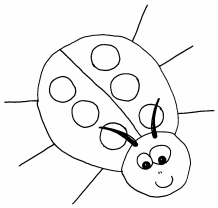
- Testing use to estimate the number of defects remaining in software!
  - Density is number of defects per unit of code
  - Defect seeding methods can be used
    - seed 100 (intentionally insert bugs!)
    - if you find 10 seeded and 20 non-seeded then you can estimate the number of remaining defects to be about 200



Data Verification Testing

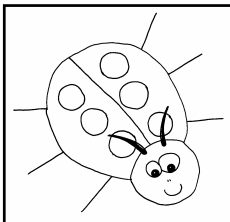
- Testing data storage and integrity in applications which store and manipulate data.
  - is data valid or legal and accurate
  - is data of the correct type
  - is data from the correct record
  - self verifying data concepts





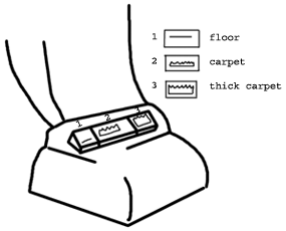
Soap Opera Testing

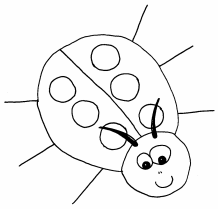
- Used when testing applications which calculate pension or insurance benefits, or income tax
  - fictitious events regarding relationships and circumstances of individuals
    - marriage, remarriage, birth, death, divorce, sex change, hire, fire, quit, rehire, leaves
  - sequencing contrived but possible



Use Case Testing

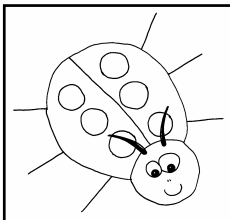
- Test procedure is modeled after the Use Cases which were used to specify and design the system
- Part of Rational Unified Process (RUP)





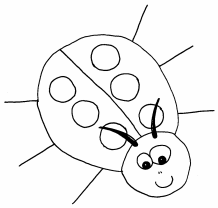
Database Testing

- Identify database related errors
- Errors in:
  - database servers
  - data warehouses
  - data marts
  - find bugs in SQL statements



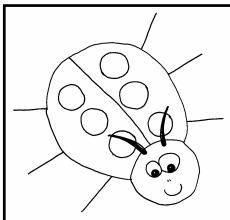
Robustness Testing

- Robustness testing attempts to measure the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.



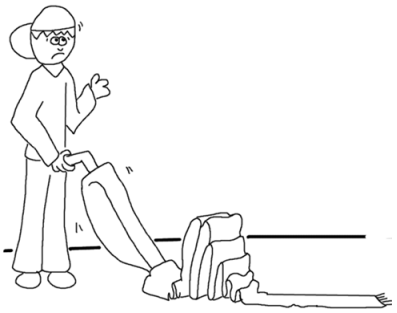
Destructive Testing

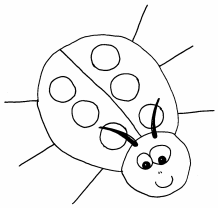
- Destructive testing involves stressing the application or it's environment until the application fails and then performing a root-cause analysis
  - measure and improve reliability
  - " … failure after 12,000 operations with 1000 users active …"



Boundary Testing

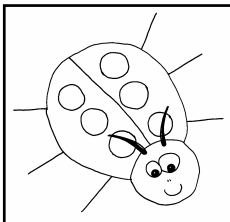
- Test with extreme input values
  - Lower and upper boundaries
  - Any edge conditions
  - Above and below extreme values
- Test to generate extreme output values
  - May or may not require extreme input values
  - Zero divide, overflow





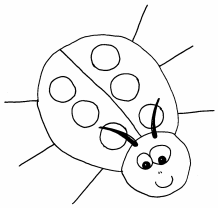
DAT Testing

- Deployment Acceptance Testing
  - Typical testing of web application on fully installed target hardware and software or on a staging site which is equivalent to the actual target or customer site.
  - Functional test suite is run to ensure operation is OK before we go live



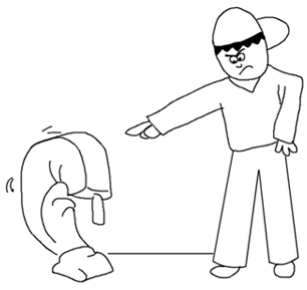
Real World User Testing

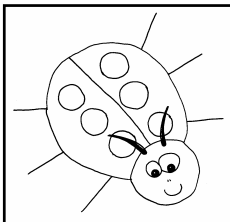
- End user centric testing
  - Simulate how real customers use the software
  - This is not the same as a usage scenario
    - Test is based on your knowledge of the typical sequences of operations by end user
    - Study logs or customer support info if available
    - Uncovers some of the most useful bugs!



Assertion Testing

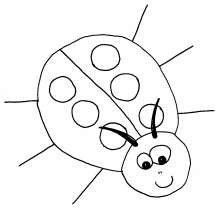
- "Asserts"
  - Added as in-line code
  - Ensure that software is operating in the correct state with correct values of certain variables, tables, parameters or other data
  - Assertions often implemented as MACROS
  - Generally disabled in commercial builds





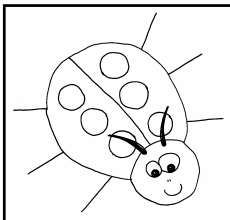
Assertion Testing

- Cautionary Warning About Assertions
  - *excellent practice for developers to ensure code works*
  - *can cause false failures if not implemented carefully (don't care condition handling)*
  - *ensure all developers use "Asserts" consistently!*



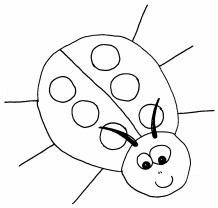
Compatibility Testing

- Similar to platform of configuration testing
  - check that an application functions correctly on various hardware and software environments
  - software
    - OS, Browser, DBMS, Network Software, Concurrent apps
  - hardware
    - CPU, Disk, Video Cards, Memory, Printers



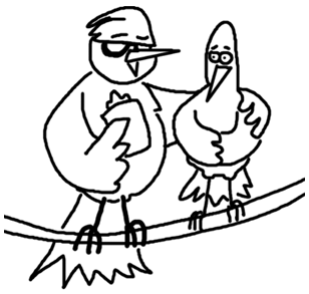
Documentation Testing

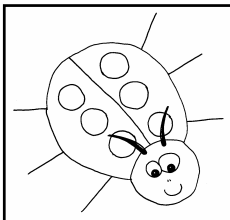
- Test software against user and reference documentation
  - factual and accurate
  - screen images are correct
  - examples work
  - marketing collateral is correct



On-Line Help Testing

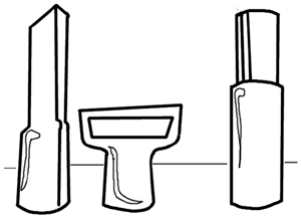
- Test functionality of Help System
  - is help factual and accurate
  - are we launching to correct page
  - does indexing work
  - are links correct
  - are tool tips correct
  - are images correct

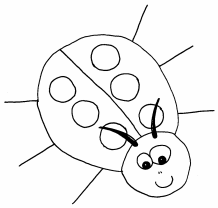




Collateral Testing

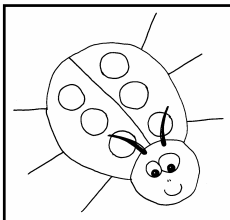
- Any software collateral shipped to the end user - or available via www should be tested to ensure it operates with software
  - Examples, Tutorials
  - Macros
  - Sample data
  - Read me files





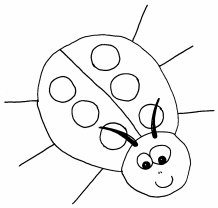
Install Testing

- Installation program can be very complex to test
  - different platforms
  - different options
  - over a previously installed version
  - not enough disk space available
  - missing or incorrect third party software



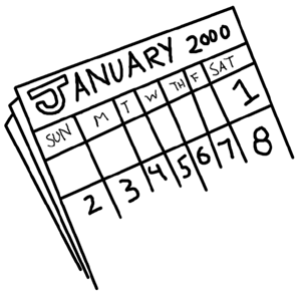
Uninstall Testing

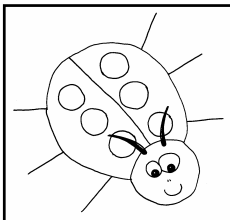
- Most applications come with an *Uninstall feature*
  - Is Uninstall clean?
  - Is registry clean?
  - Are user data files left alone?
  - Can all different configurations and options of install be Uninstalled?



Y2K Testing

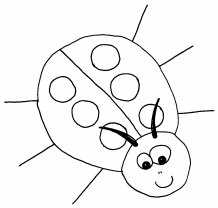
- Year 2000 testing is still needed!
  - Dominant software testing issue up to and including Year 2000
  - Testing of any date aware aspect of the application to confirm handling of forward and backward time calculations, dates, leap years
  - Examples: age computation, expiry dates





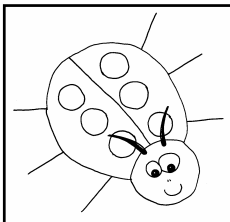
Conversion Testing

- Data conversion occurs when you upgrade from one version of an application to a subsequent version
  - Is all user data correctly converted to the new format?
  - Did we loose or any records? Any fields?
  - Are new fields initialized to correct values?



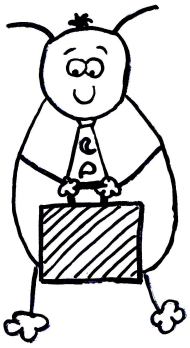
Milestone Testing

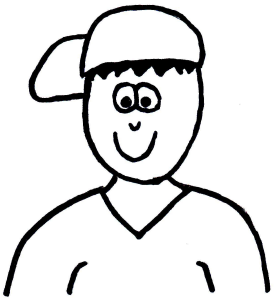
- Testing which must be passed before software is considered to have completed a milestone
  - For all project milestones stakeholders agree on milestone passage criteria and suitable testing which confirms the achievement!
  - Decide this early to avoid conflict and politics!



Thank You

- Questions?





© Robert Sabourin, 2000

AmiBug.Com, Inc.

QW2002 Paper 7Q

Douglas Hoffman
(Software Quality Methods, LLC.)

Overview of ASQ's 2002 CSQE Body of Knowledge

Key Points

- What is ASQ's CSQE about
- What are the subjects in the CSQE BOK
- How to relate tasks to jobs and performance

Presentation Abstract

The CSQE BOK was updated for 2002. The presentation describes the BOK and identifies and relates job performance measures relating to the BOK.

About the Author

Douglas Hoffman is an independent consultant with Software Quality Methods, LLC. He has been in the software engineering and quality assurance fields for nearly 30 years and now is a management consultant in strategic and tactical planning for software quality. He has just completed the writing and editing of several sections in the updated QCI CSQE Primer, the primary study guide covering all of the topics in the CSQE BOK. He also prepared and teaches CSQE examination preparation courses in Silicon Valley. His consulting experience includes management consulting, teaching, managing, and engineering across the computer systems and software industries. His recent experience has focused on creating and transforming software quality assurance and development groups, and architecting software test automation environments.

Douglas is a founder and past Chairman of the Santa Clara Valley Software Quality Association (SSQA) and the Silicon Valley Section of the American Society for Quality (ASQ). He has been a presenter and participant at dozens of software quality conferences and has been Program Chairman for several international conferences on software quality. He is a member of the ACM and IEEE and is active in the ASQ as a Senior Member, participating in the Software Division, the Santa Clara Valley Section, and the Software Quality Task Group. He has earned a BA in Computer Science, an MS in Electrical Engineering, an MBA, a Certificate from ASQ in Software Quality Engineering, and has been a registered ISO 9000 Lead Auditor.

ASQ's 2002 CSQE Body of Knowledge

Quality Week 2002

Douglas Hoffman
Software Quality Methods, LLC.
24646 Heather Heights Place
Saratoga, California 95070-9710
Phone 408-741-4830
Fax 408-867-4550
doug.hoffman@acm.org

Copyright © 2002, Software Quality Methods, LLC. No part of these graphic overhead slides may be reproduced, or used in any form by any electronic or mechanical duplication, or stored in a computer system, without written permission of the author.

"The Certified Software Quality Engineer…

… is a professional who has comprehensive understanding of software quality development and implementation; has a thorough understanding of software inspection, testing, verification, and validation; and can implement software development and maintenance processes and methods."

ASQ CSQE Certification brochure, Revised 5-02. Copyright © 2002 American Society for Quality.

ASQ

- The American Society for Quality
- Not-for-profit professional society
- Leading quality improvement organization in US for more than 50 years
- More than 117,000 individual and
- 1,100 corporate sustaining members
- 247 local Sections
- 22 industry and topic-specific Divisions.

 Copyright © 2002, SQM, LLC.

CSQE

- Certification Requirements
- The Subject Areas of the CSQE 2002 BOK
- Bloom's Levels Of Cognition
- Example of Performance Skill Levels
- Example of Mapping of Performance Levels To Job Requirements
- Describing Individual Performance Levels

 Copyright © 2002, SQM, LLC.

Certification Requirements

- Education and/or Experience
  - 8 years in quality field
  - up to 5 years credit for degrees
- Proof of Professionalism
- Examination
  - proctored, open book exam
  - 160 questions

 Copyright © 2002, SQM, LLC.

Recertification Requirements

- Recertify every 3 years
- 18 points needed
  - Professional Development
  - Employment
  - Instructor/Student
  - Meetings
  - Committees
  - Certifications
  - Proctoring
  - Publishing

 Copyright © 2002, SQM, LLC.

CSQE BOK Subject Areas

- General Knowledge, Conduct, and Ethics
- Software Quality Management
- Software Engineering Processes
- Program and Project Management
- Software Metrics, Measurement, and Analytical Methods
- Software Verification and Validation (V&V)
- Software Configuration Management

 Copyright © 2002, SQM, LLC.

General Knowledge, Conduct, and Ethics

- Quality philosophy and principles
- Standards, specifications, and models
- Leadership tools and skills
- Ethical conduct and professional development

 Copyright © 2002, SQM, LLC.

General Knowledge, Conduct, and Ethics

- Quality philosophy and principles
  - Benefits of software quality (C)
  - Prevention vs. detection (C)
  - Organizational and process benchmarking (An)
- Standards, specifications, and models (Ap)

 Copyright © 2002, SQM, LLC.

General Knowledge, Conduct, and Ethics

- Leadership tools and skills
  - Organizational leadership (Ap)
  - Team management (Ap)
  - Team tools (Ap)
  - Facilitation skills (Ap)
  - Communication skills (Ap)

 Copyright © 2002, SQM, LLC.

General Knowledge, Conduct, and Ethics

- Ethical conduct and professional development
  - ASQ Code of Ethics (E)
  - Software liability and safety issues (Ap)
  - Professional training and development (Ap)

 Copyright © 2002, SQM, LLC.

Software Quality Management

- Goals and objectives
- Methodologies
- Audits

 Copyright © 2002, SQM, LLC.

Software Quality Management

- Goals and objectives
  - Quality goals and objectives (E)
  - Outsourced services (E)
  - Planning (E)
  - Software quality management systems documentation (C)
  - Customer requirements (E)

 Copyright © 2002, SQM, LLC.

Software Quality Management

- Methodologies
  - Reviews, inspections, and testing (E)
  - Change management methods (E)
  - Cost of quality (COQ) (An)
  - Quality data tracking (E)
  - Problem reporting and corrective actions (E)
  - Quality improvement process (E)

 Copyright © 2002, SQM, LLC.

Software Quality Management

- Audits
  - Program development and administration (C)
  - Audit preparation and execution (C)
  - Audit reporting and follow up (Ap)

Software Engineering Processes

- Environmental conditions
- Requirements management
- Requirements engineering
- Analysis, design, and development methods and tools
- Maintenance management

Software Engineering Processes

- Environmental conditions
  - Life cycles (E)
  - Systems architecture (An)
- Requirements management
  - Requirements prioritization and evaluation (E)
  - Requirements change management (E)
  - Bi-directional requirements traceability (E)

 Copyright © 2002, SQM, LLC.

Software Engineering Processes

- Requirements engineering
  - Requirements types (An)
  - Requirements elicitation (C)
  - Requirements analysis and modeling (An)
  - System and software requirements specifications (An)

 Copyright © 2002, SQM, LLC.

Software Engineering Processes

- Analysis, design, and development methods and tools
  - Software design methods (Ap)
  - Types of software reuse (Ap)
  - Clean room and other formal methods (C)
  - Software development tools (Ap)

 Copyright © 2002, SQM, LLC.

Software Engineering Processes

- Maintenance management
  - Maintenance types (C)
  - Operational maintenance (C)

 Copyright © 2002, SQM, LLC.

Program and Project Management

- Planning
  - Project planning elements (Ap)
  - Goal-setting and deployment (Ap)
  - Project planning tools (An)
  - Cost and value data (Ap)

 Copyright © 2002, SQM, LLC.

Program and Project Management

- Tracking and controlling
  - Phase transition control techniques (An)
  - Interpreting and reporting COQ data (E)
  - Tracking elements and methods (E)
  - Project reviews (An)

 Copyright © 2002, SQM, LLC.

Program and Project Management

- Risk management
  - Risk management planning methods (S)
  - Risk probability (E)
  - Product release decisions (E)
  - Software security, safety, and hazard analysis issues (An)

 Copyright © 2002, SQM, LLC.

Software Metrics, Measurement, and Analytical Methods

- Metrics and measurement theory
- Process and product measurement
- Analytical techniques

 Copyright © 2002, SQM, LLC.

Software Metrics, Measurement, and Analytical Methods

- Metrics and measurement theory
  - Metrics and measurement theory (C)
  - Basic measurement theory and techniques (Ap)
  - Psychology of metrics (C)

 Copyright © 2002, SQM, LLC.

Software Metrics, Measurement, and Analytical Methods

- Process and product measurement
  - Process, product, and resource metrics (Ap)
  - Commonly used metrics (Ap)
  - Software quality attributes (C)
  - Defect detection effectiveness measures (Ap)
  - Program performance and process effectiveness (An)

 Copyright © 2002, SQM, LLC.

Software Metrics, Measurement, and Analytical Methods

- Analytical techniques
  - Data integrity (S)
  - Quality tools (An)
  - Sampling theory and techniques (An)

 Copyright © 2002, SQM, LLC.

Software Verification and Validation (V&V)

- Theory
- Reviews and inspections
- Test planning and design
- Test execution and evaluation

 Copyright © 2002, SQM, LLC.

Software Verification and Validation (V&V)

- Theory
  - V&V planning procedures and tasks (S)
  - V&V program (An)
  - Evaluating software products and processes (S)
  - Interfaces (C)

 Copyright © 2002, SQM, LLC.

Software Verification and Validation (V&V)

- Reviews and inspections
  - Types (Ap)
  - Items (Ap)
  - Processes (Ap)
  - Data collection, reports, and summaries (Ap)

 Copyright © 2002, SQM, LLC.

Software Verification and Validation (V&V)

- Test planning and design
  - Types of tests (S)
  - Test tools (C)
  - Test strategies (S)
  - Test design (Ap)
  - Test coverage of specifications (S)
  - Test environments (S)
  - Supplier components and products (Ap)
  - Test plans (Ap)

Douglas Hoffman Copyright © 2002, SQM, LLC. 31

Software Verification and Validation (V&V)

- Test execution and evaluation
  - Test implementation (Ap)
  - Test documentation (Ap)
  - Test reviews (S)
  - Code coverage metrics (Ap)
  - Customer deliverables (S)
  - Severity of anomalies (E)

Douglas Hoffman Copyright © 2002, SQM, LLC. 32

Software Configuration Management

- Configuration infrastructure
- Configuration identification
- Configuration control
- Configuration status accounting
- Configuration audits
- Release and distribution issues

Software Configuration Management

- Configuration infrastructure
  - Configuration management (C)
  - Library/repository processes (C)
  - Defect tracking and library tools (C)
- Configuration identification
  - Configuration items (Ap)
  - Baselines (C)
  - Configuration identification methods (C)
  - Software builds (S)

Software Configuration Management

- Configuration control
  - Item and baseline control (Ap)
  - Proposed modifications (C)
  - Review and configuration control boards (CCBs) (Ap)
  - Concurrent development (Ap)
  - Traceability (Ap)
  - Version control (Ap)
  - Configuration item interfaces (Ap)

 Copyright © 2002, SQM, LLC.

Software Configuration Management

- Configuration status accounting
  - Status reporting (C)
  - Changes to configuration items and baselines (C)
  - Documentation control (C)

 Copyright © 2002, SQM, LLC.

Software Configuration Management

- Configuration audits
  - Functional configuration audit (C)
  - Physical configuration audit (C)
- Release and distribution issues
  - Product release process issues (C)
  - Packaging, production, and distribution (K)

Levels of Cognition[1]

- Knowledge
- Comprehension
- Application
- Analysis
- Synthesis
- Evaluation

[1]Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). Taxonomy of educational objectives handbook 1: Cognitive domain. New York: McKay.

Levels of Cognition

Level		*Verbs*
Knowledge	-	write, list, name, define, label, state
Comprehension	-	explain, describe, summarize, illustrate, paraphrase
Application	-	use, solve, apply, construct, demonstrate, compute
Analysis	-	analyze, compare, contrast, separate
Synthesis	-	create, design, invent, develop
Evaluation	-	judge, recommend, critique, justify

Douglas Hoffman Copyright © 2002, SQM, LLC. 39

Example of Performance Skill Levels

Area	Knowledge	Comprehension	Application	Analysis	Evaluation	Synthesis
Audits	Answers questions during audits	Understands purpose and underlying meaning of audit questions	Able to participate as an auditor within an audit team	Participates as Lead Auditor given the audit plan	Creates the audit plan and is Lead Auditor	Trains Lead Auditors and participates in audit improvements
Leadership	Able to participate in projects	Acts independently in projects	Leads routine projects	Plans and leads routine projects	Plans and leads a large or cross-organizational project	Plans and manages changes in organizational culture

- Row for each Skill Area
- Column for each Level of Performance
- Cell describes behavior that demonstrates mastery
- Color code by columns

Douglas Hoffman Copyright © 2002, SQM, LLC. 40

Example Mapping Of Performance Levels To Job Requirements

Area / Level	Associate QE	Quality Engineer	Senior QE	Fellow
General Knowledge, Conduct, and	Ethics			
Quality philosophy and principles	In assigned work area.			
Standards, specifications, and models	N/A	In assigned work area.		
Leadership tools and skills	N/A			
Ethical conduct and professional development				

- Column for each job title and grade
- Color indicates Performance Skill Level expected
- Text amplifies or explains

Example Describing Individual Performance Levels

Area	<name>	Quality Engineer	Remarks
General Knowledge, Conduct, and Ethics			
Quality philosophy and principles	Knows quality principles in assigned work area.		
Standards, specifications, and models	Does not use available standards	In assigned work area.	Send to ISO 9000 overview class
Leadership tools and skills	Excellent leadership skills.		Get into mentor program
Ethical conduct and professional development	High ethics. Took leadership class.		

- Individual performance shown in second column
- Job Requirements column included for reference
- Remarks possibly color coded for strengths and weaknesses

Summary

- CSQE covers a wide field
- CSQE includes level of cognition
- You can
  - decide what applies to you
  - define Performance Skill Levels
  - define Job Requirements
  - describe Individual Performance

 Copyright © 2002, SQM, LLC.

Overview of ASQ's[1] Certified Software Quality Engineer (CSQE) Body of Knowledge

Douglas Hoffman
Software Quality Methods, LLC.
24646 Heather Heights Place
Saratoga, California 95070-9710
Phone 408-741-4830 Fax 408-867-4550
http://www.SoftwareQualityMethods.com
doug.hoffman@acm.org

Abstract

In 2002, the American Society for Quality (ASQ) has restructured and updated the Body of Knowledge (BOK) used for their Certification in Software Quality Engineering (CSQE). This paper describes the certification, outlines the updated BOK content, and highlights many of the changes[2]. It also provides a method of adapting such a BOK to describe key skills and levels of performance in a group. The following topics are covered in the paper:

- Certification Requirements
- The Subject Areas of the CSQE 2002 Body of Knowledge
  - General Knowledge, Conduct, and Ethics
  - Software Quality Management
  - Software Engineering Processes
  - Program and Project Management
  - Software Metrics, Measurement, and Analytical Methods
  - Software Verification and Validation (V&V)
  - Software Configuration Management
- Levels of Cognition (from Bloom's Taxonomy, 1956)
- Performance Skill Levels
- Mapping of Performance Levels to Job Requirements
- Describing Individual Performance Levels

---

[1] Much of the reference material contained in this document comes from ASQ's CSQE Certification brochure, ASQ Item B0110, Revised 5-02. Copyright © 2002 American Society for Quality. Reprinted with permission.
See the side bar About the ASQ for a brief description of ASQ. Further information on the American Society for Quality (ASQ) and certifications can be found at their web site at http://www.asq.org or by calling them at 800-248-1946.

[2] Special thanks to Bill Wortman and Wes Richardson at QCI for the mapping of the 2002 BOK with the 1996 BOK. (Quality Council of Indiana http://www.qualitycouncil.com)

Copyright © 2002, Software Quality Methods, LLC.

The purpose of this paper is to outline ASQ's CSQE BOK and describe how a mapping of such knowledge and skill areas into performance measures can be useful. By identifying relevant skill areas and performance measures, a quality engineering team can clearly understand what levels of performance are expected and how to evaluate the levels of performance being shown.

Readers should come away with a better understanding of:

- The topics contained in ASQ's CSQE BOK
- The level of understanding needed for certification
- How to map BOK topics into job requirements and individual skill sets
- A spreadsheet technique for displaying QA tasks and levels of performance
- A spreadsheet technique for mapping tasks and performance to job requirements
- A further technique for describing an individual's demonstrated performance level

The following detailed material is provided in appendices:

I. Outline of the Subject Areas of the CSQE 2002 Body of Knowledge
II. Mapping of the 2002 CSQE BOK with the 1996 BOK
III. CSQE Reference Materials

Background

"The Certified Software Quality Engineer…

…is a professional who has comprehensive understanding of software quality development and implementation; has a thorough understanding of software inspection, testing, verification, and validation; and can implement software development and maintenance processes and methods."[3]

I often illustrate the difference between software test and quality assurance by explaining about a course I teach to prepare engineers to take the CSQE examination. The course is four days long (28 hours), and I introduce and define the subject areas in the BOK. Software Testing is covered in about two hours. The rest of the time is spent on Software Quality Assurance.

About the ASQ

The American Society for Quality (ASQ), a not-for-profit professional society, has been the leading quality improvement organization in the United States for more than 50 years. ASQ has more than 117,000 individual and 1,100 corporate sustaining members worldwide. Individual members belong to one of 247 local Sections located throughout the United States, Canada, Mexico, Puerto Rico, and an International Chapter. ASQ also has 22 industry and topic-specific Divisions.

ASQ was incorporated as the American Society for Quality Control in 1946 as the result of the merger of several local quality societies that had formed after wartime statistical quality control classes. The classes were held to improve and maintain the quality of defense materials during World War II. To meet the needs of a changing marketplace, the organization changed its name to the American Society for Quality in 1997.

[3] ASQ CSQE Certification brochure, Item B0110, Revised 5-02. Copyright © 2002 American Society for Quality. Reprinted with permission. All rights reserved.

Copyright © 2002, Software Quality Methods, LLC.

Since 1968, nearly 80,000 certifications have been awarded to professionals through ASQ's programs. The BOK and CSQE certification was first made available in 1996. Forty countries offer ASQ's programs as their method for certification. ASQ conducts professional certification programs in eleven different areas of expertise including quality engineers, quality managers, and software quality engineers. ASQ reviews and updates the BOK for each area on a regular basis to keep up with the evolving information and technologies available in each field. Computer Science in particular is evolving and changing quickly, necessitating ongoing reevaluation and adaptation of software quality assurance concepts, tools, and techniques.

Many software quality assurance organizations have come from software development or user oriented software testing. Their principles and techniques have evolved from the emerging fields of Computer Science or Software Testing, and many have been created without benefit of research or experience in quality assurance. ASQ's programs are especially valuable for these software quality professionals because they are founded on principles and techniques from the broader, general science of quality assurance.

Although the CSQE BOK may not be universally applicable to software quality organizations, it provides an excellent starting place for detailing a BOK appropriate for a given organizational context. Performance Skill Levels can then be defined for each subject area in the BOK to describe corresponding behaviors. Job Requirements can be listed to establish performance expectations in each area for the job titles and grades used for quality engineers. (A Senior Software Quality Engineer would be expected to perform many of the same tasks as an Associate Software Engineer, but at a much higher level.) Likewise, an individual's performance can be described in each of the areas to identify strengths and weaknesses.

Certification Requirements

There are three requirements areas a candidate must fulfill to earn ASQ's CSQE:

- Education and/or Experience
- Proof of Professionalism
- Examination covering the CSQE BOK

Education and/or Experience

One must have eight years of on-the-job experience in one or more of the areas of the CSQE BOK. A minimum of three years of this experience must be in a decision-making position. "Decision-making" is defined as the authority to define, execute, or control projects/processes and to be responsible for the outcome. This may or may not include management or supervisory positions.

The RAB

The Registrar Accreditation Board (RAB), a separately incorporated affiliate of ASQ, is engaged in ISO 9000 and ISO 14000 accreditation and certification activities. The RAB is a partner with the American National Standards Institute (ANSI) in the National Accreditation Program (NAP) which accredits registrars and training course providers. The NAP process ensures customers and other stakeholders that companies have implemented proper management systems as defined by the ISO 9000 and ISO 14000 standards. The RAB independently operates the U.S. certification programs for both ISO 9000 and ISO 14000 auditors. These programs provide assurance that individuals are qualified to audit management systems against the requirements of recognized standards.

Copyright © 2002, Software Quality Methods, LLC.

If a person has been previously certified by ASQ as a Quality Engineer, Quality Auditor, Reliability Engineer, or Quality Manager, experience used to qualify for certification in these fields applies to certification as a Software Quality Engineer.

If you have completed a degree* (and have the diploma) from a college, university, or technical school with accreditation accepted by ASQ, part of the eight-year experience requirement will satisfied, as follows (only one degree may be claimed):

Diploma From	Experience
Technical or trade school	One year
Associate degree	Two years
Bachelor's degree	Four years
Master's or doctorate	Five years

*Degrees or diplomas from educational institutions outside the United States must be equivalent to degrees from U.S. educational institutions.

Proof of Professionalism

Proof of professionalism may be demonstrated in one of three ways:

1. Membership in ASQ, an international affiliate society of ASQ, or another society that is a member of the American Association of Engineering Societies or the Accreditation Board for Engineering and Technology
2. Registration as a Professional Engineer
3. The signatures of two persons who are ASQ members, members of an international affiliate society, or members of another recognized professional society, verifying that you are a qualified practitioner of the quality sciences

Examination Covering the CSQE BOK

Each certification candidate is required to pass a written examination that consists of multiple choice questions that measure comprehension of the BOK. The CSQE examination is a one-part, 160-question, four-hour exam and is offered in the English language only. Because the BOK for certification is affected by new technologies, policies, and the changing dynamics of manufacturing and service industries, changed versions of the examination based on the current BOK are used at each offering.

Examinations are conducted twice a year, in June and December, by local ASQ sections and international organizations. Some special examinations may be added in conjunction with ASQ sponsored conferences for attendees of the conference. All examinations are open-book. Each participant must bring his or her own reference materials. Use of reference materials and calculators is explained in the detailed guidance provided to applicants.

Copyright © 2002, Software Quality Methods, LLC.

Topics in the Body of Knowledge

The CSQE 2002 BOK is divided into seven areas, as outlined below. (The complete, ten page outline of the BOK as provided by ASQ is attached in Appendix I.) Each subject area covers important aspects of the field of software quality engineering that a professional software quality engineer should be familiar with.

Although a CSQE is not expected to have mastered all of the BOK (or all of any subject area), they must demonstrate their understanding of the field by getting passing marks on an intensive examination covering all of the subjects. There are four hours allotted to answer 160 multiple-choice questions in a proctored, open book, open notes examination. This gives an average of 90 seconds per question, which leaves very little time to refer to books or notes during the exam.

I. General Knowledge, Conduct, and Ethics (16 Questions)

These topics relate to quality philosophies, principles, standards, specifications, and models in general, and particularly as they relate to software quality assurance. This knowledge area includes general quality tools and skills useful for effective teamwork and leadership.

II. Software Quality Management (30 Questions)

These topics focus on the quality management systems for software. They include setting goals and objectives, techniques for evaluating and managing software quality, process audits, and quality improvement processes.

III. Software Engineering Processes (26 Questions)

Software engineering processes include development processes, system architectures, software tools, and methods for software requirements, analysis, design, and development. The area is covered from project concept and development, through maintenance and obsolescence.

IV. Program and Project Management (24 Questions)

Topics include software development planning, project tracking, project controls, and risk management. Techniques and concepts are applied to both software projects (the development and release of a software product) and software programs (ongoing operation and maintenance of software systems).

V. Software Metrics, Measurement, and Analytical Methods (24 Questions)

These topics focus on software product and process metrics, measurement theory, technology, psychological aspects of software metrics, common software metrics, measurement and analytical techniques, and quality tools.

Copyright © 2002, Software Quality Methods, LLC.

VI. Software Verification and Validation (V&V) (24 Questions)

Topics covered in this area include more than formalized software testing, reviews, and inspections. They cover techniques for verification (checking to see that each phase of work is done right) and validation (checking to see that the right work is done and requirements are met) including project planning, and technical analysis. Subject matter for software testing techniques encompasses test planning, test types, test tools, strategies, test design, environmental factors, documentation, implementation, reviews, execution, and evaluation.

VII. Software Configuration Management (16 Questions)

Configuration management topics, critically important contributors to software quality, include the infrastructure components, configuration identification, component control, traceability, status reporting, document control, audits, and release issues.

Copyright © 2002, Software Quality Methods, LLC.

Bloom's Six Levels of Cognition[4]

A software quality professional is not expected to master all subject areas to the same extent. In addition to content specifics, the subtext detail in the outline for the BOK indicates the intended complexity level of test questions for each topic (and thus, the depth of understanding expected). The six levels of comprehension are based on Bloom's "Levels of Cognition," and are presented below from least complex to most complex.

Knowledge

(Also commonly referred to as recognition, recall, or rote knowledge.) Being able to remember or recognize terminology, definitions, facts, ideas, materials, patterns, sequences, methodologies, principles, etc. Activities at this level are often described using verbs like define, list, label, name, state, or write.

Comprehension

Being able to read and understand descriptions, communications, reports, tables, diagrams, directions, regulations, etc. Activities at this level are often described using verbs like describe, explain, illustrate, paraphrase, or summarize.

Application

Being able to apply ideas, procedures, methods, formulas, principles, theories, etc., in job-related situations. Activities at this level are often described using verbs like apply, compute, construct, demonstrate, solve, or use.

Analysis

Being able to break down information into its constituent parts and recognize the parts' relationship to one another and how they are organized; identify sublevel factors or salient data from a complex scenario. Activities at this level are often described using verbs like analyze, categorize, compare, contrast, or separate.

Synthesis

Being able to put parts or elements together in such a way as to show a pattern or structure not clearly there before; identify which data or information from a complex set is appropriate to examine further or from which supported conclusions can be drawn. Activities at this level are often described using verbs create, design, develop, hypothesize, or invent.

Evaluation

Being able to make judgments regarding the value of proposed ideas, solutions, methodologies, etc., by using appropriate criteria or standards to estimate accuracy, effectiveness, economic benefits, etc. Activities at this level are often described using verbs like critique, judge, justify, or recommend.

---

[4] Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). Taxonomy of educational objectives handbook 1: Cognitive domain. New York: McKay.

Copyright © 2002, Software Quality Methods, LLC.

Examples of Performance Skill Levels[5]

By listing and defining each subject area expected to be understood by a software quality professional, the BOK also defines the scope of activities for quality engineers. A similar list or BOK can be created to identify the context specific subject areas applicable to any particular software quality organization. The subject areas, levels of understanding, specific job functions, and performance activities are unique to each organization and evolve over time. We can define performance expectations and develop a guide for gaging individual performance by listing the organization and job specific subject areas, and describing typical activity exemplars for each performance level. The incomplete table below provides an example of subject areas and activity descriptions. (**Table 1: Example Performance Skill Levels**)

The first column of the table lists the subject areas for the organization's BOK. The subsequent columns each represent successive levels of understanding and performance. Each row of the table describes a subject area and the various levels of understanding demonstrated by the types of behaviors listed. The columns are shaded uniquely to simplify identification of job requirements and individual performance levels.

The example shows six Performance Skill Levels corresponding to Bloom's Levels of Cognition. There can be more or fewer Performance Skill Levels, and they can be defined as appropriate for the organization. For example, three levels of performance could have columns titled "Familiarity," "Proven Skill," and "Creative Expertise," with descriptions of behaviors in the cells for each skill area.

Mapping of Performance Levels to Job Requirements

Various jobs within an organization encompass different task activities, and various levels of performance are expected for different grades or levels of experience. We can define the expectations of performance for various job titles in a software quality organization using the BOK and the table depicting Performance Skill Levels appropriate for each grade. The result can be represented in a tabular form, much the same as for Performance Skill Levels. (**Table 2: Example Job Performance Requirements**)

The first column of the table lists the subject areas appropriate for the performance of duties for the job functions. This can include the entire BOK or a subset of it, depending on the tasks required for the job functions. Subsequent columns are created for each job grade. The cells each describe the types of activities a person with this job title is expected to perform for this subject area. The appropriate color is shown if the description is identical to the Performance Skill Levels. Text is added to clarify or describe the particular performance required. No color is shown if the subject area is not applicable or not required for the job function and grade. A typical table covers all the grades for a particular job title and the subset of BOK subjects applicable to the job.

[5] The tabular method of describing and mapping of a BOK to job descriptions and performance is based on work done by James Bach (www.satisfice.com) and made available by STLabs.

Copyright © 2002, Software Quality Methods, LLC.

Describing Individual Performance Levels

A table can also be used to describe an individual's performance by using the first column for the BOK subject areas and a second column for examples describing the actual behaviors demonstrated. This information can be compared with the Job Performance Requirements for the person's job title and grade to identify the individual's strengths and weaknesses relative to the job requirements. (**Table 3: Example Individual Performance Levels**)

The color code in the second column identifies the Performance Skill Level observed. Text in the cells can amplify or explain the observations. The third column is a direct copy of the applicable Job Requirements for the individual. Remarks may be color coded to indicate areas of strength or where improvement is needed.

Copyright © 2002, Software Quality Methods, LLC.

Area \ Level	Knowledge	Comprehension	Application	Analysis	Synthesis	Evaluation
General Knowledge, Conduct, and Ethics						
Quality philosophy and principles	Defines and lists terminology, philosophies, and principles.	Summarizes and explains descriptions and methods related to quality principles.	Applies quality philosophies, principles, and methods in job-related situations.	Compares and contrasts quality philosophies and principles.	Creates quality philosophies and principles as required.	Makes judgments and recommendations regarding quality philosophies and principles.
Standards, specifications, and models	Identifies, lists, and defines terminology, standards, and models.	Describes and illustrates relevant standards and specifications.	Applies ideas, procedures, methods, formulas, principles, theories, etc., in job-related situations.	Compares standards, specifications, and models for applicability to given situation.	Develops standards, specifications, and models as required.	Critiques, recommends, and justifies quality philosophies and principles.
Leadership tools and skills	Lists leadership terminology, definitions, and principles.	Explains and illustrates leadership tools and techniques.	Applies appropriate leadership principles, tools, and skills.	Compares and contrasts appropriate leadership tools and techniques.	Develops leadership tools and techniques appropriate for job situations.	Critiques, recommends, and justifies leadership tools and approaches.
Ethical conduct and professional development	Defines a professional code of ethics and can identify and locate methods of professional development.	Explains a professional code of ethics and options for professional development.	Abides by a professional code of ethics and develops professionally.	Analyzes professional development requirements and options.	Designs and develops professional development programs.	Critiques, recommends, and justifies professional development approaches.

Table 1: Example Performance Skill Levels

Copyright © 2002, Software Quality Methods, LLC.

Area \ Job Title	Associate QE	Quality Engineer	Senior QE	QE Fellow
General Knowledge, Conduct, and Ethics				
Quality philosophy and principles	In assigned work area.			
Standards, specifications, and models	N/A	In assigned work area.		
Leadership tools and skills	N/A			
Ethical conduct and professional development				

* Colors correspond to columns in Table 1: Example Performance Skill Levels

Table 2: Example Job Performance Requirements

Area	<name>	Quality Engineer	Remarks
General Knowledge, Conduct, and Ethics			
Quality philosophy and principles	Knows quality principles in assigned work area.		
Standards, specifications, and models	Does not use available standards	In assigned work area.	Send to ISO 9000 overview class
Leadership tools and skills	Excellent leadership skills.		Get into mentor program
Ethical conduct and professional development	High ethics. Took leadership class.		

* The Quality Engineer column is included as reference.

Table 3: Example Individual Performance Levels

Copyright © 2002, Software Quality Methods, LLC.

Conclusion

ASQ's CSQE BOK covers a very broad range of subjects applicable to software quality engineering. The BOK is generic in the sense that it may be applied to any software quality organization, even though the particular subjects and task emphasis relevant to specific organizations are different. The subjects are broken into seven general areas, and the CSQE BOK provides a detailed outline describing the concepts, tools, and techniques in each area. Along with the description of the knowledge area, the outline also lists the expected level of understanding a CSQE is expected to have.

A similar BOK can be developed that applies to a specific software quality organization. This BOK can also be used to identify the tasks associated with various job functions and the levels of performance of the tasks required at different job grades. Individual performance can then be described in terms of the subject areas and tasks appropriate for their job. The individual performance can then be compared to the job performance requirements to identify the individual's strengths and weaknesses.

Copyright © 2002, Software Quality Methods, LLC.

Software Quality Engineer Certification (CSQE) Body of Knowledge[6]

The following is an outline of topics that constitute the ASQ's 2002 Body of Knowledge for Software Quality Engineer.

I. GENERAL, KNOWLEDGE, CONDUCT, and ETHICS (16 Questions)
   A. Quality philosophy and principles
      1. Benefits of software quality
         Describe how software quality engineering can benefit an organization. (Comprehension)
      2. Prevention vs. detection
         Describe how quality engineering methodologies can reduce the length of time for testing and can influence other defect detection methods. (Comprehension)
      3. Organizational and process benchmarking
         Identify, analyze, and model best practices at the macro (organizational) and micro (process and project) levels. Identify and develop business objectives, use metrics to monitor their achievement, and provide feedback to close the process improvement loop. (Analysis)
   B. Standards, specifications, and models
      Identify and use software process and assessment models, including ISO 9001, ISO 15504, IEEE software standards, IEEE/EIA 12207, SEI Capability Maturity Model Integrated (CMMI), etc., in a variety of situations. (Application)
   C. Leadership tools and skills
      1. Organizational leadership
         Define, describe, and apply leadership tools and techniques, including analyzing current situations, proposing, justifying, implementing, and managing change (using change-agent tools), developing and implementing quality initiatives, obtaining cross-functional commitment and collaboration, ensuring knowledge transfer, motivating personnel, etc. (Application)
      2. Team management
         Define and use various team management techniques, including identifying and assigning roles and responsibilities (e.g., champion, sponsor, facilitator, leader, coach), identifying and assessing team member skills, interpreting team dynamics and stages of team development, handling dominant or disruptive team members, recognizing how diversity in teams strengthens the creative process, etc. (Application)
      3. Team tools
         Define, describe, and use tools such as brainstorming, nominal group technique (NGT), joint application development (JAD), rapid application development (RAD), etc. (Application)
      4. Facilitation skills

[6] Copyright © 2002 American Society for Quality. All rights reserved.

Use various tools to manage and resolve conflict. Use negotiation techniques to produce win-win outcomes. Identify and use time and meeting management tools to maximize performance. (Application)

5. Communication skills
   Define, describe, and apply various communication elements used in verbal, written, and presentation formats, including interviewing and listening skills. Apply communication elements to create effective process and procedural documents, including identifying roles and responsibilities. (Application)

D. Ethical conduct and professional development
   1. ASQ Code of Ethics
      Determine appropriate behavior in situations requiring ethical decisions, including identifying conflicts of interest and recognizing/resolving ethical issues related to software licensing and use. (Evaluation)
   2. Software liability and safety issues
      Identify legal issues related to software product liability and safety, including negligence, customer notification requirements, and other legal or regulatory issues. (Application)
      [NOTE: Other aspects of product safety and hazard analysis are covered in IV.C.4.]
   3. Professional training and development
      Define, describe, and apply training needs analysis methods for software quality professionals, and manage training resources and materials. (Application)

II. SOFTWARE QUALITY MANAGEMENT (30 Questions)

A. Goals and objectives
   1. Quality goals and objectives
      Describe, analyze, and evaluate quality goals and objectives for programs, projects, and products. (Evaluation)
   2. Outsourced services
      Define, analyze, and evaluate the impact of acquisitions, subcontractor services, and other external resources on the organization's goals and objectives. (Evaluation)
   3. Planning
      Identify, apply, and evaluate scheduling and resource requirements necessary to achieve quality goals and objectives. (Evaluation)
   4. Software quality management (SQM) systems documentation
      Identify and describe various elements related to SQM system documentation. (Comprehension)
   5. Customer requirements
      Analyze and evaluate customer requirements and their effect on programs, projects, and products. (Evaluation)
      [NOTE: Changes in requirements are covered in III.B.3. The focus in this section is to ensure that customer requirements are evaluated properly.]

B. Methodologies
   1. Review, inspection, and testing
      Define, describe, evaluate, and differentiate between these defect detection methods. (Evaluation)
   2. Change management methods

Copyright © 2002, Software Quality Methods, LLC.

Identify and apply various methods appropriate for responding to changes in technology, organizations, environment, human performance, etc. (Evaluation)
[NOTE: Change-agent tools are covered in I.C.1.]

3. Cost of quality (COQ)
   Define, differentiate, and analyze COQ categories (prevention, appraisal, internal failure, external failure) and their impact on products and processes. (Analysis)
   [NOTE: Interpreting and reporting COQ data are covered in IV.B.2.]
4. Quality data tracking
   Define, describe, select, and implement information systems and models used to track quality data in various situations. (Evaluation)
5. Problem reporting and corrective action procedures
   Define, describe, analyze, and distinguish between these procedures for software defects, process nonconformances, and other quality system deficiencies. (Evaluation)
6. Quality improvement processes
   Define, describe, analyze and distinguish between various defect prevention, detection, and removal processes, and evaluate process improvement opportunities in relation to these tools. (Evaluation)

C. Audits
1. Program development and administration
   Identify roles and responsibilities for various audit participants, including team leader, team members, auditee, auditor, etc. (Comprehension)
2. Audit preparation and execution
   Define and distinguish between various audit types, including process, compliance, supplier, system, etc. Define and describe various steps in the audit process, from scheduling the audit through the closing meeting and subsequent follow-up activities. Define and identify various tools and procedures used in conducting audits. (Comprehension)
3. Audit reporting and follow up
   Identify, describe, and apply the steps of audit reporting and follow up, including the need for and verification of corrective action. (Application)

III. SOFTWARE ENGINEERING PROCESSES (26 Questions)

A. Environmental conditions
1. Life cycles
   Compare and evaluate the characteristics of spiral, waterfall, incremental, rapid prototyping, V-model, etc. Differentiate these life cycles, describe what they are designed to do, what their benefits are, and in what situations they should be used. (Evaluation)
2. Systems architecture
   Identify, describe, evaluate, and distinguish between system architectures, including client server, n tier, B to B, B to C, and B to E, web (internet/intranet/extranet) and wireless development, messaging and collaboration software, etc. (Analysis)

B. Requirements management
1. Requirements prioritization and evaluation

Copyright © 2002, Software Quality Methods, LLC.

Describe, assess, prioritize, and evaluate the requirements for verifying software correctness, consistency, completeness, and testability. Determine what should be covered in a requirements statement, how to specify a requirement, etc. (Evaluation)

2. Requirements change management
   Define, describe, and evaluate various elements of managing requirements change, including what processes should be followed, when requirements need to change, what review processes to use, etc. Define the effect of changing requirements at various stages of the project life cycle. (Evaluation)
3. Bi-directional requirements traceability
   Describe, select, and evaluate various traceability elements, including requirements to design, design to code, and requirements to test. Describe and apply traceability tools and mechanisms, such as system verification diagrams, traceability matrices, etc. (Evaluation)
   [NOTE: Traceability of configuration items is covered in VII.C.5.]

C. Requirements engineering
   1. Requirement types
      Define, describe, and analyze various requirement types such as security, regulatory, quality, feature and product functionality, etc., and the significant elements of each. (Analysis)
   2. Requirements elicitation
      Define and describe various elicitation methods, including using tools such as quality function deployment (QFD), joint application development (JAD), customer needs analysis, etc. Describe the key steps necessary for gathering product requirement details, and identify common causes of failure to comply with requirements. (Comprehension)
   3. Requirements analysis and modeling
      Describe, select, and analyze tools such as data flow diagrams (DFDs), entity relationship diagrams (ERDs), use cases, etc. Describe how they are used at different phases of development and requirements specifications. (Analysis)
   4. System and software requirements specifications
      Define and distinguish between these two types of specifications and their purpose, and describe their relationship to each other. (Analysis)

D. Analysis, design, and development methods and tools
   1. Software design methods
      Define and use various design methods, including object-oriented analysis and design (OOAD), structured analysis and design (SAD), unified modeling language (UML), etc. Identify the steps used in program design and explain their uses. (Application)
   2. Types of software reuse
      Define, describe, and differentiate the use of various reuse methods including reengineering, reverse engineering, plug-and-play, etc., and describe the design paradigms that address these concepts. (Application)
   3. Clean room and other formal methods
      Define and describe these methods and their benefits. (Comprehension)
   4. Software development tools
      Identify, describe, use, and distinguish between various tools used for modeling, code analysis, documentation, relational databases, etc. (Application)

Copyright © 2002, Software Quality Methods, LLC.

E. Maintenance management
   1. Maintenance types
      Describe the characteristics of corrective, adaptive, and perfective maintenance types and their benefits and risks. (Comprehension)
   2. Operational maintenance
      Describe the various categories of and activities involved in providing operational services to the customer, managing application portfolios, and providing basic software maintenance. (Comprehension)

IV. PROGRAM AND PROJECT MANAGEMENT (24 Questions)

A. Planning
   1. Project planning elements
      Describe and use factors such as forecasts, resources, schedules, etc., to develop, initiate, and accomplish project goals. (Application)
   2. Goal-setting and deployment
      Identify and use milestones, objectives achieved, task duration, and other goal-setting and deployment methods. (Application)
   3. Project planning tools
      Define, apply, and analyze various methods of managing risk, estimating costs, scheduling resources, etc. using tools such as PERT charts, critical path method (CPM), work breakdown structure (WBS), etc. (Analysis)
      [NOTE: Gantt charts are covered in IV.B.1.]
   4. Cost and value data
      Identify and use various methods for calculating project-related data such as earned value, development investment costs, etc. (Application)

B. Tracking and controlling
   1. Phase transition control techniques
      Develop and use various control techniques for tracking projects, including entry/exit criteria, phase gate reviews, Gantt charts, etc. (Analysis)
   2. Interpreting and reporting cost of quality (COQ) data
      Review, interpret, and report COQ data and evaluate how each category is affected by continuous improvement strategies. (Evaluation)
      [NOTE: The definitions and distinctions between these categories are covered in II.B.3.]
   3. Tracking elements and methods
      Describe, assess, and apply different tracking methods, including establishing metrics for costs, deliverables, productivity, etc., creating and evaluating status reports and life-cycle phase reports, measuring changes in earned value, evaluating changes in business conditions, etc. (Evaluation)
      [NOTE: Calculating earned value is covered in IV. A. 4.]
   4. Project reviews
      Define, use, and differentiate various types of reviews, including post-project, senior management, team, etc., and use closed-loop methodologies to improve projects as a result of lessons learned. (Analysis)

C. Risk management
   1. Risk management planning methods

Copyright © 2002, Software Quality Methods, LLC.

Define, integrate, and analyze various risk management methods, including assessing, preventing, and mitigating risk with respect to critical aspects of a project and its supporting strategies. (Synthesis)

2. Risk probability
   Describe and evaluate various risk warning signs, assess risk probability and impact, and develop contingency plans. (Evaluation)
3. Product release decisions
   Identify situations and factors that require trade-offs on product release decisions. Develop and analyze various ways of bringing a project back on track when problems occur that affect quality, scheduling, customer requirements, product functionality, etc. (Evaluation)
4. Software security, safety, and hazard analysis issues
   Identify, review, and evaluate various factors related to software security, safety-critical software, and hazard analyses. Identify and describe rationales for developing safety plans and for implementing hazard analyses. (Analysis)
   [NOTE: The legal aspects of product safety are covered in I.D.2.]

V. SOFTWARE METRICS, MEASUREMENT, AND ANALYTICAL METHODS (24 Questions)

A. Metrics and measurement theory

1. Definitions
   Define, describe, and explain various terms related to metrics and measurement, including error, reliability, internal vs. external validity, explicit vs. derived measures, etc. (Comprehension)
2. Basic measurement theory and techniques
   Define, describe, and use basic measurement scales (nominal, ordinal, ratio, interval), the central limit theorem and related terms, including mean, median, mode, standard deviation, variance, etc. (Application)
3. Psychology of metrics
   Define and describe various uses of metrics. Compare and contrast how metrics affect people and how people affect metrics. (Comprehension)

B. Process and product measurement

1. Process, product, and resource metrics
   Describe and use various metrics to assess processes, products, and resources. (Application)
2. Commonly used metrics
   Define and use metrics to measure various aspects of software, including software complexity, lines of code (LOC), non-commented lines of code (NCLOC), design defects, requirements volatility, system performance, etc. (Application)
   [NOTE: Code coverage metrics are covered in VI.D.4.]
3. Software quality attributes
   Identify and describe various criteria for measuring attributes such as maintainability, verifiability, reliability, usability, reusability, testability, expandability, etc. (Comprehension)
4. Defect detection effectiveness measures
   Define, describe, and use defect detection measures such as cost, yield, customer impact, etc., and track their effectiveness. (Application)

Copyright © 2002, Software Quality Methods, LLC.

5. Program performance and process effectiveness
   Identify and use various methods of examining performance and effectiveness. (Analysis)

C. Analytical techniques
1. Data integrity
   Define, use, and interpret various techniques to ensure the quality of metrics data, its accuracy, completeness, timeliness, etc. (Synthesis)
2. Quality tools
   Define, select, and use quality analysis and problem-solving tools such as flow charts, Pareto charts, cause and effect diagrams, check sheets, scatter diagrams, control (run) charts, histograms, root cause analysis, affinity diagrams, tree diagrams, process decision program charts (PDPCs), matrix diagrams, interrelationship digraphs, prioritization matrices, activity network diagrams. (Analysis)
3. Sampling theory and techniques
   Describe, differentiate, and analyze various sampling techniques for use in auditing, testing, product acceptance, etc. (Analysis)

VI. SOFTWARE VERIFICATION AND VALIDATION (V&V) (24 Questions)

A. Theory
1. V&V planning procedures and tasks
   Identify and select various methods for verification and validation, including static analysis, structural analysis, mathematical proof, simulation, etc. Identify and analyze which tasks should be iterated as a result of proposed or completed modifications. (Synthesis)
2. V&V program
   Describe and analyze methods for managing and reviewing a V&V program, including technical accomplishments, resource utilization, program status, etc. (Analysis)
3. Evaluating software products and processes
   Analyze and select various ways of evaluating documentation, source code, test and audit results, etc., to determine whether user needs and project objectives have been satisfied. (Synthesis)
4. Interfaces
   Identify various interfaces used with hardware, user, operator, and software applications. (Comprehension)

B. Reviews and inspections
1. Types
   Define, describe, and use various types of reviews and inspections, including desk-checking, walk-throughs, Fagan and Gilb inspections, technical accomplishments, resource utilization, future planning, etc. (Application)
2. Items
   Identify, describe, and use various review and inspection items, including proposals, project charters, specifications, code, tests, etc. (Application)
3. Processes
   Define, describe, and use various review and inspection processes to examine objectives, criteria, techniques, methods, etc. (Application)

Copyright © 2002, Software Quality Methods, LLC.

4. Data collection, reports, and summaries
   Define, describe, and use terms related to data collection, including preparation rates, defect density yield, phase containment, etc. (Application)

C. Test planning and design
   1. Types of tests [6B1]
      Select, apply, and develop various types of test, including functional, performance, regression, certification, environmental load, stress, worst case, perfective, exploratory, etc. (Synthesis)
   2. Test tools
      Define and describe the application and capabilities of commonly used test tools such as acceptance test suites, utilities (for memory, screen capture, string-finding, file viewer, file comparison, etc.), and diagnostics (for hardware, software, configuration, etc.). (Comprehension)
   3. Test strategies
      Identify, analyze, and apply various test strategies, including top-down, bottom-up, black-box, white-box, simulation, automation, etc. (Synthesis)
   4. Test design
      Identify, describe, and apply various types of test design including fault insertion, fault-error handling, equivalence class partitioning, boundary value, etc. (Application)
   5. Test coverage of specifications
      Identify, apply, and develop various test coverage specifications, including functions, states, data and time domains, etc. (Synthesis)
   6. Test environments
      Identify various environments and use tools such as test libraries, drivers, stubs, harnesses, etc., in those environments, and describe how simulations can be used in test environments. (Synthesis)
   7. Supplier components and products
      Identify the common risks and benefits of incorporating purchased software into other software products. Use various methods to test supplier components and products in the larger system. (Application)
   8. Test plans
      Identify, describe, and apply methods for creating and evaluating test plans including system, acceptance, validation, etc., to determine whether project objectives are being met. (Application)

D. Test execution and evaluation
   1. Test implementation
      Define, describe, and use various implementation elements, including scheduling, freezing, dependencies, V-model, error repair models, acceptance testing, etc. (Application)
   2. Test documentation
      Define, describe, and use various documentation procedures, including defect recording and tracking, test report completion metrics, trouble reports, input/output specifications, etc. (Application)
   3. Test Reviews
      Describe, develop, and analyze various methods of reviewing test efforts, including technical accomplishments, future planning, risk management, etc. (Synthesis)

Copyright © 2002, Software Quality Methods, LLC.

4. Code coverage metrics
   Define and apply various metrics including branch-to-branch, condition, domain, McCabe's cyclomatic complexity, boundary, etc. (Application)
   [NOTE: Other types of metrics are covered in V.B.2.]
5. Customer deliverables
   Identify and select various methods for testing the accuracy of customer deliverables, including packaged or downloaded products, license keys, user documentation, marketing and training materials, etc. (Synthesis)
6. Severity of anomalies
   Identify and select various methods for evaluating severity of anomalies in software operations. (Evaluation)

VII. SOFTWARE CONFIGURATION MANAGEMENT (16 Questions)

A. Configuration infrastructure

1. Configuration management
   Describe the roles and responsibilities of the configuration management group. (Comprehension)
2. Library/repository processes
   Define and identify processes used in a library system including dynamic, static, controlled, etc., and their related procedures. (Comprehension)
3. Defect tracking and library tools
   Define and describe configuration management tools used for defect tracking, library management tools, etc. (Comprehension)

B. Configuration identification

1. Configuration items
   Define, select, and use various items, including documentation, code interfaces, training materials, customer-supplied equipment, etc. (Application)
2. Baselines
   Define and identify when configuration baselines are created and used. (Comprehension)
3. Configuration identification methods
   Define and describe how these methods relate to schemes, naming conventions, versions, serializations, etc. (Comprehension)
4. Software builds
   Define and describe the primary purpose of software builds and their relation to configuration management functions. Describe and use various methods for controlling builds, including automation, new-version builds, etc. (Synthesis)

C. Configuration control

1. Item and baseline control
   Define, describe, and apply various control processes, including version control, traceability requirements, specifications, concurrent development, verifying milestones, etc. (Application)
2. Proposed modifications
   Describe how to assess proposed modifications, enhancements, or additions in terms of their impact on an existing or planned system. (Comprehension)
3. Review and configuration control boards (CCBs)

Copyright © 2002, Software Quality Methods, LLC.

Define, describe, and differentiate the roles and responsibilities of and procedures used by these boards. (Application)

4. Concurrent development
   Describe how configuration management control principles can be used in concurrent development processes. (Application)
5. Traceability
   Identify and apply various tools and methods for establishing and maintaining traceability design, including backward and forward traceability, naming conventions, etc., and explain how they are related to configuration management objectives. (Application)
   [NOTE: Traceability through product development is covered in III.B.3. The focus for this area is on traceability and evolution of configuration items in code archives and other configuration management elements.]
6. Version control
   Define, describe, and use version control methods such as source code version management and others, and how such methods can be used effectively by both small and large development teams. (Application)
7. Configuration item interfaces
   Define, describe, and apply management control processes for configuration item interfaces. (Application)

D. Configuration status accounting
   1. Status reporting
      Describe various processes for establishing, maintaining, and reporting the status of configuration items. (Comprehension)
   2. Changes to configuration items and baselines
      Describe the processes that should be used when changes are proposed to configuration items and baselines. (Comprehension)
   3. Documentation control
      Define and describe related procedures for document distribution, approval, storage, retrieval, revision, etc. (Comprehension)

E. Configuration audits
   1. Functional configuration audit
      Describe the primary purpose of these types of audits in relation to product specifications and in contrast to physical configuration audits. (Comprehension)
   2. Physical configuration audit
      Describe the primary purpose of these types of audits in relation to product specifications and in contrast to functional configuration audits. (Comprehension)

F. Release and distribution issues
   1. Product release process issues
      Identify and describe product release issues such as planning, scheduling, hardware and software dependencies, etc. (Comprehension)
   2. Packaging, production, and distribution
      Define and describe these components in relation to product release requirements and related issues. (Knowledge)

Copyright © 2002, Software Quality Methods, LLC.

Mapping of 2002 CSQE BOK with the 1996 BOK[7]

CSQE BOK 2002	CSQE BOK 1996
I. General Knowledge, Conduct and Ethics (16 Questions)	**I. General Knowledge, Conduct and Ethics (24 Questions)**
I.A. Quality philosophy and principles	I.B. Quality Philosophies and Principles
I.A.1. Benefits of software quality	I.B.1. Benefits of software quality
[No similar topic found in BOK 2002]	I.B.2. Quality philosophies (e.g., Juran, Deming, Crosby)
I.A.2. Prevention vs. detection	I.B.3. Prevention vs. Detection philosophies
[No similar topic found in BOK 2002]	I.B.4. Software Total Quality Management principles and applications
I.A.3. Organizational and process benchmarking	I.B.5. Organization and process benchmarking (i.e., identifying, analyzing, and modeling best practices)
I.B. Standards, specifications, and models	I.A. Standards
I.B. Standards, specifications, and models - Identify and use software process and assessment models including ISO 9001, ISO 15504, IEEE software standards, IEEE/EIA 12207, SEI Capability Maturity Model Integrated (CMMI), etc., in a variety of situations.	I.A.1. Domestic and international standards and specifications (e.g., ISO 9000, IEEE, Human Factors and Ergonomics Society, graphical user interface guidelines) I.A.2. Software quality and process initiatives, ventures, and consortia (e.g., SEI, SPICE, bootstrap, ESPRIT)
I.C. Leadership tools and skills	I.C. Organizational and Interpersonal Techniques
I.C.1 Organizational leadership I.C.2. Team management I.C.4. Facilitation skills	I.C.5. Facilitation (e.g., team management, customer-supplier relationships)
I.C.1 Organizational leadership I.C.2. Team management I.C.3 Team Tools I.C.4. Facilitation skills	I.C.6. Principles of team leadership and facilitation
I.C.1 Organizational leadership I.C.2. Team management I.C.4. Facilitation skills	I.C.7. Meeting management
I.C.1 Organizational leadership I.C.2. Team management I.C.4. Facilitation skills	I.C.8. Conflict resolution
I.C.1 Organizational leadership I.C.2. Team management	I.C.9. Organization and implementation of various types of quality teams
I.C.5. Communication skills	I.C.1. Verbal communication and presentation I.C.2. Written communication I.C.3. Effective listening I.C.4. Interviewing

[7] Copyright © 2002 Quality Council of Indiana Reprinted with permission. All rights reserved.

CSQE BOK 2002	CSQE BOK 1996
I.D. Ethical conduct and professional development	I.E. Professional Conduct and Ethics II.C. Organizational and Professional Software Quality Training
I.D.1. ASQ Code of Ethics	I.E.1. ASQC Code of Ethics I.E.2. Conflict of interest issues for a software quality engineer I.E.3. Ethical issues involving software product licensing
I.D.2. Software liability and safety issues	I.E.4. Legal issues involving software product liability and safety (e.g., negligence, customer notification, recall, regulations)
I.D.3. Professional training and development	II.C.1. Quality training subject areas (e.g., inspection, testing, configuration management, project management) II.C.2. Available training resources, materials, and providers II.C.3. Professional societies, technical associations, and organizations for software quality engineers
II. Software Quality Management (30 Questions)	**II. Software Quality Management (16 Questions)**
II.A. Goals and Objectives	II.A. Planning
II.A.1. Quality goals and objectives II.A.3. Planning	II.A.1. Product and project software quality goals and objectives
II.A.2. Outsourced Services	IV.A.6. Supplier management methodologies
II.A.3. Planning	II.A.3. Quality and customer support activities
II.A.4. Software quality management (SQM) systems documentation	[New topic in BOK 2002]
II.A.5. Customer requirements	II.A.2. Customer requirements for quality
II.B. Methodologies	II.B. Tracking
II.B. Change methodologies	III.B. Process and Technology Change Management
II.B.1. Review, inspection, and testing	III.A.3. Defect prevention, detection, and removal methods
II.B.2. Change management methods	III.B.1. Software process and technology change management theory and methods
[No similar topic found in BOK 2002]	III.B.2. Process maturity models
[No similar topic found in BOK 2002]	III.B.3. Software process assessment and evaluation techniques
[No similar topic found in BOK 2002]	III.B.4. Software process modeling (e.g., entry and exit criteria, task definition, feedback loops)
II.B.3. Cost of Quality (COQ)	IV.B.3. Cost of Quality categories (e.g., prevention, appraisal, internal failure, external failure)

Copyright © 2002, Software Quality Methods, LLC.

CSQE BOK 2002	CSQE BOK 1996
II.B.4. Quality data tracking	II.B.1. Scope and objectives of quality information systems II.B.2. Categories of quality data and their uses II.B.4. Techniques for implementing information systems to track quality-related data II.B.5. Records and data collection, storage, maintenance, and retention
II.B.5. Problem reporting and corrective action procedures	II.B.3. Problem reporting and corrective action procedures (e.g., software defects, process nonconformances)
II.B.6. Quality improvement processes	III.A.3. Defect prevention, detection, and removal methods I.B.3. Prevention vs. Detection philosophies III.B.6. Barriers to the implementation or success of quality improvement efforts and quality systems
II.C. Audits	VII. Software Audits (16 Questions)
II.C.1. Program development and administration - Identify roles and responsibilities for various audit participants, including team leader, team members, auditee, auditor, etc.	VII.C. Audit Planning VII.C.1. Audit team member responsibilities VII.C.2. Management (auditee and auditor) responsibilities concerning audits VII.C.3. Hosting external audits
II.C.1. Program development and administration - Identify roles and responsibilities for various audit participants, including team leader, team members, auditee, auditor, etc.	VII.C.4. Audit program development and administration VII.B.4. Audit process (e.g., objectives, criteria, techniques and methods, participant roles)
[No similar topic found in BOK 2002]	VII.C.5. Auditing requirements (e.g., industry and government standards)
II.C.2 Audit preparation and execution	VII.A. Audit Types VII.B. Audit Methodology
II.C.2 Audit preparation and execution – (a) Define and distinguish between various audit types, including process, compliance, supplier, system, etc.	VII.A.1. Performing internal audits (e.g., quality system, product, process, project, customer) VII.A.2. Performing external audits (e.g., supplier qualifications, certification of supplier systems, auditing testing done by independent agencies) VII.B.1. Purpose, objectives, frequency, and criteria of the overall audit program and individual software audits
II.C.2 Audit preparation and execution – (b) Define and describe various steps in the audit process, from scheduling the audit through the closing meeting and subsequent follow-up activities.	VII.B.3. Audit steps (planning, preparation, execution reporting, corrective action, verification, follow-up)

Copyright © 2002, Software Quality Methods, LLC.

CSQE BOK 2002	CSQE BOK 1996
II.C.2 Audit preparation and execution – (c) Define and identify various tools and procedures used in conducting audits.	VII.B.2. Procedures, tools, and issues related to conducting audits in specific areas (e.g., software development, project management, configuration management)
II.C.3 Audit reporting and follow-up – Identify, describe, and apply the steps of audit reporting and follow up, including the need for and verification of corrective action.	VII.B.3. Audit steps (planning, preparation, execution reporting, corrective action, verification, follow-up)
III. Software Engineering Processes (26 Questions)	**III. Software Processes (24 Questions)**
III.A. Environmental Conditions	III.A. Development and Maintenance Methods
	III.A.1. Software development procedures
III.A.1. Life cycles	III.A.2. Life cycle or process models, e.g. waterfall, spiral, etc.
III.A.2. Systems Architecture	[New topic in BOK 2002]
III.B. Requirements Management	[New topic in BOK 2002]
III.B.1. Requirements prioritization and evaluation	VI.C.5 Methods for evaluating requirements for correctness, consistency, completeness, and testability
III.B.2. Requirements change management	VI.C.9 Methods for assessing all proposed modifications, enhancements, or additions to determine the effect each change will have on the system
III.B.3. Bi-directional requirements traceability	VI.B.13 Traceability mechanisms (e.g., system verification diagrams) VI.C.3 Methods for evaluating software life cycle products and processes (e.g., physical traces, documentation, source code, plans, test and audit results) to determine if user needs and project objectives are satisfied VI.C.4 Methods for performing requirements traceability (e.g., requirements to design, design to code)
III.C. Requirements Engineering	III.A. Development and Maintenance Methods
III.C.1. Requirement types III.C.3. Requirements analysis and modeling	III.A.4. Requirement analysis and specification methods (e.g., data flow diagram, entity-relationship diagram)
III.C.2. Requirements elicitation	III.A.5. Requirements elicitation methods and techniques (e.g., Quality Function Deployment, Joint Application Development, context-free questioning, needs analysis, focus groups)
III.C.4. System and Software Requirements Specifications	[New topic in BOK 2002]

Copyright © 2002, Software Quality Methods, LLC.

CSQE BOK 2002	CSQE BOK 1996
III.D. Analysis, Design, and Development Methods and Tools	III.A. Development and Maintenance Methods
III.D.1. Software design methods Define and use various design methods, including object-oriented analysis and design (OOAD), structured analysis and design (SAD), unified modeling language (UML), etc. Identify the steps used in program design and explain their uses.	III.A.6. Software design methods (e.g., structured analyses and design, Jackson Design method, Warnier-Orr method, object-oriented) - Information Domain - Structured Analysis - Warnier-Orr method - Jackson Design method - Object Oriented
III.D.2. Types of software reuse	III.A.7. Issues related to reuse, re-engineering, and reverse engineering
III.D.3. Clean room and other formal methods	III.A.3. Defect Prevention Methods – Clean room and defect prevention
III.D.3. Clean room and other formal methods III.D.4. Software development tools	III.B.5 Software environments (e.g., development methodologies, tools, data, infrastructure)
III.E. Maintenance Management	III.A. Development and Maintenance Methods
III.E.1. Maintenance types III.E.2. Operational maintenance	III.A.8. Maintenance processes (e.g., re-engineering, reverse engineering, change management, retirement)
III.E.1. Maintenance types	IV.A.4. Maintenance types (e.g., corrective, adaptive, perfective)
III.E.2. Operational maintenance	IV.A.5. Software maintenance and adaptability program planning
IV. Program and Project Management (24 Questions)	**IV. Software Project Management 16 Questions)**
IV.A. Planning	IV.A. Planning
IV.A.1. Project planning elements	IV.A.1. Project planning factors (e.g., quality, costs, resources, deliverables, schedules)
IV.A.2. Goal-setting and deployment	IV.A.3. Goal-setting and deployment methodologies
IV.A.3. Project planning tools	IV.A.2. Project planning methods and tools (e.g., work breakdown structures, documentation, forecasting, estimation) Estimating, WBS, Sizing V.C.3. Commonly used metrics (e.g. complexity, reliability, defect density, phase containment, size) - Boehm – Construction Cost Model (COCOMO) - Albrecht Function Points
IV.A.3. Project planning tools (e.g. planning, cost estimating) IV.B.3. Tracking elements and methods (e.g. tracking, reporting)	IV.C.1. Project management tools (e.g., planning, tracking, cost estimating, reporting)
IV.A.4. Cost and value data	[New topic in BOK 2002]

Copyright © 2002, Software Quality Methods, LLC.

CSQE BOK 2002	CSQE BOK 1996
IV.B. Tracking and Controlling	IV.B. Tracking
IV.B.1. Phase transition control techniques IV.B.4. Project reviews	IV.B.1. Phase transitioning control techniques (e.g., reviews and audits, Gantt Charts, PERT, budgets)
IV.B.2. Interpreting and reporting cost of quality (COQ) data	IV.B.2. Methods of collecting Cost of Quality data IV.C.2. Methods of reporting Cost of Quality data
IV.B.3. Tracking elements and methods	IV.B.4. Cost, progress, and deliverable tracking (e.g., status reports, life cycle phase reports)
IV.C. Risk Management	
IV.C.1. Risk management planning methods	I.D.3. Risk management (e.g., project, product, process)
[No similar topic found in BOK 2002]	I.D.4. Problem-solving processes
IV.C.2. Risk probability	[New topic in BOK 2002]
IV.C.3. Product release decisions	IV.C.3. Trade-off involved in product release decisions (e.g., cost, quality, schedule, customer, test sufficiency, stability)
IV.C.4. Software security, safety, and hazard analysis issues	II.A.4. Issues related to software security, safety, and hazard analysis
V. Software Metrics, Measurement, and Analytical Methods (24 Questions)	**V. Software Metrics, Measurement and Analytical Methods (24 Questions)**
V.A. Metrics and measurement theory	V.A. Measurement Theory
[No similar topic found in BOK 2002]	V.A.1. Goal, question, metric paradigm for selecting metrics
V.A.2. Basic measurement theory and techniques	V.A.2. Basic measurement theory and techniques
V.A.1. Definitions	V.A.3. Definitions of metrics and measures
[No similar topic found in BOK 2002]	V.A.4. Designing measures
V.A.3. Psychology of metrics	V.A.5. Psychology of metrics (e.g., how metrics affect people and how people affect metrics)
V.B. Process and product measurement	V.C. Software Measurement
[No similar topic found in BOK 2002]	V.C.1. Prediction techniques of future maintainability
V.B.1. Process, product, and resource metrics	V.C.2. Applications of measurements to process, product, and resources
V.B.2. Commonly used metrics	V.C.3. Commonly used metrics (e.g. complexity, reliability, defect density, phase containment, size) - Overview - Methodology - Halstead SW Science - Size – Lines of Code - DeMarco Bang
V.B.3. Software quality attributes	V.C.4. Software quality attributes (e.g., reliability, maintainability, usability, testability)
V.B.4. Defect detection effectiveness measures	V.C.5. Defect detection effectiveness (e.g., cost yield, escapes, customer impact)

Copyright © 2002, Software Quality Methods, LLC.

CSQE BOK 2002	CSQE BOK 1996
V.B.5. Program performance and process effectiveness	[New topic in BOK 2002]
V.C. Analytical techniques	V.B. Analytical Techniques
V.C. Analytical Techniques	I.D. Problem-Solving Tools and Processes
V.C.1. Data integrity	V.B.1. Issues involving data integrity, completeness, accuracy, and timeliness
V.C.2. Quality tools	V.B.2. Basic statistical concepts and graphical techniques for analysis and presentation of software data (e.g., distributions, confidence intervals, statistical inference) V.B.3. Quality analysis tools (Pareto chart, Flow charts, Control charts, Check sheets, Scatter diagrams, histograms)
V.C.2. Quality tools	I.D.1. Root cause analysis I.D.2. Tools (e.g., affinity diagram, tree diagram, matrix diagram, interrelationship digraph, prioritization matrix, activity network diagram)
V.C.3. Sampling theory and techniques	V.B.4. Sampling theory and techniques as applied to audits, testing, and product acceptance
VI. Software Verification and Validation (V&V) (24 Questions)	**VI. Software Inspection, Testing, Verification and Validation (24 Questions)**
VI.A. Theory	VI.C. Verification and Validation (V & V)
VI.A.1. V&V planning procedures and tasks – (a) Identify and select various methods for verification and validation, including static analysis, structural analysis, mathematical proof, simulation, etc.	VI.C.1 V & V planning procedures
VI.A.1. V&V planning procedures and tasks – (b) Identify and analyze which tasks should be iterated as a result of proposed or completed modifications.	VI.C.10 Methods for determining which V&V tasks should be iterated based upon proposed modifications and enhancements
VI.A.2. V&V program	VI.C.2 Methods for reviewing V & V program (e.g., technical accomplishments, resource utilization, future planning, risk management, impact analysis of proposed changes)
VI.A.3. Evaluating software products and processes	VI.C.3 Methods for evaluating software life cycle products and processes (e.g., physical traces, documentation, source code, plans, test and audit results) to determine if user needs and project objectives are satisfied
VI.A.4. Interfaces	VI.C.6 Methods for evaluating interfaces with hardware, user, operator, and other software applications
VI.B. Reviews and inspections	VI.A. Inspection

Copyright © 2002, Software Quality Methods, LLC.

CSQE BOK 2002	CSQE BOK 1996
VI.B.1. Types	VI.A.1. Inspection types (e.g., peer reviews, inspections, walk-throughs) VI.A.4. Methods for reviewing inspection efforts (e.g., technical accomplishments, resource utilization, future planning)
VI.B.2. Items	[New topic in BOK 2002]
VI.B.3. Processes	VI.A.2. Inspection process (e.g., objectives, criteria, techniques and methods, participant roles)
VI.B.4. Data collection, reports, and summaries	VI.A.3. Inspection data collection, reports, and summaries
VI.C. Test planning and design	VI.B. Testing
VI.C.1. Types of tests	VI.B.1 Types of tests (e.g., functional, performance, usability, stress, regression. real-time response) VI.B.2 Test Levels (e.g., unit, integration, system, field)
VI.C.2. Test tools VI.C.6. Test environments	VI.B.7 Test environments (e.g., tools and methodologies, test libraries, drivers/stubs, equipment compatibility test laboratories)
VI.C.3. Test strategies	VI.B.3 Test strategies (e.g., top down, bottom up, automated testing, I/0 first, beta testing, black box, white box)
VI.C.4. Test design	VI.B.4 Test design (e.g., test cases, fault insertion and error handling, equivalence class partitioning, usage scenarios, customer defect reports)
VI.C.5. Test coverage of specifications	VI.B.6 Test coverage of specifications (e.g., functions, states, data and time domains, localization, internationalization)
VI.C.7. Supplier components and products	VI.B.11 Methods for testing supplier components and products
VI.C.8. Test plans	VI.C.7 Methods for evaluating test plans (e.g., system acceptance, validation) to determine if software satisfies software and system objectives
VI.D. Test execution and evaluation	VI.B. Testing
VI.D.1. Test implementation	VI.B.9 Test management (e.g., scheduling, freezing, resources, dependencies, analysis of test results)
VI.D.2. Test documentation	VI.B.8. Test documentation (e.g., test plans, logs, test designs, defect recording, test reports)
VI.D.3. Test reviews	VI.B.10 Methods for reviewing testing efforts (e.g., technical accomplishments, resource utilization, future planning, risk management)

Copyright © 2002, Software Quality Methods, LLC.

CSQE BOK 2002	CSQE BOK 1996
VI.D.4. Code coverage metrics	V.C.3. Commonly used metrics (e.g. complexity, reliability, defect density, phase containment, size) - McCabe's Cyclomatic Complexity VI.B.5 Test coverage of code (e.g., branch-to-branch, path, individual predicate, data)
VI.D.5. Customer deliverables	VI.B.12 Methods for testing the accuracy of customer deliverables including user documentation, marketing and training materials
VI.D.6. Severity of anomalies	VI.C.8 Methods for evaluating the severity of anomalies in software operation
VII. Software Configuration Management (16 Questions)	**VIII. Software Configuration Management (16 Questions)**
VII.A. Configuration infrastructure	[New topic in BOK 2002]
VII.A.1. Configuration management	[New topic in BOK 2002]
VII.A.2. Library/repository processes	VIII.A.3. Library control procedures
VII.A.3. Defect tracking and library tools	VIII.A.5. Configuration management tools
VII.B. Configuration identification	VIII.A. Planning and Configuration Identification
VII.B.1. Configuration items VII.B.2. Baselines	VIII.A.1. Technical and managerial factors that guide software product partitioning into configuration items and components
VII.B.3. Configuration identification methods VII.B.2. Baselines	VIII.A.4. Configuration identification methods (e.g., schemes, reidentification, naming conventions, versions and serialization, baselines)
VII.B.4 Software builds	VIII.B.2. Patching issues (e.g., testing, traceability, source updating)
VII.C. Configuration control	VIII.B. Configuration Control, Status Accounting, and Reporting
VII.C.1. Item and baseline control	[New topic in BOK 2002]
VII.C.2. Proposed modifications	VI.C.9 Methods for assessing all proposed modifications, enhancements, or additions to determine the effect each change will have on the system VIII.B.3. Trade-offs between cost, cycle time, and integrity of software product and rigor and formality of change control VIII.B.6. Techniques for assessing impacts of proposed software changes
VII.C.3. Review and configuration control boards (CCBs)	VIII.B.5. Software configuration/change control board processes
VII.C.4. Concurrent development	[New topic in BOK 2002]
VII.C.5. Traceability	[New topic in BOK 2002]
VII.C.6. Version control	VIII.B.4. Source and object code control procedures
VII.C.7. Configuration item interfaces	[New topic in BOK 2002]
VII.D. Configuration Status Accounting	VIII.B. Configuration Control, Status Accounting, and Reporting

Copyright © 2002, Software Quality Methods, LLC.

CSQE BOK 2002	CSQE BOK 1996
VII.D.1. Status reporting	[New topic in BOK 2002]
VII.D.2. Changes to configuration items and baselines	[New topic in BOK 2002]
VII.D.3. Documentation control	VIII.B.1. Documentation control (e.g., issuing, approval, storage, retrieval, revision)
VII.E. Configuration Audits	[New topic in BOK 2002]
VII.E.1. Functional configuration audit VII.E.2. Physical configuration audit	VII.A.3. Functional and physical configuration audits - Functional - Physical
VII.F. Release and distribution issues	[New topic in BOK 2002]
VII.F.1. Product release process issues	VIII.A.2. Release process issues (e.g., supporting multiple versions, feature vs. corrective releases, hardware and software dependencies)
VII.F.2. Packaging, production, and distribution	[New topic in BOK 2002]

Copyright © 2002, Software Quality Methods, LLC.

Software Quality Engineer Certification (CSQE) References[8]

The technical content and test questions for the CSQE are derived from the listed reference materials. The list shows the references by subject area. References are repeated for each relevant area. A list of general reference material follows the last subject area.

General Knowledge, Conduct, and Ethics

ANSI/ISO/IEE TICKIT Guidelines

Humphrey, Watts. Managing the Software Process, Addison-Wesley, 1989.

Juran, Joseph M., Juran on Quality by Design: The New Steps for Planning Quality into Goods and Services, New York: McGraw-Hill1992.

Juran, Joseph M., The Quality Control Handbook, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., Juran's Quality Handbook, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., Metrics and Models in Software Quality Engineering, 3rd ed., Kansas: Addison-Wesley, 1995.

Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, Testing Computer Software, 2nd ed., New York: Van Nostrand Reinhold, 1999.

Pressman, Roger S., A Manager's Guide to Software Engineering, New York: McGraw-Hill, 1992. ISBN 0070508208

Pressman, Roger S., Software Engineering: A Practitioner's Approach, 5th ed., New York: McGraw-Hill, 2000.

Russell, J.P., and Terry Regel, After the Quality Audit: Closing the Loop on the Audit Process, 2nd ed., Milwaukee: ASQ Quality Press, 2000.

Scholtes, Peter R., The Team Handbook, 2nd ed., Revised Madison, Wisconsin: Joiner Associates, 1996.

Schulmeyer, G. Gordon, and James I. McManus, Handbook of Software Quality Assurance, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

---

[8] Copyright © 2002 American Society for Quality. All rights reserved.
These books cover significant parts of the Body of Knowledge.
The ASQ Certification Board does not endorse any one particular reference source.

Software Quality Management

Arter, Dennis, Quality Audits for Improved Performance, 2nd ed., Milwaukee: ASQC Quality Press, 1994.

Demarco, Tom, Controlling Software Projects: Management, Measurement, and Estimation, NY: Yourdon Press, 1982. ISBN 0917072324.

Dobbins, James H., Software Quality Assurance and Evaluation ASQC Quality Press, 1990. ISBN 0873890590

Dunn, Robert H., Software Quality: Concepts & Plans, Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN 0138202834

Gryna, Frank M., Quality Planning and Analysis: From Product Development through Use, Boston, MA: McGraw-Hill, 2001.

Humphrey, Watts, Managing the Software Process, Massachusetts: Addison-Wesley, 1989.

Humphrey, Watts, A Discipline for Software Engineering, Massachusetts: Addison-Wesley, 1995. ISBN 0201546108

Juran, Joseph M., and Frank M. Gryna, Quality Planning and Analysis, 3rd ed., New York: McGraw-Hill Publishing Co., 1993. ISBN 0070331839

Juran, Joseph M., The Quality Control Handbook, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., Juran's Quality Handbook, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., Metrics and Models in Software Quality Engineering, 3rd ed., Kansas: Addison-Wesley, 1995.

Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, Testing Computer Software, 2nd ed., New York: Van Nostrand Reinhold, 1999.

Paulk, Mark C., et al. The Capability Maturity Model—Guidelines for Improving the Software Process, Carnegie Mellon University Software Engineering Institute, 1995.

Pressman, Roger S., A Manager's Guide to Software Engineering, New York: McGraw-Hill, 1992. ISBN 0070508208

Pressman, Roger S., Software Engineering: A Practitioner's Approach, 5th ed., New York: McGraw-Hill, 2000.

Russell, J.P., ed., ASQ Quality Audit Division, The Quality Audit Handbook, 2nd ed., Milwaukee: ASQ Quality Press, 2000.

Schulmeyer, G. Gordon, and James I. McManus, Handbook of Software Quality Assurance, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999. P759

Copyright © 2002, Software Quality Methods, LLC.

Software Engineering Processes

Arter, Dennis, Quality Audits for Improved Performance, 2nd ed., Milwaukee: ASQC Quality Press, 1994.

Booch, Grady, Objected-Oriented Analysis and Design with Applications 2nd ed., CA: Benjamin/Cummings Publishing Co., 1994. ISBN 0805353402

Booch, Grady, et al., Unified Modeling Language User's Guide Ed 1 Massachusetts: Addison-Wesley, 1999. ISBN 0201571684

Dunn, Robert H., Software Quality: Concepts & Plans, Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN 0138202834

Dunn, Robert H., and Richard S. Ullman, TQM for Computer Software (System Design and Implementation) 2nd ed.,: McGraw-Hill, 1994. ISBN 0070183147

Humphrey, Watts, Managing the Software Process, Massachusetts: Addison-Wesley, 1989.

Humphrey, Watts, A Discipline for Software Engineering, Massachusetts: Addison-Wesley, 1995. ISBN 0201546108

Juran, Joseph M., The Quality Control Handbook, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., Juran's Quality Handbook, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., Metrics and Models in Software Quality Engineering, 3rd ed., Kansas: Addison-Wesley, 1995.

Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, Testing Computer Software, 2nd ed., New York: Van Nostrand Reinhold, 1999.

McConnell, Steve, Rapid Development, Washington: Microsoft Press, 1996. ISBN 1556159005

Paulk, Mark C., et al. The Capability Maturity Model—Guidelines for Improving the Software Process, Carnegie Mellon University Software Engineering Institute, 1995.

Pressman, Roger S., A Manager's Guide to Software Engineering, New York: McGraw-Hill, 1992. ISBN 0070508208

Pressman, Roger S., Software Engineering: A Practitioner's Approach, 5th ed., New York: McGraw-Hill, 2000.

Rumbaugh, James, et al., Object-oriented Modeling and Design, NJ: Prentice Hall, 1991. ISBN 0136298419

Copyright © 2002, Software Quality Methods, LLC.

Scholtes, Peter R., The Team Handbook, 2nd ed., revised Madison, Wisconsin: Joiner Associates, 1996.

Schulmeyer, G. Gordon, and James I. McManus, Handbook of Software Quality Assurance, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

Tingey, Michael, Comparing ISO 900,0 Malcolm Baldrige, and SEI CMM for Software, Prentice Hall, 1997. ISBN 0133762602

Program and Project Management

Booch, Grady. Object Solutions: Managing the Object-Oriented Project, CA: Addison-Wesley Publishing Co., Inc., 1996. ISBN 0805305947

Dunn, Robert H., Software Quality: Concepts & Plans, Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN 0138202834

Futrell, Robert T., Donald F. Shafer, and Linda I. Shafer, Quality Software Project Management, New Jersey: Prentice Hall, 2002. ISBN 0130912972

Gryna, Frank M., Quality Planning and Analysis: From Product Development through Use, Boston, MA: McGraw-Hill, 2001.

Humphrey, Watts. Managing the Software Process, Massachusetts: Addison-Wesley, 1989.

Juran, Joseph M., and Frank M. Gryna, Quality Planning and Analysis, 3rd ed., New York: McGraw-Hill Publishing Co., 1993. ISBN 0070331839

Juran, Joseph M., The Quality Control Handbook, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., Juran's Quality Handbook, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., Metrics and Models in Software Quality Engineering, 3rd ed., Kansas: Addison-Wesley, 1995.

Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, Testing Computer Software, 2nd ed., New York: Van Nostrand Reinhold, 1999.

King, David, Project Management Made Simple: A Guide to Successful Management of Computer Systems Projects, New Jersey: Yourdon Press, 1992. ISBN 0137177291

Paulk, Mark C., et al. The Capability Maturity Model—Guidelines for Improving the Software Process, Carnegie Mellon University Software Engineering Institute, 1995.

Copyright © 2002, Software Quality Methods, LLC.

Pressman, Roger S., A Manager's Guide to Software Engineering, New York: McGraw-Hill, 1992. ISBN 0070508208

Pressman, Roger S., Software Engineering: A Practitioner's Approach, 5th ed., New York: McGraw-Hill, 2000.

Schulmeyer, G. Gordon, and James I. McManus, Handbook of Software Quality Assurance, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

Software Metrics, Measurement, and Analytical Methods

ANSI/ISO/IEE TICKIT Guidelines

Dunn, Robert H., Software Quality: Concepts & Plans, Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN 0138202834

Dunn, Robert H., Software Defect Removal, NY: McGraw-Hill, 1984. ISBN 0070183139

Dunn, Robert H., and Richard S Ullman, TQM for Computer Software (System Design and Implementation) 2nd ed., McGraw Hill, 1994. ISBN 0070183147

Humphrey, Watts. Managing the Software Process, Massachusetts: Addison-Wesley, 1989.

Humphrey, Watts, A Discipline for Software Engineering, Massachusetts: Addison-Wesley, 1995. ISBN 0201546108

Juran, Joseph M., The Quality Control Handbook, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., Juran's Quality Handbook, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., Metrics and Models in Software Quality Engineering, 3rd ed., Kansas: Addison-Wesley, 1995.

Pressman, Roger S., A Manager's Guide to Software Engineering, New York: McGraw-Hill, 1992. ISBN 0070508208

Pressman, Roger S., Software Engineering: A Practitioner's Approach, 5th ed., New York: McGraw-Hill, 2000.

Scholtes, Peter R., The Team Handbook, 2nd ed., Revised Madison, Wisconsin: Joiner Associates, 1996.

Schulmeyer, G. Gordon, and James I. McManus, Handbook of Software Quality Assurance, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

Copyright © 2002, Software Quality Methods, LLC.

Software Verification and Validation

Dobbins, James H., Software Quality Assurance and Evaluation ASQC Quality Press, 1990. ISBN 0873890590

Dunn, Robert H., Software Defect Removal, NY: McGraw-Hill, 1984. ISBN 0070183139

Dunn, Robert H., and Richard S Ullman, TQM for Computer Software (System Design and Implementation) 2nd ed., McGraw Hill, 1994. ISBN 0070183147

Humphrey, Watts. Managing the Software Process, Massachusetts: Addison-Wesley, 1989.

Juran, Joseph M., The Quality Control Handbook, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., Juran's Quality Handbook, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., Metrics and Models in Software Quality Engineering, 3rd ed., Kansas: Addison-Wesley, 1995.

Kaner, Cem, Jack Falk, and Hung Quoc Nguyen, Testing Computer Software, 2nd ed., New York: Van Nostrand Reinhold, 1999.

Myers, Glenford J., The Art of Software Testing, 1st ed., New York: John Wiley & Sons, 1979. ISBN 0471043281

Myers, Glenford J., Software Reliability: Principles and Practices, NY: Wiley, 1976. ISBN 0471627658

Pressman, Roger S., A Manager's Guide to Software Engineering, New York: McGraw-Hill, 1992. ISBN 007-0508208

Pressman, Roger S., Software Engineering: A Practitioner's Approach, 5th ed., New York: McGraw-Hill, 2000.

Schulmeyer, G. Gordon, and James I. McManus, Handbook of Software Quality Assurance, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

Copyright © 2002, Software Quality Methods, LLC.

Software Configuration Management

ANSI/ISO/IEE TICKIT Guidelines

Dobbins, James H., Software Quality Assurance and Evaluation ASQC Quality Press, 1990. ISBN 0873890590

Dunn, Robert H., Software Quality: Concepts & Plans, Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN 0138202834

Dunn, Robert H., Software Defect Removal, NY: McGraw-Hill, 1984. ISBN 0070183139

Dunn, Robert H., and Richard S Ullman, TQM for Computer Software (System Design and Implementation) 2nd ed., McGraw Hill, 1994. ISBN 0070183147

Humphrey, Watts. Managing the Software Process, Massachusetts: Addison-Wesley, 1989.

Juran, Joseph M., The Quality Control Handbook, 4th ed., New York: McGraw-Hill, 1988.

Juran, Joseph M., Juran's Quality Handbook, 5th ed., New York: McGraw-Hill, 1999.

Kan, Stephen H., Metrics and Models in Software Quality Engineering, 3rd ed., Kansas: Addison-Wesley, 1995.

Osborne, Wilma M., Software Configuration Management: An Overview, Gaithersburg, MD: US Dept. of Commerce, National Computer Systems Lab, NIST Special Publication, 500-161, 1989.

Paulk, Mark C., et al. The Capability Maturity Model—Guidelines for Improving the Software Process, Carnegie Mellon University Software Engineering Institute, 1995.

Pressman, Roger S., A Manager's Guide to Software Engineering, New York: McGraw-Hill, 1992. ISBN 0070508208

Pressman, Roger S., Software Engineering: A Practitioner's Approach, 5th ed., New York: McGraw-Hill, 2000.

Schulmeyer, G. Gordon, and James I. McManus, Handbook of Software Quality Assurance, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

Copyright © 2002, Software Quality Methods, LLC.

General References

Brooks, Frederick P. Jr., The Mythical Man-Month: Essays on Software Engineering, Massachusetts: Addison-Wesley Publishing Co., 1975. ISBN 0201006502

Daughtrey, Taz, ed., Fundamental Concepts for the Software Quality Engineer, ASQ Quality Press, 2002. H1115

DeMarco, Tom, and Timothy Lister, Peopleware: Productive Projects and Teams, NY: Dorset House Publishing Co, 1977. ISBN 0932633056

Escoe, Adrienne, The Practical Guide to People-Friendly Documentation, ASQ Quality Press, 2001.

Lockheed Martin Advanced Concepts Center, Succeeding with the Booch and OMT Methods: A Practical Approach, CA: Addison-Wesley, 1996. ISBN 0805322795

Rumbaugh, James, Ivar Jacobson, and Grady Booch, eds., The Unified Modeling Language Reference Manual, CA: Addison-Wesley, 1999. ISBN 020130998X

Yourdon, Edward, Decline and Fall of the American Programmer, New Jersey: Prentice-Hall, Inc., 1992. ISBN 0132036703

Yourdon, Edward, Rise and Resurrection of the American Programmer, New Jersey: Prentice-Hall, Inc., 1998. ISBN 013956160

Copyright © 2002, Software Quality Methods, LLC.

QW2002 Panel Session 8Q

Dr. Selim Aissi
Senior Security Architect
(Intel Corporation)

Web Services QoS, Reliability, And Security

Panel Moderator: Dr. Selim Aissi

Panelists (alphabetically)

Mr. Don Adam, Principal Security Architect, TIBCO
Mr. Dale Moberg, Chief Architect, Cyclone Commerce
Mr. Mark O'Neill, CTO, Vordel
Mr. Narendra Patil, President & CEO, InfoLead

Panel Description

The team of Web Services leading experts will have a discussion about QoS and Security aspects of Web Services technologies. The Panel discussion will be centered around this list of Web Services Reliability Indicators:

- Security Aspects (authentication, authorization, privacy, integrity)
- Functional, Regression, Stress, and Monitoring testing techniques
- XML Validity
- Messaging Validity (SOAP validity) and security
- Service Discovery Performance
- Web Services Availability
- Web Services Load-Balancing and Workload-Partitioning

About the Panel

Dr. Selim Aissi has been involved in the development of Safety-Critical Embedded Systems in the R&D, military, automotive, and wireless appliances for over twelve years. He worked at the University of Michigan, General Dynamics (M1A2 Abrams Battlefield Tank Division), General Motors (Embedded Controller Excellence Center), Applied Dynamics International, and Intel Corporation. Dr. Aissi has played several management and architecture roles at Intel, and he is currently a Senior Security Architect at Intel's Corporate Technology Group in Hillsboro, Oregon, USA He serves as the Intel representative to several Web Services Standards Working Groups, including the OASIS CPPA and the UDDI Technical Committees. He holds a Ph.D. in Aerospace Engineering from the University of Michigan.

Mr. Don Adam is Principal Security Architect for TIBCO Software. In this role he has oversight for security strategy and direction in all products, services and solutions. He was previously CTO of TriStrata Inc. and Principal Architect - Security and Networks at Sun Microsystems. Prior to Sun he served in the USAF as Chief of Intelligence Systems Engineering, Pacific Air Forces and Technology Advisor - Air Force Intelligence Agency.

Mr. Dale Moberg has been involved in defining or enabling secure, distributed standards or systems for business collaboration since 1993. He has been an editor, chair or author in IETF WGs (EDIINT), RosettaNet, OASIS, ebXML Messaging, Collaboration Profile Agreement, Java Community Process JSR (JAXR and 157), and participates in several W3C working groups. Most of his activities have been concerned with integrating security standards for B2B PKI based security operations, such as digital signatures for nonrepudiation of origin and nonrepudiation of receipt. He is also interested in pragmatic approaches to getting different PKI infrastructures to work together effectively, and to manage the PKI lifecycle. He is currently chair of the OASIS ebXML Joint Committee and also chair of the Collaboration Protocol Profiles and Agreements Technical Committee. He works for Cyclone Commerce as Chief Architect. Prior to 1993, he taught in colleges or universities for 10 years, and spent 7 years as a researcher at a university (object oriented modeling, distributed simulation, and practical uses for knowledge representation).

Mr. Mark O'Neill is the CTO of Vordel, the producer of the VordelSecure product for Web Services security. Mark's security background began in academic cryptography at Trinity College Dublin's School of Mathematics, then continued at Eirtrade Services Ltd, Ireland's largest EDI value-added network, where Mark oversaw the migration of EDI services to the Internet, using crytography toolkits. Mark founded Delphium Technologies to provide security programming to blue-chip clients, and then joined Vordel in May 2000. Mark is published in XML Journal, Web Services Journal, and PriceWaterhouseCoopers Journal of Crypography, and is a frequent speaker at security and XML conferences. Mark is the author of Web Services Security, a book to be published by McGraw-Hill in late 2002.

Mr. Narendra Patil is President & CEO of Infolead, Inc. responsible for strategic management, emerging technology product offerings, product positioning, and R&D efforts. Mr. Patil is very actively involved in the product engineering efforts related to Infolead's ASQ (Automated Software Quality) tools and is the key contributor to the product architecture. Before establishing Infolead, Mr. Patil came from two premier organizations within Sun Microsystems, Inc., the JavaSoft and Developer Tools divisions, where he initiated and promoted the development of key ASQ tools, which are being used very extensively. Mr. Patil holds a BS in Computer Science from India and MBA from University of Phoenix, USA.

QW2002 Paper 9Q

Danny Faught
(Tejas Software Consulting)

A Survey of Freeware Test Tools

Key Points

- Introduces the concept of freeware licensing.
- Surveys broad categories of freeware test tools.
- Demonstrates a handful of freeware tools in action.

Presentation Abstract

A survey of the many freeware test tools that are available, including live demos of the most promising freewar tools.

About the Author

Danny R. Faught is an independent software quality consultant, calling his practice Tejas Software Consulting. He has been in the industry for ten years. He has a BS in Computer Science from the University of North Texas. He is a senior member of the American Society for Quality, and is also a member of the Project Management Institute and the Fort Worth Chamber of Commerce. Danny is the maintainer of testingfaqs.org and is cofounder of the swtest-discuss mailing list. He serves on the Practicality Gauntlet for STQE magazine. He speaks and writes frequently on software quality topics.

A Survey of Freeware Test Tools

Danny R. Faught
Tejas Software Consulting
www.tejasconsulting.com

Quality Week 2002

Objectives

- Familiarize you with the types of test tools that are available as freeware.
- Demonstrate a few of the tools in action.
- Point you to where you can find the tools and use them for yourselves.

© 2002 Danny R. Faught

Topics du jour

- What is freeware?
  - allpairs demo
- Scripting languages
  - stress_driver demo
- Static analysis
  - sloccount demo
- Unit testing
  - MinUnit example
- Wrapup

© 2002 Danny R. Faught

What is freeware?

© 2002 Danny R. Faught

“Freeware” is a broad term

- Freeware generally refers to any software that can be used free of charge
  - Maybe binary-only, no source code available, no support or documentation if there are problems, no rights to redistribute
  - Maybe open source, fully documented, with rights granted to modify the source, redistribute it, and even sell it, with both commercial support and a broad user community that can help with problems

© 2002 Danny R. Faught

Types of licenses

- No license specified
- Public domain
- Free (including copyleft, and GPL-compatible)
- Open Source
- Custom
- Not freeware: shareware

© 2002 Danny R. Faught

Free software: four essential freedoms

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. (freedom 3). Access to the source code is a precondition for this.

(http://www.gnu.org/philosophy/free-sw.html)

© 2002 Danny R. Faught

Sample licenses

- GNU GPL & LGPL
- BSD template, with or without advertising clause
- X11
- Artistic
- Apache License
- IBM Public License
- Mozilla Public License
- MIT License

© 2002 Danny R. Faught

allpairs demo

- Allpairs is a test design tool that helps you design tests for pair-wise combinations of features or configurations.
- Written by James Bach, Satisfice, Inc.
- Available at http://satisfice.com/testmethod.shtml

© 2002 Danny R. Faught

Freeware references

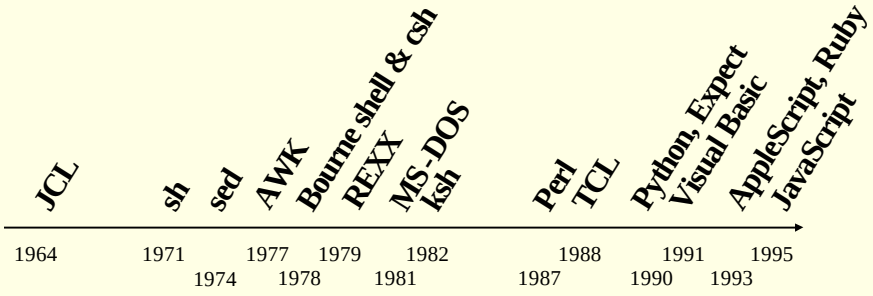
- "What Flavor is Your Freeware?" http://tejasconsulting.com/newsletter/2002June-July.html#feature
- *Free as in Freedom: Richard Stallman's Crusade for Free Software*, Sam Williams, 2002, ISBN 0-596-00287-4
- *The Cathedral & the Bazaar*, Eric S. Raymond, 1999, ISBN 1-56592-724-9
- *Open Sources*, Chris DiBona, Sam Ockman, & Mark Stone, eds., 1999, ISBN 1-56592-582-3
- Open Source Approved licenses http://opensource.org/licenses/
- GNU's "Various Licenses and Comments about Them" http://www.fsf.org/licenses/license-list.html

© 2002 Danny R. Faught

Scripting languages

© 2002 Danny R. Faught

Script language timeline



Includes information from "History of Scripting," http://scriptics.com/doc/scriptHistory.html

© 2002 Danny R. Faught

Proprietary scripting languages

- JCL
- MS-DOS batch files
- Visual Basic
- AppleScript

Now on to the freeware...

(Many of those that follow have both freeware and proprietary incarnations.)

© 2002 Danny R. Faught

Unix shells

- sh
  - Primitive versions appeared on an early version of Unix
  - The Bourne shell (by Stephen Bourne) appeared in 1978
- Alternatives have been popping up ever since, including:
  csh, ksh, POSIX shell, bash, zsh, tcsh, rc, with numerous variations of each

© 2002 Danny R. Faught

Sed & awk

- Sed is an abbreviation for “stream editor,” an evolution of the grep utility that can find lines of text and also modify them
- Awk is named for its authors – Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger.
  - A small, general purpose scripting language
  - One of the main precursors of Perl (along with sed)

© 2002 Danny R. Faught

REXX

- An acronym for “Restructured Extended Executor Language”
- An ANSI-standard procedural language that was also designed to be a general-purpose macro language for applications
- Seems to have a big following on AmigaDOS and OS/2

© 2002 Danny R. Faught

Perl

- "Practical Extraction and Report Language," or perhaps, "Pathologically Eclectic Rubbish Lister"
- Combines some of the best features of C, sed, awk, and sh
- My personal favorite!

© 2002 Danny R. Faught

TCL & Expect

- TCL stands for "tool command language" and is pronounced "tickle"
  - The Tk graphics extension is probably the most popular extension for TCL
- Expect, a TCL extension, is a tool for automating and testing interactive applications
  - Named after the "expect" command used in early modem scripts

© 2002 Danny R. Faught

Recent additions

- Python, Ruby, JavaScript, and many others
- I'm not familiar with these, so they're left as an exercise for the reader
- Python and Ruby have been touted as cleaner languages than Perl, so they may be worth a look

© 2002 Danny R. Faught

What's missing?

- What scripting languages do you use that we haven't discussed yet?

© 2002 Danny R. Faught

stress_driver demo

- Stress_driver is a generic stress testing tool written in Perl, allowing you to schedule multiple invocations of a test program.
- Written primarily by Danny Faught, Tejas Software Consulting, and owned by Hewlett-Packard.
- Available at http://tejasconsulting.com/tools/sd.html

© 2002 Danny R. Faught

Language resources (1 of 2)

- "History of Scripting," http://scriptics.com/doc/scriptHistory.html
- Comp.unix.shell FAQs, http://faqs.org/faqs/by-newsgroup/comp/comp.unix.shell.html
- *Learning the bash Shell*, 2nd Edition, ISBN 1-56592-347-2
- *Effective awk Programming*, 3rd Edition, Arnold Robbins, 2001, ISBN 0-596-00070-7
- *sed & awk*, 2nd Edition, Arnold Robbins, Dale Dougherty, 1997, ISBN 1-56592-225-5
- awk FAQ, http://www.faqs.org/faqs/computer-lang/awk/faq/
- sed FAQ, http://www.faqs.org/faqs/editor-faq/sed/index.html
- *The Rexx Language*, Michael Cowlishaw, 1990, ISBN 0137806515
- Rexx FAQ, http://www.mindspring.com/~dave_martin/RexxFAQ.html

© 2002 Danny R. Faught

Language resources (2 of 2)

- http://www.perl.com (and many others)
- *Programming Perl*, 3rd Edition, Larry Wall, Tom Christiansen, Jon Orwant, ISBN 0-596-00027-8 (and many others from O'Reilly)
- comp.lang.tcl FAQ, http://www.geocities.com/lvirden/tcl-faq/
- *Exploring Expect*, Don Libes, 1995, ISBN 1-56592-090-2
- The Expect Home Page, http://expect.nist.gov/
- Python, http://www.python.org/
- Ruby, http://www.ruby-lang.org/
- comp.lang.javascript meta FAQ, http://faqs.org/faqs/computer-lang/java/javascript/

© 2002 Danny R. Faught

Static analysis

© 2002 Danny R. Faught

Web static analysis

- Tidy
  - Fixes mistakes in html code automatically and tidies up sloppy editing.
- W3C HTML Validation Service
  - A service that checks documents like HTML and XHTML for conformance to W3C Recommendations and other standards.
- Link Valet
  - WWW Link checker.

© 2002 Danny R. Faught

Static error checking (Java)

- JiveLint
  - A command line tool employing static analysis on your JAVA source code.
- Jlint
  - Jlint will check your Java code and find bugs, inconsistencies and synchronization problems by doing data flow analysis and building the lock graph.

© 2002 Danny R. Faught

Static error checking (C)

- PScan
  - Scans C source files for problematic uses of printf style functions.
- CQUAL
  - A type-based analysis tool that provides a lightweight, practical mechanism for specifying and checking properties of C programs.
- BLAST
  - A software model checker for C programs.

© 2002 Danny R. Faught

Code metrics

- sclc
  - Counts lines of code (and comments, and non-comment source lines, and assembly-equivalent source lines) for Ada, Assembly, Awk, C, C++, Eiffel, Java, Lisp, Pascal, Perl, Tcl, shell, and make.
- metrics
  - Tools to generate Halstead, McCabe, and LOC metrics for C code.

© 2002 Danny R. Faught

sloccount demo

- sloccount
  - A set of tools for counting physical Source Lines of Code (SLOC) in Ada, Assembly, awk, Bourne shell and variants, C, C++, C shell, COBOL, C#, Expect, Fortran, Haskell, Java, lex/flex, LISP/Scheme, Makefile, Modula-3, Objective-C, Pascal, Perl, PHP, Python, Ruby, sed, SQL, TCL, Yacc/Bison

© 2002 Danny R. Faught

Static analysis resources

- Tidy - http://tidy.sourceforge.net/,
  - also Tidy Online - http://valet.htmlhelp.com/tidy/
- W3C HTML Validation Service - http://validator.w3.org/
- Link Valet - http://www.htmlhelp.com/tools/valet/
- JiveLint - http://www.bysoft.se/sureshot/javalint/
- Jlint - http://www.artho.com/jlint/
- PScan - http://www.striker.ottawa.on.ca/~aland/pscan/
- CQUAL - http://www.cs.berkeley.edu/~jfoster/cqual/
- BLAST – http://www-cad.eecs.berkeley.edu/~rupak/blast/
- sclc- http://www.enteract.com/~bradapp/clearperl/sclc-cdiff.html
- metrics - http://sources.isc.org/devel/tools/metrics.txt
- sloccount - http://www.dwheeler.com/sloccount/

© 2002 Danny R. Faught

Unit testing

© 2002 Danny R. Faught

Lots of unit test tools out there

- Wow - I found 61 different freeware unit test frameworks!
- I found tools for these languages:
  - AppleScript, Ada, C, C++, C#, CA-OpenROAD, Curl, LISP, .Net, Delphi, Eiffel, Flash, GemStone, Haskell, HTML, Jade, Java, JavaScript, K, KSQL, Macromedia Director, Objective-C, Ocaml, Perl, PHP, PowerBuilder, Python, REALbasic, REBOL, Ruby, Scheme, Shell, Suneido, Visual Basic, XML.

© 2002 Danny R. Faught

Simple and simplest

- Many unit testing frameworks are based on JUnit for Java.
- The frameworks tend to be fairly simple. Here is one that is perhaps the simplest of them all...

© 2002 Danny R. Faught

MinUnit for C

```
/* file: minunit.h */
 #define mu_assert(message, test)
  do { if (!(test)) return message; } while (0)
 #define mu_runtest(test)
  do { char *message = test(); tests_run++; \
  if (message) return message; } while (0)
 extern int tests_run;
```

- by John Brewer, http://www.jera.com/techinfo/jtns/jtn002.html

© 2002 Danny R. Faught

MinUnit example (1 of 2)

```
#include "minunit.h"

int tests_run = 0;

int foo = 7;
int bar = 4;

static char * test_foo() {
 mu_assert("error, foo != 7", foo == 7);
 return 0;
}
```

© 2002 Danny R. Faught

MinUnit example (2 of 2)

```
static char * test_bar() {
     mu_assert("error, bar != 5", bar == 5);
     return 0;
 }

 static char * all_tests() {
     mu_run_test(test_foo);
     mu_run_test(test_bar);
     return 0;
 }
...
```

© 2002 Danny R. Faught

Unit testing references

- See http://xprogramming.com/software.htm for a long list of pointers to unit test frameworks (some come with no documentation)
- The seminal JUnit article, “JUnitTest Infected: Programmers Love Writing Tests”, http://junit.sourceforge.net/doc/testinfected/testing.htm

© 2002 Danny R. Faught

Wrapup...

© 2002 Danny R. Faught

Other tool categories...

- Security
- Coverage analysis
- GUI testing
- Test implementation/runtime tools
- Test suites
- Bug tracking

© 2002 Danny R. Faught

General references

- Testing Tools Supplier List, will eventually list hundreds of freeware test tools, mailing list for periodic announcements about new items- http://testingfaqs.org/
- free-testing-tools mailing list, not much activity yet - http://groups.yahoo.com/group/free-testing-tools/
- TestToolWiki, also not much there yet- http://www.satisfice.com/scribble/

© 2002 Danny R. Faught

Comments welcome

- Comments and questions about this presentation are welcome!
- Contact me at:

  faught@tejasconsulting.com

  +1 817 294 3998
- Sign up for my free email newsletter – see the "News" section of my web site, http://tejasconsulting.com

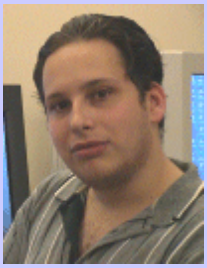
© 2002 Danny R. Faught

Thanks for listening!

- Please fill out the evaluations.

-Danny Faught,
Software Alchemist

© 2002 Danny R. Faught



QW2002 Paper 2T1

Mr. Ibrahim El-Far (Flordia Institute of Technology), Ms. Florence Mottay (J.D. Edwards), & Dr. Herbert Thompson (System Integrity)

Experiences in Testing Pocket PC Applications

Key Points

- Learn key concepts in model-based testing
- a model-based technique to Windows CE Software
- Gain insight into our experience with state models

Presentation Abstract

Promising model-based approaches have emerged in the field of software testing during the past decade or so. In this presentation, we focus on our recent experiences with using finite state machines to test a host of applications created by Microsoft for the Pocket PC platform. Creating a state model of each of these applications facilitated the tasks of building test automation and verifying test results. We explain the process we followed, present our results, and share some of the lessons we learned from our

About the Author

Ibrahim K. El-Far is a doctoral candidate in computer sciences at the Florida Institute of Technol-ogy under the academic supervision of professor James A. Whittaker. He has a Bachelor of Sci-ences and a Master of Sciences in Computer Science from the American University of Beirut, Beirut, Lebanon and the Florida Institute of Technology, Melbourne, Florida, USA, respectively. His interests are in investigating software models for testing, test automation and tools, adequacy criteria, test cost and effectiveness, and software testing education. He has over five years of ex-perience in model-based testing using finite state machines at the Center for Software Engineer-ing Research at Florida Tech, where he has supervised the development of experimental model-based testing tools, advised model-based testing groups, and taught model-based testing in vari-ous formats to a variety of students. You can contact Ibrahim at ielfar@acm.org, visit for more information, or write him at the Software Engineering Program, Computer Sciences Department, Florida Institute of Technology, 150 West University Boulevard, Melbourne, Florida 32901 USA.

Florence E. Mottay is a quality engineer at J.D. Edwards. She received her Master of Science in mathematics from the Florida Institute of Technology. In previous years, Ms. Mottay worked as a research assistant at the Center for Software Engineering Research, Florida Institute of Technol-ogy, Melbourne. Her research interests are in software testing, formal languages, mathematical models, and e-commerce. She received awards for excellence in mathematics by the United States Achievement Academy (1997) and for academic excellence by the American Association of University Women (1998).

Herbert H. Thompson is the director of security technology at System Integrity. He received his doctoral degree in mathematics from the Florida Institute of Technology in 2002. In previous years, Dr. Thompson has worked for Microsoft Corporation as a test engineer. His research inter-ests are in software engineering, security, and applying mathematics to computer science prob-lems. You can contact him at hethomps@fit.edu or visit .

Experiences in Testing the Pocket PC

Ibrahim K. El-Far
Florida Institute of Technology

Herbert H. Thompson
System Integrity

Florence E. Mottay
J.D. Edwards

 © 2002 The authors, all rights reserved.

Overview

- Scope of the Project
- Introduction to Model Based testing
- Case Study: the Pocket Inbox
- Conclusions

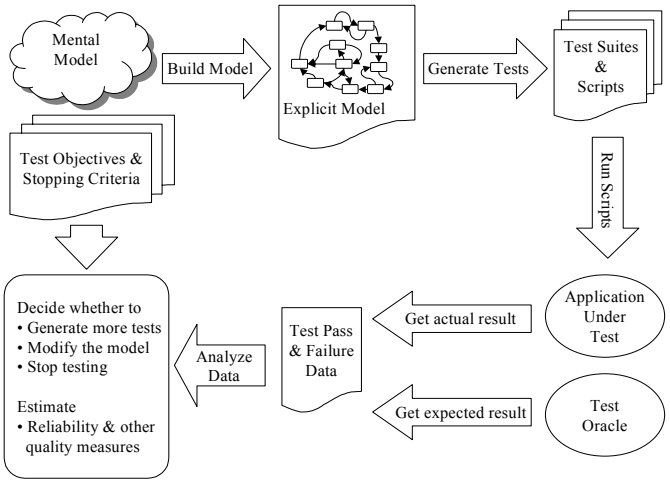
 © 2002 The authors, all rights reserved.

Project Scope

- Pocket PC : Operating platform developed by Microsoft for handheld devices.
- Contracted to develop software models for five applications: Contacts, Calendar, Inbox, Pocket Word and Connectivity Manager.

 © 2002 The authors, all rights reserved.

Model Based Testing: The Big Picture



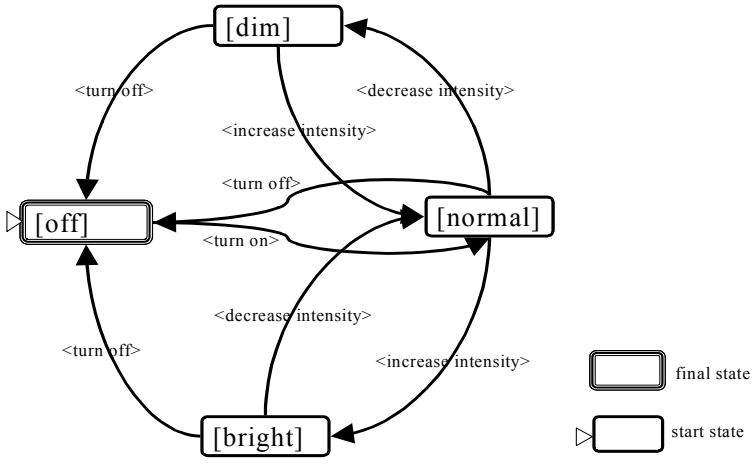
 © 2002 The authors, all rights reserved.

Modeling with Finite State Machines

- A state is a condition of the software where a specific set of inputs can be applied.
- Two states are considered equal if the same inputs can be applied and they produce the same result.

 © 2002 The authors, all rights reserved.

Building Models : An Example



 © 2002 The authors, all rights reserved.

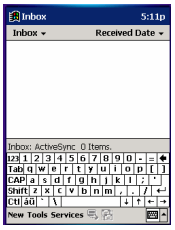
Case Study: the Pocket Inbox

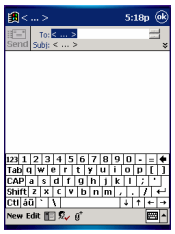
- General Information
  - 5,000 transitions
  - 1,500 states
  - Most complex application we tested
  - Design choices
    - Limited number of messages
    - Input peculiarities

 © 2002 The authors, all rights reserved.

Case Study: Exploration

- Explore the application in order to discover and build a mental representation of its functionality
  - For demonstration purposes, we only use a partial model





 © 2002 The authors, all rights reserved.

Case Study: Inputs

- Identify all user inputs
  1. New: to go to a new message window
  2. Ok: to leave the New Message screen and save the current message
  3. Send: to leave the New Message screen and send the current message
  4. Space: space character
  5. AlphanumericChar: any alphanumeric character

 © 2002 The authors, all rights reserved.

Case Study: Input Applicability

- Identify the individual characteristics used to define the states of the application?

  1. **New**: this input is applicable at any time. The user can press the "New" button when the general inbox screen is showing, when the window is New Message and whether he/she has entered text inside a New Message window.
  2. **Ok**: this input is applicable when the window is a New Message. Whether the "To" field is empty or not does not make a difference.
  3. **Send**: this input is applicable only when the window is a New Message and the "To" field is not empty.
  4. **Space**: this input is applicable when the window is a New Message.
  5. **AlphanumericChar**: this input is applicable when the window is a New Message

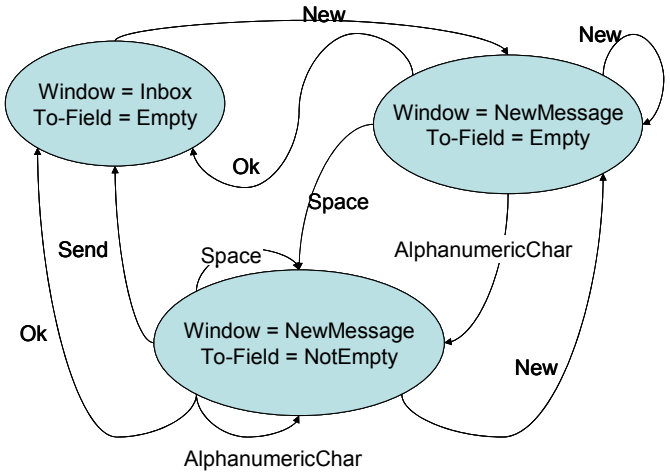
 © 2002 The authors, all rights reserved.

Case Study: State Variables

- Window = Inbox, NewMessage
  - *This operational mode records which window the user is on. For this model, only two windows are considered as possible.*
- To Field = Empty, NotEmpty
  - *This operational mode records if the value of the "To" field is empty or not. It is useful to determine whether the "Send" is enabled or not.*

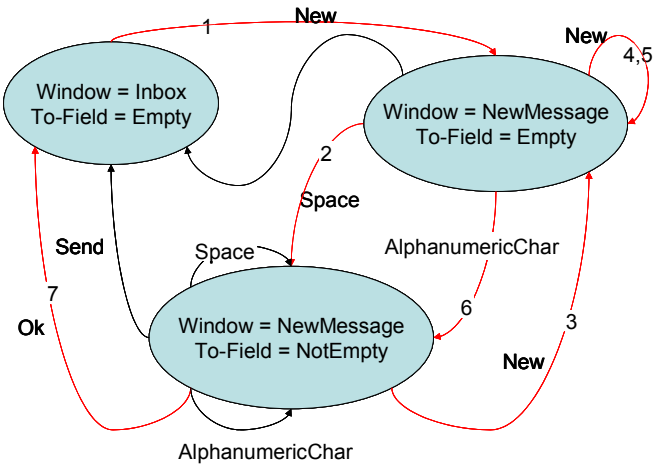
 © 2002 The authors, all rights reserved.

Case Study: Transitions



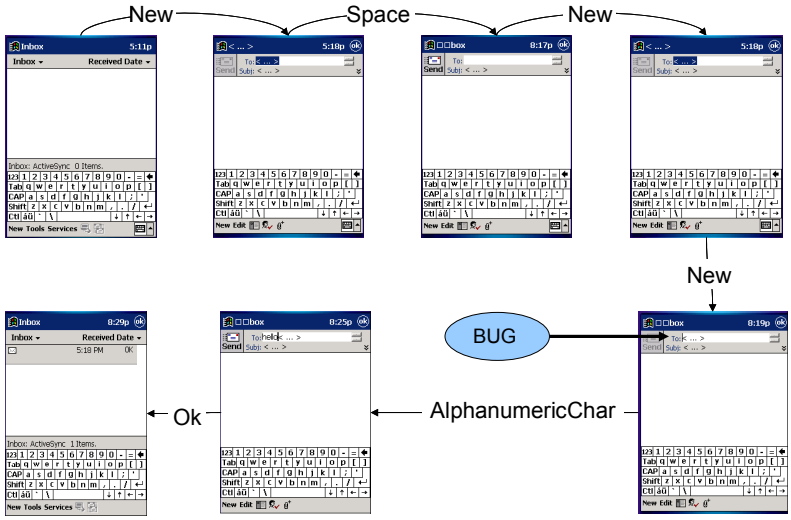
 © 2002 The authors, all rights reserved.

Case Study: Test Cases



 © 2002 The authors, all rights reserved.

Case Study: Test Cases



 © 2002 The authors, all rights reserved.

Conclusions

- Model-based testing needs to be coupled with exploratory techniques with the dual benefit of attaining a better, more current understanding of the system and harvesting many bugs along the way.
- Models are beneficial, not only as a point of reference for testing purposes, but also as a living specification of the functionality it represents and as a basis for test automation.
- Having a good automated test oracle is vital to the effectiveness of automated testing in general and model-based testing in particular.

 © 2002 The authors, all rights reserved.

Conclusion

- As long as finite state machines are used, there are inescapable critical issues to be dealt with: model building and maintenance, state explosion, and model correctness. There is need for more practical pointers on how to work around, or at least reduce the impact of these factors.
- Finally, studies need to be performed on answering the question: is model-based testing worth the effort when it comes to finding faults? Our preliminary results show that, for a very good, close-to-release, stable product, the number of faults uncovered by model-based testing is slightly disappointing if we severely limit the time during which we can run tests. The strongpoint of model-based testing is that it finds bugs with different characteristics: those that require long complicated sequences of inputs to be exposed.

 © 2002 The authors, all rights reserved.

- Contact Information
  - Email : hthompson@testingresearch.com
  - Web : http://www.testingresearch.com
  - Telephone : (321)-795-4531

 © 2002 The authors, all rights reserved.

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

Experiences in Testing Pocket PC Applications

Ibrahim K. El-Far
Florida Institute of Technology
Herbert H. Thompson
System Integrity
Florence E. Mottay
J.D. Edwards

Keywords

Pocket PC; model-based testing; finite state models; case study

1. Introduction

1.1 Remarks on Testing with Software Models

Model-based testing (or MBT) techniques are deeply rooted in such fields as phone switches and computer hardware components. However, their value remains vague in the software industry, despite their apparent intuitive appeal. Perhaps this can be attributed to poor understanding of the underlying principles and concepts of testing with models. Conceivably, it can also be attributed to a troublesome paradigm shift from what is widely practiced today. On the other hand, the fact remains that there is an obvious shortage of useful or insightful case studies. Further, there is barely any work that faithfully details the goals, activities, and risks involved to the average test professional who is expected to work with these methods. Indeed, many of these professionals today are oblivious to the very existence of MBT, and, those who are aware of it are, at best, highly doubtful of its value and the kind of returns on investment it presents. Recently, there has been a rise in the number of researchers and testers willing to take the time to investigate the models and methods of the paradigm (El-Far 2001).

These investigations seem to have started to pay off. Over the past few years, there have been many success stories about employing models to steer various testing activities such as test generation and test result evaluation (El-Far and Whittaker 2001). Such reports have generated a lot of enthusiasm with the popularization of object-oriented technologies and the advent of model-based design and specification methods and tools. As a result, we have been witnessing a rapid growth of the relevant body of literature since the 1990s. As with any developing field, the literature is affected by the lack of a common body of knowledge and a standard set of terms that are precisely defined or that everyone uses consistently. However, there are numerous lessons to be learned and many observations to be made on the model-based testing paradigm as a whole, notwithstanding the differences among various types of models.

For example, the literature has some pointers as to the benefits of model-based testing, many of which seem to agree with intuition (Robinson, 2000). For instance, the underlying model is a formal, precise expression of a tester's understanding of how the software is supposed to work. When such an understanding is written out to a structure that others can review, update, modify, and influence with their own understanding of the software under test, many problems can be solved. The model becomes a point of reference for the testing team, an aid to presenting results to non-technical staff, and a form of documentation that reflects the most recent build of the system – a living specification. Another benefit that is typical of several models is that they have a

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

substantial and rich theoretical background that makes numerous tasks such as generating large suites of tests fairly easy to automate. Examples of this are the theories of graphs (Gross & Yellen, 1998) and automata (Ullman & Hopcroft, 1979) for finite state machines and stochastic theory for Markov-chain models (Kemeny & Snell, 1976).

It is unfortunate that, due to many reasons, that we rarely see reports of failure or articles that contain warnings of pitfalls and tips on what to expect. For this reason, we are usually left to deduce most drawbacks from reports and from hard experience. Perhaps the thorniest such issue is one that plagues all forms of automated testing, namely, the oracle problem: how do we build an automated mechanism that checks the outcome of tests against the required behavior? The absence of an oracle is an obstacle to the automated execution of long tests or large suites of tests, both acclaimed by the field as major benefits of the approach (El-Far, 2001). Another significant drawback is the substantial investments, time and personnel, that typically have to go into building, reviewing, and maintaining models. Even with the smallest models, precious time will be lost before testers start to reap any fruit. Consequently, short development cycles, major delays in development, postponing testing activities until after components are developed can all potentially reduce the value of using models.

An interesting observation that can be drawn from the literature is that success reports seem to always come from only a few application domains: phone switch software (Avritzer & Larson, 1993), embedded software such as that in hardware controllers (Agrawal & Whittaker, 1993), and graphical user interfaces (Rosaria & Robinson, 2000), to mention some of the more typical domains. This is very encouraging for those who are considering employing models in testing these and other similar systems, although they would have to keep in mind that the results at our disposal are certainly not beyond doubt. So, not too surprisingly, when we were about to embark on a testing endeavor of some Pocket PC applications, we were enthusiastic and encouraged by what we know from the works of others and our earlier finite-state model based testing experiences. We shall elaborate on this later, but, first, we will briefly introduce the project in concern.

1.2 The Project at a Glance

Pocket PC is a Microsoft platform for handheld devices such as palmtops (Microsoft Inc. Official Website). It is powered by Windows technologies and has the look and feel of a scaled down version of a member of the Windows family of operating systems. Pocket PC devices ship with a collection of built in utilities. These are small applications that are design to be familiar to the Windows desktop user. They include, for example, Pocket Word (a simplified word processor) and Pocket Outlook (email manager and organization utility). Other applications can be added by the user or by a third party vendor.

Several months before the planned release date, when the product had reached a reasonable degree of stability, Microsoft contracted our group at the Center for Software Engineering Research in the Florida Institute of Technology to test five standalone components packaged with Pocket PC: Contacts, Calendar, Inbox, Connectivity Manager, and Pocket Word, all of which we will describe in some detail in a later section. They were particularly interested in seeing us apply finite-state model based techniques that were developed in part by researchers at the Center and that we will be briefly explaining in the next section.

Microsoft supplied us with tools to help carry out various MBT activities, and they established communication channels through which we were able to report bugs, request development support, and resolve conflicts and ambiguities. We started by gathering a team with a rich, varied background. They all had been receiving some sort of formal university education in the fields of

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

computer sciences, software engineering, and mathematics for a while, and they were fairly distributed across educational levels from those just starting their undergraduate studies to those pursuing doctoral research. Four out of the five working on the project had received proper instruction in software testing and two had previous experience in applying the technique in other projects administered by the Center (Jorgensen, 2000).

Both product and project conditions seemed to be in favor of employing a finite state model based technique. First, all the products had graphical user interfaces and seemed to be state rich, making them ideal for modeling using finite state models according to earlier case studies. Second, the products were in general small; they had significantly less inputs and features than what one would expect in a similar desktop applications. We believed, at first sight, that the environment in which the Pocket PC application were deployed to be relatively well behaved. For instance, there were only a few other applications with which they would interact, and most of these interactions could be manageably monitored and recorded. In addition, by virtue of its design, which was intended to support only a few devices and did not have any backward compatibility issues, the operating system was small, free of clutter functions, and well tweaked for its purposes. We had better chances, therefore, to accurately configure our tests and account for most environmental conditions.

The project spanned two academic semesters, which would amount to eight effective months of testing, more than many groups in the industry world normally have to develop a product never mind test it. All members were contractually obligated to work for at least twenty hours every week, but many ended up devoting up to thirty-five hours to this project. Given the fact that we were supposed to test five applications, however, this meant that we had just enough time, but not a whole lot.

We did have some worries about a number of practical issues, most notably input simulation and test outcome evaluation. We were not exactly clear on how to execute our tests in an embedded system; typically, in such cases, some type of simulator would be needed. As to evaluation, we were also not clear on how to verify the state of the application against our models and how to monitor and record any other needed application information. Both these concerns were addressed and resolved through development and test-tool support from Microsoft, details of which could not be disclosed as per our legal obligations toward the company.

1.3 This Paper

Working on this project was rewarding in terms of the lessons learned. Our experience reinforced some of the common beliefs about some of the benefits of model-based testing. On the other hand, many questions about the returns on investment, bug count, and model-adaptability to specific contexts were raised with no satisfying answer.

This paper summarizes this experience. First, we briefly visit our technique for finite state model based software testing. Then, we describe the applications under test and briefly outline the plan that we followed to test each of them. A summary of the proceedings of the testing effort is followed by a list of some the lessons learned.

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

2. Background

2.1 Definitions and Terminology

2.1.1 Software States

A software state is loosely defined as a condition of the software in which a certain collection of inputs can be applied. For example, consider a typical combination safe. For our purpose, let us say we walk into a room with that has such a safe. Consider two general states that the safe can be in:

1. All tumblers are aligned and we can turn the handle to open the safe.
2. The correct sequence has not been applied to the combination dial and the safe handle can not be turned.

Here we have clear criteria to define states in terms of applicable inputs. One state is defined by the fact that we can turn the handle and open the safe. In the other state this input is not available to us. We can easily extend this criterion for state definition to software. Consider a typical GUI email application. In most such applications if there is no entry in the "To" line the send option is disabled. Intuitively we can say that when text is in the "To" line, the software is in one state which is different from the state the software is in when text is not present because different inputs are available to the user. In the following sections we will cement this notion of a software state through examples and formal definition.

2.1.2 An Example Software Under Test

Consider a hypothetical light switch. The lights can be turned on and off using one input. The intensity of the light can be adjusted using two inputs for lowering and increasing the intensity. There are three levels of light intensity: dim, normal, and bright. If the lights are bright, increasing the intensity should not affect the intensity. The case is similar for dim light and decreasing the intensity. The simulator starts with the lights off. Finally, when the lights are turned on, the intensity is normal by default, regardless of the intensity of the light when it was last turned off. Obviously, the simulator can be in only one of four distinct states at any one time: the lights are either off, dim, normal, or bright.

2.1.3 Finite State Machines

Formally a finite state machine representing a software system is defined as a quintuple *(I, S, T, F, L)*, where

- *I* is the set of inputs of the system (as opposed to input sequences).
- *S* is the set of all states of the system.
- *T* is a function that determines whether a transition occurs when an input is applied to the system in a particular state.
- *F* is the set of final states the system can end up in when it terminates.
- *L* is the state into which the software is launched.

A finite state machine can only be in one state at any one time. The occurrence of a transition from one state to another is exclusively dependent on an input in *I*.

2.1.4 Example Finite State Machine

One way to model this is to use a finite state machine that is defined as follows:

```
"Light Switch" = (I, S, T, F, L), where:
```

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

```
o   I = {<turn on>, <turn off>, <increase intensity>, <decrease intensity>}
o   S = {[off], [dim], [normal], [bright]}
o   T:
        o   <turn on> changes [off] to [normal]
        o   <turn off> changes any of [dim], [normal], or [bright] to [off]
        o   <increase intensity> changes [dim] and [normal] to [normal] and
            [bright], respectively
        o   <decrease intensity> changes [bright] and [normal] to [normal]
            and [dim], respectively
        o   The inputs do not affect the state of the system under any con-
            dition not described above
o   F = [off]
o   L = [off]
```

Figure 1: "Light Switch" Finite State Machine Definition

2.1.5 REPRESENTATION

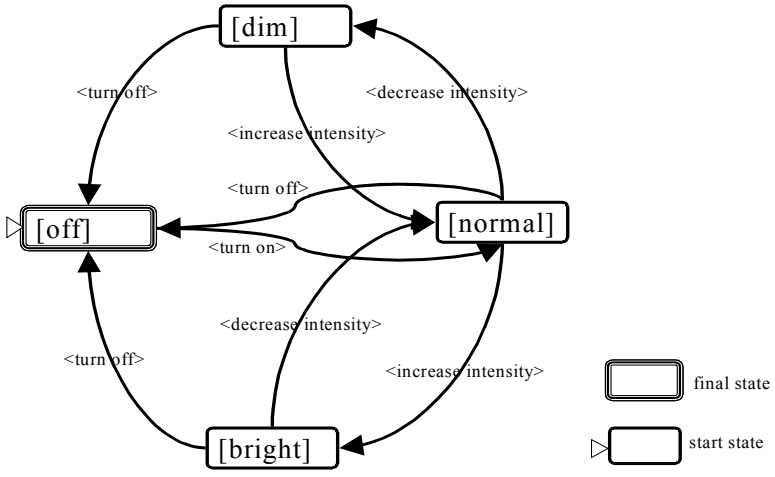
Finite state machine models can be represented as graphs, also called state transition diagrams, with nodes representing states, arcs representing transitions, and arc-labels representing inputs causing the transitions. Usually, the starting and final states are specially marked. Automata can also be represented as matrices, called state transition matrices. There are two useful forms of state transition matrices that are illustrated for the "Light Switch" along with the corresponding state transition diagram.

	[off]	**[dim]**	**[normal]**	**[bright]**
[off]			<turn on>	
[dim]	<turn off>		<increase intensity>	
[normal]	<turn off>	<decrease intensity>		<increase intensity>
[bright]	<turn off>		<decrease intensity>	

(i)

	<turn on>	**<turn off>**	**<increase intensity>**	**<decrease intensity>**
[off]	[normal]			
[dim]		[off]	[normal]	
[normal]		[off]	[bright]	[dim]
[bright]		[off]		[normal]

(ii)



(iii)

Figure 2: The presentation screen of the inbox application

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

2.2 Why Finite State Models are Useful

Consider a common testing scenario: a tester applies an input and then appraises the result. The tester then selects another input, depending on the prior result, and once again reappraises the next set of possible inputs. At any given time, a tester has a specific set of inputs from which to choose. This set of inputs varies depending on the exact state of the software. This characteristic of software makes state-based models a logical fit for software testing: software is always in a specific state and the current state of the application governs what set of inputs from which testers can select. If one accepts this description of software then a model that must be considered is the *finite state machine*.

Finite state machines have been around even before the inception of software engineering. There is a stable and mature theory of computing at the center of which are finite state machines and other variations. Chow (1978) wrote one of the earliest, generally available articles addressing the use of finite state models to design and test software components.

Finite state models are an obvious fit with software testing where testers deal with the chore of constructing input sequences to supply as test data; state machines (directed graphs) are good models for describing sequences of inputs. This, combined with a wealth of graph traversal algorithms (Robinson 1999 TCS), makes generating tests less of a burden than manual testing. On the downside, complex software implies large state machines, which are nontrivial to construct and maintain.

2.3 Finite State Model-Based Testing Activities

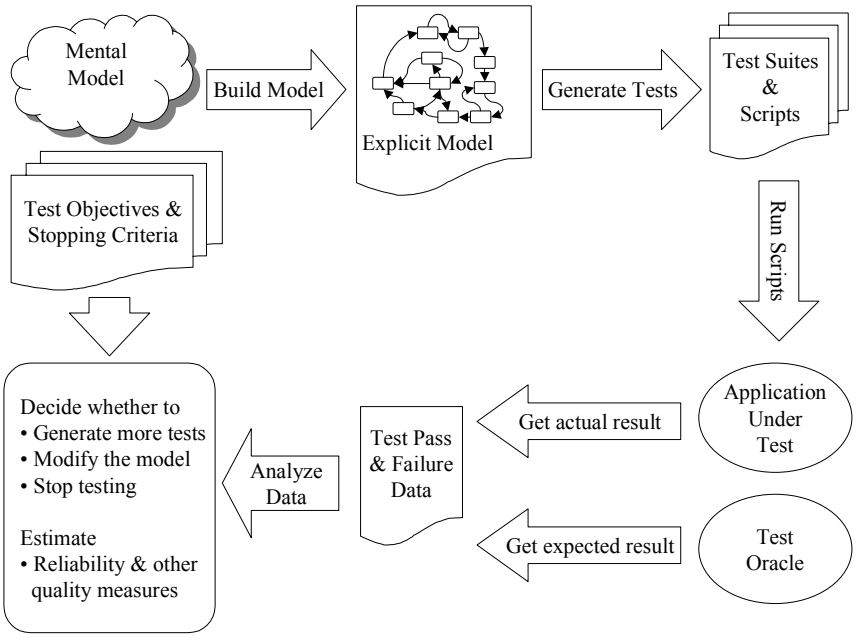


Figure 3: Some Model-based Testing Activities

Figure 3 above describes the finite state model based testing process. Perhaps the most difficult step is encapsulating our mental model of the software into a concrete structure. In the next section, we discuss a framework for expressing software models and representing states as a collection of software attributes.

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

2.4 A Compact Representation of Finite State Models

Directed graphs representing the functionality of a software component can be an effective tool in software testing. Figure 2 shows an example of such a graph. For any software system of non-trivial size, however, these representations are inadequate. Here we seek to define a compact way of defining a software state in terms of critical characteristics of the software. For example, consider the email application described in section 2.1.1 above. Recall that the "Send" button of the application is disabled if no text is entered in the "To" field. Suppose this application has two windows: the first, which lists all received email and the second, which allows the user to compose an email. Here we can identify two critical conditions that must be met in order to apply the "Send" input:

1. The application must be in the "compose" window.
2. Text must be present in the "To" field.

To encapsulate this information, each of these characteristics of the system is referred to as a state variable. For example, for this application we may want to define our state variables as Window and Text_in_To_Field. Associated with each state variable is a set of values. In this case it would be appropriate to define:

Window = View, Compose
Text_in_To_Field = Yes, No

A state then in terms of these values can be thought of as the combination of the variables above with one value for each. The only state in this case for which we can apply the "Send" input is:
{Window = Compose and Text_in_To_Field = Yes}
The total number of potential states is the cross-product of the number of state variable values. In our example the total number of possible states is thus 4 (2* 2) because we have 2 values for each state variable. However, the number of valid states is almost never equal to this total. This is one of the problems of model-based testing in that a significant amount of time is generally spent identifying impossible states. In this example, there are only 3 possible states, which are:
{Window = View and Text_in_To_Field = No}
{Window = Compose and Text_in_To_Field = No}
{Window = Compose and Text_in_To_Field = Yes}

Another significant issue in model-based testing is state explosion. State explosion generally happens when we increase the number of state variables and/or values. Consequently, adding only one value can result in an out-of-control number of valid states, especially for large models. Consider in this case adding just one value to the Window state variable. This action will increase the number of potential states from 4 to 6.

3. The Testing Effort: The Inbox

The Inbox application is a small-scale version of outlook. It is the largest application we had to test. The whole model consisted of almost 5,000 transitions and approximately 1,500 states. This application was interesting to model, in that it had a diverse range of features. An example of this diversity is the number of different windows in the application. The user can either view the inbox, outbox, deleted-items or drafts. One can edit a message from all these screens except from the deleted-items window. To limit state explosion, we had to limit the number of messages that could be found in each of the windows to three(except the deleted items window).

We encountered some interesting challenges while modeling this application and automating tests. For example, we actually needed to start running test suites with the keyboard visible on the screen to ensure that inputs were accessible by automation. Such problems are often

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

met while using MBT techniques and these design choices that the developer makes in order to be user-friendly sometimes force the testers to model around them.

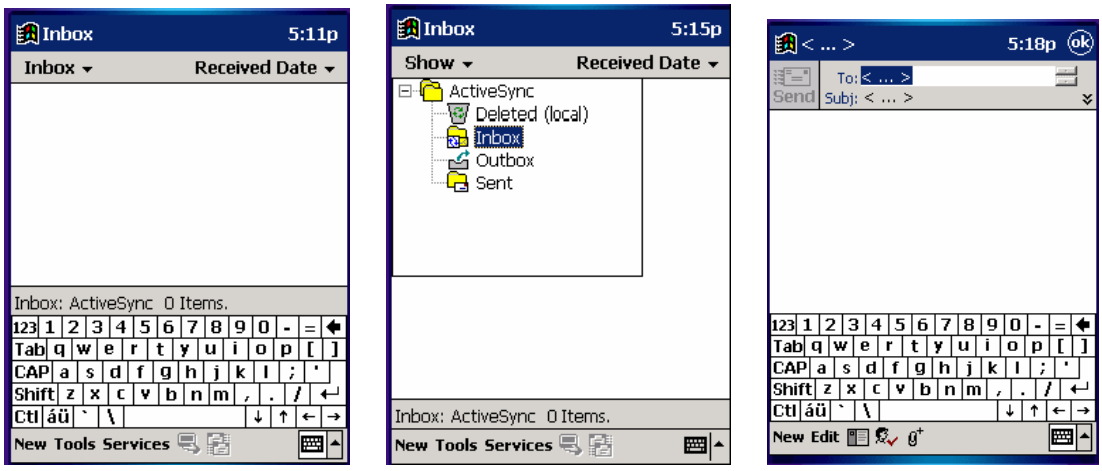
Most of the defects we found were in this application. This is understandable as it was the most complex and the one with the most features modeled. This also confirms our intuition that modeling more details often increases the chances of finding defects. One has to be careful, though, as one potential pitfall when using MBT techniques is to construct an overly detailed model. Such a model is seldom readable, hard to maintain, and makes building automation much more difficult.

Next, we present the steps necessary for our finite state model-based technique. For each step, we use examples from our work on the inbox application. The complete model would be confusing as it is too big; thus, we show only a partial model.

1. Explore the application in order to discover and build a mental representation of its functionality.

   For the Inbox application, we first studied the different screens[*].

   Figure 4(a), shows the first screen that users see when they enter the inbox application. The second screen, figure 4(b), exposes the menu that allows a user to browse through the inbox folders. Figure 4(c), shows the new message screen that appears after the user clicks on "New". On this screen, the user can type a message and send it. The "Send" button will only be enabled when the user has typed some text in the "To" field. To save a message the user will have to click the "Ok" button.



(a) (b) (c)

Figure 4: Different Screens of the Inbox Application: (a) Presentation Screen of the Inbox Application (b) Presentation Screen and the Inbox Menu (c) New Message Window

[*] The screenshots in this paper were captured from the publicly available Pocket PC emulator. The actual (similar) beta versions we worked with fall under non-disclosure agreement.

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

2. Identify all user inputs. Decisions on whether to abstract physical inputs are made based on what we need to test. For example, two inputs that are visibly the same and that can be simulated with the same script may be abstracted as one input.

   While we were testing the inbox application, we made a lot of abstractions. Figure5 shows a screenshot of the New Message window. The "Ok" input located in the upper-left corner of the window has the same effect as pressing the Enter button. Since we could not find any significant differences between these two inputs, we decided to only consider "Ok" as an input and not Enter.

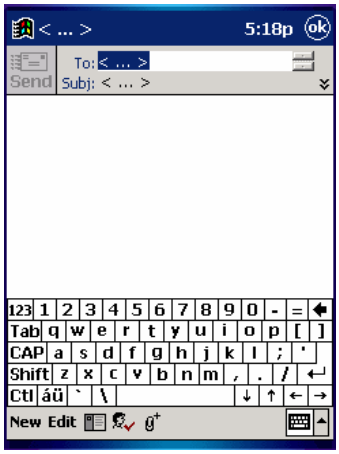


Figure 5: Abstraction of Inputs

   Here is the full list of inputs with short explanation for each of them for the small model presented in this section.

   1) New: to go to a new message window
   2) Ok: to leave the New Message screen and save the current message
   3) Send: to leave the New Message screen and send the current message
   4) Space: space character
   5) AlphanumericChar: any alphanumeric character

3. Identify the individual characteristics used to define the states of the application. When can this input be applied by the user and what are the system characteristics that affect its applicability? What are the properties of the system that cause different responses to the same input under seemingly similar conditions? From this information, define the rules that describe valid sequences of inputs.

   For each model, we explored the application in more depth than in step 1 to uncover input's applicability. Next, we describe the conditions in which each input is applicable.

   1) New: this input is applicable at any time. The user can press the "New" button when the general inbox screen is showing, when the window is New Message and whether he/she has entered text inside a New Message window.

   2) Ok: this input is applicable when the window is a New Message. Whether the "To" field is empty or not does not make a difference.

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

3) Send: this input is applicable only when the window is a New Message and the "To" field is not empty.

4) Space: this input is applicable when the window is a New Message.

5) AlphanumericChar: this input is applicable when the window is a New Message.

Applicability of inputs then allows the tester to derive valid sequences of inputs. For our model and assuming that the starting state is the general inbox window, an example of a valid sequence is: New-AphanumericChar-Space-Send.

4. Generate the set of valid software states and transitions with the aid of specialized tools.

Following is the list of operational modes and inputs that are necessary in order to construct our scaled down model of the inbox. Its state transition diagram will then be shown.

State variables

Window = Inbox, NewMessage
This operational mode records which window the user is on. For this model, only two windows are considered as possible.

To Field = Empty, NotEmpty
This operational mode records if the value of the "To" field is empty or not. It is useful to determine whether the "Send" is enabled or not.

State transition diagram

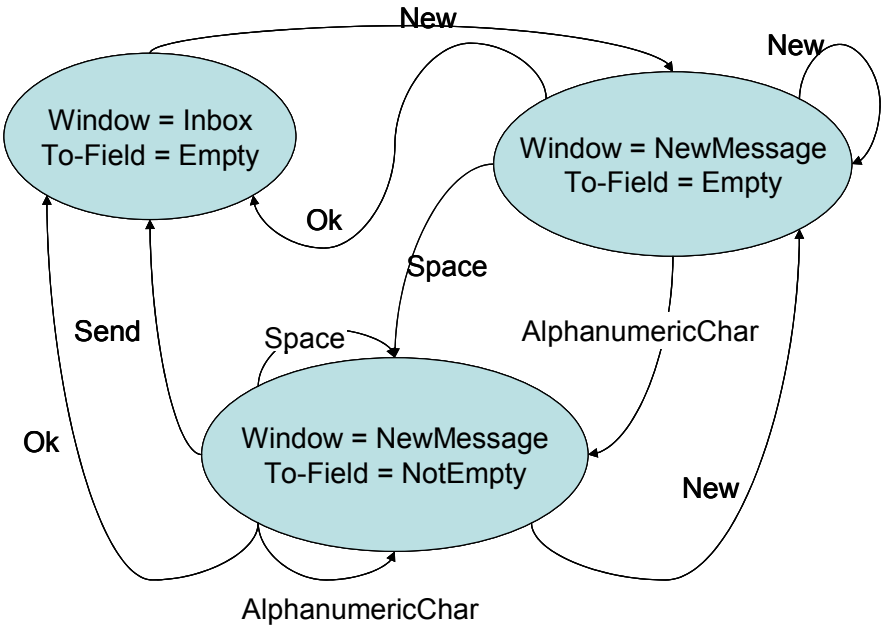


Figure 6: Graph Representation of the Model

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

Examples of input sequences are New-Space-Ok, New-New-Ok etc.

5. Generate and run test cases (traversal paths in the graph). This step particularly benefits from the well-established graph-theoretical body of knowledge.
   To generate and run test cases, we used tools provided by Microsoft that we cannot disclose in this paper. However, to better understand some of the sequences in this model, the following shows one such possible scenario. Figure 8 below shows the path that is illustrated through the screen shots below in figure 9.

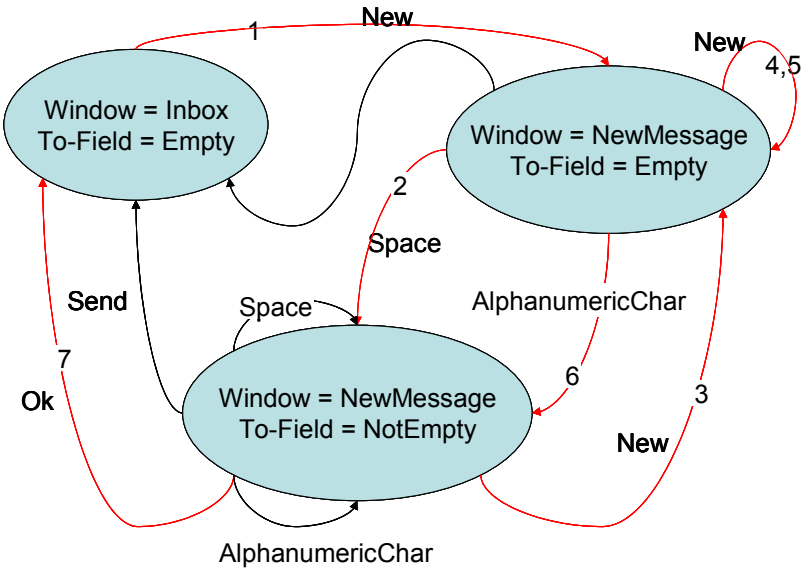


Figure 7: Traversal Path

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

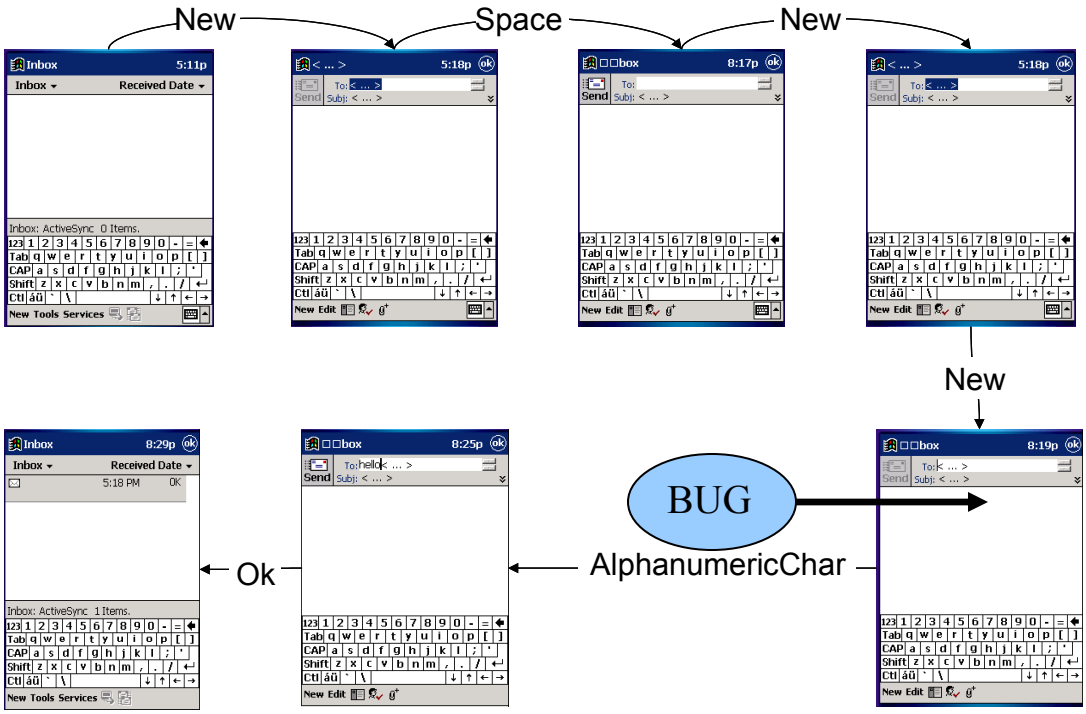


Figure 8: Scenario in Screenshots

This series of screenshots is an example of test sequences that were run. This sequence also demonstrates an inconsistency in one of the inputs. The first two "New" inputs show a New Message screen with the "To" field highlighted. This allows the user to type text directly into the "To" field. The third "New" though shows a New Message with the cursor in front of the "< … >". It means that the user will type text in front of that default value, which will stay unless the user manually removes it (see the screenshot resulting from the "AlphanumericChar" input). This issue was accepted as a defect by Microsoft and even if it seems like a small problem, it would still inconvenience the user. The Inbox application was the largest application we had to test and we found a number of inconsistencies that were accepted as defects. Uncovering such failures illustrates an advantage of model-based testing techniques over other testing techniques; sequences of inputs that are unusual and do not seem to be potentially defective are executed by finite state machines.

4. Conclusions

- Model-based testing needs to be coupled with exploratory techniques with the dual benefit of attaining a better, more current understanding of the system and harvesting many bugs along the way.
- Models are beneficial, not only as a point of reference for testing purposes, but also as a living specification of the functionality it represents and as a basis for test automation.
- Having a good automated test oracle is vital to the effectiveness of automated testing in general and model-based testing in particular.
- As long as finite state machines are used, there are inescapable critical issues to be dealt with: model building and maintenance, state explosion, and model correctness. There is need for more practical pointers on how to work around, or at least reduce the impact of these factors.

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

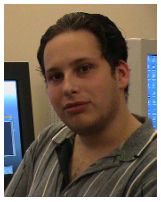
- Finally, studies need to be performed on answering the question: is model-based testing worth the effort when it comes to finding faults? Our preliminary results show that, for a very good, close-to-release, stable product, the number of faults uncovered by model-based testing is slightly disappointing if we severely limit the time during which we can run tests. The strongpoint of model-based testing is that it finds bugs with different characteristics: those that require long complicated sequences of inputs to be exposed.

Acknowledgements

This project was sponsored in part by a grant from Microsoft Inc in 2000. Thanks are due to Ahmed Salem, Matt Wagner, Danko Panic, and Mohammed Al-Ghafees for their work and contribution to the project.

About the Authors

Ibrahim K. El-Far is a doctoral candidate in computer sciences at the Florida Institute of Technology under the academic supervision of professor James A. Whittaker. He has a Bachelor of Sciences and a Master of Sciences in Computer Science from the American University of Beirut, Beirut, Lebanon and the Florida Institute of Technology, Melbourne, Florida, USA, respectively. His interests are in investigating software models for testing, test automation and tools, adequacy criteria, test cost and effectiveness, and software testing education. He has over five years of experience in model-based testing using finite state machines at the Center for Software Engineering Research at Florida Tech, where he has supervised the development of experimental model-based testing tools, advised model-based testing groups, and taught model-based testing in various formats to a variety of students. You can contact Ibrahim at ielfar@acm.org, visit http://www.testingresearch.com/ for more information, or write him at the Software Engineering Program, Computer Sciences Department, Florida Institute of Technology, 150 West University Boulevard, Melbourne, Florida 32901 USA.



Herbert H. Thompson is the director of security technology at System Integrity. He received his doctoral degree in mathematics from the Florida Institute of Technology in 2002. In previous years, Dr. Thompson has worked for Microsoft Corporation as a test engineer. His research interests are in software engineering, security, and applying mathematics to computer science problems. You can contact him at hethomps@fit.edu or visit http://www.testingresearch.com/.

Florence E. Mottay is a quality engineer at J.D. Edwards. She received her Master of Science in mathematics from the Florida Institute of Technology. In previous years, Ms. Mottay worked as a research assistant at the Center for Software Engineering Research, Florida Institute of Technology, Melbourne. Her research interests are in software testing, formal languages, mathematical models, and e-commerce. She received awards for excellence in mathematics by the United States Achievement Academy (1997) and for academic excellence by the American Association of University Women (1998).

References

[Agrawal & Whittaker 1993] K. Agrawal and James A. Whittaker. Experiences in applying statistical testing to a real-time, embedded software system. *Proceedings of the Pacific Northwest Software Quality Conference*, October 1993.

Copyright © 2002 The Authors. All rights reserved. This paper to appear in the Proceedings of the Fifteenth International Internet & Software Quality Week Conference (QW 2002), September 2002.

[Avritzer & Larson 1993] Alberto Avritzer and Brian Larson. Load testing software using deterministic state testing." *Proceedings of the 1993 International Symposium on Software Testing and Analysis (ISSTA 1993)*, pp. 82-88, ACM, Cambridge, MA, USA, 1993.

[Chow 1978] Tsun S. Chow. Testing design modeled by finite-state machines. *IEEE Transactions on Software Engineering*, 4(3): 178-187, May 1978.

[El-Far & Whittaker 2001] Ibrahim K. El-Far and James A. Whittaker. Model-based software testing. *To appear in the Encyclopedia of Software Engineering*, 2001.

[El-Far 2001] Ibrahim K. El-Far. Enjoying the perks of model-based testing. *Proceedings of 2001 Software Testing Analysis & Review Conference (STARWEST)*, San Jose, California, USA, October/November 2001.

[Gross & Yellen 1998] Jonathan Gross and Jay Yellen. *Graph theory and its applications*. CRC, Boca Raton, FL, USA, 1998.

[Hopcroft & Ullman 1979] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, 1979.

[Jorgensen 2000] Alan Jorgensen and James A. Whittaker. An API Testing Method. *Proceedings of the International Conference on Software Testing Analysis & Review (STAREAST 2000)*, Software Quality Engineering, Orlando, May 2000.

[Kemeny & Snell 1976] J. G. Kemeny and J. L. Snell. *Finite Markov chains*. Springer-Verlag, New York 1976.

[Robinson 1999 STAR] Harry Robinson. Finite state model-based testing on a shoestring. *Proceedings of the 1999 International Conference on Software Testing Analysis and Review (STARWEST 1999)*, Software Quality Engineering, San Jose, CA, USA, October 1999.

[Robinson 1999 TCS] Harry Robinson. Graph theory techniques in model-based testing. *Proceedings of the 16th International Conference and Exposition on Testing Computer Software (TCS 1999)*, Los Angeles, CA, USA, 1999.

[Robinson 2000] Harry Robinson. Intelligent test automation. *Software Testing Quality Engineering (STQE) Magazine*, October/November 2000.

[Rosaria & Robinson 2000] Steven Rosaria and Harry Robinson. Applying models in your testing process. *Information and Software Technology*, 42(12): 815-824, September 2000.

QW2002 Paper 2T2

Dr. Josef Withalm
(Siemens Ag)

Quality Assurance Aspects in Developing a Component-based System

Key Points

- Component-Based
- E-business
- Quality Assurance

Presentation Abstract

In my contribution, I examine the influence component technologies can have on established QM-systems. With an example from the field of tourism, I describe how an e-business architecture is built and which component technologies can be used in order to fulfil the demands of both, QM and e-business.

About the Author

Born in 1947, Dr. J. Withalm studied at the University of Technology and graduated in Technical Mathematics, in Vienna in 1972 with the degree "Diploma Engineer". During his studies he worked with an Austrian company as a system analyst on financial software packages. Since 1973 he is employed by Siemens Austria. His areas of responsibility between 1973 an 1980 were as a system analyst working on mathematical, topological system design algorithms for mainframe Hardware design. Another speciality was the design of topological algorithms for railway signal technique. Besides this he continued his studies at the University of Technology and made his Ph.D. in 1979. Parallel to those activities Dr.Withalm has always been very active on Quality Assurance matters and was head of Siemens PSE's Quality Assurance department sind 1983.

In 1991 he started lectures in object oriented SW-Development on the University of Bratislava.In 1998 he started a second series of lectures in distributed objects on the Fachhochschule for telematic in Salzburg. In Autumn 2000 he began a further series of lectures in quality management on the Fachhochschule Technikum Vienna.

Since 1994 he is head of the business field software engineering and Tourism. Dr. Withalm headed 3 CMM assessments teams in 1993 - 1994 and was co-assessor in a CMM assessment in spring 1998.

SIEMENS

Program and System Engineering PSE

Quality Assurance Aspects in Developing a Component-based System

Dr. Josef Withalm

Curriculum vitae

Born in 1947, Dr. J. Withalm studied at the University of Technology and graduated in Technical Mathematics in 1972 with the degree “Diplomingenieur”. During his studies, he worked with an Austrian company as a systems analyst for financial software packages. Since 1973 he has been employed with Siemens Österreich. His areas of responsibility between 1973 and 1980 were system analysis on mathematical topological system design algorithms for mainframe hardware design. Another specialty was the design of topological algorithms for railway signaling. At the same time he continued his studies at the University of Technology, graduating with a Ph. D. degree in 1979. In parallel, Dr. Withalm has always been very active in the field of quality assurance, becoming the head of Siemens PSE’s Quality Assurance department in 1983.

In 1991 he began to hold lectures on object-oriented SW development at the university of Bratislava, Slovakia. In 1998 he started a second series of lectures on the topic of distributed objects at the College for Telematics at Salzburg, Austria. Salzburg auf.

Since 1994, he has been the head of the PSE SW Engineering and Tourism sub-division.

Between 1993 and 1994, Dr. Withalm directed 3 CMM assessment teams and acted as co-assessor in a CMM assessment in the spring of 1998.

Quality assurance aspects in developing a component-based system

Overview

- Siemens/PSE environment
- Project definition
- QA requirements
- e-business architecture
- Technological requirements
- Lessons learned

Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

Siemens facts & figures

- Sales in 2001: € 87 billion
- World market for the electrical and electronics industry: € 1.745 billion
- Staff in 2001: 484.000
- Represented in over 190 countries
- Manufacturing plants in 43 countries

Siemens /
PSE environment

Project definition

QA requirements

e-business
architecture

Technological
requirements

Lessons learned

Siemens in the USA

- 35 companies, including:
  - Siemens Information and Communication Networks
  - Siemens Westinghouse Power Corporation
  - Siemens Business Services
  - Siemens Financial Services
  - Osram Sylvania
- Total staff: 80.000
- 780 locations across all 50 states
- Sales in the Americas in 2001: € 26,1 billion

U.S. companies

- Siemens Airfield Solutions, Inc
- Siemens Information and Communication Networks, Inc.
- Siemens Real Estate, Inc.
- Siemens Shared Services LLC
- Siemens Building Technologies, Inc
- Siemens Carrier Networks, LLC
- Siemens Transportation Systems, Inc.
- Siemens Enterprise Networks, LLC
- Siemens Business Services
- Efficient Networks, Inc.
- Siemens Corporate Research, Inc
- Siemens VDO Automotive Corporation
- Optisphere Networks, Inc.
- Siemens Diesel Systems Technology, LLC
- Siemens Technology-To-Business Center LLC
- Siemens Foundation
- Siemens Information and Communication Mobile LLC
- Siemens Procurement and Logistics Services, LLC

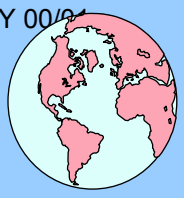
U.S. companies

- Unisphere Networks, Inc
- Siemens Venture Capital, Inc.
- Siemens Medical Solutions USA, Inc.
- Siemens Dematic Corp. Rapistan Material Handling Automation Division
- Siemens Hearing Instruments, Inc.
- Siemens Westinghouse Power Corporation
- Siemens Dematic Postal Automation L.P.
- OSRAM SYLVANIA
- Acuson
- Siemens Dematic Electronics Assembly Systems, Inc.
- Siemens One
- Siemens Energy & Automation, Inc.
- Siemens Medical Solutions Health Services Corporation
- Siemens Corporation
- Siemens Power Transmission & Distribution, Inc.
- Siemens Financial Services, Inc.

Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

PSE environment

- Approx. 5.200 employees (including subsidiaries)
- 40 years of practical experience
- 21 locations in 7 European countries and the USA
- Total sales approx. € 493 million in the FY 00/01
- Export rate approx. 95%
- Projects in 53 countries



Siemens /
PSE environment

Project definition

QA requirements

e-business
architecture

Technological
requirements

Lessons learned

PSE Locations in the USA

Quality assurance aspects in developing a component-based system

Overview

- Siemens/PSE environment
- Project definition
- QA requirements
- e-business architecture
- Technological requirements
- Lessons learned

Siemens /
PSE environment

Project definition

QA requirements

e-business
architecture

Technological
requirements

Lessons learned

Project description

- establishing an agent platform for the following applications:
  - SAPA (solving allotment problems by mobile agents)
  - Smart-Up (Small and Medium Sized Enterprises Alliance through Research in Tourism)
  - LuSe (worldwide luggage search)

Program and
System Engineering
PSE

SIEMENS

Siemens /
PSE environment

Project definition

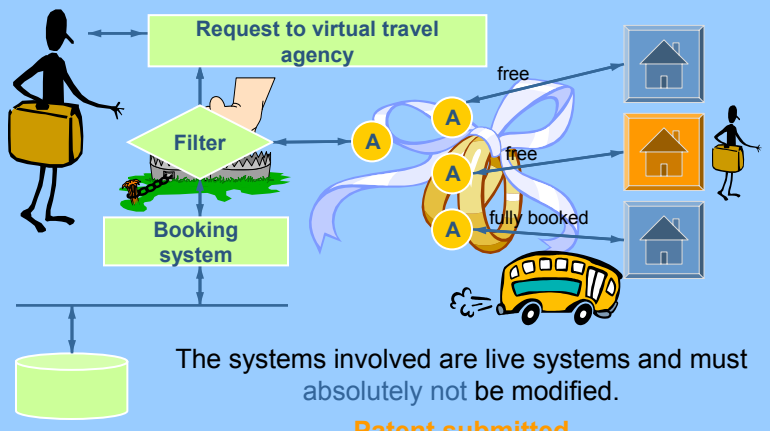
QA requirements

e-business
architecture

Technological
requirements

Lessons learned

Task – Agents between travel agent and hotel



The systems involved are live systems and must absolutely not be modified.

Patent submitted

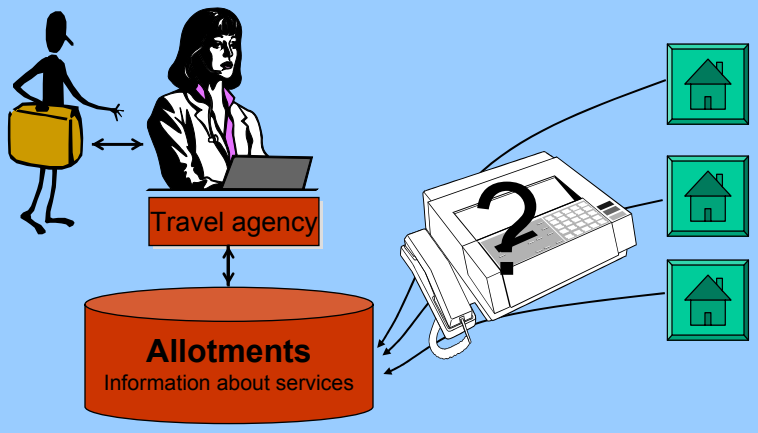
Summary of benefits of SAPA

- Benefits for hotel owners
  - fast and flexible room assignment
  - risk of overbooking decreases
  - fewer different allotments required
- Benefits for providers of Computerized Reservation Systems (CRS)
  - data up-to-date
  - lower costs for data management
- Benefits for tourists/travelers
  - guaranteed room booking
  - genuine last-minute booking
  - no rebooking to other hotels due to overbooking

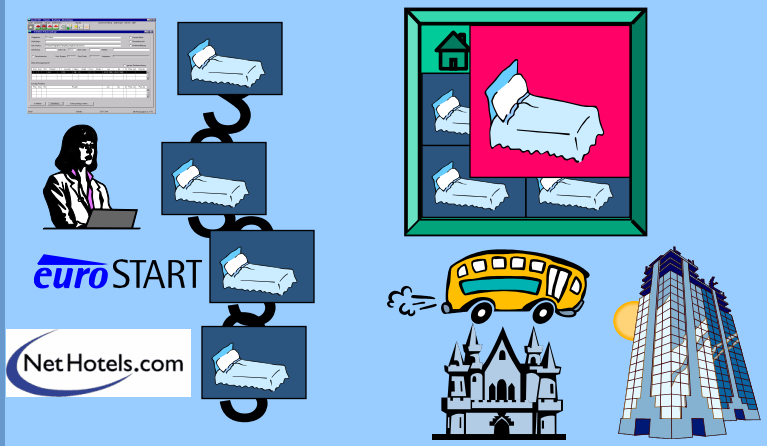
Motivation

- Computerized reservation and booking systems (CRSs) currently require allotments to be made
  - Hotels want to use this sales channel
- Disadvantages of allotments
  - Hotel cannot sell all of its services on its own
  - No guarantee that rooms will be sold
  - High commission payments for placement
  - Data is not always up-to-date
  - Management of allotments cost-intensive also for CRSs

Initial situation
Traditional travel agency



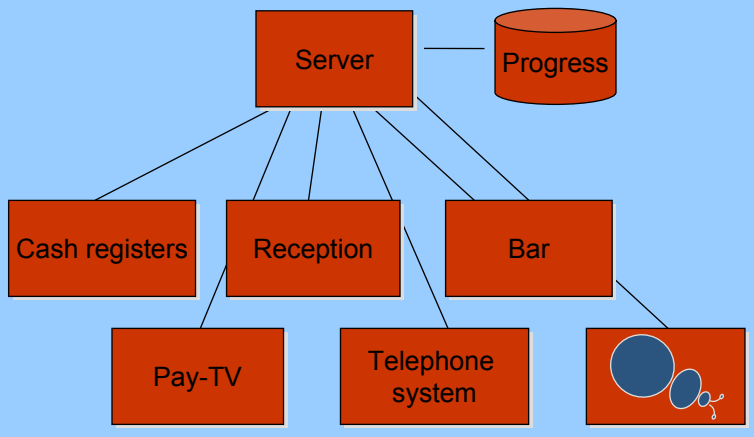
The allotment problem



Implemented parts

Agent
Technology

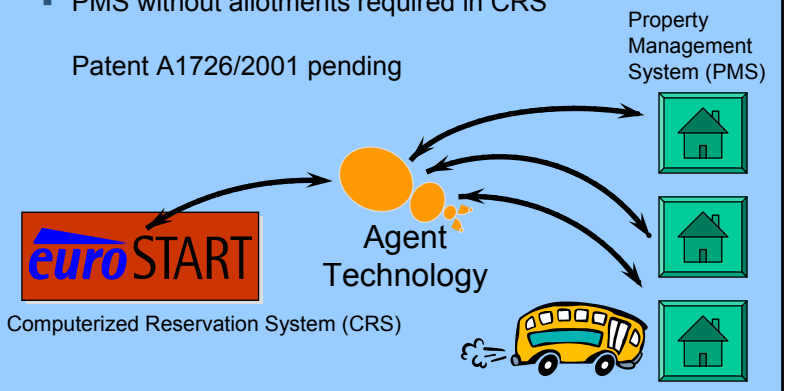
HOGATEX Architecture



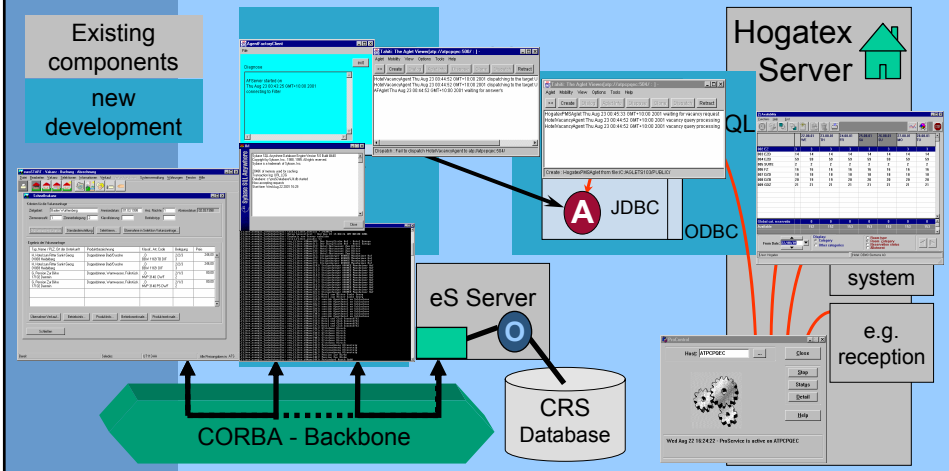
α-version CRS for PMS

- PMS without allotments required in CRS

  Patent A1726/2001 pending



Architecture α-version CRS for PMS



GUI of α-version CRS for PMS

Some companies requested proposals

- ebookers
- Nethotels
- Sigma
- Start-Amadeus
- Egypt Tourism Ministry
- SBS in Spain

Siemens /
PSE environment

Project definition

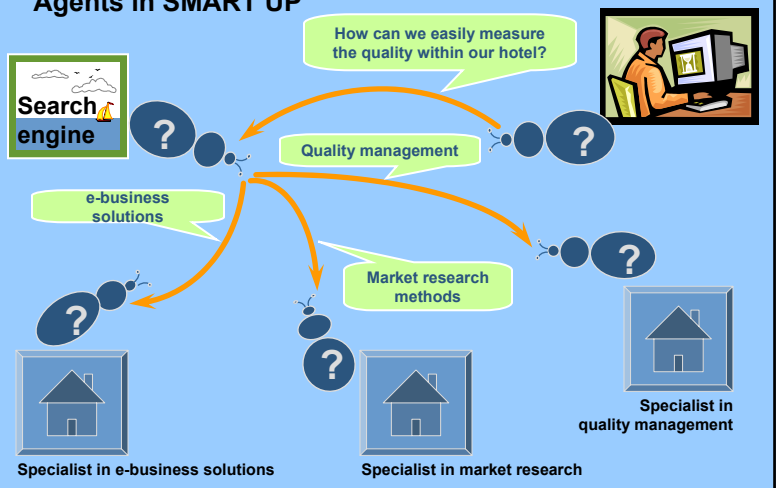
QA requirements

e-business
architecture

Technological
requirements

Lessons learned

Agents in SMART UP



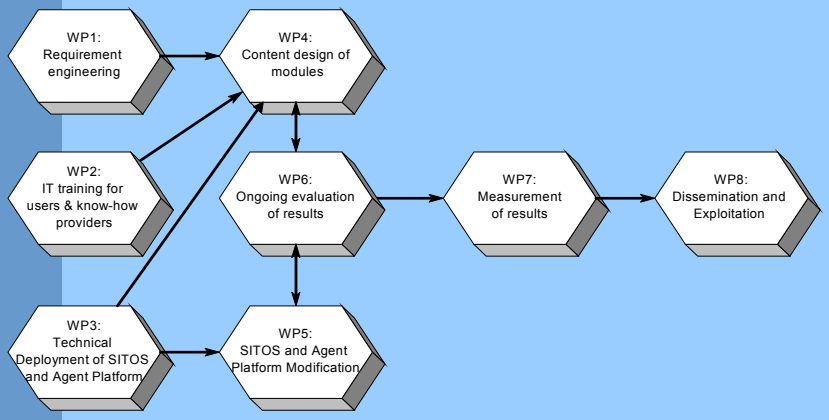
Summary of benefits of SMART UP

- To provide SMEs in the tourism industry in Europe with the tools required for them to become competitive again
  - through a cooperation of SMEs, research institutions, and industrial partners
  - through an exchange of information and experience
  - through benchmarking and interactive learning
  - through establishing long-term relationships
- Funded by the EU

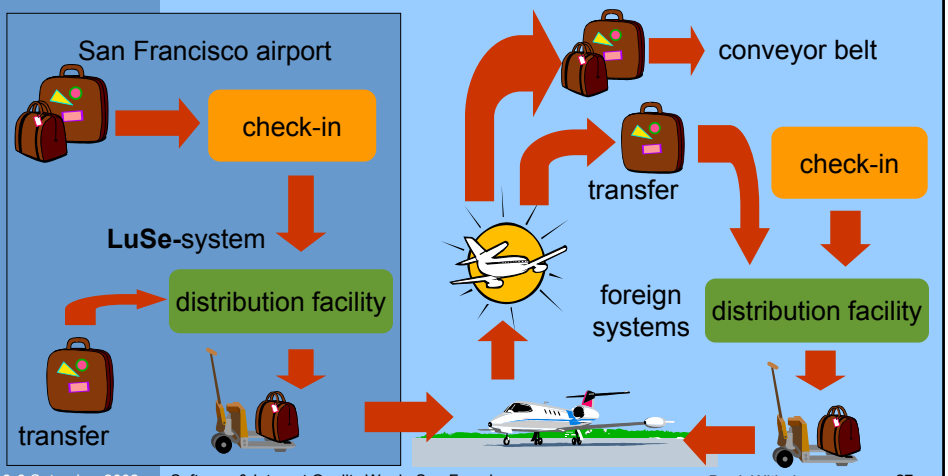
SMART UP
The partners

- OEHV - Association of Austrian hotel owners
- Sigma - provides travel agents with IT infrastructure and training in using such infrastructure
- Siemens - provides the HW/SW platform
- 4 University institutes from 4 European countries
  - University of Innsbruck, Austria, Center for Tourism and Service Economics
  - Dublin Institute of Technology, Ireland, Faculty of Tourism and Food School of Hospitality Management and Tourism
  - University of St. Gallen, Switzerland
  - University of Surrey ,UK, School of Management Studies for the Service Sector

Workpackage linkages of SMART UP



LuSe/Solution



Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

Summary of benefits of LuSe

- Airport
  - less storage cost
  - less damages payments/ „Which airport is responsible?“
- Customer
  - more exact and quicker search process, more information on results
  - can influence further route of luggage
- Insurance
  - less payments
- Airline
  - better customer satisfaction

Quality assurance aspects in developing a component-based system

Overview

- Siemens/PSE environment
- Project definition
- QA requirements
- e-business architecture
- Technological requirements
- Lessons learned

Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

QA requirements/Constructive

- Business model
  - what is the optimum for provider, supplier and user
- Q characteristics
  - which of them must be fulfilled to make the business model work
- e-business architecture
  - with which architecture can the Q characteristics be accomplished
- Technology
  - which technology matches the selected e-business architecture
- Deployment
  - how is the selected technology deployed best

Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

QA requirements/Q characteristics

- Focus quite different at SAPA:
  - Availability: 365 days a year and 24 hours a day
  - Performance highest priority; nobody, particularly not the new “users” like to wait
  - Functionality: full support for transaction security (ACID: Atomicity, Consistency, Isolation, Durability) because of bookings
  - User friendliness: irrelevant here as GUIs of CRS and PMS are used

Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

QS requirements/Methodology

- stdSEM must be supplemented with an e-project management part
- Vision/strategy shared by all partners
- Uniform development methodology used by all partners
- Buy/make “smaller components”
- Short development cycles
- Clear specifications for integration
- Clear specifications for acceptance

Siemens / PSE environment

Project definition

QA requirements

e-business architecture

Technological requirements

Lessons learned

QA requirements/Acceptance

Concrete example: SAPA

- SAPA components in CRS (Computerized Reservation System)
- Agent platform – developed by an OEM
- SAPA components in PMS (Property Management System)

Siemens / PSE environment

Project definition

QA requirements

e-business architecture

Technological requirements

Lessons learned

QA requirements/Deployment

- system is being used on 3 different servers
- must interoperate with the existing SW

Quality assurance aspects in developing a component-based system

Overview

- Siemens/PSE environment
- Project definition
- QA requirements
- e-business architecture
- Technological requirements
- Lessons learned

Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

The importance of architecture for e-business solutions

- Several challenges:
  - Usage unpredictable, thus no performance forecasts possible
  - Difficulties relating to manageability and security
- To create robust solutions, it is necessary to cobble together various products (or their subcomponents)
  - that are in a constant state of evolution
  - both in terms of technical and market faces

E-Business
Common pitfalls in designing e-business applications without an overall architectural concept / 1

- Making purely technical choices in setting up a Web solution
  - Leaves a brittle solution with inadequate performance and scalability
- Focussing on a single product and the features it offers
  - Influences the design to an extent that compromises performance, scalability and flexibility

E-Business
Common pitfalls in designing e-business applications without an overall architectural concept / 2

- Architecture is an ongoing, solutions-oriented and component-based philosophy
- The goal is to build solutions that optimize both current performance and future flexibility

E-Business
Benefits of an architectural approach / 1

- Shorter time-to-market
  - Avoiding integration threads that need constant manipulation and updating
- Better protection of investments against technology shifts
  - Avoiding “getting trapped” in obsolete e-business technology
- Ability to scale with increasing demand
  - Avoiding unanticipated spikes in demand

E-Business
Benefits of an architectural approach / 2

- Achieving robustness and continuity of the solution while at the same time minimizing disruption to the existing architecture
  - However, changes or upgrades of infrastructure elements are inevitable
- Minimizing personnel and maintenance costs
  - Existing personnel can concentrate on value-added services

E-Business
Benefits of an architectural approach / 3

- Providing an architecture that adapts to business change
  - Any kind of business-oriented solution must be built with continuous change in mind
- Enterprise is enabled to leverage technologies of the past, present, and future
  - Providing freedom to enlist emerging technologies as they become available

E-Business
Benefits of an architectural approach / 4

- Basis for enlightened make or buy decisions
  - Clear understanding of the relationship between the various components
- Freedom from extensive coding and multiple middleware services
  - Developers can identify and implement the right Enterprise Application Integration (EAI) technology for Internet applications

Electronic Business
Creating an e-business architecture/Identification of key requirements / 1

- What type of applications is being built?
  - Batch, continuous, interactive, transactional, rule-based, expert system, simulation, real-time, workflow, reporting, publish & subscribe
- Which features are required?
  - Scalability, security, reliability, ACID properties, integrity, flexibility, portability, manageability, performance

Electronic Business
Creating an e-business architecture/Identification of key requirements / 2

- What is already available today?
  - Existing platforms, products and protocols
    - Platforms – operating systems
    - Products – SW development environments, databases, middleware, application and web servers, network and application management products, application packages
    - Protocols – network operating systems (Novell, TCP/IP, SNA), RPC, message queuing, Corba, distributed DB protocols

Electronic Business
Creating an e-business architecture/Identification of key requirements / 3

- Existing technology
  - home-grown application
  - third-party SW or packaged application
  - mobile computing devices
- Integration could take place on the
  - API level
  - protocol level, through a third-party or home-grown gateway
  - database level

Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

Validating the e-business architecture / 1

- Typically involves:
  - Architecture review
  - Prototype
  - Pilot application

Siemens /
PSE environment

Project definition

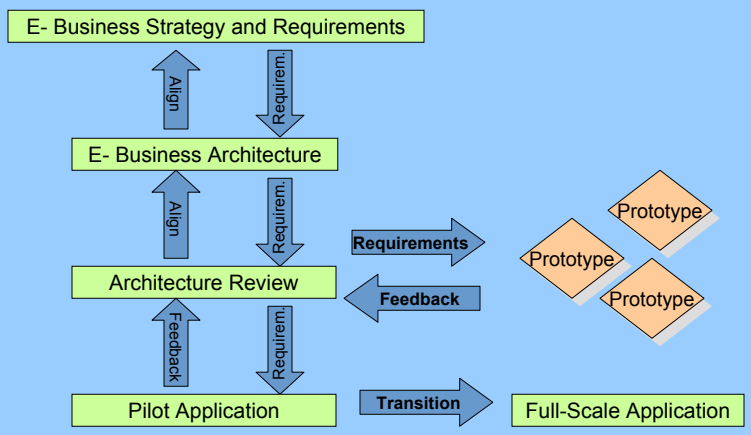
QA requirements

e-business
architecture

Technological
requirements

Lessons learned

Validating the e-business architecture / 2
E-business solution development lifecycle



Validating the e-business architecture / 3
Architecture review/committee

- is conducted by a small, focused steering committee
  - comprised of architects, security administrators, business analysts/subject matter experts
  - database administrators, and systems and network administrators

Validating the e-business architecture / 4 Tasks

- understand the overall system requirement
- understand the acceptable risk levels
- isolate the core business processes that are shared by other parts of the business
- identify areas of high data volume, high transaction volume and high availability
- isolate technology and components within the architecture into known and unknown entities

Validating the e-business architecture / 5 Further factors to consider

- business alignment
- core business components
- technology readiness
- technology standards
- system integration
- cost of ownership
- cost containment

Validating the e-business architecture / 6
Prototype / 1

- Mitigation of risks
- Technological requirements and constraints
- Is the performance acceptable within the confines of the requirements?
  - Does it exhibit the ability to scale?
  - Does the fail-over functionality work, and is it reliable?
  - Is the communication protocol able to securely traverse through firewalls or proxy servers?

Validating the e-business architecture / 7
Prototype / 2

- Does it integrate or interoperate with legacy systems?
- Does the stability of the technology measure up to expectations?
- What is the level of complexity involved in understanding and using the technologies in question?
- Do the technologies work together?
- Budgetary constraints
- Does the solution stay within the allotted budget?

Validating the e-business architecture / 8 Pilot application

- Validation of the implementation of an e-business strategy on a small scale
- Validation of the technical solutions of the e-business architecture and the application
- A prototype validates a specific technology
- A pilot application validates the overall business solution

Implementing the e-business architecture

- Define the skill and team requirements
  - Component-based architecture is centered on OO technology
  - The team should consist of an architect, a data/database analyst, and a business analyst

Siemens / PSE environment

Project definition

QA requirements

e-business architecture

Technological requirements

Lessons learned

Implementing the e-business architecture Designing a new e-business application using the architecture / 1

- Use-case methodology coupled with the Unified Modelling Language Notation
- System Engineering Method (SEM)
- Design and Development Environment (Rational Rose Visual Modeling Tool, StP, ...)

Siemens / PSE environment

Project definition

QA requirements

e-business architecture

Technological requirements

Lessons learned

Implementing the e-business architecture Designing a new e-business application using the architecture / 2

- Design pattern
  - a recurring design solution to the same problem (ranging from user interface to business logic)
- Creating and reusing business components
  - components encapsulate a collection of logically related business objects
  - these components seamlessly communicate among themselves through published interfaces (CORBA, COM, RMI)

Siemens / PSE environment

Project definition

QA requirements

e-business architecture

Technological requirements

Lessons learned

Implementing the e-business architecture Designing a new e-business application using the architecture / 3

- Compliance with standards
- Scalability and performance
  - Component-based design
    - provides the ability to replicate identical copies across physical computers to accommodate growth in accordance with business needs and budgetary constraints
    - components can also be redistributed across physical computers for the same reason

Siemens / PSE environment

Project definition

QA requirements

e-business architecture

Technological requirements

Lessons learned

Implementing the e-business architecture Designing a new e-business application using the architecture / 4

- Stateless servers
  - the design of session management should coincide with the stateless server design to ensure cohesive scalability
  - scale best and work best with network load balancing
- Facilitating the deployment phase
  - avoid any special software customization when an e-business partner, vendor, supplier, or customer signs up

Implementing the e-business architecture Designing a new e-business application using the architecture / 5

Application design

- With the help of the use-case methodology
  - business requirements are captured through a series of interviews with users and stakeholders
  - the interviewees could include staff members from business partners

Implementing the e-business architecture Designing a new e-business application using the architecture / 6

- Discussion of issues such as access security, business events, growth plans, usage platforms, operation requirements, and capacity planning requirements
- Interactive development process
  - Chunks of logically related requirements are designed, built, tested, and assembled progressively

Implementing the e-business architecture Designing a new e-business application using the architecture / 7

The team

- will need to perform only minor configuration changes
- can quickly replicate or redistribute business components across computers
- The use of XML technology allows companies to securely share information with new business partners without requiring any special software or hardware deployment

Implementing the e-business architecture Designing a new e-business application using the architecture / 8

- Integration with legacy systems
  - can occur either at the application level or at the database level
    - these points of contact may form bottlenecks and/or single points of failure
    - replicate these interface points

Implementing the e-business architecture Designing a new e-business application using the architecture / 9

- Retiring older applications
  - If the e-business application enhances much of the business logic in an older legacy system
    - consider phasing out the older system in favor of the new
    - to avoid duplicated maintenance and confusion

Implementing the e-business architecture Designing a new e-business application using the architecture / 10

- Extending existing business models using wrapper codes
- Extending existing business models using gateways (e.g. Java gateway for CICS)
- Partially replacing existing business models
- Scalability and performance
- Technology constraints

Siemens /
PSE environment

Project definition

QA requirements

e-business
architecture

Technological
requirements

Lessons learned

e-business deployment

- Change management
- Component-based deployment
- Security
- Concerted deployment
- Software and data migration
- Training
- Fine-tuning the application
- Documenting the environment

Deployment issues / 1

- Change management
  - divide roles among e-business, operating systems, and the database
  - software upgrades
  - changes to e-business components
  - new browsers
  - keep several versions running while a migration plan is put in place
  - hardware replacement
  - operating system upgrades
  - third party software rollouts

Deployment issues / 2

- Component-based deployment
  - must pass stringent quality tests to ensure zero defects
  - any version upgrades or retiring of components must be carefully orchestrated

Deployment issues / 3

- Security
  - Firewalls block protocols such as the Internet Inter-ORB Protocol, Java Remote Method Protocol, and RPC from passing through unless specifically allowed by security administrator
  - Some organizations resort to tunnelling these protocols through HTTP, which, however, causes considerable performance problems
  - Many organizations prohibit downloading unsigned Java applets and Active X controls

Deployment issues / 4

- Block access to unauthorized web sites
- Perform anti-virus scans on web servers, e-mail servers, and desktops
- Forbid dial-outs to Internet service providers from within the organization
- DMZ (demilitarised zone)network provides a safe area for communication
  - taking place between internal and external systems
  - any damage caused by hackers is confined within the DMZ
  - sensitive components should not be deployed in the DMZ

Deployment issues / 5

- Concerted deployment
  - Many e-business solutions interoperate with other systems
    - These applications may need to be deployed in unison

Deployment issues / 6

- Software and data migration
  - May require “data scrubbing”
    - Is typically done programmatically, in some cases human intervention will be required

Deployment issues / 7

- Training
  - Operators and help desk personnel
    - Probably not familiar with the new systems
- Fine-tuning the application
  - Use metrics to continuously gauge the success
- Documenting the environment

Siemens / PSE environment

Project definition

QA requirements

e-business architecture

Technological requirements

Lessons learned

Production management for e-business applications

- New roles and responsibilities
- Management tools
- Troubleshooting

Production management for e-business applications / 1

- New roles and responsibilities
  - security, administration, capacity planning, architecture, technology strategies, and business strategies
  - e-business begins to force a high dependency between business and IT
- Management tools
  - business component deployments, remote problem tracking and notification, and better sharing of application management data between business partners

Production management for e-business applications / 2

- Troubleshooting
  - leads to a high level of dependency on relationships
  - numerous business partners any of whom may identify new problems in the areas of performance, availability, and connectivity
  - will require close co-operation among business partners

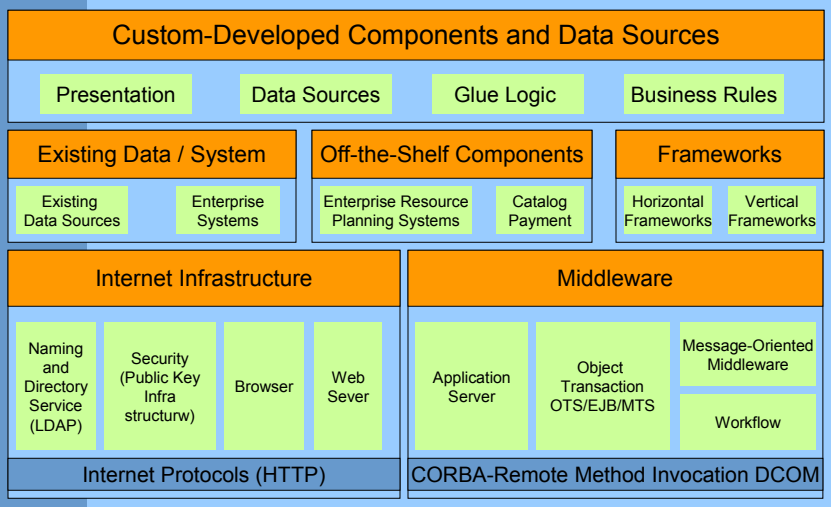
Quality assurance aspects in developing a component-based system

Overview

- Siemens/PSE environment
- Project definition
- QA requirements
- e-business architecture
- Technological requirements
- Lessons learned

SIEMENS

E-business technologies



SIEMENS

Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

Technologies used

- Corba required for
  - Encapsulating legacy systems
  - „Highway“ for agents
- Software agents required because of their properties:
  - Autonomous
  - Pro-active (goal-based)
- XML required because of:
  - Standardized data exchange
- SOAP enables
  - Interoperability with other distributed systems
- EJB facilitates for the programmer all problems with
  - Transaction acidity
  - Load balancing
- JSP/Servlets/Java Applets/JDBC
  - Enables database access from a web site
  - Only used for test system

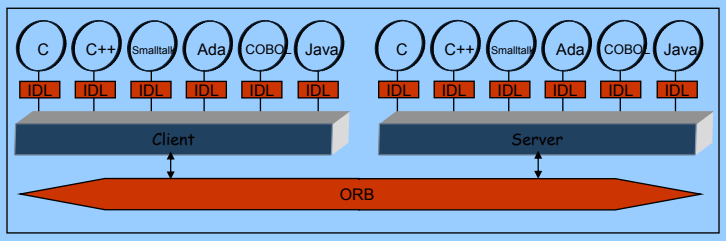
CORBA / 1
OMG (Object Management Group) / 1

- Largest software consortium in the world
  - More than 800 members
    - system suppliers such as Siemens, IBM, HP, DEC, Oracle...
    - but also “end users”, such as Boeing, AT&T, Daimler Benz...
    - from all fields, such as telecom, health, finances...
- through IIOP (Internet Inter ORB Protocol)

CORBA / 2
OMG (Object Management Group) / 2

- Mission:
  - To offer a common platform-independent framework architecture for the development of object-oriented applications on distributed computer systems: OMA (Object Management Architecture)
- CORBA (Common Object Request Broker Architecture)
  - Core part of OMA, synonymous with system architecture
  - Language-independent through IDL (Interface Definition Language)
  - Platform-independent

CORBA / 3
Independence of programming languages



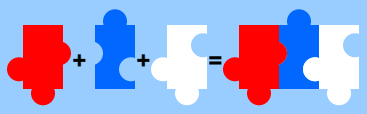
- IDL lets you specify the following:
  - The methods supported by your interface, including input/output parameters and data types
  - Attributes of a component
  - Classes to inherit from
  - Exceptions to be generated
  - Events

CORBA / 4
Distributed objects according to CORBA

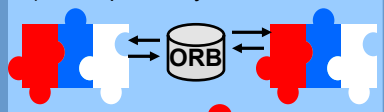
- Clients do not need to know
  - where the distributed object is
  - under which operating system it executes
  - how the server object is implemented
- Clients only know
  - the interface of the server object

CORBA / 5
Properties of components

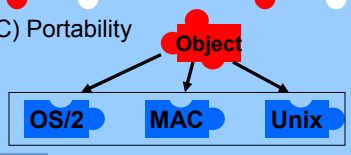
A) Plug-and-Play



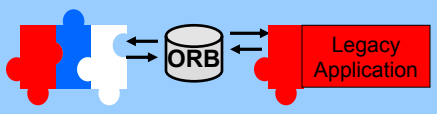
B) Interoperability



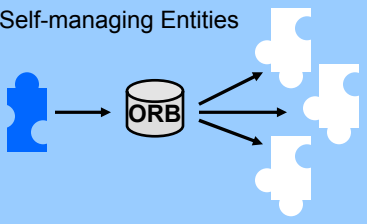
C) Portability



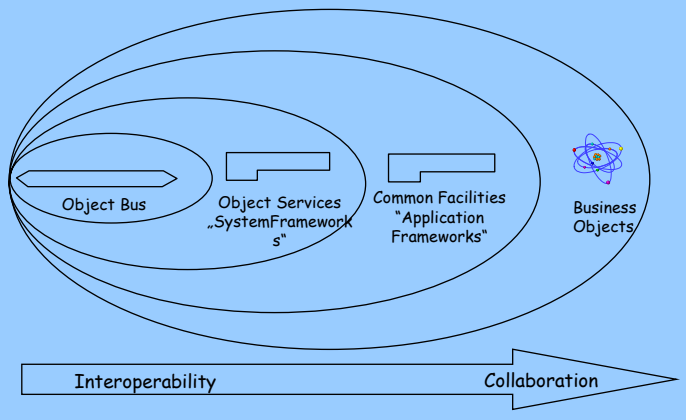
D) Coexistence



E) Self-managing Entities



CORBA / 6
Goal: Business Objects



CORBA / 7
Pros

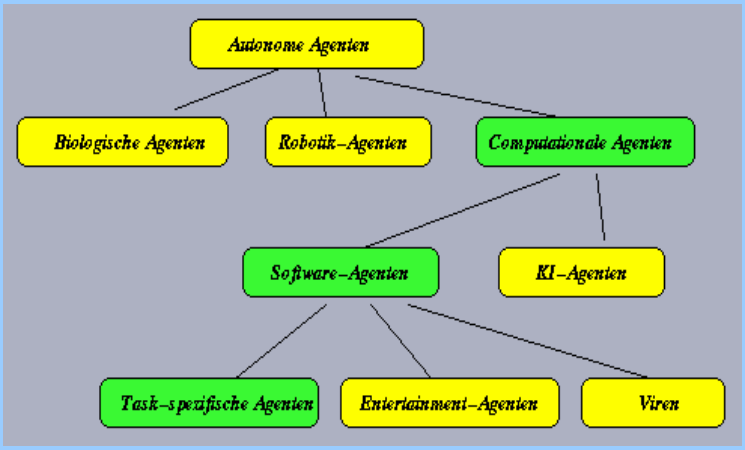
- Fast access to known objects (defined in IDL)
- Standardized IDL and ORB infrastructure
- CORBA is being widely used as backbone for server applications.

CORBA / 8
Cons

- No independent adjustment to changing databases (special events, coordination tasks)
- Very strict access rules (no fuzzy searching possible)
- No offline operation (e.g. monitoring and waiting for changes)
- No learning mechanism (customer behavior has to be recorded by the server software)

Source: http://ivs.cs.uni-magdeburg.de/~dumke/STV/ST3-06.html

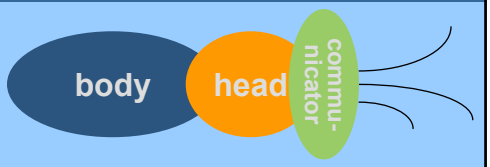
Agents / 1
General classification of agents



Agents / 2
Properties

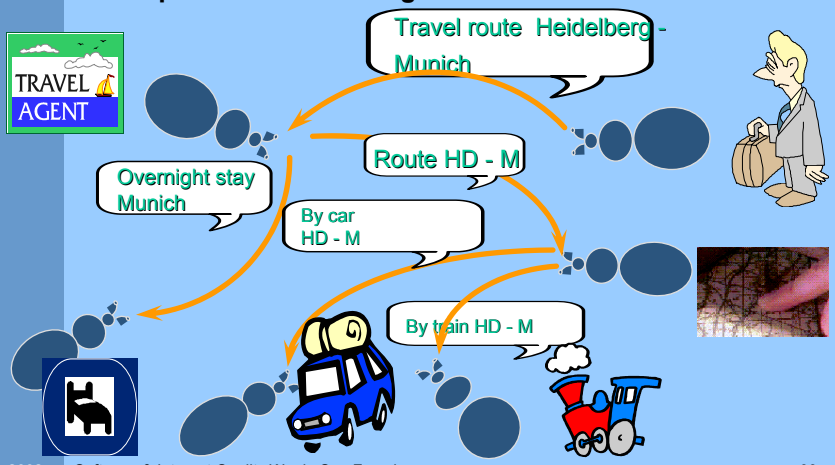
- autonomous
  act on their own
- perceptive
  perceive their environment
- re-active (event-based)
  react to changes in their environment
- pro-active (goal-based)
  attempt to change the real world
- cooperative
  cooperate to achieve common goals as well as their own goals
- adaptive
  learn from experience

Agents / 3
What is an agent?



- Body
  - application-specific basic functionality of the agent
  - can be an existing system with an API
- Head
  - manages the agent's goals, plans
  - controls the body via the API
  - manages cooperation with other agents
- Communicator
  - is responsible for exchanging messages with other agents
  - is based on traditional communication protocols, e.g. TCP/IP

Agents / 4
Cooperation between agents



Agents / 5
Pros - cons

- Agent technology pros
  - automation
  - adaptability
  - feedback
  - learning
  - teaching help functions
  - remote control
- Agent technology cons
  - danger through exaggeration
  - user has to adjust to agent control
  - requirements for business model design
  - security
  - confidentiality

Agents / 6
Summary

- By providing specific services, the agent system takes over a number of coordination tasks in the communication between the partners
- The agent system's implementation provides for a standardization of communications
- Search and negotiation rules allow fuzzy searches
- The agent system ensures easy scalability and strictly delimits responsibilities between the partners
- The agent system allows for a high degree of automation and provides a uniform interface.
- Agents are prepared for future use in combination with other user interfaces (voice input, mobile telephony, PDAs)
- The agent system supplements the Corba backbone by allowing to better connect to dynamic heterogeneous and globally distributed databases

Application servers / 1

- Application servers consist of:
  - a combination of several different technologies, including
    - Web servers, ORBs, MOM, DBs...
- Application servers can refer also to technologies other than distributed objects
- We focus only on the ones based on distributed object technology
  - such as CORBA, Java RMI or COM+

Application servers / 2
ORBs / 1

- feature varying degrees of complexity
  - the simplest ones make it possible to connect
    - client applications and
    - distributed objects

    } are called ORBs
  - make it easy to find and use objects distributed on client
  - are less well suited for transaction controlled environments with high data volumes

Application sserver / 3 ORBs / 2

- provide a communication backbone for distributed objects
  - but normally not the robust infrastructure required to support large numbers of users and mission-critical operations
  - application developers must access services such as transaction, persistence, multi-threading on their own

Component transaction monitors

- are application servers that have developed from a mixture of
  - traditional TP monitors and
  - ORB technologies
- provide infrastructure able to automatically manage
  - transactions, object distribution, multi-threading, security, persistence, and resources (=Corba services)

Enterprise Java Beans Definition

- is a server-side standard component model for component transaction monitors
- Original definition by SUN:
  - is a component architecture for the development and use of component-based business applications
  - applications written using the Enterprise Java Beans architecture are scalable, transaction-oriented, and multi-user capable
  - once written, such applications can be used on any server platform supporting the Enterprise Server Beans specification

Distributed objects / 1

- Different parts of a system can run on different computers
- Business logic and data can be accessed remotely
- Customers, business partners, and other remote parties can use a business system from virtually any place at any time
- The latest development in distributed processing are distributed objects

Distributed objects / 2

Distributed object technologies such as

- Corba
- Java RMI
- COM+

make it possible to

- run objects on one computer,
- which are then used by client applications on other computers

Distributed objects / 3

- Distributed objects have developed from an old type of 3-tier architecture
- which is used in TP monitor systems such as CICS by IBM or Tuxedo by BEA
- Such systems separate the representation, the business logic, and the database into three separate layers

Distributed objects / 4

- Distributed object technologies make it possible to
  - replace procedural COBOL and PL/1 applications on the middle layer with business objects
- Often, more complex, multi-tier architectures are used
  - here, different objects are located on different servers and interact to complete a task
- Enterprise Java Beans make it particularly easy to create such n-tier architectures

Server-side components / 1

- OO languages such as C++, Java and Smalltalk are used to
  - write SW that is flexible, scalable, and reusable
    - the three axioms of OO development
- In business systems, OO languages are used to
  - improve GUI development
  - simplify data access
  - encapsulate the business logic
- Encapsulating the business logic into business objects is the latest focus of the IT industry

Server-side components / 2

- Business is always moving; the products, processes, and goals of a company are bound to change in the course of time
- If it is possible to encapsulate the software that models a business into a business object, the software will then be flexible, scalable, and reusable and thus be able to develop on its own in line with the business

Server-side components / 3

- A server-side component model defines
  - an architecture for developing distributed business objects
- and combines
  - the accessibility of distributed object systems with
  - the changeability of the business logic in the form of an object
- Server-side component models are used on the application servers of the middle layer
  - that manage the components at runtime and make them available to remote clients

Server-side components / 4

- can be sold and purchased as independent chunks of executable software, just like any other component
- comply with a standard component model and can be executed without direct modification on a server that supports this component model
- often support attribute-based programming
  - the run-time behavior of operational components can be modified
  - without changes to the component's program code

Server-side components / 5

- Depending on the component model, the server administrator is able to set the behavior of a server-side component
  - with respect to transaction, security, and even persistence
  - by assigning certain values to these attributes

Server-side components / 6

- When new products are being developed and corporate processes change
  - it is possible to re-assemble, change, and extend server-side components in such a way
  - that the business system will reflect these changes
- A business system can be regarded as a collection of server-side components that model concepts such as customers, products, reservations, warehouses, etc.

Server-side components / 7

- Each component is like a building block that can be combined with other components to form a business logic
  - Products can be stored in a warehouse or delivered to a customer
  - A customer can make a reservation or buy a product
- You can assemble or disassemble components, use them in other components, and change their definitions
- A business system based on server-side components is
  - flexible because it consists of objects, and
  - accessible because the components can be distributed

Enterprise Bean / 1
Component

There are two different types:

- Entity Bean
- Session Bean

Enterprise Bean / 2
Entity Beans

- model business processes that can be expressed in the form of nouns
  - for example: customer, piece of equipment, entry in stock list, location,...
- thus model objects from the real world
  - hence are persistent data records in some kind of database

Enterprise Bean / 3
Session Beans

- are an extension of the client application and are responsible for managing processes or tasks
  - for example: they are typically used to manage certain activities such as a reservation; in doing so, they rely on Entity Beans
- All these operations are reflected in the database by actions being performed on the corresponding Entity Beans

Classes and interfaces / 1

- In order to implement an Enterprise Bean, it is necessary to define two interfaces and one or two classes
  - Remote interface
  - Home interface
  - Bean class
  - Primary key class

Classes and interfaces / 2
Remote Interface

- defines the business methods of the beans
  - i.e. the methods offered in the real world to complete a task
- extends a javax.ejb.EJBObject, which in turn extends java.rmi.Remote
- the undefined entity that actually implements this interface is called an EJB.Object

Classes and interfaces / 3
Home interface

- defines the lifecycle methods of the beans
  - Create, Remove, Find
- extends javax.ejb.EJBHome, which in turn extends java.rmi.Remote
- the object that implements the home interface is called the local EJB object

Classes and interfaces / 4
Bean class

- implements the business methods of the beans
  - but normally not their remote or home interfaces
- must, however, feature
  - methods with signatures that match the ones in the remote interface, and
  - methods that match some of the methods of the home interface
- an Entity Bean must implement javax.ejb.EntityBean
- a Session Bean must implement javax.ejb.SessionBean

} Both extend javax.ejb.EnterpriseBean

Classes and interfaces / 5
Primary key

- makes a database pointer available
  - only Entity Beans require a primary key
- must implement java.io.Serializable

Classes and interfaces / 6
Context

- Enterprise Beans are something in between a client software and a database
  - the client never interacts directly with a bean class
  - always uses just the home and remote interfaces of a bean
  - interacts with automatically generated stubs
- A bean that needs the services of another bean thus is another client
  - using the same stubs instead of interacting directly with the bean class

Web services

Navigation 1
Navigation 2
Navigation 3
Navigation 4
Navigation 5
Navigation 6
Navigation 7
Navigation 8

- a web service is a collection of functions
  - that can be published, located, and invoked across the Web
- these functions can be anything from simple requests to complicated business processes
- once a web service is deployed
  - other software (including other web services) can discover and invoke the deployed service dynamically at runtime
- So, in a nutshell, Web services are components for the truly distributed era.

Web services - Standards

- SOAP — Simple Object Access Protocol
- WSDL — Web Service Description Language
- UDDI — Universal Description, Discovery, and Integration

SOAP/1

- is a lightweight protocol based on XML
  - for the exchange of information in a decentralized environment
  - typically across the Internet or intranets.
- there are three key parts of SOAP

SOAP/2
Key parts

- an envelope that defines a framework for description
  - what is in a message and how to process it
- a set of encoding rules for expressing
  - instances of application-defined data types
- a convention for representing
  - remote procedure calls (RPCs) and responses

SOAP/3

- SOAP plays a role similar to
  - CORBA‘s Internet InterORB Protocol (IIOP)
  - COM‘s RPC
  - Java‘s Remote Method Invocation (RMI)
  - HTTP in ordinary Web browsing

SOAP/4

- SOAP is intended to be simple enough for a programmer to be able
  - to implement in a couple of days
  - using the favorite programming language and operating system
- this simplicity comes from deliberately neglecting the more complex parts of other distributed computing protocols

SOAP/5

- Although the choice of HTTP and XML encoding makes it look anything but simple
  - SOAP clients really just send XML call messages over HTTP
  - to servers and can get XML response messages back in return.

WSDL

- WSDL is an XML-based specification schema
  - for describing contracts between a set of Web services that are to exchange messages
- it declares the what, where, and how of the Web services
- with a role very much like that of the IDL in Corba and COM+
- WSDL plays a critical linking role, conferring the self-definition property of Web services

UDDI/1

- UDDI standards define how to build and use a directory of Web services
- the UDDI directory is an online means
  - of consistent publishing and consuming of Web services
- it addresses the key question of how to actually find web services
  - using different selection criteria

UDDI/2

- UDDI provides a set of XML documents
  - for describing and classifying Web services.
- these services are described in Internet registries that can be searched
- the focus is on middleware connectivity and using XML itself
  - to describe the system that companies use to interface with one other
- UDDI is based around the concept of standard registry services
  - that provide yellow, white, and green page business-centric functionality

UDDI/3

- White pages
  - provide general information about businesses, such as names, addresses, and contact details
- Yellow pages
  - let clients search for businesses that provide services to particular industry segments or offer specific services

UDDI/4

- green pages
  - provide technical information on services being offered
    - including how to call them and where they are located
    - this information is returned as a „Type model“ (tModel)
  - the t Model is basically the same as the WSDL description for the service.
  - the t Model includes information about
    - the location of the service
    - the service name
    - the operations it supports, and their parameters

Quality assurance aspects in developing a component-based system

Overview

- Siemens/PSE environment
- Project definition
- QA requirements
- e-business architecture
- Technological requirements
- Lessons learned

Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

Lessons learned / Project management

- One overall project manager and one QA manager
- Three subprojects
  - The second subproject was designed “only” for the procurement and acceptance testing of the Aglet system (OEM)

Siemens / PSE environment
Project definition
QA requirements
e-business architecture
Technological requirements
Lessons learned

Lessons learned / Responsibilities of the overall project manager

- Coordination of joint approach/vision with managers from partner companies
- Definition of useful deliverables in agreement with the customer
  - e.g. vacancy query that is available to employees of a partner company

Siemens /
PSE environment

Project definition

QA requirements

e-business
architecture

Technological
requirements

Lessons learned

Lessons learned / Subprojects

- Each subproject had its own organization and functionality
  - No competition with other subprojects
- Q feature requirements resulted in
  - High-priority focus on test cases and coordination with overall project management

Siemens /
PSE environment

Project definition

QA requirements

e-business
architecture

Technological
requirements

Lessons learned

Lessons learned /
Q characteristics (availability, performance)

- Choose architecture to meet these requirements
- Technologies, too
- Reviews/tests/prototypes, too
- Encourage use of already developed components (filters, agent platform, PMS client) that only need to be customized
  - agreed quality is achieved
  - considerably lower error rate
  - errors detected very early on
  - error diagnosis and clearance considerably easier

Siemens /
PSE environment

Project definition

QA requirements

e-business
architecture

Technological
requirements

Lessons learned

Lessons learned / Acceptance testing

- 3 sub-acceptance test were required
- a special team was formed to deal with overall acceptance testing
- security aspects played a major role
  - EJB support for acceptance testing of bookings on hotel PCs

SIEMENS

Program and System Engineering PSE

**Thank you
for your attention!**

QW2002 Paper 3T1

Dr. Mark Blackburn, Mr. Robert Busser & Aaron Nauman (T-VEC Technologies/SPC, Inc.) & Dr. Ramaswamy Chandramouli (National Institute of Standards and Technology)

Interface-driven Model-based Test Generation of Java Test Drivers

Key Points

- Test Automation Technology and Experience
- Interface-driven Model-Based Test Automation
- Java and SQL Test Driver Generation for Testing Database Functionality

Presentation Abstract

The combination of model-based verification and test automation has helped reduce cost, provide early identification of requirement defects, and improve test coverage [RR00; KSSB01; BBNKK01; BBN01d; Sta00; Sta01]. This paper extends prior work in model-based verification and describes an interface-driven analysis approach that combines requirement modeling to support automated test generation of Java test scripts for executing against a database. The paper describes the concepts of models and test driver mappings using examples for testing security functionality of an Oracle database using Java and standard Structured Query Language (SQL) test drivers. Recommendations are provided for performing the modeling of textual requirements in conjunction with interface analysis to support reuse of models and their associated test driver mappings. These recommendations were derived while extending an early experimental model of one small set of related requirements to several other groups of interrelated requirements. The resulting insights are useful for understanding how to scale models and the associated test driver mappings to an industry-sized verification project. Although the modeling and testing is focused on security capabilities of a database, the results and recommended approaches are general for testing any application.

About the Author

Dr. Blackburn is a Software Productivity Consortium Fellow, President of T-VEC Technologies, Inc. and co-inventor of the T-VEC system. He has twenty years of software systems engineering experience in development, management and applied research of process, methods and tools. He is currently involved in consulting, strategic planning, proposal and business development, as well as developing and applying methods for model-based approaches to support requirement defect removal and test automation. He has also been involved in applied research and technology demonstrations in requirement and design specification, formal methods, and formal verification, object technology, web-based knowledge engineering, domain engineering, and reverse engineering. He earned a BS in Mathematics from Arizona State, MS in Mathematics from Florida Atlantic University, and a Ph.D. in Information Technology from George Mason University.

Mr. Robert D. Busser is co-founder of T-VEC Technologies, Inc. and co-inventor of the T-VEC system. He has over twenty years of software systems engineering experience in development, and management in the area of advanced software engineering, and expertise in software engineering processes, methods and tools. He is the chief architect of the T-VEC system. He has extensive experience in requirement and design methods, real-time systems, model-based development and test generation tools, model analysis, and verification. He has extensive knowledge about model transformation systems, theorem prover and constraint solving systems. In addition, he has extensive avionics engineering experience and has been involved in several FAA certifications. He has experience applying this knowledge in the development of highly-reliable software systems and the development of state of the art requirements-based software modeling and testing technologies. Mr. Busser has a B.S. in Electrical and Electronics Engineering from Ohio University.

Mr. Aaron M. Nauman has a wide range of systems and applications development experience in both real-time (telecommunications) and information systems domains. He is currently involved in the development of model transformation, and software verification through specification-based automated testing. His experience includes all aspects of product development from requirements analysis through test implementation. Additionally, he has experience in object-oriented technologies, distributed and client/server systems, web-based and components-based software and systems integration. He is a representative on the OMG UML Action Semantics working group. Mr. Nauman graduated Summa Cum Laude from North Carolina State University with a B.S. in Computer Science.

Dr. Ramaswamy Chandramouli is a computer scientist at NIST with over 15 years of experience in both Private Sector and Federal Agencies. His professional interests include Distributed System Security, Access Control Models and Security Specifications. He was one of the authors of “Role Based Access Control Protection Profile” which was the first Common Criteria (V 2.0) Protection Profile to be certified in the U.K. He was also the lead author of the paper titled “Comparison of Role Based Access Control Features in commercial DBMS” which won the Best Professional Paper award at the the 21st National Information Systems Security Conference held at Crystal City, VA, Oct 1998. Dr.Chandramouli holds an MS degree in Operations Research from the University of Texas and a Ph.D. in Information Technology from George Mason University.

Interface-Driver Model-based Test Generation of Java Test Drivers

Mark Blackburn, Robert Busser, Aaron Nauman
T-VEC Technologies/SPC

Ramaswamy Chandramouli (Mouli)
National Institute of Standards and Technology

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Common Symbols, Abbreviations and Acronyms

Tool

Manual process

Machine readable artifact

Textual document

Object mapping

Tabular model

API	Application programming interface
COTS	Commercial off-the-shelf
GUI	Graphical user interface
Java	High-level programming language
JDBC	Java Database Connectivity
MCDC	Modified Condition Decision Coverage
NIST	National Institute of Standards and Technology
NRL	Naval Research Laboratory
ODBC	Open Database Connectivity
Perl	High-level programming language
TAF	Test Automation Framework
TTM	T-VEC Tabular Modeler
SCR	Software Cost Reduction
SQL	Structured Query Language
SRS	System/software requirement specification
UML	Unified Modeling Language

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Outline

- Background/context
  - Benefits achieved through use on other applications
- General description of process-oriented perspectives of the method, tools and roles
  - Lifecycle perspective
  - Generic discussion of interfaces
  - Models
  - Test driver support
- Example requirements and interfaces for database security
- Other applications
- Summary and conclusions

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Background/Context

- National Institute of Standards and Technology (NIST) initiated program to develop methods and tools for automating Security Functional Testing
  - Provided end-to-end support including model development, model analysis, automated test generation, automated test execution in multiple environments, and results analysis
- NIST expanded scope of requirements to assess scalability
- Applied best practices method from use on prior large projects
  - Develop verification models based on refinement of requirements in terms of component interfaces
  - Verification/test engineer work in parallel with developer to refine requirements, stabilize interfaces while identifying requirement defects
    - Parallels concepts of eXtreme Modeling (like eXtreme Programming)
  - User tabular modeling – easiest to learn and use

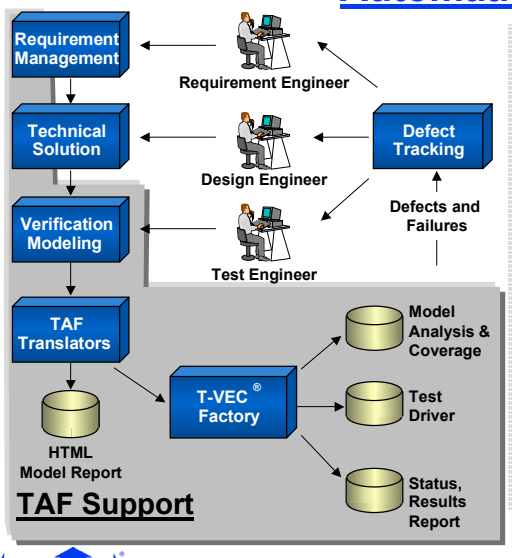
Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Contributions

- Pragmatic guidance for combining interface analysis and requirement modeling
- Recommendations for defining interfaces that provide better support for testability are valid for all forms of testing
- Interface-driven modeling has benefits for testing a released product, but it has been applied during development with many additional benefits
  - Ideally, test engineers work in parallel with developers to stabilize interfaces, refine requirements, and build models to support iterative test and development
  - Test engineers write requirements for products (which in some cases are very poorly documented) in models, as opposed to hundreds or thousands of lines of test scripts

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Test Automation Framework Life Cycle Automation



User Benefits
Full Life Cycle Automation

- LM Aero
  - Early defect discovery reduces rework
  - Up to 50% Reduction in Test Planning
  - Up to 90% Reduction in Test Creation
- LM Astronautic - Mars Polar Lander Bug
- Rockwell - 52% Lifecycle Cost Reduction
- Member - 50% Reduction in Verification for Life Critical System
- Requirement Management
- Drives Early Stabilization of Interfaces
- Model Analysis
  - Early identification of model defects (inconsistencies, contradictions)
- Test Vector Generation
- Test Driver Generation
  - For host, target, or simulation
  - Available for most any language
  - Supports early model validation
- Automated results comparison
- Defect Tracking
- Requirement-to-test traceability

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Demonstrated to Work in Other Application Domains

- Web-applications (client) interface to server (database)
  - Models built in TTM (SCR) and test drivers generated in WinRunner scripting language for web applications
- Client/server mainframe (database) application
  - Models built in TTM (SCR) and test drivers generated in DynaComm/Elite (3270 terminal emulation) scripting language for PC clients
- Mars Polar Lander
- Life critical: medical, aerospace, avionics
- Non critical: telecomm, automotive
- Defense

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

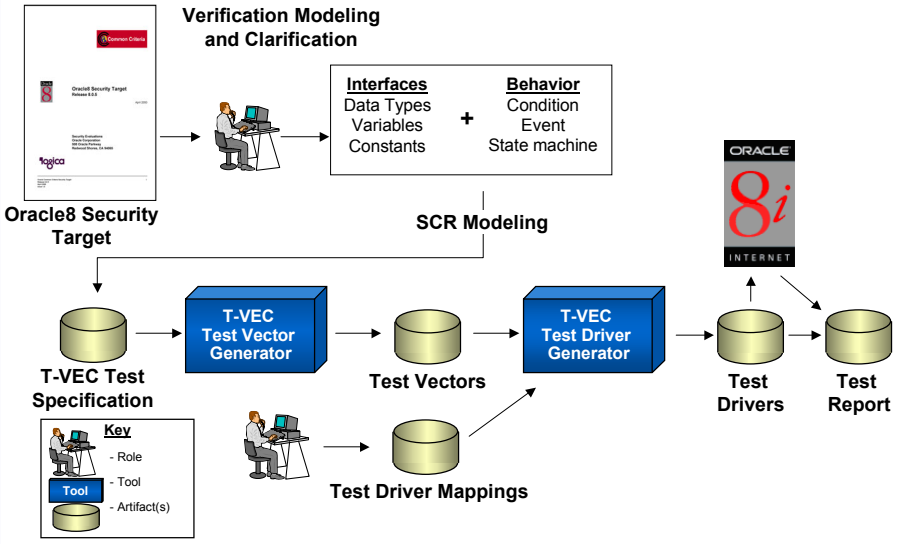
Key Results and Benefits*

- Found Mars Polar Lander bug
- Early requirement defect identification reduces rework
- Testable requirements help eliminate rework
- Test planning time reduced by up to 50%
- Automated test generation reduces manual effort by up to 90%*
- Test cases development and execution optimized to eliminate test redundancy
- Known level of requirements coverage can be planned and measured
- **Significant cost reductions**

***Source:** Safford, Key applications of Test Automation Framework (TAF) , Twelfth Annual Software Technology Conference, 2000.
Kelly, et. al., Requirements Testability and Test Automation, Lockheed Martin Joint Symposium, June 2001.
Mars Polar Lander Fault Identification Using Model-based Testing, Proceeding in IEEE/NASA 26th Software Engineering Workshop, November 2001.

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Process Flow



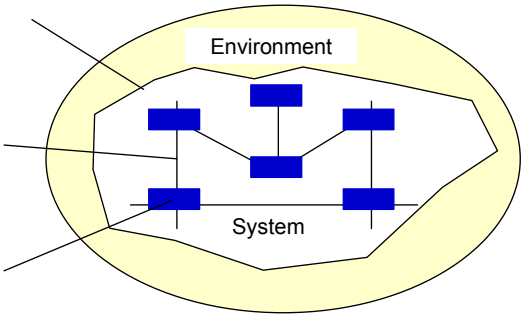
Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Modeling Perspectives

Requirement Specification: defines the boundary between the environment and the system

Functional Specification: defines the interfaces within the system

Design Specification: defines the component

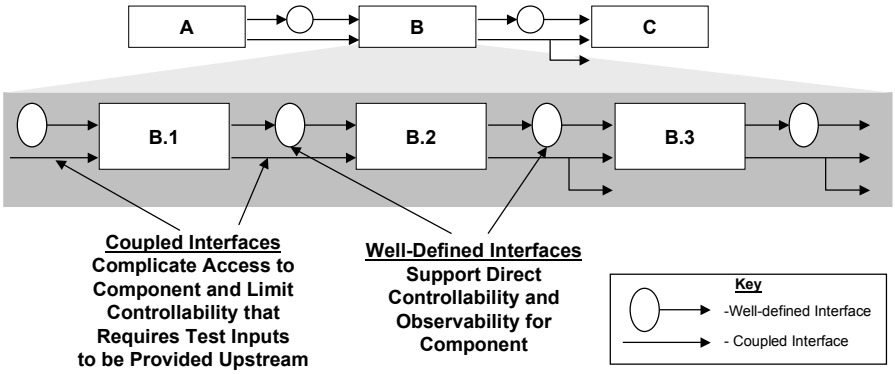


D. Cooke et al., 1996

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Defining Component Interface Helps Stabilize Architecture

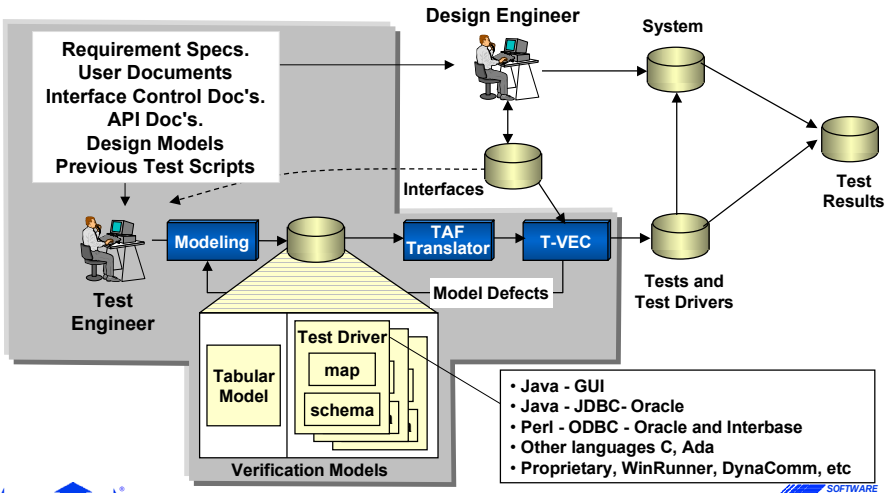
- Verification efforts have helped drive better/early definition of component interfaces to stabilize architecture early



Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

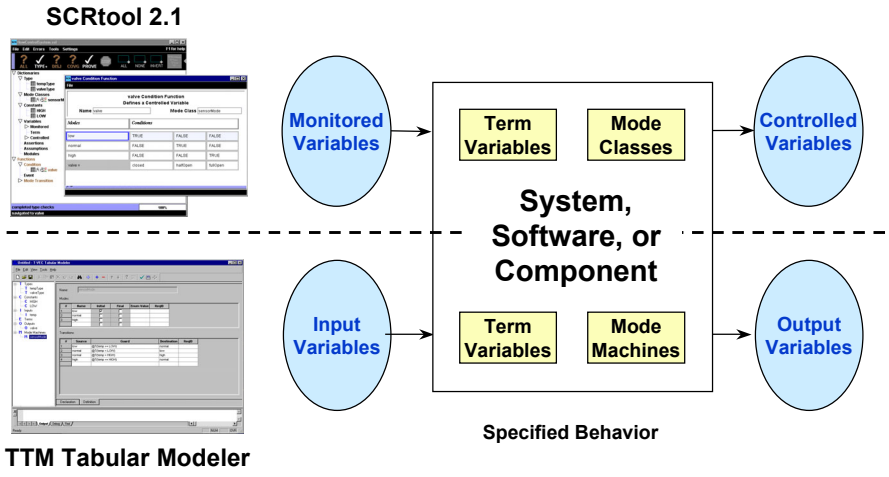
Developing Verification Models

- Verification model is refinement of requirements defined in terms of interfaces and other related documentation



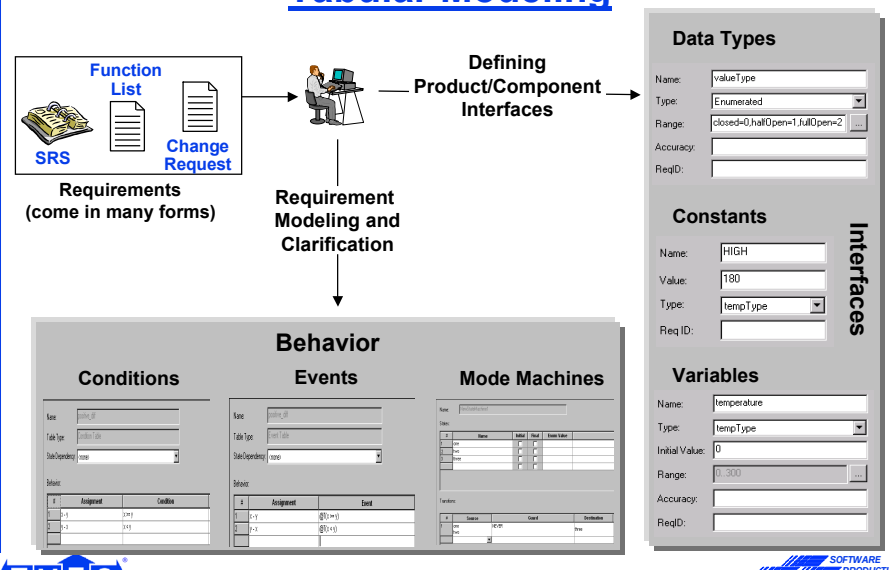
Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Tabular Modeling Based on SCR Method Is Easiest for Test Engineers



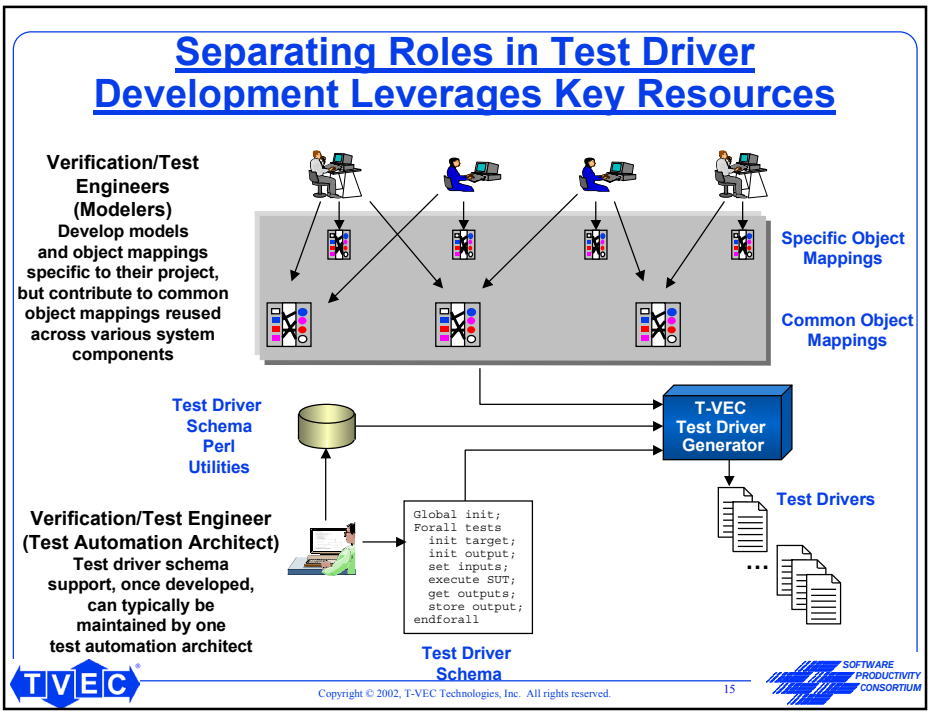
Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

How to Model Requirements Using Tabular Modeling



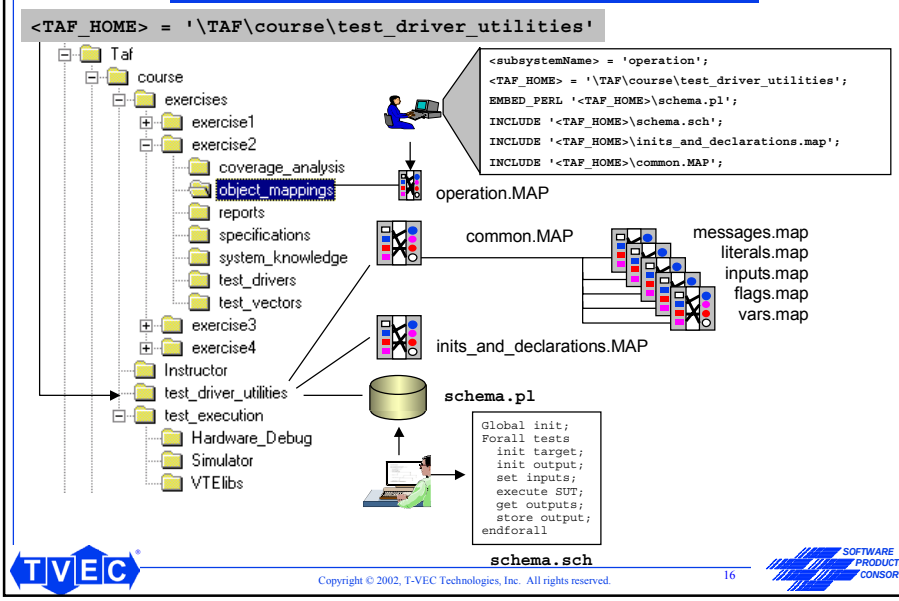
Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Separating Roles in Test Driver Development Leverages Key Resources



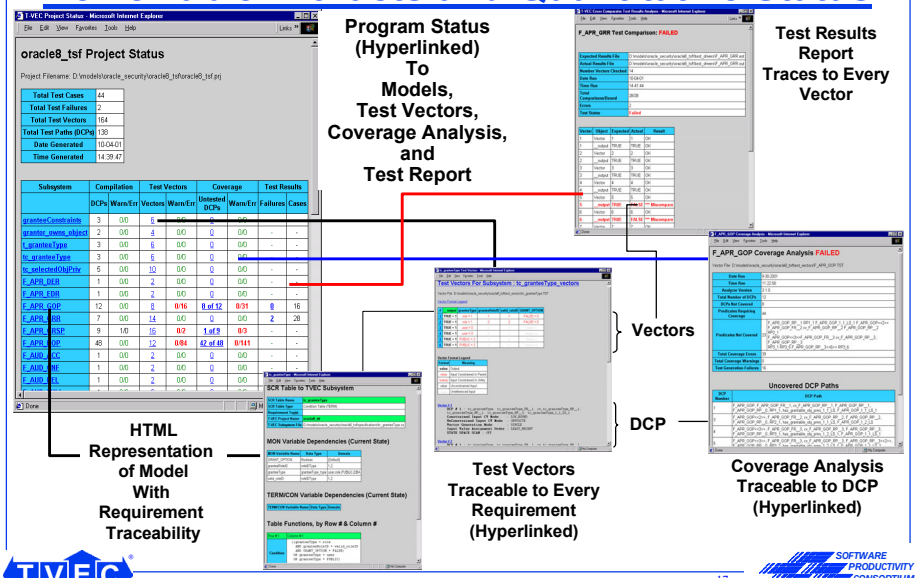
Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Test Driver Organization Facilitates Reuse and Leverages Expertise



Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

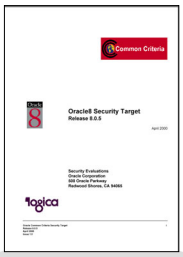
Benefits of Full Traceability: Reviewable Artifacts and Quantitative Status



Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

What's a Security Requirement?

Oracle8 Security Target



Oracle claimed support for ISO/IEC 15408 Security Target

Grant Object Privilege (F.APR.GOP):
A normal user (the grantor) can grant an object privilege to another user, role or PUBLIC (the grantee) only if:
a) the grantor is the owner of the object; or
b) the grantor has been granted the object privilege with the GRANT OPTION.

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Model Behavior for Grant Object Privileges

Grant Object Privilege

#	Assignment	Condition
1	TRUE	grantee = VALID_GRANTEE AND (grantor = VALID_GRANTOR AND (grantorOwnsObject = TRUE OR (grantorHasPrivilege = TRUE AND hasGrantOption = TRUE))) AND (((granteeType = user OR granteeType = PUBLIC) AND tcUserObjectPrivileges) OR (granteeType = role AND tcRoleObjectPrivileges))
2	FALSE	grantee = VALID_GRANTEE AND grantor = VALID_GRANTOR AND NOT (grantorOwnsObject = TRUE OR (grantorHasPrivilege = TRUE AND hasGrantOption = TRUE)) AND (((granteeType = user OR granteeType = PUBLIC) AND tcUserObjectPrivileges) OR (granteeType = role AND tcRoleObjectPrivileges))

tcUserObjectPrivileges

#	Assignment	Condition
1	TRUE	(objectPrivilege = ALTER OR objectPrivilege = DELETE OR objectPrivilege = INDEX OR objectPrivilege = INSERT OR objectPrivilege = SELECT OR objectPrivilege = UPDATE OR objectPrivilege = REFERENCES OR objectPrivilege = ALL)
2	FALSE	(objectPrivilege = NONE)

tcRoleObjectPrivileges

#	Assignment	Condition
1	TRUE	(objectPrivilege = ALTER OR objectPrivilege = DELETE OR objectPrivilege = INSERT OR objectPrivilege = SELECT OR objectPrivilege = UPDATE OR objectPrivilege = ALL)
2	FALSE	(objectPrivilege = NONE OR objectPrivilege = INDEX OR objectPrivilege = REFERENCES)

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

What's an Interface?
Data Dictionary View and SQL Commands

Oracle8 Reference &
Oracle8 SQL Reference

Example from
Security Management

Requirement Summary	Data Dictionary Items	SQL Command	Dependant Command
Disable roles	DBA_ROLE_PRIVS SESSION_ROLES	SET ROLE	GRANT, ALTER,
Enable roles	DBA_ROLE_PRIVS SESSION_ROLES	SET ROLE	GRANT
Grant object privileges	DBA_TAB_PRIVS	GRANT	CREATE, INSERT, SELECT
Grant/revoke role privileges	DBA_ROLE_PRIVS	GRANT/ REVOKE	CREATE, INSERT, SELECT
Grant system privileges	DBA_SYS_PRIVS	GRANT	CREATE, INSERT, SELECT
Revoke privileges	DBA_TAB_PRIVS	REVOKE	GRANT
Every object uniquely identified, even if deleted	ALL_OBJECTS		CREATE, INSERT, SELECT, DELETE

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

SQL Command and Data Dictionary

- SQL operations directly related to granting of object privileges is of the form

```
GRANT <privilege> ON <object> TO <user | role | PUBLIC> [WITH GRANT OPTION]

Where <privilege> can be: ALTER, EXECUTE, INDEX, INSERT, READ, REFERENCES, SELECT,
   UPDATE, ALL, and the GRANT OPTION is optional
```

- Related SQL operations

```
CREATE USER command and initialize user's privileges

To log on to Oracle, a user must have CREATE SESSION system privilege
```

- Data Dictionary is DBA_TAB_PRIVS.
  - Lists all grants on objects in database
  - Has attributes that indicate, GRANTEE (user to whom access was granted), object owner, name of the object, GRANTOR (user who performed the grant operation), privilege, and indication of whether privilege can be granted to another user

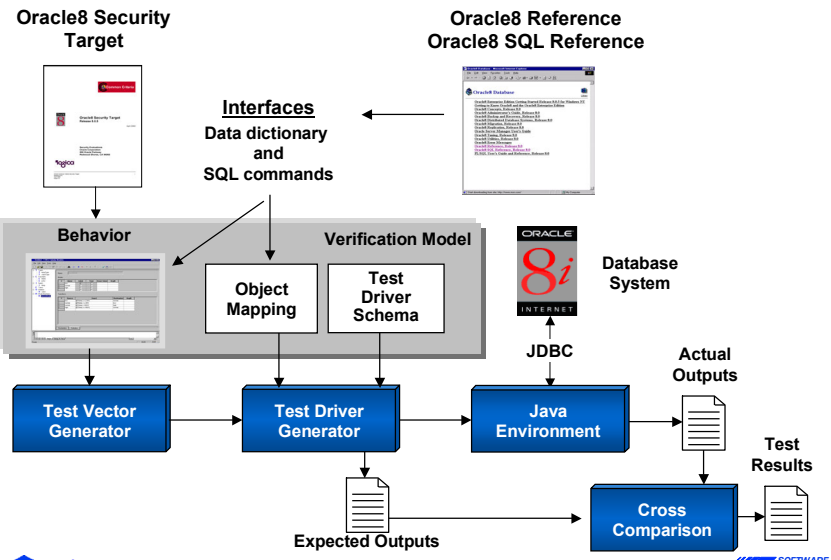
Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Test Driver Capabilities

- Retrieve global test configuration settings that direct test driver mechanisms (log directory, output file directory, system user and password,etc.)
- Retrieve test vector parameters during test execution
- Log test operation
- Create test output file
- Establish an Oracle database connection and SQL execution through JDBC wrapper
- Specify test interface along with helper methods
- Initialize global constants
- Execute test

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Detailed Process Flow Summary



Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Packaging of Infrastructure Java Classes

- Test driver support packaged using a Java package com.tvec.support, which contains the following classes:
- ConfigManager – provides access to the global test configuration settings
- Constants – set of constants used by the tests
- Context – used to retrieve (and set) test vector parameters
- Logger – provides classes to write log files and output files
- SQLUtils – provides database access
- TestImpl – abstract class with the test interface and helper methods
- TestRunner – framework for running classes that implement TestImpl

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Operational Scenario

- TestRunner class contains entry point for running tests that implement TestImpl
- Executing TestRunner performs as follows:
  - Read global configuration file to determine log file directory, output file directory, and maximum number of users
  - Initialize test database (delete existing test table space, create a new test table space)
  - For each test vector:
    - Create default data users, tables, roles, and profiles
    - Call TestImpl.setupTest to setup the test environment further
    - Call TestImpl.runTest to perform the test and return a Boolean result
    - Write the result of the test to the output file
    - Call TestImpl.cleanupTest to do standard cleanup needed to restore test environment.
    - Perform cleanup of standard users, tables, roles, and profiles

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Conclusions/Summary

- Interface driven approach to modeling:
  - Helps drive early stabilization of interfaces and aids in identifying model variables
  - Helps identify common test driver support capabilities that can be constructed once and reused across many related tests
- Parallel development of verification modeling beneficial in development and helps identify requirement defects early to reduce rework
  - Concept characterized as eXtreme modeling (similar to eXtreme programming)
- Java-based test driver capabilities more difficult to setup than Perl (as used on previous projects)
  - If test drivers created and maintained manually, there might be some benefit in using Java, but when automated Perl is easier to build, read/review, and maintain

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Interface-driven Model-based Test Generation of Java Test Drivers

Mark Blackburn, Robert Busser, Aaron Nauman, T-VEC Technologies/SPC
Ramaswamy Chandramouli, National Institute of Standards and Technology

This paper extends prior work in model-based verification and describes interface-driven analysis that combines textual requirement modeling to support automated test generation of Java test scripts for executing against a database. It describes concepts of models and test driver mappings using examples for testing security functionality of an Oracle database using Java and standard Structured Query Language test drivers. Although the modeling and testing is focused on database security capabilities, the described concepts are general for testing most applications.

Keywords: Test Automation Technology and Experience, Interface-driven Model-Based Test Automation, Java and SQL Test Driver Generation, Security Testing, Database Testing

1 Introduction

The combination of model-based verification and test automation has helped reduce cost, provide early identification of requirement defects, and improve test coverage [RR00; KSSB01; BBNKK01; BBN01d; Sta00; Sta01]. This paper extends prior work in model-based verification and recommends the use of interface-driven analysis with requirement modeling to support automated test generation. The interface analysis provides key information that results in **test driver mappings** that specify the relationships between model variables and the interfaces of the system under test. The paper describes the concepts of models and test driver mappings using examples for testing security functionality of an Oracle database using Java and Structured Query Language (SQL) test drivers[1]. Recommendations are provided for performing the modeling of textual requirements in conjunction with interface analysis to support reuse of models and their associated test driver mappings. These recommendations were derived while extending an early experimental model of one small set of requirements to several other groups of interrelated requirements. The resulting insights have been applied to other industry applications and are useful for understanding how to scale models and the associated test driver mappings to support industry-sized verification projects.

1.1 Background

The National Institute of Standards and Technology (NIST) selected the T-VEC test generation system to assess the feasibility of automating security functional testing [Cha99; BBNC01]. T-VEC[2] supports test vector generation, test driver generation, requirement test coverage analysis, and test results checking and reporting [BB96]. **Test vectors** include inputs as well as the expected outputs with requirement-to-test traceability information. The test driver mappings and the test vectors are inputs to the test driver generation, which produces test drivers that are

[1] One of the key requirements for the environment required the testing to be executed against the Oracle database engine through a Java/JDBC connection.

[2] Aissi provides a historical perspective on test vector generation, and recognizes T-VEC as one of the leading commercial tools [Ais02].

Copyright © 2002, Software Productivity Consortium and T-VEC Technologies, Inc. All rights reserved.

executed against the implemented system during test execution. T-VEC is a component tool of the Test Automation Framework (TAF) that integrates various government and commercially available model development and test generation tools to support defect prevention and automated testing of systems and software.

Although the modeling and testing examples are focused on security functionality of a database, the results and recommended approaches are general for testing most applications. TAF has been applied to other applications in various domains including critical applications for aerospace (Mars Polar Lander) [BBNKK01], medical devices, flight navigation, guidance, autopilots, display systems, flight management and control laws, engine controls, and airborne traffic and collision avoidance. TAF has also been applied to non-critical applications like workstation-based Java applications with GUI user interfaces, databases, client-server, web-based, automotive, and telecommunication applications. The related test driver generation has been developed for many languages (e.g., C, C++, Java, Ada, Perl, PL/I, SQL, etc.) as well as proprietary languages, COTS test injection products (e.g., DynaComm®, WinRunner®) and test environments. Most users of the approach have reduced their verification/test effort by 50 percent [KSSB01, Saf00].

1.2 Contributions

This paper provides pragmatic guidance for combining interface analysis and requirement modeling. These recommendations for defining interfaces that provide better support for testability are valid for all forms of testing. Although this paper describes why interface-driven modeling has benefits for testing a released product, it has been applied during development with many additional benefits, which are described in Section 2.4.

1.3 Organization of Paper

Section 2 provides an overview of the method and tools, while providing concept definitions, guidance on interface definitions and analysis, and organizational roles and best practices. Section 3 discusses the security requirements and database interface details using examples. Section 4 discusses the test driver mapping and associated Java support required for test driver generation. Section 5 provides conclusions concerning the use of Java for automated test driver support and summarizes the benefits of interface-driven model-based testing.

1.4 Related Work

The Software Cost Reduction (SCR) modeling concepts are briefly explained in this paper. For more information on modeling there are papers that describe related modeling approaches [HJL96; PM91; Sch90], with examples that support automated test generation [BBN01a; BBN01b; BBN01c; BBNC01, BBNKK01]. Asisi provides a historical perspective on test vector generation and describes some of the leading commercial tools [Asi02]. Pretschner and Lotzbeyer briefly discuss Extreme Modeling that includes model-based test generation [PL01], which is similar to uses of TAF as discussed in Section 2.4. There are various approaches to model-based testing and Robinson hosts a website that provides useful links to authors, tools and papers [Rob00].

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

2 Method and Tool Overview

The TAF support, as shown in Figure 1, involves three main roles of development, including Requirement Engineer, Design/Implementation Engineer, and Test Engineer. A requirements engineer performs requirement analysis and typically documents the requirements in text. A designer/implementer develops the technical solution, which includes system/software architecture, design and implementation. Test engineers clarify the requirements in the form of a **verification model**, which specifies behavioral requirements in terms of the interfaces for the system under test.[3] This is in contrast to a "pure" requirement model, which specifies the requirements in terms of *logical entities* representing the environment of the system under test [PM91; Sch90; HJL96]. Verification modeling from the interfaces is analogous to the way a test engineer develops tests in terms of the specific interfaces of the system under test. TAF translators convert verification models into a form where the T-VEC system generates test vectors and test drivers, with requirement-to-test traceability information that allows failures to be traced backwards to the requirement.

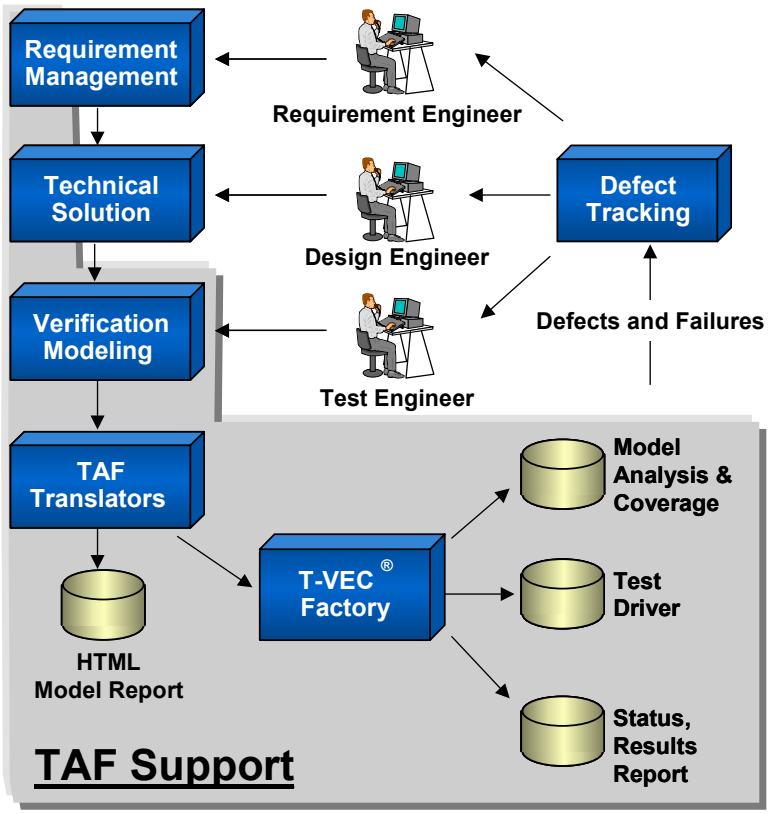


Figure 1. Test Automation Framework Life Cycle Automation

2.1 Verification Modeling Process

Figure 2 provides a detailed perspective of the verification modeling process flow. A test engineer is supplied with various inputs. Although it is common to start the process with poorly defined

[3] A design engineer typically defines the interfaces, and component interfaces are typically documented in a application programming interface (API) or other interface documents.

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

requirements, inputs to the process can include requirement specifications, user documentation, interface control documents, application program interface (API) documents, previous designs, and old test scripts. A verification model is composed of a model and one or more test driver mappings. A test driver consists of object mappings and a schema (pattern). Object mappings relate the model objects to the interfaces of the system under test. The schema defines the algorithmic pattern to carry out the execution of the test cases. The one selected for use in this paper is a Java test driver schema tailored to interface with an Oracle database through a Java Database Connectivity (JDBC) application programming interface (API).

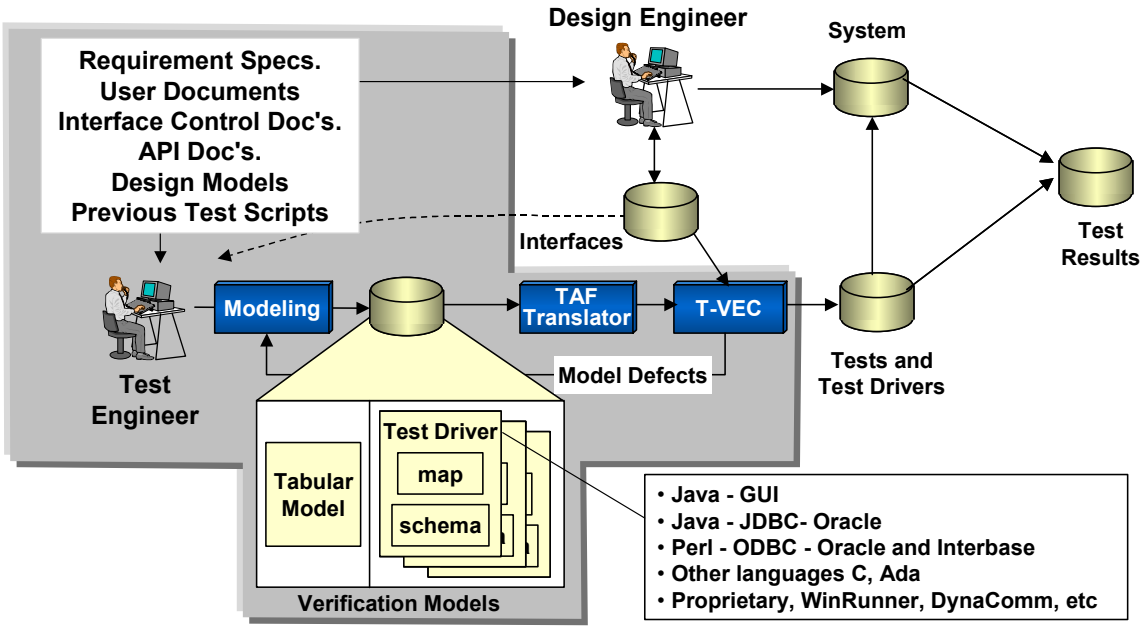


Figure 2. Verification Model Details

Models are typically developed incrementally. The models are translated and T-VEC generates test vectors. T-VEC also detects untestable requirements (i.e., requirements with contradictions). The generation of test vectors and defect detection does not use the test driver information. Test drivers are produced from the test vectors using the test driver mappings and schema information. Detail is provided in Section 4.

2.2 Why Tabular Models?

Table-based requirement modeling like the Software Cost Reduction (SCR) method [HJL96] has been very effective and relatively easy to learn for test engineers [KSSB01]. Although design engineers commonly develop models based on state machines or other notations like the Unified Modeling Language (UML), users and project leaders observed that test engineers find it easier to develop requirements for test in the form of tables (See [BBN01a] for details). The modeling notations supported by tools for the SCR method have well-defined syntax and semantics allowing for a precise and analyzable definition of the required behavior. This paper provides examples using functional tabular modeling, based on the SCR method.

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

2.3 Why Interface-Driven Modeling?

It may seem appropriate to first develop models from the requirements, but when developing models for the purpose of testing, the models should be developed in conjunction with analysis of the interfaces to the component or system under test. Modeling the behavioral requirements is usually straightforward and easier to evolve once the interfaces and operations are understood because the behavioral requirements, usually defined in text, must be modeled in terms of variables that represent objects accessible through interfaces.

2.3.1 Modeling Perspectives

Models are described using specification languages, usually supported through graphical modeling environments. Specification languages provide abstract descriptions of system and software requirement and design information. Cooke et al. developed a hierarchical scheme that classified specification language characteristics [CGDDTK96]. Independent of any specification language, Figure 3 illustrates three categories of specifications based on the purpose of the specification. Cooke et al. indicates that most specification languages usually are based on a hybrid approach that integrates different classes of specifications.

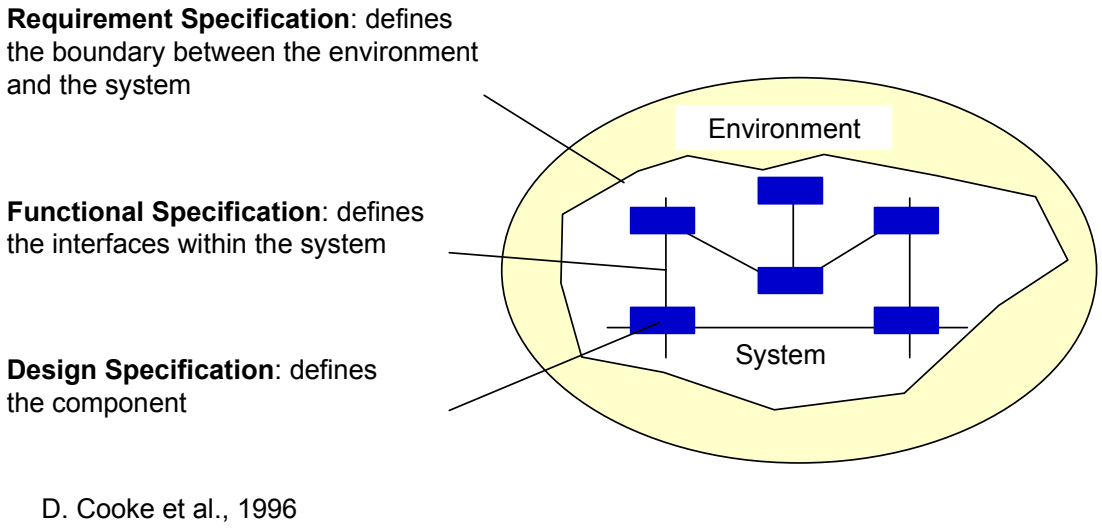


Figure 3. Specification Purposes

Requirement specifications define the boundaries between the environment and the system and, as a result, impose constraints on the system. Functional specifications define behavior in terms of the interfaces between components, and design specifies the component itself. A specification may include behavioral, structural, and qualitative properties. Behavioral properties define the relationships between inputs and outputs of the system [Sim69]; structural properties provide the basis for the composition of the system components; and qualitative requirements [YZCG84] define nonfunctional requirements. Often, languages support certain elements of requirement and functional specifications and are termed functional requirements, as opposed to nonfunctional requirements [Rom85].

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

A verification model, in the context of this paper, is best classified as a functional specification. The requirements are defined in terms of the interfaces of the components. The term interface is used loosely in this paper. An **interface** is a component's inputs and outputs, along with the mechanism to set inputs, including state and history information, and retrieve the resulting outputs. Some components or systems may require sequences of function calls to initialize a component or system, as well as additional calls to place the system in a particular state prior to setting the inputs for testing.

2.3.2 Database Interfaces

For database security requirements the interfaces include the data dictionary (sometimes referred to as database tables) that hold security information and reflect the results of security operations. For each set of modeled requirements it is important to determine the data dictionary views and the SQL commands associated with the requirements, and determine how those database tables are modified to reflect the "correct" or "incorrect" results. Once the interfaces and the SQL operations that affect those tables are understood, it's usually easy to develop the test driver mappings and models hand-in-hand.

2.3.3 Interface Accessibility

It is best to understand the interfaces of the system under test prior to modeling the behavioral requirements to ensure that the interfaces for the resulting test driver map to actual inputs or outputs of the system under test. If the interfaces are not formalized or completely understood requirement models can be developed, but associated object mappings required to support test driver generation must be completed once the interfaces have been formalized. This can make the object mapping process more complex, because the model entities may not map to the component interfaces. In addition, if the component interfaces are coupled to other components, the components are typically not completely controllable through separate interfaces. This too can complicate the modeling and testing process. Consider the following conceptual representation of the set of components and interfaces shown in Figure 4.

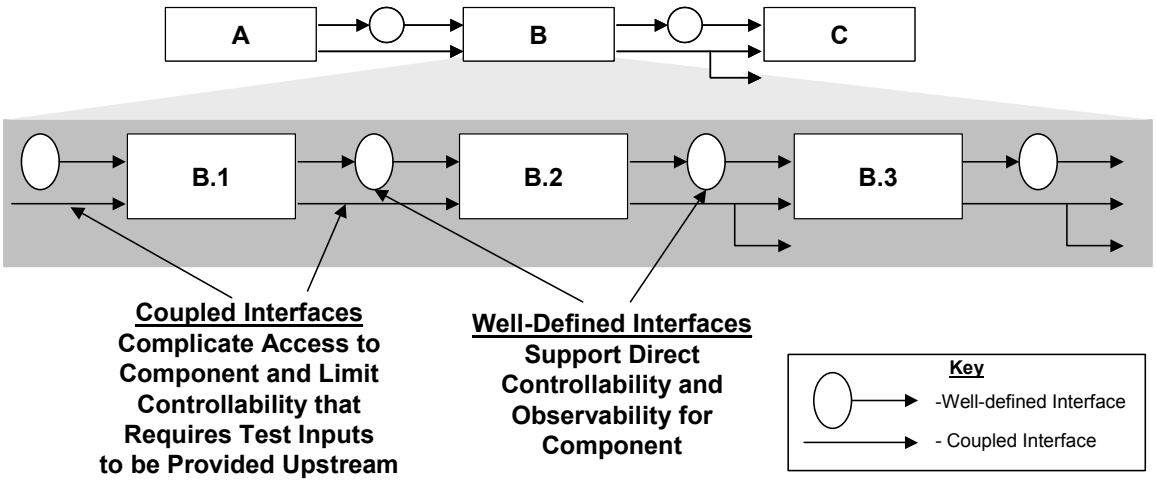


Figure 4. Conceptual Components of System

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

To support a systematic verification approach that can be performed in stages where each component is completely verified with respect to the requirements allocated to it, the interfaces to the component should be explicitly and completely accessible, either using global memory, or better through get and set methods/procedures as reflected in Figure 4. For example, if the inputs to the B.2 component of higher-level component B are completely available for setting the inputs to B.2, and the outputs from the B.2 functions can be completely observed, then the functionality within B.2 can be completely specified and systematically verified. However, if interfaces from other components, such as B.1 are not accessible, then some of the functionality of the B.2 component is coupled with B.1, and the interfaces to B.2, must also include interfaces to B.1, or to other upstream components, such as component A. This interface coupling makes the test driver interfaces more complex to describe, but also forces the behavioral modeling to be described in terms of functionality allocated to combinations of components. The coupling reduces the reuse of components, and increases the regression testing effort due to the coupled aspects of the system components. The problems associated with testing highly coupled systems can be problematic for model-based testing, but also negatively impacts any type of testing. As discussed in Section 2.4, we have observed that interface-driven modeling has helped foster better system design by reducing the coupling, but also helps provide better support for testing.

Systematic test coverage can typically be achieved directly from the verification model if the components of the system can be tested individually. Component integration testing can later be performed from higher-level models to ensure that the integration of the components (i.e., the contractual obligation of the integration) is systematically and completely verified.

2.4 Organizational Best Practices

Interface-driven modeling can be applied after development is complete as is the case for security testing of an Oracle database. However, significant benefits have been realized when it was applied during development. Ideally, test engineers work in parallel with developers to stabilize interfaces, refine requirements, and build models to support iterative test and development. Test engineers write the requirements for the products (which in some cases are very poorly documented) in the form of models, as opposed to hundreds or thousands of lines of test scripts. They generate the tests vectors and test drivers automatically. During iterative development, if the component behavior, the interface, or the requirements change, the models are modified, and test cases and test drivers are regenerated, and re-executed. The key advantages are that testing proceeds in parallel to development. Users like Lockheed Martin state that test is being reduced by about fifty percent or more, while describing how early requirement analysis significantly reduces rework through elimination of requirement defects (i.e., contradiction, inconsistencies, feature interaction problems) [Saf00, KSSB01]. This typical and pragmatic use of TAF parallels eXtreme Programming (XP) [Bec99] where tests are created before the program. However, others refer to this model-based method as Extreme Modeling (XM) [PL01; BBWL00], which applies the principles to write tests prior to coding. With XP test code is developed manually, but with XM the requirements are modeled and the tests are generated.

3 Security Requirements and Database Interfaces

The Oracle 8 Common Criteria Security Target defines Security Functional Requirements [Ora00 – Chapter 5]. The data dictionary views are defined in Oracle8 Reference, which is provided with

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

the Oracle software. The SQL commands are defined in the Oracle8 SQL Reference, also provided with the Oracle software.

Prior efforts focused on developing verification models for the security functionality, referred to as "Granting Object Privilege Capability (GOP)" [BBNC01]. While extending the model to support Identification & Authentication, Security Management, and Session Management, we observed that it reduces work when "low-level" primitive models and their associated test driver mappings are developed first so that the low-level models and test driver mappings can be reused as primitives in higher-level requirement models. Developing from the lowest-level interfaces is not an absolute requirement, but if this approach is applied to a larger verification effort, the resulting verification model leverages reusable model elements that are directly related to reusable test driver interface mappings.

3.1 Security Requirement Interfaces Analysis

Prior to, or in conjunction with, modeling the requirements, the database interfaces associated with the requirements are analyzed to identify common tables, SQL commands, and common test driver mappings that can be extended and maintained as the product evolves. Model variables are used to represent database tables, objects, privileges and relationships. Consider the example of Granting Object Privilege. The requirements state:

```
A normal user (the grantor) can grant an object privilege to another user, role
or PUBLIC (the grantee) only if:
  a) the grantor is the owner of the object; or
  b) the grantor has been granted the object privilege with the GRANT OPTION.
```

The SQL operations that are directly related to the granting of the object privileges include:

```
GRANT <privilege> ON <object> TO <user | role | PUBLIC> [WITH GRANT OPTION]

Where <privilege> can be: ALTER, EXECUTE, INDEX, INSERT, READ, REFERENCES,
SELECT, UPDATE, ALL, and the GRANT OPTION is optional.

And, where <object> is a database schema object like a table, view, sequence,
procedure, function, package, or snapshots.

And, where <user> is a database user, <role> is a defined database role, and
<PUBLIC> represents all users.
```

However, there are some initial privilege and dependent SQL commands that are related to the GRANT SQL command. These involve the creation of a user, role, or session.

- When a user is created with the CREATE USER command, the user's privilege is empty.
- To log on to Oracle, a user must have CREATE SESSION system privilege. After creating a user, the user must be granted this privilege.

There are numerous other cases where additional constraints restrict grant privileges on various object types. These details are beyond the scope of this paper, and are not discussed.

The data dictionary table that is affected, or can be used to determine if a particular GRANT operation is successful, is DBA_TAB_PRIVS. This data dictionary view lists all grants on objects

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

in the database. It has attributes that indicate, the GRANTEE (user to whom access was granted), object owner, name of the object, GRANTOR (user who performed the grant operation), privilege, and an indication of whether the privilege can be granted to another user.

3.2 Security Models and Interface Specifications

As shown in Figure 5, the behavioral requirements are derived from the requirement text in the Oracle Security Target, like Grant Object Privilege. The requirements are defined in terms of the model variables that represent the interface defined in terms of the data dictionary and SQL commands. The interfaces are declared as model variables using the modeling tool. The mapping for the model variable defines how to affect that variable within the test execution environment. For example, a GRANT SQL command must be issued to affect an object's privilege.

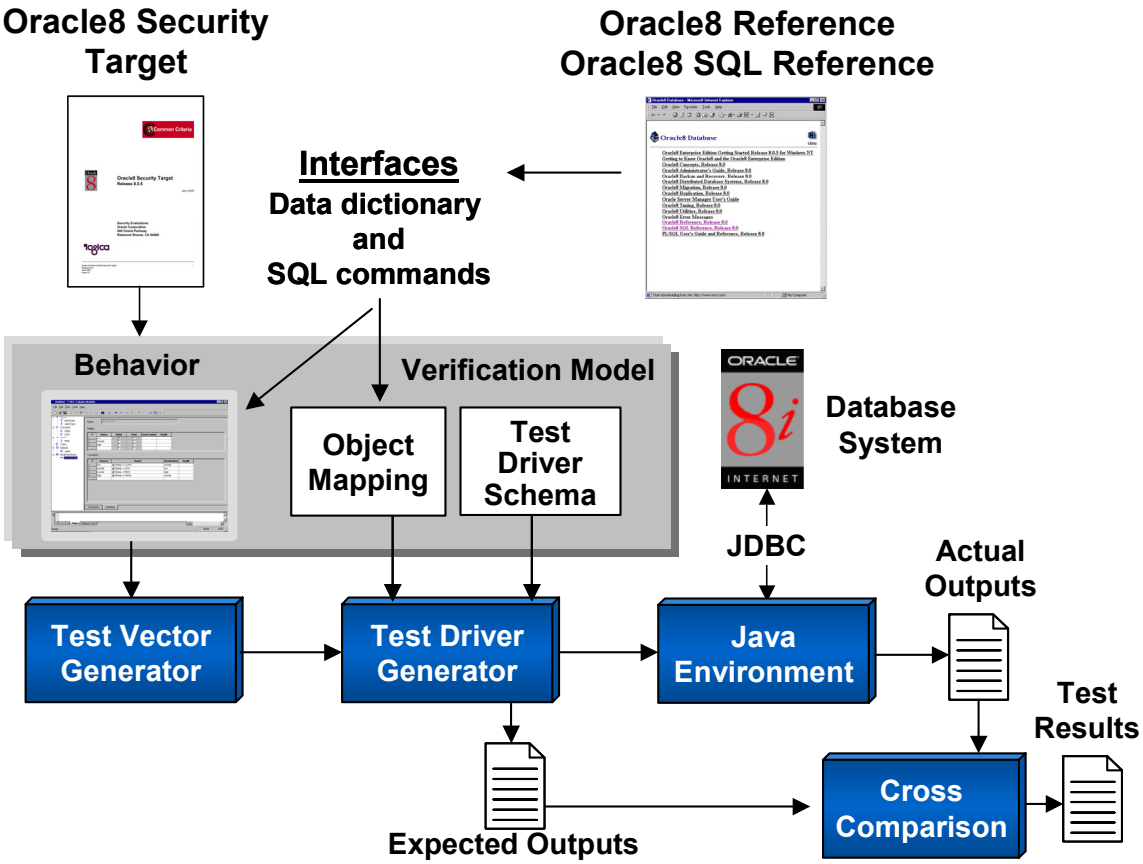


Figure 5. Detailed Process Flow

As shown in Figure 5, the model is input to the test vector generator, and the resulting test vectors are combine with the object mappings and test driver schema (details provided in Section 4) to produce a Java test driver. The executing test driver communicates with the Oracle database through a JDBC connection to carry out the tests. The actual outputs for each test are captured by the test driver during test execution and stored for post processing. Finally a cross comparison tool compares the expected outputs against the actual outputs and produces a test results log that indicates the pass/fail status for each test vector.

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

3.2.1 Modeling Security Properties

Each security property is modeled as a Boolean object in a manner similar to Grant Object Privileges as shown in Figure 6. The conditions associated with the TRUE output or the positive sense for the model is the valid set of conditions required for Granting Object Privilege. Each test case for the TRUE case should result in valid actions with respect to the security relationships established for that case. The FALSE cases are negative conditions, which establish realistic database relationship, but the corresponding test attempts to execute invalid operations, from a security perspective that should be denied as an invalid security response. Some operations cause failures because the database responds with an error message when improper or unauthorized actions are requested. This general approach is used to model each security requirement to ensure that proper security exists for authorized actions, while unauthorized actions are not permitted.

Row 1 of the model for Grant Object Privilege, shown in Figure 6, with the assignment TRUE describes the conditions in which the grant object privilege should be permitted. When the grantee and the grantor are valid database users, then an object privilege should be granted if the grantor owns the object, or if the grantor has been granted object privileges with the GRANT OPTION. In addition, the model defines additional conditions where the grantee (reflected by granteeType) can be a user, PUBLIC or role. The term variable tcUserObjectPrivileges references another condition table that enumerates the set of objectPrivileges (e.g., ALTER, DELETE, INDEX, INSERT, etc.) that are valid, and should be tested. If the granteeType is a role, then the term tcRoleObjectPrivileges defines a subset of the valid ObjectPrivileges that apply to roles.

Grant Object Privilege

#	Assignment	Condition
1	TRUE	grantee = VALID_GRANTEE AND (grantor = VALID_GRANTOR AND (grantorOwnsObject = TRUE OR (grantorHasPrivilege = TRUE AND hasGrantOption = TRUE))) AND (((granteeType = user OR granteeType = PUBLIC) AND tcUserObjectPrivileges) OR (granteeType = role AND tcRoleObjectPrivileges))
2	FALSE	grantee = VALID_GRANTEE AND grantor = VALID_GRANTOR AND NOT (grantorOwnsObject = TRUE OR (grantorHasPrivilege = TRUE AND hasGrantOption = TRUE)) AND (((granteeType = user OR granteeType = PUBLIC) AND tcUserObjectPrivileges) OR (granteeType = role AND tcRoleObjectPrivileges))

tcUserObjectPrivileges

#	Assignment	Condition
1	TRUE	(objectPrivilege = ALTER OR objectPrivilege = DELETE OR objectPrivilege = INDEX OR objectPrivilege = INSERT OR objectPrivilege = SELECT OR objectPrivilege = UPDATE OR objectPrivilege = REFERENCES OR objectPrivilege = ALL)
2	FALSE	(objectPrivilege = NONE)

tcRoleObjectPrivileges

#	Assignment	Condition
1	TRUE	(objectPrivilege = ALTER OR objectPrivilege = DELETE OR objectPrivilege = INSERT OR objectPrivilege = SELECT OR objectPrivilege = UPDATE OR objectPrivilege = ALL)
2	FALSE	(objectPrivilege = NONE OR objectPrivilege = INDEX OR objectPrivilege = REFERENCES)

Figure 6. Example Model for Grant Object Privilege

3.3 Relationship of Security Requirements and Interfaces

Table 1 provides a summary for several modeled requirements. Each row provides a brief summary of a requirement, the related data dictionary views, associated SQL command that are

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

primarily used to affect the operation, and related commands that are referred to as dependent commands.

For example, the Grant Role Privilege command, like the Grant Object Privilege command describes the requirements for granting and revoking role privileges. The primary data dictionary table from which the results of the granted role privilege can be retrieved is the DBA_ROLE_PRIVS (database administrator role privileges). The SQL commands that are used to grant/revoke privileges are GRANT and REVOKE, and the related SQL commands include CREATE, INSERT, SELECT and others. The operations and test driver commands required to support Grant Role Privilege overlap Grant Object Privilege. More importantly, much of the functionality for other requirements like DISABLE and ENABLE roles subsume many of the tested requirements developed for GRANT and REVOKE roles.

Table 1. Detailed Security Specification Analysis

Requirement Summary	Data Dictionary Items	SQL Command	Dependant Command
Disable roles	DBA_ROLE_PRIVS SESSION_ROLES	SET ROLE	GRANT, ALTER,
Enable roles	DBA_ROLE_PRIVS SESSION_ROLES	SET ROLE	GRANT
Grant object privileges	DBA_TAB_PRIVS	GRANT	CREATE, INSERT, SELECT
Grant/revoke role privileges	DBA_ROLE_PRIVS	GRANT/ REVOKE	CREATE, INSERT, SELECT
Grant system privileges	DBA_SYS_PRIVS	GRANT	CREATE, INSERT, SELECT
Revoke privileges	DBA_TAB_PRIVS	REVOKE	GRANT
Every object uniquely identified, even if deleted	ALL_OBJECTS		CREATE, INSERT, SELECT, DELETE

4 Test Driver Generation

This section provides a brief summary of test driver generation. The details of the models, test vectors and test drivers are beyond the scope of this paper. In addition, to understand the test driver support requires some understanding of Java, SQL and operational details of an Oracle database. Additional details including the security requirement models (in HTML), test vectors, object mappings, test driver schema, test drivers and instructions for installing and executing the test drivers against an Oracle database are available for download from: http://www.software.org/pub/taf/Reports.html.

4.1 Creation of Test Oracle Database

The test driver dynamically creates and deletes database information in the form of users, roles, database tables and values. Although most manual database-related testing is performed using populated databases, model-based test generation systematically populates the database with test data derived from the model. This allows automated test execution without manual assistance. The models are constructed in a way that is independent of any specific populated database. There are some specific database conditions that must be established prior to the execution of the tests. For

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

example a database administrator must install the database and the Oracle database test execution requires the "TEMPORARY" tablespace to be available during execution.

4.2 Test Driver Application Programming Interface and Language

The test driver API discussed in this paper is based on JDBC API using Java that makes SQL calls to the database. In prior work Perl test drivers used an Object Database Connectivity (ODBC) API to inject SQL calls to both Oracle and Interbase databases. Although each language provides suitable support for performing the test execution, we believe that there is more effort involved in developing the Java/JDBC support as opposed to the Perl/ODBC support for test driver generation.

4.3 Java Test Driver Support

The test driver generation support capabilities are provided by a Java infrastructure to:

- Retrieve global test configuration settings that can be configured to direct the test driver mechanisms to use user-specified options such as log directory, output file directory, system user and password, etc.
- Retrieve test vector parameters during test execution
- Log test operation
- Create test output file
- Establish an Oracle database connection and SQL execution through JDBC
- Specify an interface to which each test must conform along with helper methods
- Provide global constants
- Provide a framework for test execution

4.3.1 Test Driver Packaging

The test driver support is packaged using a Java package com.tvec.support, which contains the following classes:

- ConfigManager – provides access to the global test configuration settings
- Constants – set of constants used by the tests
- Context – used to retrieve and set test vector parameters
- Logger – provides classes to write log files and output files
- SQLUtils – provides database access
- TestImpl – abstract class with the test interface and helper methods
- TestRunner – framework for running classes that implement TestImpl

4.3.2 Operational Scenario

The TestRunner class contains the entry point for running tests that implement TestImpl. Executing TestRunner performs as follows:

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

1. Read the global configuration file to determine the log file directory, the output file directory, and the maximum number of users
2. Initialize the test database, which deletes existing test table space, create a new test table space
3. Get the test vectors from TestImpl by calling TestImpl.getTestVectors. For each test vector:
    1. Create default data based on the user-specified number of standard users
    2. Call TestImpl.setupTest to setup the test environment further
    3. Call TestImpl.runTest to perform the test and return a Boolean result
    4. Write the result of the test to the output file
    5. Call TestImpl.cleanupTest to do standard cleanup needed to restore test environment.
    6. Perform cleanup of standard users, tables, roles, and profiles
4. Exit.

4.3.3 TestImpl Interface

The TestImpl class contains four methods that must be implemented when creating a test, including: setupTest, runTest, cleanupTest, and getTestVectors**.**

- setupTest performs additional database configuration beyond the creation of the standard users
- runTest performs test execution
- cleanupTest restores the database to a known state to support the next test vector
- getTestVectors retrieves the inputs for the current test.

4.3.4 SQLUtils

The SQLUtils class handles the database connectivity and SQL execution. It maintains a user-authenticated connection that is used to execute SQL commands. The connection is only lost when a disconnection-related operation or another connect call is performed.

5 Summary

This paper provides pragmatic guidance for combining interface analysis and requirement modeling to support model-based test automation. The model-based testing method and tools described in this paper have been demonstrated to significantly reduce cost and effort for performing testing, while being demonstrated to identify requirement defects that reduce costly rework. These recommendations for defining interfaces that provide better support for testability are valid for all forms of testing. Although this paper describes why interface-driven modeling has benefits for testing a released product, it has been applied during development with many additional benefits. Organizations see the benefits of using interface driven model-based testing to help stabilize the interfaces of the system early, while identifying common test driver support capabilities that can be constructed once and reused across related tests. In addition, parallel development of verification modeling is beneficial in development and helps identify requirement defects early to reduce rework. This concept has been characterized as eXtreme modeling, which is similar to eXtreme programming.

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

Although this paper discusses modeling and test automation for security requirements, the tools and method are generally applicable because they have been used in several other application domains. Finally, this paper discusses the use of Java test drivers, but in prior work Perl test drivers were developed for both Oracle and Interbase databases. Although each language provides suitable support for performing the test execution, we believe that there is more effort involved in developing the Java support as opposed to the Perl support for test driver generation.

6 References

[Asi02] Aissi, S.,Test Vector Generation: Current Status and Future Trends, Software Quality Professional, Volume 4, Issue 2, March 2002.

[Bec99] Beck, K., Extreme Programming Explained: Embrace Change. Addison Wesley, 1999.

[BBN01a] Blackburn, M.R., R.D. Busser, A.M. Nauman, Removing Requirement Defects and Automating Test, STAREAST, May 2001.

[BBN01b] Blackburn, M. R., R.D. Busser, A.M. Nauman, How To Develop Models For Requirement Analysis And Test Automation, Software Technology Conference, May 2001.

[BBN01c] Blackburn, M. R., R.D. Busser, A.M. Nauman, Eliminating Requirement Defects and Automating Test, Test Computer Software Conference, June 2001.

[BBNC01] Blackburn, M.R., R.D. Busser, A.M. Nauman, R. Chandramouli, Model-based Approach to Security Test Automation, In *Proceeding of Quality Week 2001*, June 2001.

[BBNKK01] Blackburn, M.R., R.D. Busser, A.M. Nauman, R. Knickerbocker, R. Kasuda, Mars Polar Lander Fault Identification Using Model-based Testing, Proceeding in IEEE/NASA 26th Software Engineering Workshop, November 2001.

[BBN01d] Busser, R. D., M. R. Blackburn, A. M. Nauman, Automated Model Analysis and Test Generation for Flight Guidance Mode Logic, Digital Avionics System Conference, 2001.

[BBWL00] Boger, M., T. Baier, F. Wienberg, and W. Lamersdorf. Extreme modeling. In Proc. Extreme Programming and Flexible Processes in SW Engineering (XP'00), 2000.

[CGDDTK96] Cooke, D., A. Gates, E. Demirors, O.Demirors, M. Tankik, B. Kramer, Languages for the Specification of Software, Journal of Systems Software, 32:269-308, 1996.

[Cha99] Chandramouli R., Methodology for Automated Security Testing", NIST Request for Proposal, Nov 1999.

[HJL96] Heitmeyer, C., R. Jeffords, B. Labaw, Automated Consistency Checking of Requirements Specifications. *ACM TOSEM*, 5(3):231-261, 1996.

[KSSB01] Kelly, V. E.L.Safford, M. Siok, M. Blackburn, Requirements Testability and Test Automation, Lockheed Martin Joint Symposium, June 2001.

[PL01] Pretschner, A., H. Lotzbeyer, Model Based Testing with Constraint Logic Programming: First Results and Challenges, Proc. 2nd ICSE Intl. Workshop on Automated Program Analysis, Testing and Verification (WAPATV'01), Toronto, May 2001.

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.

[PM91] Parnas, D., J. Madley, Functional Decomposition for Computer Systems Engineering (Version 2), TR CRL 237, Telecommunication Research Inst. of Ontario, McMaster University, 1991.

[Rob00] Robinson, H., http://www.model-based-testing.org/.

[Rom85] Roman, G.C., A Taxonomy of Current Issues in Requirements Engineering, IEEE Computer, 18(4):14-23, 1985.

[RR00] Rosario, S., H. Robinson, Applying Models in Your Testing Process, Information and Software Technology, Volume 42, Issue 12, 1 September 2000.

[Sch90] van Schouwen, A.J., The A-7 Requirements Model: Re-Examination for Real-Time System and an Application for Monitoring Systems. TR 90-276, Queen's University, Kinston, Ontario, 1990.

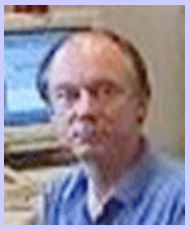
[Sta00] Statezni, David. Test Automation Framework, State-based and Signal Flow Examples, *Twelfth Annual Software Technology Conference,* April 30 - May 5, 2000.

[Sta01] Statezni, David. T-VEC's Test Vector Generation System, *Software Testing & Quality Engineering*, May/June 2001.

[Saf00] Safford, Ed L. Test Automation Framework, State-based and Signal Flow Examples, *Twelfth Annual Software Technology Conference,* April 30 - May 5, 2000.

[YZCG84] Yeh, R.T., P. Zave, A.P. Conn, G.E. Cole, Software Requirements: New Directions and Perspectives, in Handbook of Software Engineering, Editors C. R. Vick and C. V. Ramamoorthy), Van Nostrand Reinhold, 1984.

Copyright © 2002, T-VEC Technologies, Inc. All rights reserved.



QW2002 Paper 3T2

Dr. Stacy Prowell & Mr. William Thomas Swain
(The University of Tennessee)

Tool Support for Model Based Statistical Testing

Key Points

- Model-based statistical testing is both efficient and cost-effective.
- Model-based testing supports the use of test automation tools.
- Significant tool support for model-based testing has been developed.

Presentation Abstract

This talk gives an overview of statistical testing based on a usage model, focusing on opportunities for automation in the process. In particular, an overview of the Java Usage Model Builder Library (JUMBL), a Java class library to support statistical testing, is given. The JUMBL supports the development and analysis of usage models, generation and transformation of tests, and analysis of test

About the Author

Dr. Prowell is a Research Associate Professor in the Department of Computer Science at the University of Tennessee where his work focuses of rigorous software specification and statistical testing. Prior to joining the Univerity of Tennessee, Dr. Prowell worked for Q-Labs on industrial application of process improvement and statistical testing techniques.

Mr. William Thomas Swain As manager of the SQRL, Mr. Swain coordinates project and contract activities. In addition he assists industrial sponsors in application of previous and ongoing SQRL research.

From 1991 to 2000, Mr. Swain held Senior Software Engineering and Technical Leadership positions at CTI PET Systems, doing product research and development for medical imaging systems based on Positron Emission Tomography. At CTI he performed technical leadership roles in system architecture definition, system testing, and software process improvement.

From 1990 to 1991, he was responsible for demonstration of electronic document management technology at Integrated Computer Systems (ICS). From 1986 to 1990, he managed development of PC-based machinery vibration monitoring systems at Computational Systems Incorporated (CSI). At Perceptics Corporation from 1985 to 1986, he was Product Manager for Visual Recognition Systems. As Director of Digital Systems at Technology for Energy Corporation (TEC) from 1979 to 1985, he was responsible for system development of power plant monitoring and emergency response systems. From 1975 to 1978, he performed functional design and design qualification for minicomputer-based reactor protection systems at Combustion Engineering Power Systems. While in the U.S. Air Force (1972-1975), he was a Development Engineer specializing in technical intelligence related to electro-optics. Mr. Swain received a M.S. degree in nuclear engineering from the University of Tennessee in 1972.

Tool Support for Model Based Statistical Testing

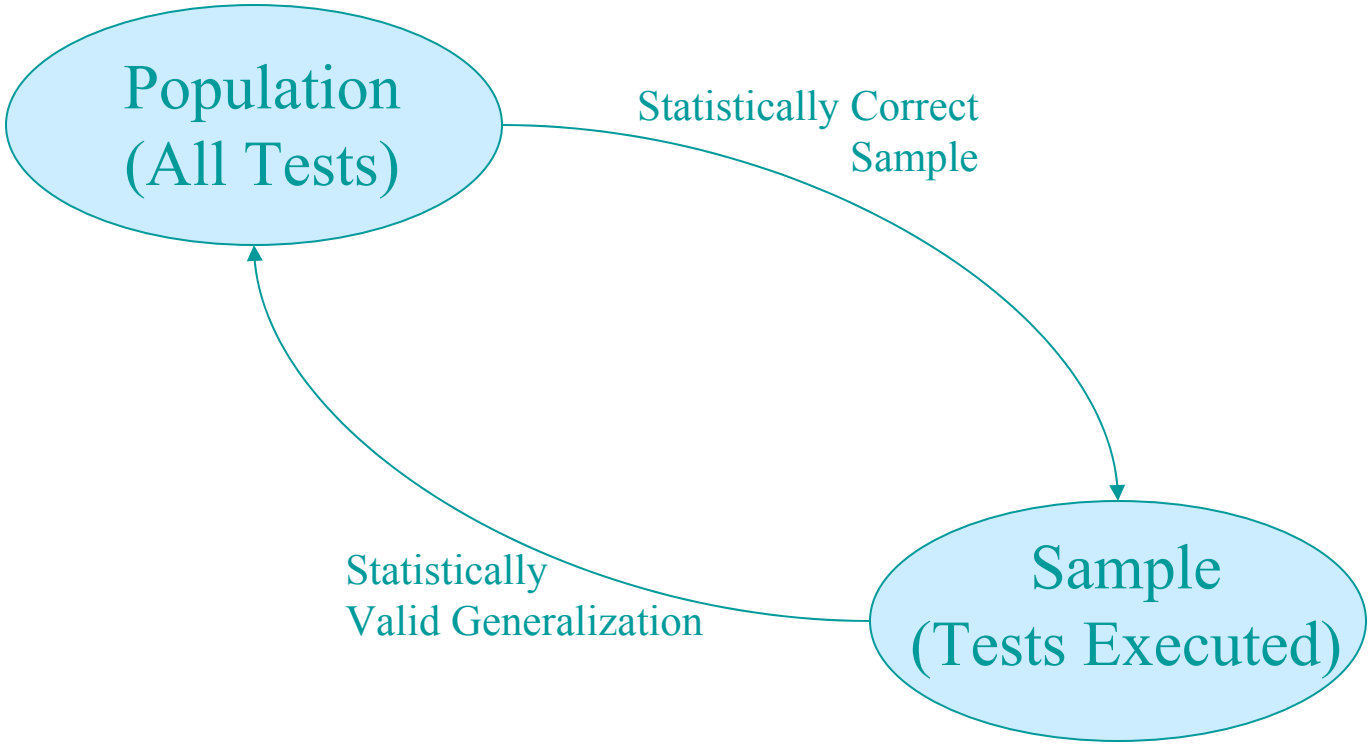
Stacy Prowell
sprowell@cs.utk.edu

Tom Swain
swain@cs.utk.edu

The University of Tennessee
Software Quality Research Laboratory
http://www.cs.utk.edu/sqrl/

Jump to first page

Software Testing



Jump to first page

Usage Based Testing

- User-perceived quality
  - weighted by importance to user
- Test software as it will be used
  - weighted by frequency
- Infer operational characteristics
  - reliability / confidence
  - expected loss
  - risk assessment

Jump to first page

SQRL

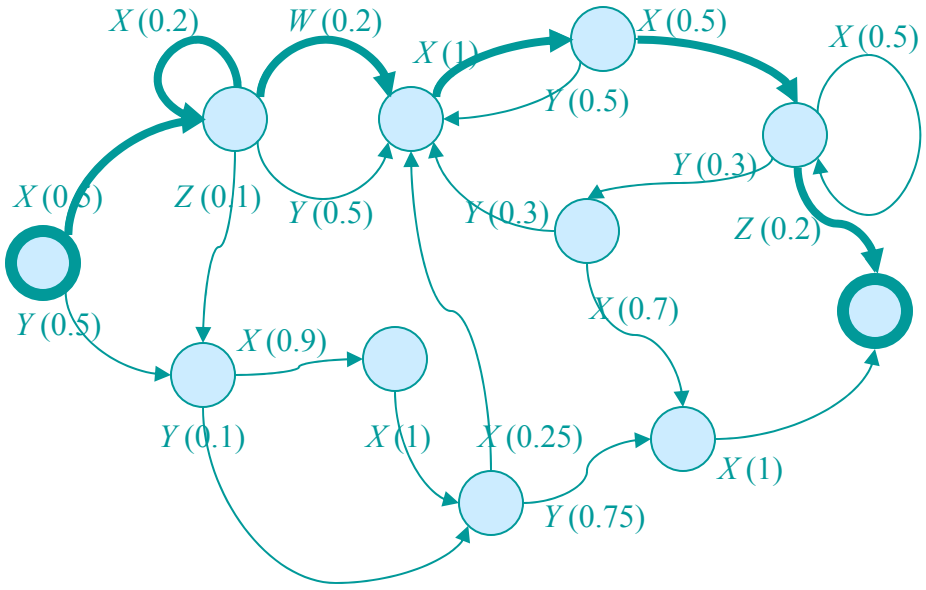
Statistical Testing Process

- **Modeling**: Characterize Population
- **Sampling**: Construct Sample
- **Executing**: Execute Sample on SUT
- **Evaluating**: Observe and Record Results
- **Inferring**: Infer Population Characteristics

Jump to first page

Markov Chain Usage Model

- Need a model of population: Finite-State Markov Chain



$$\Pr[X \quad X \quad W \quad X \quad X \quad Z] = 0.0002$$

- Nodes represent "states of use"
- Arcs represent stimuli
- Probabilities represent use

A "use" (or test) is any path from the source to the sink.

Represents the population of all tests.

Jump to first page

Applicability

This approach has been used for:

- Embedded real-time systems (Raytheon)
- Both hardware and software (IBM SSD)
- GUI applications (CTI)
- Web-based applications (DeRoyal)
- Networking (Nortel)

Automation Opportunities

Tool support is useful for:

- Constructing models
- Analyzing a model
- Generating tests from a model
- Executing tests on the SUT
- Inferring quality from observed behavior

Jump to first page

Developing Automation

- Drivers:
  - Support research
  - Support application
- Choices:
  - Get abstractions right (TML)
  - Construct a class library (JUMBL)
  - Build tools (TMLE)
  - Develop in Java

Jump to first page

SQRL

JUMBL

- Java Usage Model Builder Library (JUMBL)
- Java class library and toolkit for statistical testing
- Developed and maintained by the Software Quality Research Laboratory (SQRL)
- Has a command line / GUI interface

Jump to first page

Model Building and Editing

The Model Language (TML)

- Supports definition of models and related information
- Hierarchical modeling
- Reuse of components
- Constraints
- Automated testing information
- JUMBL: Native support

Jump to first page

Model Building and Editing

```
($ assume(1) $)
model security
source [S0 Enter]
                        "S“         [S1 Ready]

[S1 Ready]
     ($ 40 $)           "S,C“       [S1 Ready]
     ($ 240 $)          "B“         [S2 Entry Error]
     ($ 720 $)          "G“         [S3 1 OK]
                        "T“         [S5 Alarm]

[S2 Entry Error]
     ($ 4000 $)         "S,B,G“     [S2 Entry Error]
     ($ 96000 $)        "C“         [S1 Ready]
                        "T“         [S8 Alarm and Entry Error]

…
Sink [S9 Exit]
end
```

Jump to first page

SQRL

Model Building and Editing

Other Formats

- Graphlet / yFiles (GML)
- Spreadsheets (CSV)
- XML (MML / TCML)
- AT&T Graphviz (DOT)
- Visio 2002 (in development)
- Others…
- JUMBL: Automatic detection and WriteModel to convert

Jump to first page

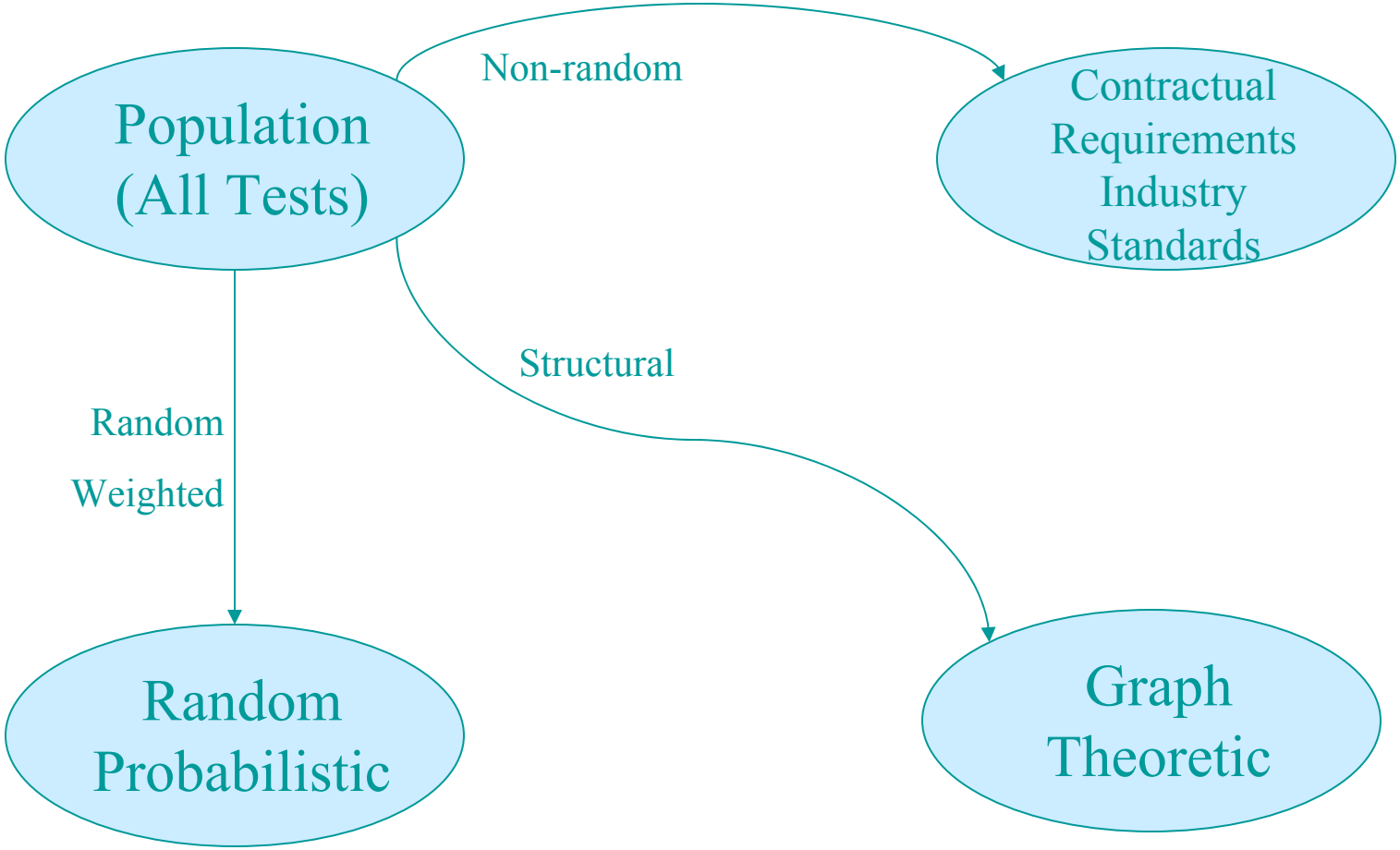
Population Analysis

- Many model statistics are available
- Some results can be obtained by direct computation
- Other results must be obtained by approximation or simulation
- JUMBL: Analyze

Jump to first page

SQRL

Sampling



Jump to first page

Test Selection

- **Importance sampling**:
  JUMBL: GenWeighted
- **Graph theoretic**:
  JUMBL: GenMinCover
- **Random walk**:
  JUMBL: GenTest
- **Manual construction**:
  JUMBL: CraftTest

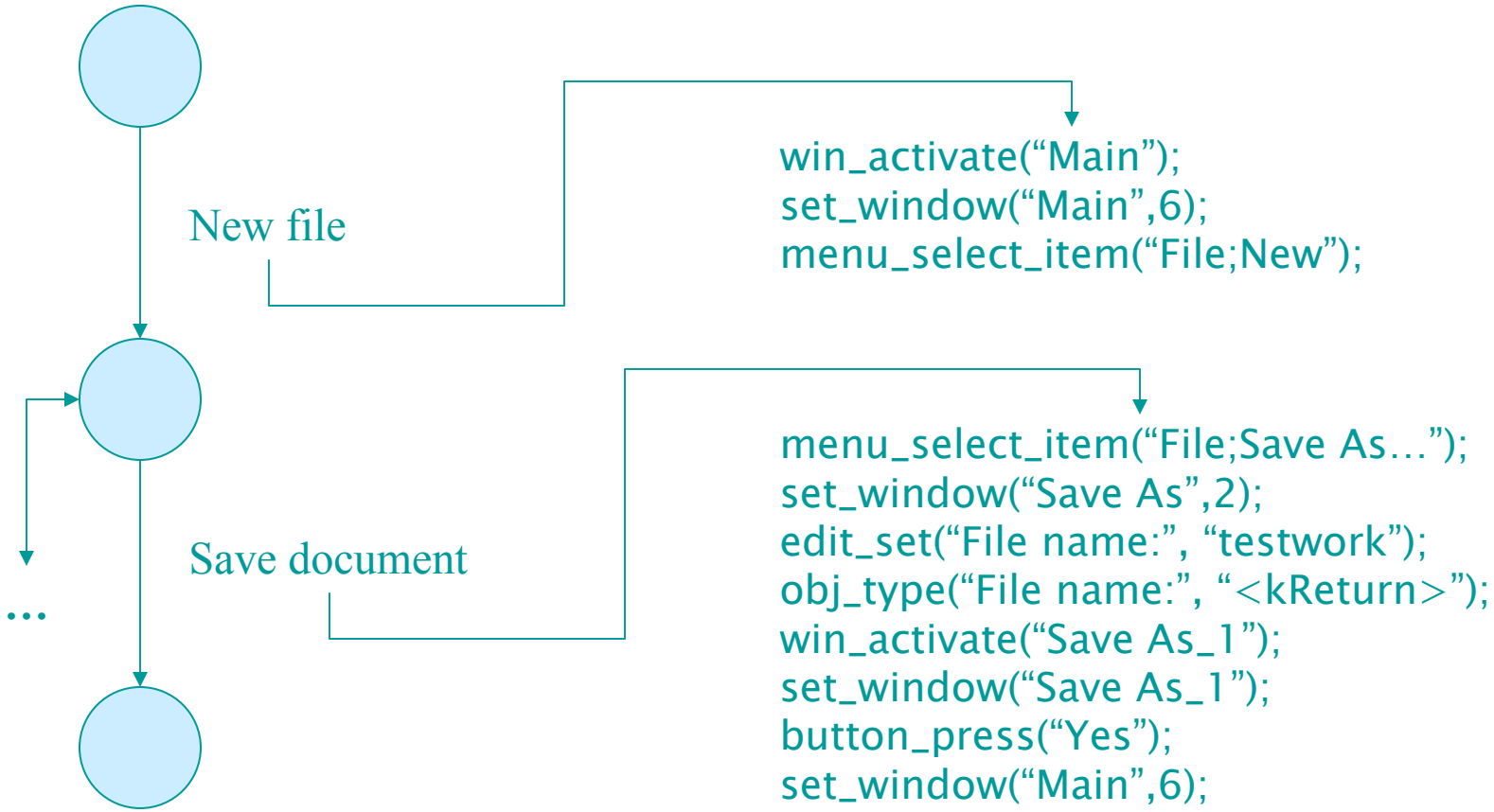
Jump to first page

Test Execution

- Many automated test tools already exist
- Convert test cases to test scripts which can be executed by a test tool
- Associate test instructions with arcs and states in a model
- JUMBL: WriteTest

Jump to first page

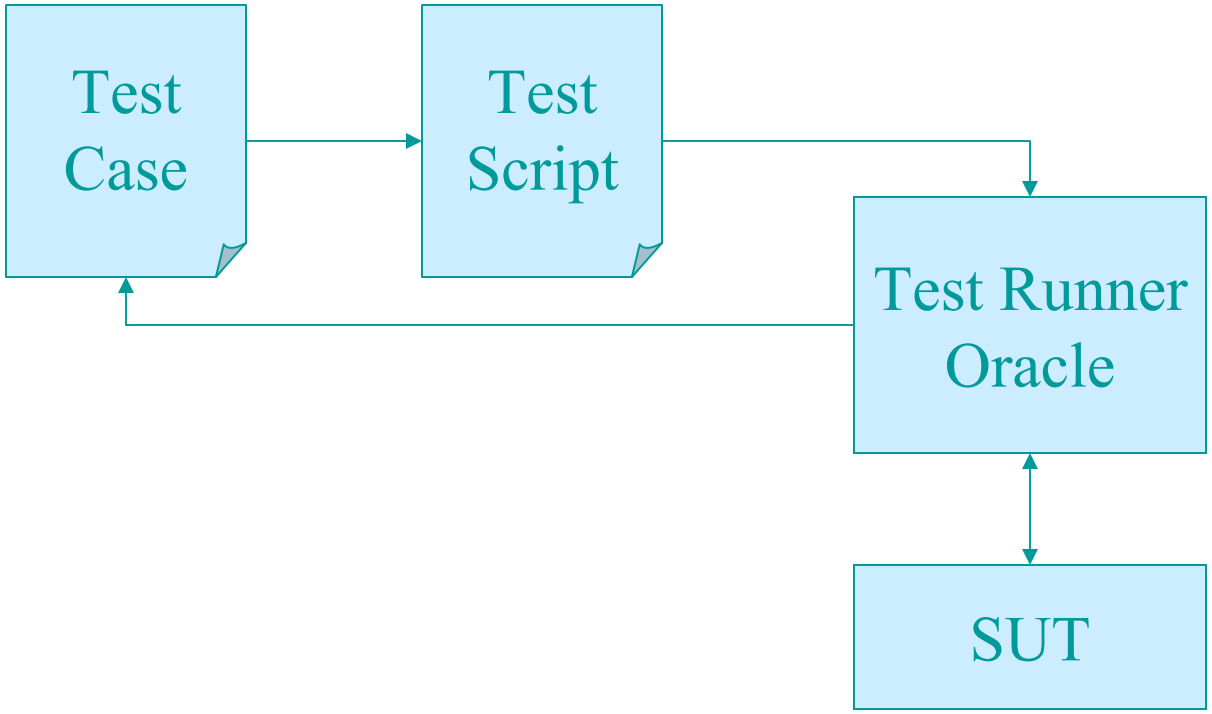
Automated Testing



SQRL

Jump to first page

Automated Testing



Jump to first page

Test Evaluation

- Must be able to decide pass / fail for each test.
- This is difficult to automate:
  - Build oracle into test script
  - Use self-checking data
  - Use known cases
  - Log results for post-test analysis
  - For some cases, let a human decide

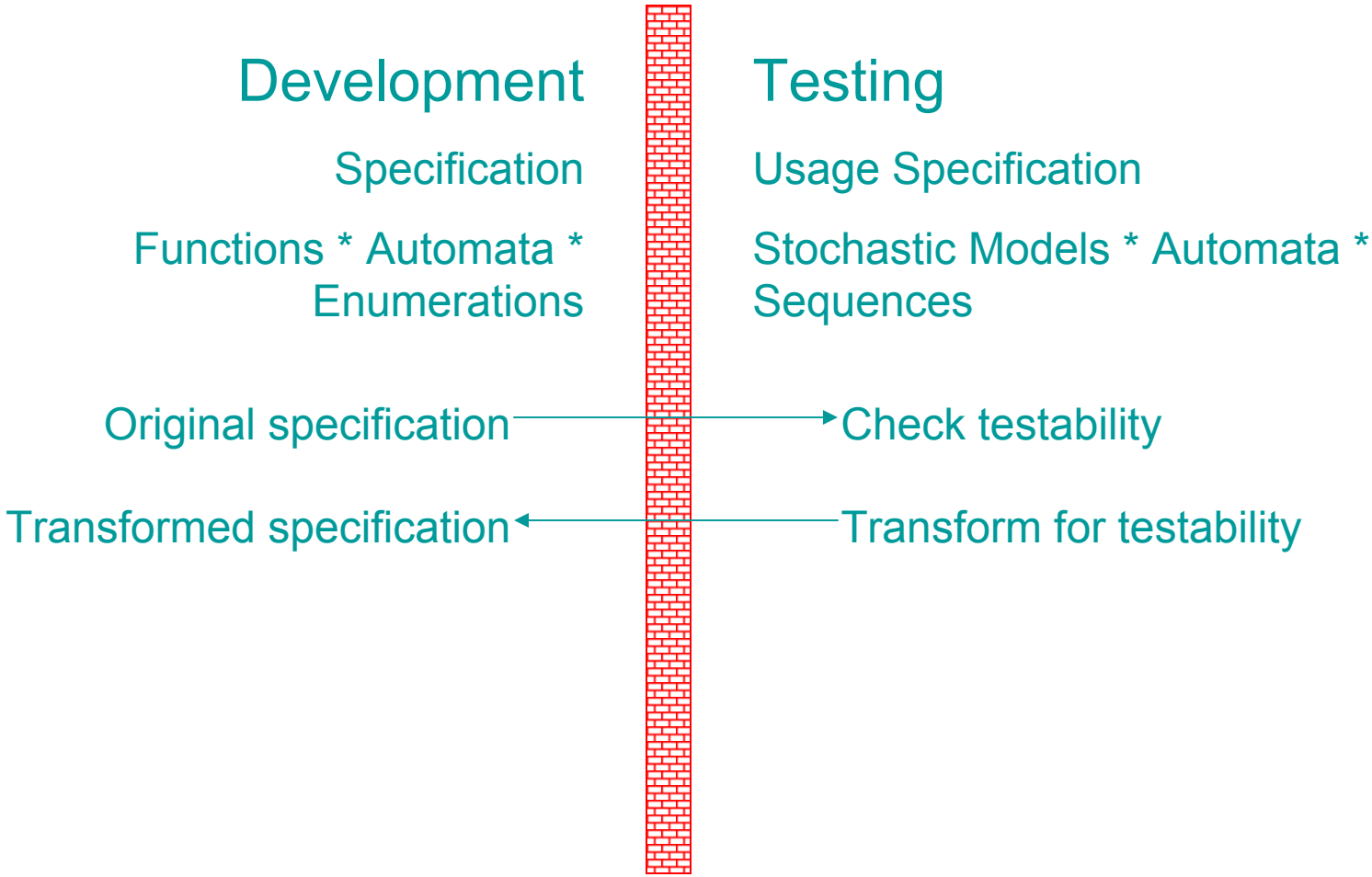
Jump to first page

Test Analysis

- Record information from tests
  JUMBL: RecordResults
- Obtain reliability, MTTF, confidence, and test sufficiency
  JUMBL: AnalyzeTest

Jump to first page

Testability



SQRL

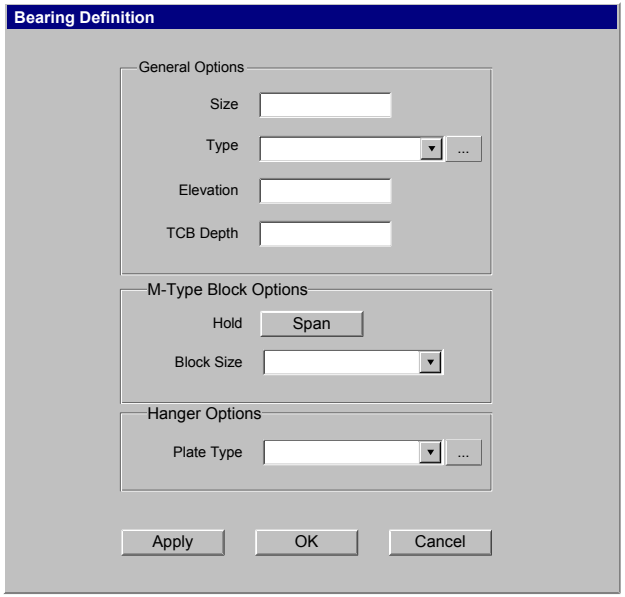
Jump to first page

Practical Modeling Issues

- GUI-based Systems
- Computational Applications
- Embedded Systems
- Combinations of the Above

GUI-based Systems

- Data Entry <-> Processing
- States
  - Allowed Input
  - Potential Behavior
- Arcs
  - Processing Steps
  - Abstraction of Complex or Routine Interaction
- Anywhere-to-Anywhere Designs



Jump to first page

SQRL

GUI State Definition

- Fields Enabled/Disabled
- Dialogs
- Stimulus/Response Modes

Jump to first page

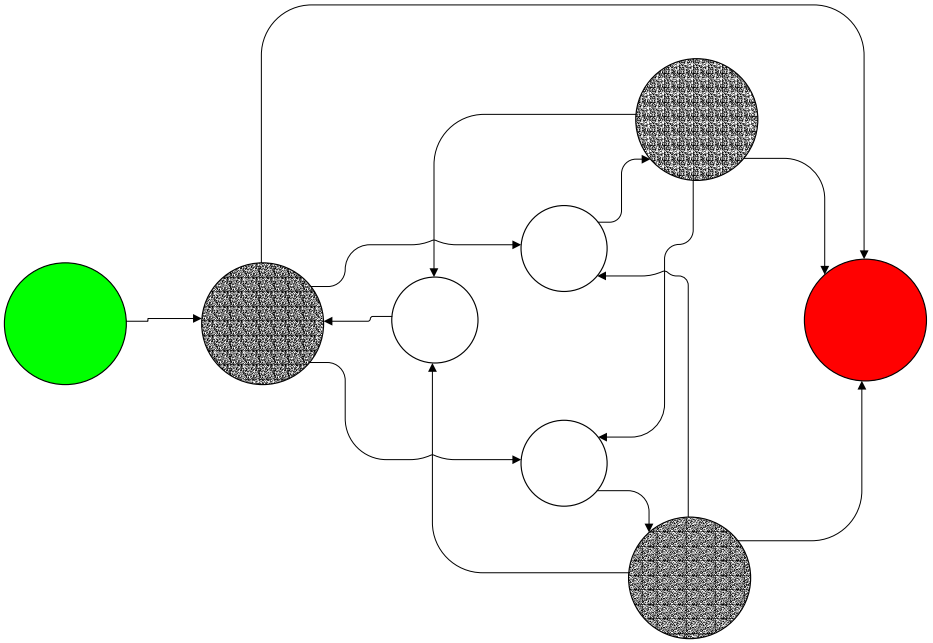
Useful Arc Abstractions

- Simple Dialogs
  - File chooser
  - Confirmation
  - etc.
- Complex Interaction or Process
  - Data Acquisition
  - Process Control
  - DB Query

Jump to first page

Anywhere-to-Anywhere Designs

- Use Submodels
- Use Gateway States



SQRL

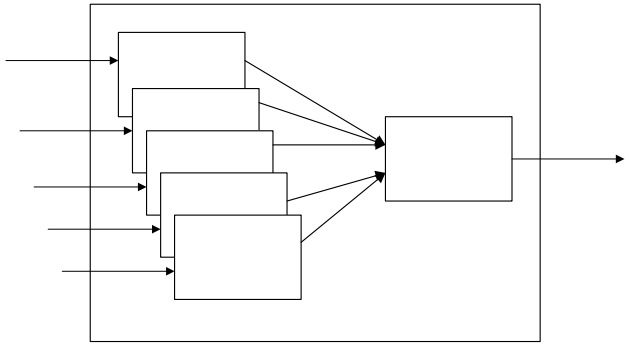
Computational Software

- Partition Input Space
  - Abstract Input Parameters into Subregions
  - Include Invalid as well as Valid Values
- Model Input Selection Process

Jump to first page

Embedded Systems

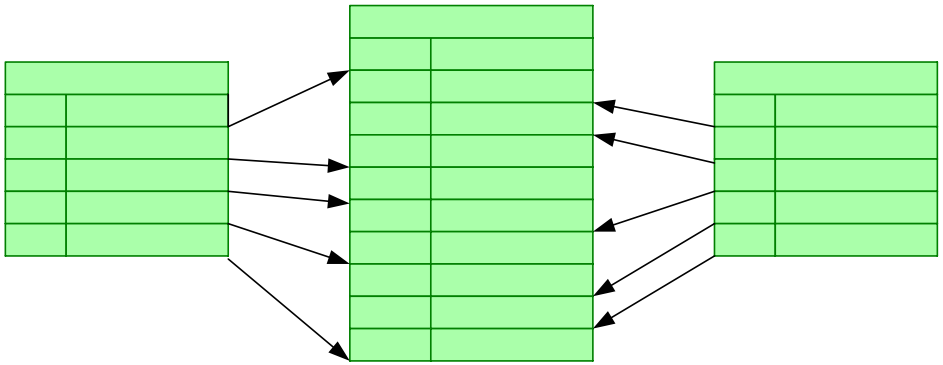
- Sequential
- Parallel or Distributed
  - Submodel per Processor
  - Submodel per Interface
  - Interleaved Test Cases



SQRL

Test Case Interleaving

Handled via the Tangle Function



Jump to first page

SQRL

Tangle Uses

- Servers to Multiple Clients
- Multiplexed Input Streams
- Non-modal GUI

Jump to first page

SQRL

Summary

- Flexible: JUMBL is built using a “plugin” architecture for formats and analytical results
- Adaptable: Easy to build support for third-party tools
- General: Support any automated test tool
- Versatile: Applicable to diverse architectures

Jump to first page

Tool Availability

- JUMBL is available from U.T. SQRL under license agreement
- Contact sprowell@cs.utk.edu for more information
- Large case study in Cleanroom Software Engineering: Technology and Process (Addison-Wesley)

QW2002 Paper 4T1

Mr. Glenn Breslin
(Independent)

Quality Process for XML Interfaces

Key Points

- XML Interfaces - Standards and Process
- Key Learnings from years of Project Managing five XML Interface Projects
- Success Factors

Presentation Abstract

XML interfaces are currently growing in interest and popularity. All major companies are currently creating XML interfaces to obtain necessary efficiencies and to collaborate with customer and vendor companies. This abstract communicates my recent experience project managing five XML interfaces. To support iterative quality XML interface development, I have created an XML Interface Standards and Process document, which I would like to share with my audience. I would very much like to communicate my key learnings (that I learned over several years) so that my audience can expedite quality XML interface design, development, testing and implementation.

About the Author

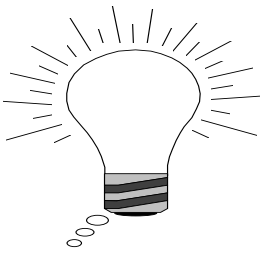
Glenn Breslin, CQA, CSTE Glenn Breslin is a Independent Software QA/Test Consultant. He manages testing efforts on large sized information system projects for Fortune 500 clients. He has been involved in software test and quality assurance for over 20 years. He has additionally functioned as a Testing Manager and Senior Project Manager at the Bank of America and at various software vendors. The Quality Assurance Institute (QAI) has awarded Glenn the Certified Quality Analyst (CQA) and Certified Software Test Engineer (CSTE) credentials.

GLENN BRESLIN PRESENTS

QUALITY PROCESS FOR XML INTERFACES

XML INTERFACE QUALITY KEY PRESENTATION TOPICS

- Process And Standards
- Key Learnings From Years of Project Managing Five XML Interface Projects
- Success Factors



INTRODUCTION

- **Who Am I ?**
- **What Will You Learn ?**
- **How Will I Proceed ?**
- **What Is Not Included ?**

Quality Process for XML Interfaces -- Glenn Breslin 7/12/2002 Page 3

XML - AN INTRODUCTION

- **What Is XML ?**
- **Why Is XML Used ?**
- **What Does XML Look Like ?**
- **Who Is Using XML ?**
- **Why Is XML So Popular ?**
- **What Environments for XML Usage?**
- **What Is XML's Future ?**

Quality Process for XML Interfaces -- Glenn Breslin 7/12/2002 Page 4

QUALITY PROCESS (1)

Legend: y - included in presentation
n - not included in presentation

- **Project Plan (y)**
- **Requirements (y)**
- **Data Map (y)**
- **Field Interface Map (y)**
- **Design Document (y)**
- **Code Development (n)**

Quality Process for XML Interfaces -- Glenn Breslin 7/12/2002 Page 5

QUALITY PROCESS (2)

- **Integration Test Planning (y)**
- **End to End Acceptance Test Plan (y)**
- **Business Events (y)**
- **Integration Test Execution (n)**
- **End to End Acceptance Test Execution (y)**
- **Implementation Planning (n)**
- **Implementation (n)**

Quality Process for XML Interfaces -- Glenn Breslin 7/12/2002 Page 6

BUSINESS REQUIREMENTS

- Introduction
- Deliverable Content
  - List and name the XML interfaces
    - From the customer to the vendor
    - From the vendor to the customer
  - Sample Interfaces
- Risks of Not Creating Requirements

DATA MAP

- Introduction
- Deliverable Content
  - Customer User Interface, Database, XML, Vendor Database, Vendor User Interface (in columns)
  - Customer…. To…… Vendor
  - Vendor …. To…… Customer
- Risks of Not Creating Data Map

DATA MAP – SIMPLE EXAMPLE

Cust UI	Cust Database	XML	Vendor Database	Vendor UI
first	a_first	fname	borr_first	first
M	b_mid	mname	borr_mid	middle
Tel	b_phone	b_tel	borr_tel	phone
dist	d_name	dist_n	district	dist nbr

FIELD INTERFACE MAP

- **Introduction**
- **Deliverable Content**
  - **PowerPoint Slide with Customer User Interface on the Left Side and the Vendor User Interface on the Right Side**
- **Risks of Not Creating Field Interface Map**

FIELD INTERFACE MAP - SIMPLE EXAMPLE

Customer UI	Vendor UI
Order Number (A) Order Type (B) Order Instr (C) Requestor (D) Contract Date (E)	Cust Order Nbr (A) Customer Name (D) Type (B) Instructions (C) Delivery Date (E)

DESIGN DOCUMENT

- **Introduction**
- **Deliverable Content**
  - **Description of each interface in detail**
  - **Mapping from Customer's element name to the XML name**
  - **XML Tag Glossary**
  - **Push vs. Pull Strategy**
  - **Required URLs**
- **Risks of Not Creating Design Document**

INTEGRATION TEST PLANNING

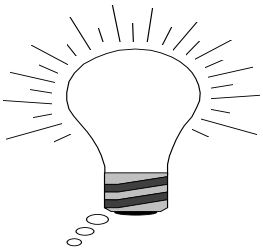
- Introduction
- Deliverable Content
  - Test Cases
  - Test Data
  - Coverage Strategy
  - Track to Design Document
- Risks of Not Planning Integration Test

END TO END ACCEPTANCE TEST PLAN

- Introduction
- Deliverable Content
  - Business Events
  - Acceptance Test Cases
  - Test Environment
  - Test Cycles
  - Actual Hard Copy Results
  - Test Case Tracking Logs
- Risks of Not Creating End to End Acceptance Test Plan

CONCLUSION

- **Recommended Quality Process**
- **Recommended Deliverables**
- **Standards**
- **Major Key Learnings**
- **Success Factors**
- **Next Steps**



QWE2002 Paper 4T2

Dr. Edward Miller
(eValid, Inc.)

Innovative WebSite Mapping Tool

Key Points

- WebSites are most commonly viewed from a browser.
- There are significant advantages to using a browser as the basis for WebSite quality checking.
- A new technology that integrates functional testing, timing/tuning, load testing, and site analysis is described.

Presentation Abstract

The eValid Test Enabled Web Browser incorporates all of the main functions of a complete WebSite Test Environment in a convenient, easy-to-use patent-pending package. eValid capabilities include functional testing and verification, detailed WebSite timing and tuning, load generation with full-browser and partial-browser playbacks, and site analysis with the built-in WebSite spider search and filter function.

This talk describes how the eValid system is architected, how it is used through the set of built-in menus, and how it can be applied to produce reliable, effective tests of WebSites. Specific examples of WebSite testing are given.

About the Author

Dr. Edward Miller is Chairman and President of Software Research, Inc., San Francisco, California, and Chief Technical Architect for software test tools development and software engineering quality questions. Dr. Miller has worked in the software quality management field for 25 years in a variety of capacities, and has been involved in the development of families of automated software, analysis and Web quality tools. He was chairman of the 1985 1st International Conference on Computer Workstations, and has participated in IEEE conference organizing activities for many years. He is the Chairman of the Quality Week Conferences since 1988. He is the author of Software Testing and Validation Techniques, an IEEE Computer Society Press tutorial text. Dr. Miller received his Ph.D. (Electrical Engineering) degree from the University of Maryland, an M.S. (Applied Mathematics) degree from the University of Colorado, and a BSEE from Iowa State University.

Innovative WebSite Mapping Tool

Dr. Edward Miller
eValid, Inc.
901 Minnesota Street
San Francisco, CA 94107 USA
(miller@soft.com)

Outline

- General Objectives
- Technology Base
- Search Engine
- Filters
- Reports
- SiteMap Representation
- Examples

eValid, Inc.

General Objectives

- Client Side Analysis
- Support ANY Server Technology
- 100% Realistic User Perspective
- Ease of Use
- Universality
- Valuable, Easy-To-Read Reports

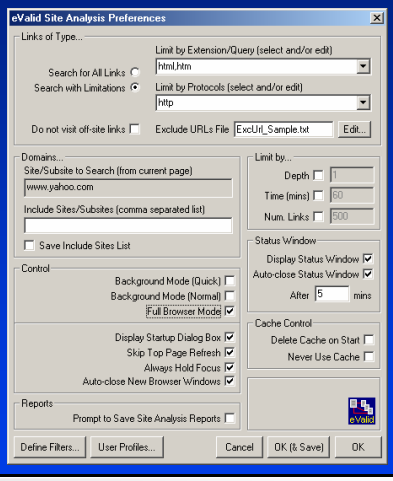
Technology Base

- eValid Browser Base
- Recursive Descent “spider”
- Universal Application (LAN + WEB)

Search Engine Architecture

- Starting URL
- Protocols
- Inclusion Criteria
  - Suffixes
  - Added URLs
- Exclusion Criteria
- Case Study

Site Analysis Preferences Menu



Search By Links

- All Possible URLs
- Extensions
- Protocols
- Included Domains
- Excluded URLs
- Visit/Don't Visit Off-Site URLs

Excluded URLs "sample.txt"

```
################################################################################
#
# ExcUrl_Sample.txt
#
# ---------------------------------------
# eValid Site Analysis: Exclude-URLs File
# ---------------------------------------
#
# (1) What This File Is For...
#
# Use this file to prevent the eValid Site Analysis process from examining
# certain selected URLs during a eValid Site Analysis session.
#
# You can provide different Exclude-URLs files for different situations.
# eValid uses the current one for the current search.
#
# (2) Why Do I Need This...
#
# You may want to prevent a search of:
#
#   (a) A URL which you know to be very large file;
#
#   (b) A URL which, if searched, would terminate subsequent processing,
#   e.g. on which is know to log you off after have logged in;
#
#   (c) Areas of a website that you know may contain problems and
#   thus are not of interest.
#
# (3) How Does It Work...
#
# The file you specify in the Site Analysis Preferences as the "Exclude URLs
# File" specifies URLs you want to exclude.  It does this by specifying strings
# that, when found in any URL, prevent the URL from being processed.
#
# These strings are case-sensitive.  "XYZ" does not block "xyz", for example.
#
# All URLs that are blocked this way using this mechanism are marked as
# [excluded = URL] in maroon in the SiteMap that is produced.
#
# (4) Format and Rules...
#
# Lines preceded by the '#' character (like these) are treated as comments.
#
# Blank lines are ignored.
#
# The URL can be complete or partial but each URL must be on a single line.
#
# The match is case-sensitive.
#
# Individual lines should NOT be "quoted".
#
# (5) Examples...
#
#   http://www.xyz.com/vlarge.pdf - The complete URL will be blocked
#   ACTION=2DLOGOFF               - Any URL with ACTION=2DLOGOFF will be blocked
#   /not_this/                    - Any URL with this string will be blocked
#
# (6) This file at present contains only comments, so no URLs will be blocked.
# Add your own URLs below this comment, or create a new text file.
#
################################################################################
```

Search Limitations

- Depth of Search (From From Top)
- Length of Elapsed Time
- Total Number of URLs Analysed
- Cache On/Off

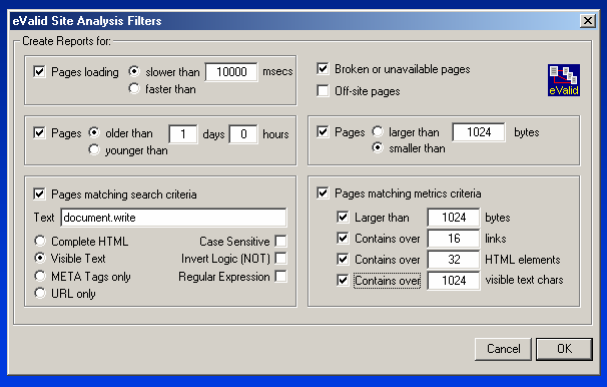
Search Mode

- Browser Mode (Foreground)
- Background Mode (Normal)
- Background Mode (Quick)
- Performance Differences
- Result Differences
- Performance Issues

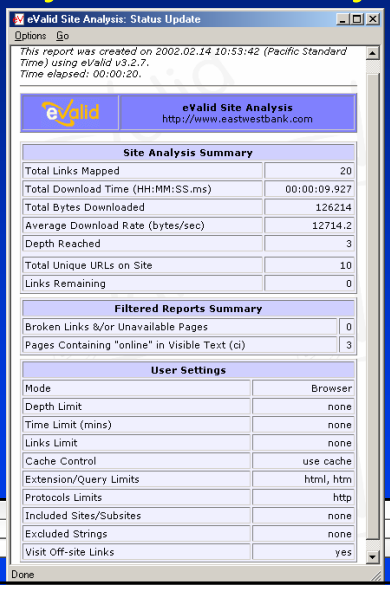
Site Analysis Filtering

- Unavailable Pages
- Slow Pages
- Old Pages
- Large Pages
- External Pages
- Matching String
- Matching Regular Expression
- Metrics
- Case Study

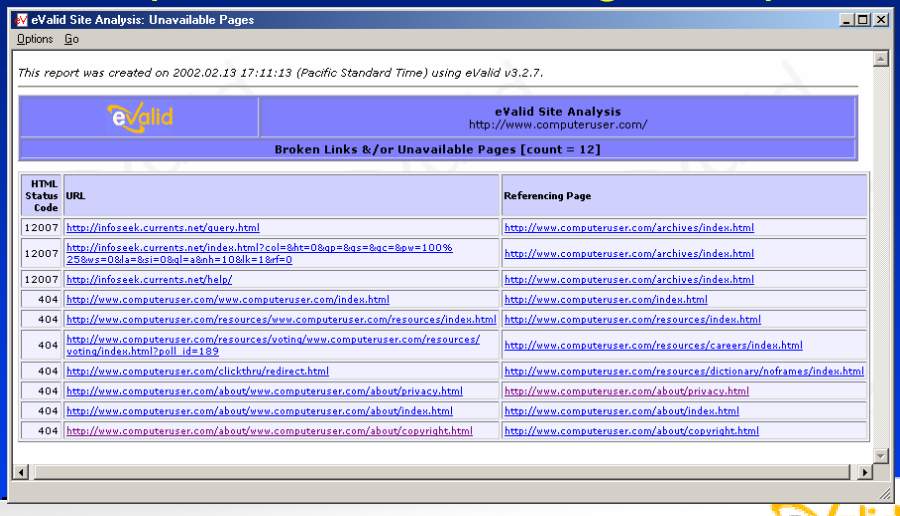
Filter Selection Menu



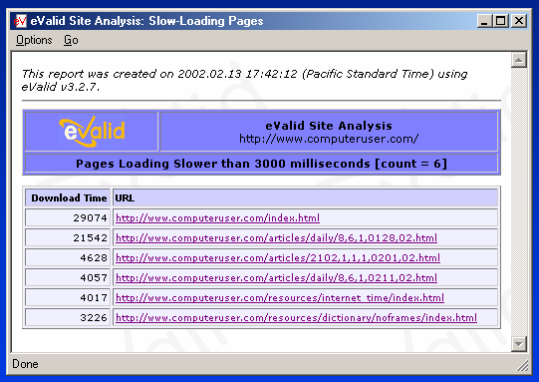
Site Analysis Summary Report



Sample Unavailable Pages Report

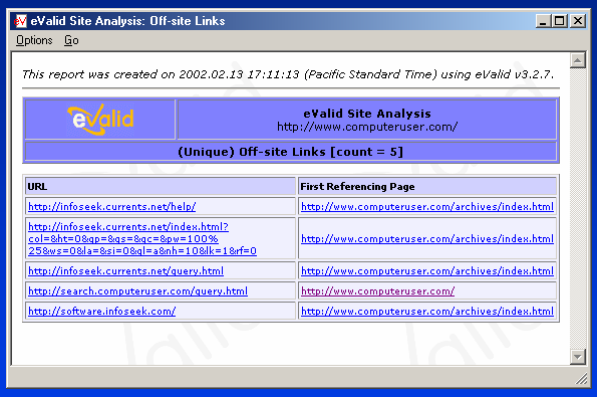


Sample Slow Loading Pages Report



Sample Too Old Pages Report

Sample Off Site Pages Report



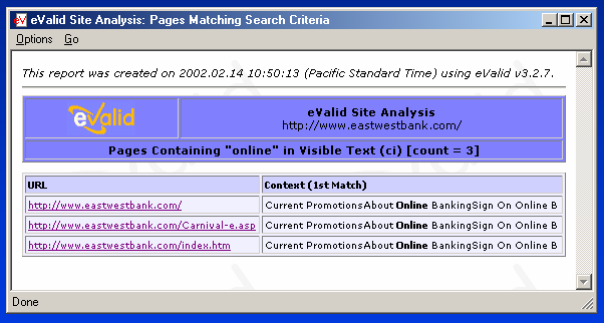
String Match Target Selection

- Complete HTML
- Visible Text
- META Tags Only
- URL Text Only

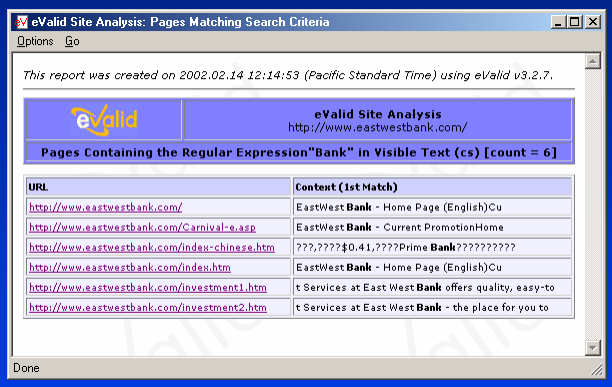
String Match Specification

- Base String
- Case Sensitivity
- Inverted Logic Sense (NOT)
- Regular Expression

Simple String Match Report



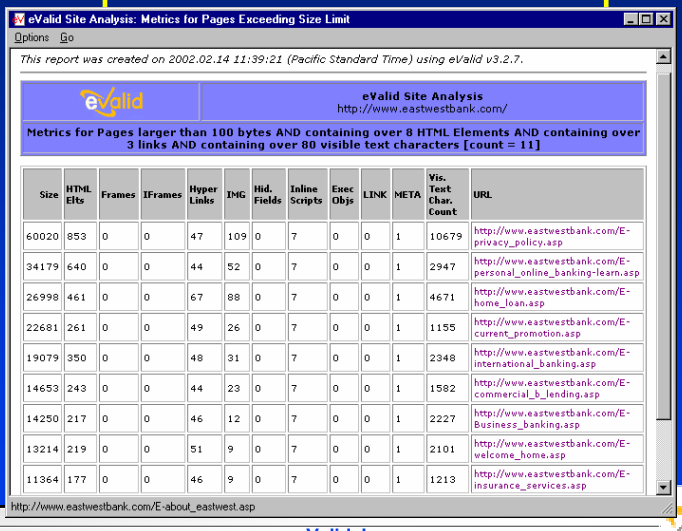
Regular Expression Match Report



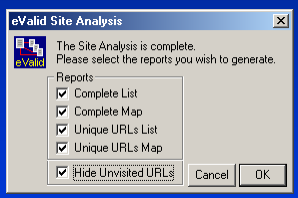
Pages Matching Metric Criteria

- Pages Larger Than ?? Bytes
- Pages Containing Over ?? Links
- Pages Containing Over ?? HTML Elements
- Pages Containing Over ?? Visible Text Bytes
- Composition Rule (AND)

Sample Metric Criteria Report



Site Analysis Report Selection Menu

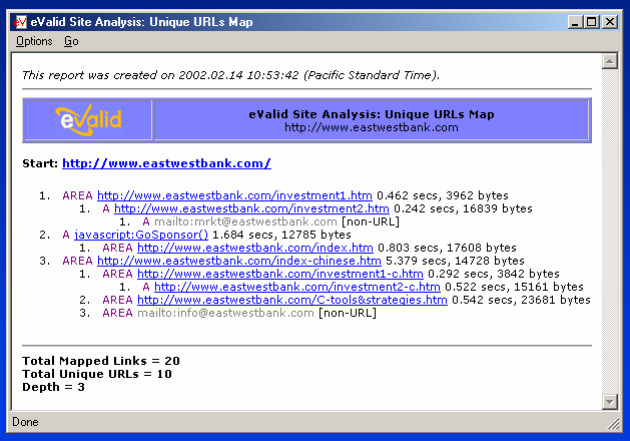


Site Analysis Report Selection

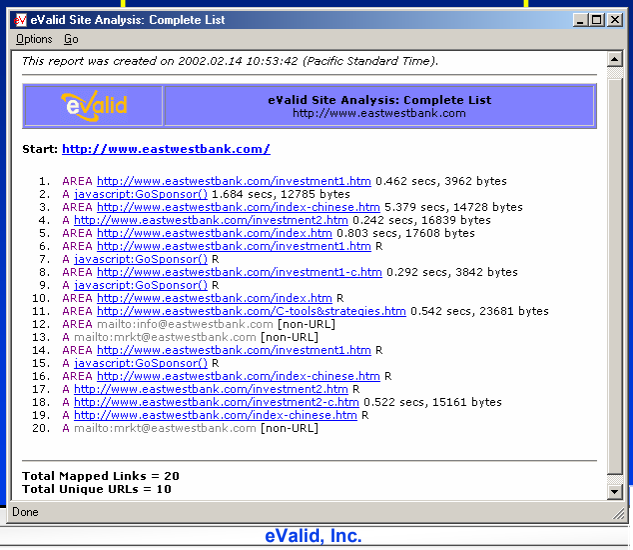
- Full SiteMap
- Unique SiteMap
- Full URL List
- Unique URL List
- Show/Don't Show Unvisited URLs

Sample Full SiteMap Report

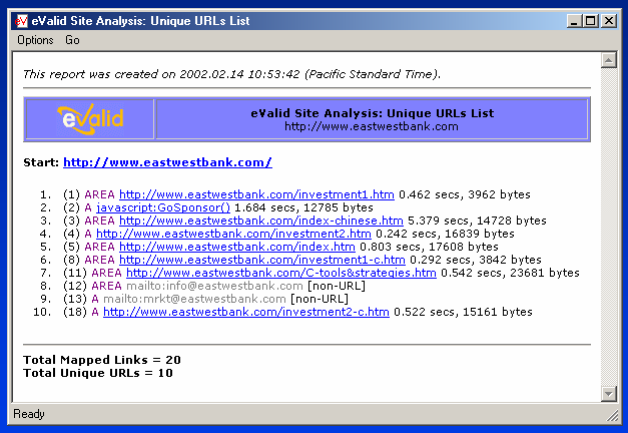
Sample Unique SiteMap Report



Sample Full URLs Report



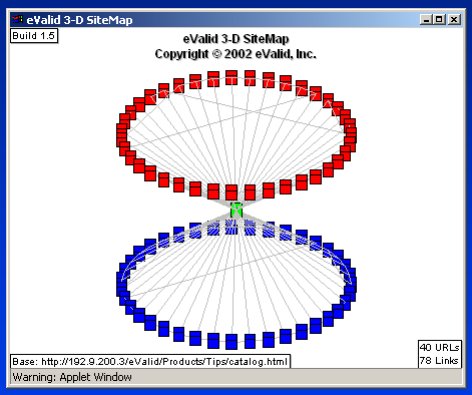
Sample Unique URLs Report



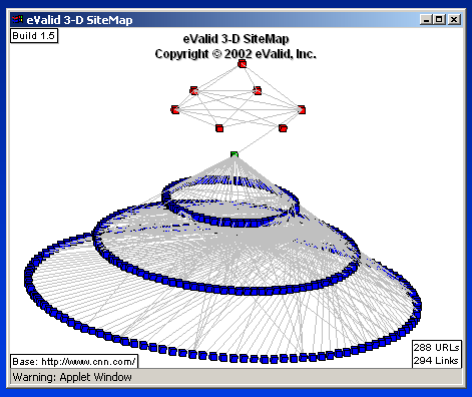
3D SiteMap Graphs

- Derived From Complete SiteMap Table
- Fully Scalable
- Fully Rotatable (Horizontal & Vertical)
- Dependence Annotation
- Redraw from Current Image

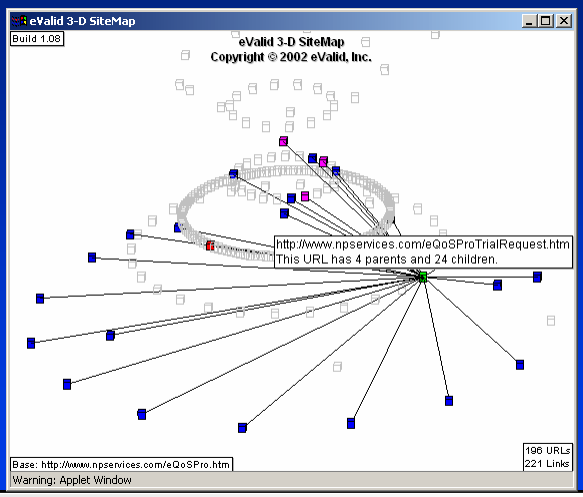
Simple 3D SiteMap



Complex 3D SiteMap

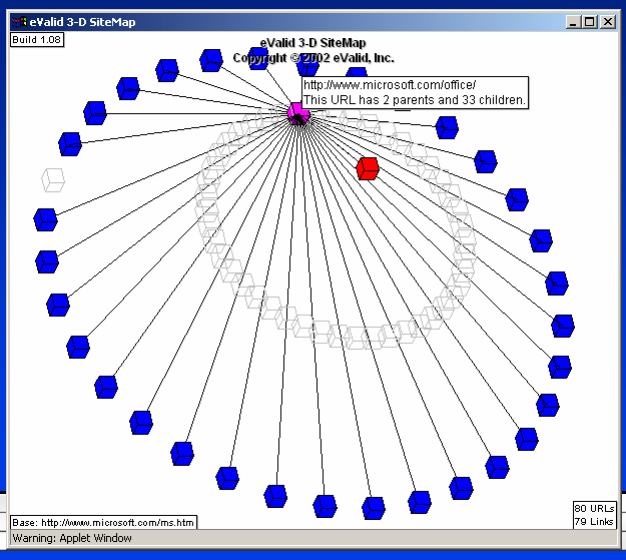


Dependency Display #1



eValid, Inc.

Dependency Display #2



Performance Issues

- RAM Usage
- Download Time
- Download Volume
- Report Preparation Time (Size)
- Other Factors

Conclusions & Recommendations

- Effective Technology
- Correct Application
- Results Analysis
- Repeatability
- Limitations

QW2002 Paper 6T1

Mr. Vijay Sikka & Mr. Anurag Khemka
(Nirixa, Inc.)

Revisiting Comparisons between Manual and Automated testing

Key Points

- Pros and cons of manual and automated testing
- Situations where either type of testing is more effective
- How to increase effectiveness of your testing efforts

Presentation Abstract

Since the dawn of software development, we all have debated the pros and cons of Manual Testing and Automated Testing. In this paper, the authors discuss the features and benefits of each and the situations where each is more appropriate. The paper includes our findings as well as compilations of discussions with several esteemed colleagues in QA and testing arena.

About the Author

Vijay Sikka is the co-founder and principal of Nirixa, Inc. a company providing comprehensive QA and testing outsourced services. Vijay's more than 15 year executive career and strengths span business development, engineering, and operations. Vijay has done marketing at IBM in Madison Avenue, New York and engineering management for 7 ½ years at Intel Corporation headquarters in Santa Clara. In 1996, Vijay founded IBrain Software, Inc. and served as its CEO until its acquisition by Entigen Corp in 1998. In 1999, Vijay started Bodha.com, Inc. and served on its board of directors until its acquisition by Peregrine Systems (NASDAQ: PRGN) in 2002. Vijay has built international teams and engineering organizations with successful multi - year operations. Vijay is an advisor and a consultant to several bay area companies.

Anurag Khemka is the co-founder and principal of Nirixa, Inc. a company providing comprehensive QA and testing outsourced services. Anurag has more than fifteen years of experience in enterprise software development, product architecture, and executive management. Anurag is the founder of MarketFirst Software, a pioneer and leader in Enterprise Marketing Automation, and currently serves there as Vice President and Chief Technology Officer. Prior to founding MarketFirst in 1996, Anurag was the director of research and development at Cambio Networks, and previous to that held positions as the chief product architect and R&D manager for UB Network's enterprise-wide, client/server based network management products.

nirixa

Revisiting Comparisons Between Manual and Automated Testing

By:
Anurag Khemka
Vijay Sikka
Nirixa, Inc.

Presentation to: QW2002

Manual vs. Automated Testing

- **Manual Testing** : is the approach of testing where human resources perform all aspect of testing and validation for the 'system under test'.
- **Automated testing** : is the approach of testing where a significant amount of testing and validation work is performed by non-human resources in an automated or semi-automated fashion.

Copyright © 2001-02 Nirixa, Inc.

nirixa

Type of Resources Needed for Various Test steps

Steps	Manual	Automated
Objective definition	Human	Human
Test planning	Human	Human
Creation of test specification	Human	Human
Development of test cases, sequences, & plans	Human	Human
Development of test scripts	-NA-	Semi-automated
Running tests	Human	Automated
Monitor and record results	Human	Automated
Compare results to an oracle or standard	Human	Automated
Coverage Analysis	Human	Semi-automated
Generate reports	Human	Semi-automated

Copyright © 2001-02 Nirixa, Inc.

nirixa

Testing Basics

- Manual testing is a must: interactive & creative testing and human brain has no replacement
- Automated testing will fail in any scenario where manual testing fails e.g. bad requirements, unclear specifications
- Automation testing only finds a bug that you anticipate, it can't get creative
- 100% Automation is a myth: 100% manual testing is still very common
- Automation helps test staff focus on deeper manual testing instead of working on repeated basic testing

Copyright © 2001-02 Nirixa, Inc.

nirixa

Manual vs. Automated Testing Score Card

Benefits	Manual Testing	Automated Testing
Time per test run	Slow	**Fast**
Reliability	Low	**High**
Redundancy	Requires duplicate effort	**Easily achievable**
Consistency	Low	**High**
Responsiveness	**High**	Low
Flexibility	**High**	Low
Initial implementation costs	**Low**	High
Cost of repetition	High	**Low**
Economies of scale	None	**High**
Skills required	**Low to medium**	High

Copyright © 2001-02 Nirixa, Inc.

nirixa

Why Automated Testing

- Manual testing is very expensive and not scalable
- If certain routine aspects of testing are automated, the test staff can do more interactive/creative testing
- Today's fast development cycles require quick time to market : Manual testing takes too much time
- Today's products are more complex, requiring extensive testing

Copyright © 2001-02 Nirixa, Inc.

nirixa

Simple Economics of Automation

- High fixed cost: deployment and licensing
- Higher test script development cost
- Promises less run time (per test cycle) cost
- Usually first run of automation is more expensive than manual testing
- Automation results in cost savings only after multiple runs are done
- To recoup the cost of automation**

  number of test cycles required = (fixed cost + cost of developing test scripts) / reduced cost per test cycle

** This simple economics ignores the value of expanded testing abilities

Copyright © 2001-02 Nirixa, Inc.

nirixa

Simple Economics of Automation

- ROI = (total benefits-total cost)/total cost
- Total cost = cost of software and hardware + cost of developing the test scripts + (run time cost per cycle * number of test cycles)
- Total benefits = reduced cost per test cycle * number of test cycles
- Additional benefits (not easily quantifiable)
  - Reduced time to market
  - Improved product quality
  - Ability to do load and performance testing

General cost guidelines:

If cost of one manual test cycle is $**n**,

Then, generally the cost of developing test scripts is **5 times n**

After automation, the cost of running one test cycle and maintaining the scripts is **n/5**

So if project has **m** cycles, max cost of Software and Hardware supported is : $ **n * (4/5*m – 5)**

e.g. if n = 10,000 and m = 20, the direct benefit of automation in this example can be $110,000

Copyright © 2001-02 Nirixa, Inc.

nirixa

When Automated Testing does not work

- If manual testing doesn't work, automation too won't
- Unclear requirements, changing specifications
- Volatile interface or environment
- Testing at high level semantics (e.g. picture and text description, semantical validation of content)
- Lack of discipline, strategy, and methodology
- Challenges offered by (Limited automation applications)
  - Multi-threaded, Multi-tiered systems
  - Distributed complex systems
  - Natural Language interfaces
  - Multitude of input device types
  - High Graphical content

Copyright © 2001-02 Nirixa, Inc.

nirixa

When Automated Testing has best Advantage

- Clearly defined inputs and outputs
- GUI (capture/playback) regression tests
- Test suite runs (conformance testing e.g. XML conformance tests, HTML validity testing)
- Build/integration validation (Smoke testing)
- Boundary condition testing / regression testing
- Load testing and performance benchmarking
- Configuration testing: same (or ported code) runs under multiple environments (OS, H/W platforms, browsers etc.)
- Testing transactional systems with mission critical applications
- Systems testing such as API set, Embedded systems, peripheral testing etc.
- Randomized input generation
- High volume exhaustive (large permutations) testing
- Sequence dependent testing etc.

Copyright © 2001-02 Nirixa, Inc.

nirixa

Pros & Cons of Automation

- Cons:
  - Automated testing does not find new bugs
  - Development & maintenance of test scripts is an expensive and tedious task, specially in an evolving product
  - Cost of automation software, initial setup/training, generation of reference oracle etc. is very high (direct & resources)
  - Real time, multi threaded systems are not always repeatable
  - Automation beyond a certain point is not cost effective
- Pros:
  - Tests run faster, provides coverage and scale
  - They are consistent
  - Improved testing and better test records
  - Eliminates human error due to boredom & fatigue
  - Better use of the test staff for creative, deeper testing
  - Can reliably run tests that humans may find difficult and complex

Copyright © 2001-02 Nirixa, Inc.

nirixa

When Manual Testing is a must

- Automation is not complete testing, manual testing is always necessary
  - Real value of automation is so team members can spend more time doing interactive (manual) testing
  - Generally automated testing does not find any new bugs
- Ad hoc or one time testing
- First time testing
- Interactive and usability testing
- Natural language interfaces
- Systems without predictive results
- Less stable, high feature churn, or brittle systems
- Unfrozen changing requirements/specification
- Time of environment dependent response

Copyright © 2001-02 Nirixa, Inc.

nirixa

Components of Automation

- An established testing methodology
  - Commitment for adoption of automation tool
  - Sufficient and trained resources to implement automation
- System under test (SUT)
- Automation software or tools
- Script for running tests
- Results monitoring tools
- Defined program inputs and expected outputs
  - Input dataset and expected results oracle
- External and environmental factors
- Managing results
  - Ability to automatically capture results in electronic format
  - Ability to compare and analyze results to what is expected
  - Reporting of final conclusions

Copyright © 2001-02 Nirixa, Inc.

nirixa

Various Test stages and applicable test methodology

Test Stage	Test Methodology
Unit Testing	Limited automation, broadly manual
White box testing	Limited automation
Functional (Black box) testing	Limited automation
Smoke testing (Regression testing)	High degree of automation
Load & scalability testing	High degree of automation
Stress and performance testing	High degree of automation
Usability testing	Manual
Ad hoc / exploratory testing	Broadly manual
Acceptance testing	Initially Manual; migrating to more automation

Copyright © 2001-02 Nirixa, Inc.

nirixa

Various Application types and applicable test methodology

Application Type	Test Methodology
GUI with predictive results	High degree of automation
GUI with dynamic content	Broadly manual, limited automation
Run time libraries (APIs)	Close to 100% automation
System level tools testing (Compilers, syntax analyzer):	High degree of automation
Semantic testing	Limited automation, Broadly manual
Embedded systems testing	Largely automated
Real time systems	Limited automation
Transactional systems	Limited automation
Multi tier enterprise software	Initially manual; migrating to more automation

Etc..

Copyright © 2001-02 Nirixa, Inc.

nirixa

A Plethora of Available Test tools

- Very sophisticated automation tools available
  - Capture / playback (including remote capabilities)
  - Handles security issues - records secure & non-secure pages
  - Strong scripting capabilities – runs with many data sets
    - Requires none or very little coding
    - Sophisticated Web features
  - Object based smart testing (not pixel based)
    - Adjust if button moves
    - Validation of object properties
  - Ability to model reference oracles & automated result comparison
  - Load generation – simulating large user sets
  - Performance measurements
  - Strong database support
    - Both for input data set
    - And storing output and results
  - Schedulable monitoring
- Can't provide recommendations in this forum
  - We have studied over 15 tools
  - Can make the study available on private basis

Copyright © 2001-02 Nirixa, Inc.

nirixa

How to Approach Automation

- Requires commitment and appropriately trained resources
- Define the purpose and scope of automation
- Define the strategy : how's and what's?
  - It is a development effort in itself
  - What is the investment and what is the expected ROI
  - Script development is a large investment
- Walk before you run: build on early small successes
- Try automation early : even at unit testing time
- Planning, resource allocation is the key

Copyright © 2001-02 Nirixa, Inc.

nirixa

Increase ROI in either Method

- Your best bet is semi-automatic
- Define your overall testing strategy
  - Methodology and tools
  - Identify proper candidates for automation
- Must be process centric
  - Planning and documentation
  - Increases efficiency
  - And reduces resources (time & effort)
- Least common denominator approach
  - Planning, specification, test design, results and reports are common steps
- Building Blocks approach
  - Have component test suite library
  - Build larger complex tests using these as building blocks
  - Reduces resources required

Copyright © 2001-02 Nirixa, Inc.

nirixa

Summary

- Automation offers true value
  - Sophisticated automation tools are worth it
- Manual testing remains primary testing vehicle
- Winners use both automated and manual testing
- Planning, strategy & process are very important
- Leveraging earlier experiences increase ROI

Copyright © 2001-02 Nirixa, Inc.

nirixa

QW2002 Paper 6T2

Mr. Darin Magoffin, Mr. Todd Hovorka &
Mr. Rich Wolkins
(PowerQuest)

A Custom Automation Framework and Test Case Management Solution

Key Points

- If you can't find or buy the Test Case and Automation tools you need, you might try bui
- Automation Management with metrics and history reporting
- Integrated Test Case management tools with Automation tools

Presentation Abstract

Interested in seeing a real Test Automation solution in motion? Automation testing is an exciting thing to be a part of. Automating the automation is even more exciting. In this framework the Test Case/Automation system is set in motion after Configuration Management builds a piece of software for a project that has been automated. Thousands of pre-programmed test cases for multiple projects can be run at night or throughout the day on multiple machines. You will see a demonstration of this process in motion, learn what obstacles were overcome and how flexible the system is.

About the Author

Darin Magoffin, Todd Hovorka and Rich Wolkins have Software Automation Testing experience totaling more than 30 years combined. Using their skills and background in software testing, they have combined their strengths to create a robust Test Case/Automation Framework that suits the unique challenges of PowerQuest's products. Their experience and skill set have been aquired at companies like WordPerfect, Novell, Corel, PowerQuest, and Gazelle Systems.

www.powerquest.com

A Custom Automation Framework and Test Case Database

Automation can be your friend

Overview

- Why did PowerQuest build their own Test Case and Automation Framework?
- Obstacles overcome and problems solved
- Automation Framework Process Flowchart
- Development Tools used
- Automation Server/Client/Log Manager overview
- See TCA (Test Case Automation) Manager, Automation Server, Automation Client, Log Manager in a real live, “No smoke and mirrors” demo
- Q/A

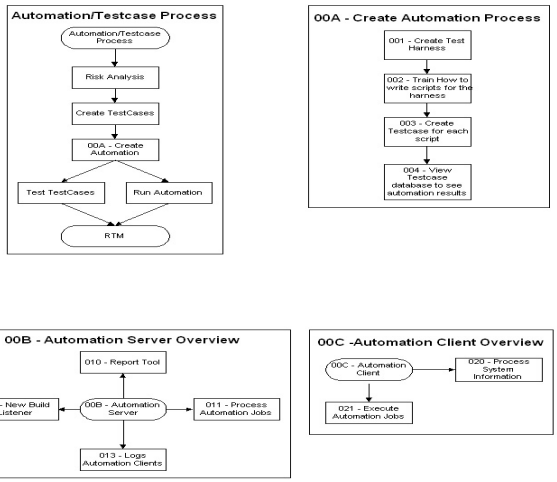
Why did PowerQuest build their own Test Case and Automation Framework?

- We had the expertise in house
- Existing Testing Frameworks, could not address unique issues that PowerQuest products have
- We needed a tightly integrated Test Case database with an automation framework
- We had existing automation that could more easily be “tweaked” to fit into a custom built system than into existing commercial systems

Obstacles overcome and problems solved

- Manager Buyoff
- Modifying already existing automation technologies to be integrated into the system
- Metrics at a project level for Test Case results both manual and automated
- Customizable Test Case matrix to allow for all combination of Operating Systems, File Systems, and Languages
- Automated submissions (Project Builds and Automation Jobs)

Automation Framework Process Flowchart



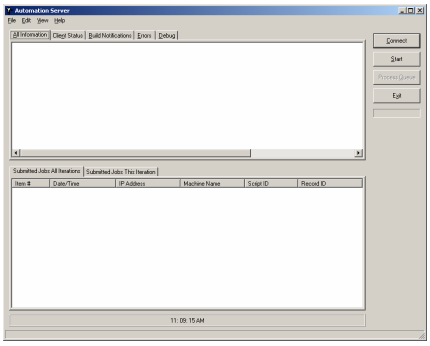
Development Tools used

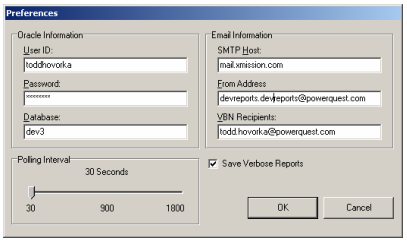
- Delphi 6 – (Automation Server, Automation Client, Verify Build Notification Client, Log Manager)
- Borland's C++ Builder 5 – (TCA Manager)
- Microsoft Visual C++ 6 – (Test harness for Calculator and PartitionMagic 8)
- AppTester – (Automation API tool that integrates with Microsoft Visual C++. It is used by the test harness to drive the applications being tested)

Automation Server Overview

- Listens for Automation Client requests
- Monitors Automation Run Queue for Pending Jobs
- Qualifies Jobs to Available Clients
- Farms out Jobs to Qualified Client Machines
- Handles all database requests
- Builds Client Profiles from system information
- Monitors & Displays all Client/Server communications

Automation Server Overview

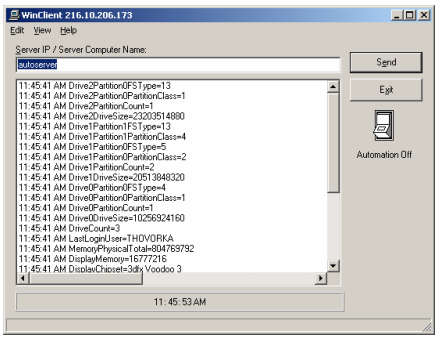


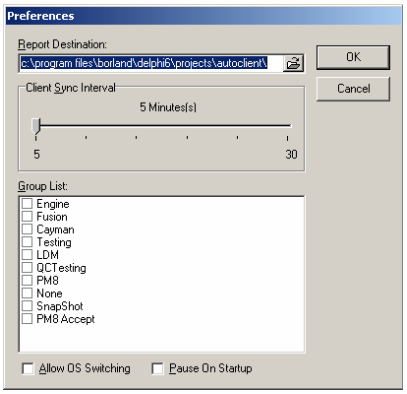


Automation Client Overview

- Gathers systems information for automation qualification and system analysis purposes
- Listens for, and executes Automation Jobs sent to it by the Automation Server

Automation Client Overview

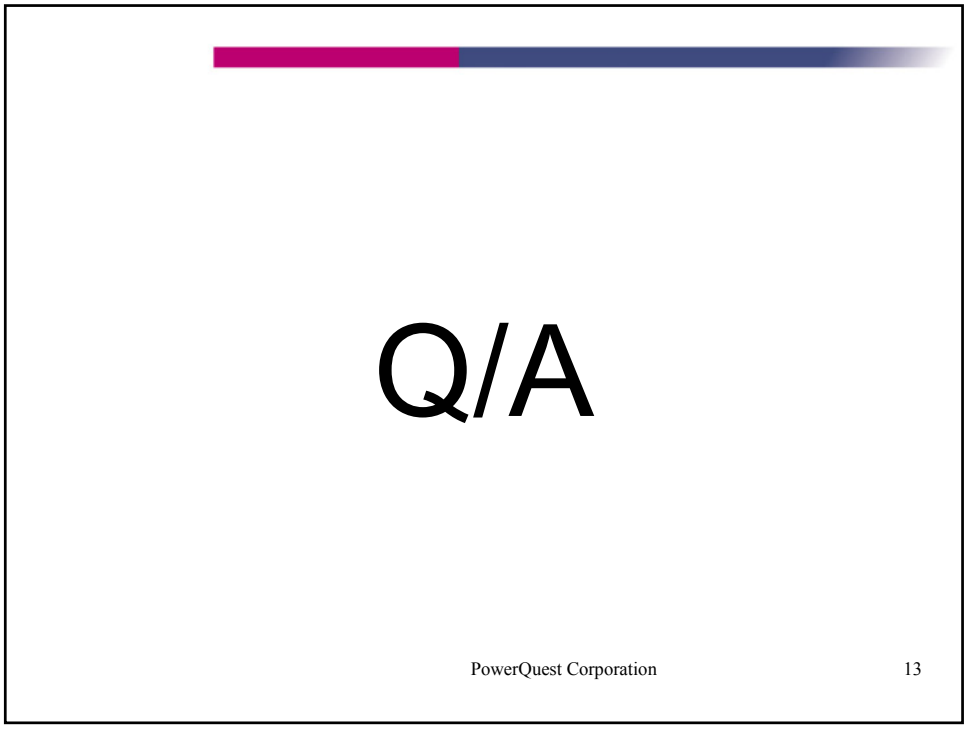




Log Manager Overview

- Implemented as an NT Service
- Monitors all specified folders for the existence of log files
- Parses all detected logs for specified tags
- Emails results to specified recipients
- Closes the automation job in the run queue
- Saves automation job results to the database

See TCA Manager, Automation Server, Automation Client, Log Manager in a real live, “No smoke and mirrors” demo


Q/A
PowerQuest Corporation
13

QW2002 Paper 7T1

Dr. Chris Overton (Keynote Systems) & Dr. Simon Robins
(Genimedia)

Streaming Media Quality: Orphan Child of "Old Media" and Internet Flakiness

Key Points

- Streaming media quality IS rigorously measurable in MOS-type metrics
- You also need low-level operational metrics tied to discrete events
- The challenge lies in bridging the gap - attemptable from both directions

Presentation Abstract

For standardized old media such as television, telephone, or even radio, there are mature infrastructures for quantifying quality, based on correlation to Mean Opinion Scores (MOS), which average over panels of human judges. This has not yet become widespread for streaming media, due to its variety, rapid evolution, novel degradations, and market immaturity. Nevertheless, we present rigorous pixel-based metrics that correlate well with MOS (Genimedia approach), as well as metrics built upward from low-level operational events (Keynote.) We discuss several important tradeoffs in the two approaches (interested audience, operational and SLA usability, intrinsic meaning, maintainability), as well as how we see the two approaches being brought together in the future. Many metrics are demoed under actual video degradations.

About the Author

Chris Overton is Keynote Statistician & Quantitative Architect. Over the last decade, he has consulted as a statistician in industry and in academic biomedicine, as a software architect and developer, and in business model development. He founded Crazy Tulip Corp. to build knowledge modeling software systems.
His responsibilities at Keynote include algorithm and tool design, data analysis & interpretation, internal & external education, and serving as academic liaison. He architected Keynote's SLA reporting engine and has helped several large companies build SLAs, including for streaming media both on the provider side and on the customer side.
Chris is the principal architect of Keynote's streaming media metrics and has published and lectured on related topics. He got his pure math PhD from Stanford in ?96 and has taught there and at the University of San Francisco.

Simon Robins has been working in software development for over 10 years, and joined Genimedia as Principal Engineer on its foundation at the start of 2001. Genimedia is dedicated to perceptual quality measurement for the monitoring, management and control of the production and delivery of digital media. He now leads the technical marketing team at Genimedia. He has a degree in theoretical physics from the University of Liverpool and a PhD in particle physics from London University.

Streaming Media Quality:

The Orphan Child of "Old Media" and Internet Flakiness

Chris Overton, Ph.D.
Keynote Systems
Simon Robins, Ph.D.
Genimedia

Plan for the talk

I. Big picture
II. Common challenges
III. Audio
IV. Video
V. State of the industry of quantifying streaming quality

Warning

- This talk's content is much easier to appreciate if you see the actual video and real-time metric tracking
  - "A picture is worth a thousand words"
  - "A video is worth a thousand pictures" (127.34 after MPEG4 compression)

Take-home message

- Quantifying streaming quality CAN be done rigorously
  - We'll discuss how – it's much trickier than one might think!
- However, it seems difficult at this time to build metrics that are both commercially viable and well-accepted by industry
- As streaming quality continues to become more of an expectation than a miracle, the current vacuum will beg to be filled

Wish list for a streaming media quality metric

1. A single number …
2. … transparently derived…
   - Having **conceptual** meaning
3. … universally accepted by industry …
4. … that tracks well with human panel judgment …
5. … and tells you what is going wrong, so you can fix it
   - Having **diagnostic** meaning

Wish list for a streaming media quality metric

1. A single number …
2. … transparently derived…
3. … universally accepted by industry …
4. … that tracks well with human panel judgment …
5. … and tells you what is going wrong, so you can fix it

THIS TALK:

- EXPLAINS HOW THIS IS UNACHIEVABLE
- SUMMARIZES CURRENT MARKET STATUS AND POTENTIAL

Speaker backgrounds & relevance to this talk

- Keynote Systems: “Streaming Media Perspective”
- Genimedia: “QoS solutions”
- This talk grew out of a discussion between the authors after they appeared on a panel at Streaming West 2002 (LA, April, 2002)
- We also take into account information shared under NDA by other vendors
- Current negative market conditions direct our emphasis to the general subject, rather than to the products
- Thus we are able to give more balanced insight into the industry of streaming quality and its measurement and assurance

A sought-after metric: MOS (mean opinion score)

- Average over ratings by several users, each on an n-point scale
- Averaging builds “consensus” and reduces variance due to atypical users
- Probability distribution of user’s ratings may appear much simpler than that of various metrics computed from hardware and software
- Goal: establish a good, **calculable** model for MOS

How well could streaming media MOS stack up with our stated goals?

- ☺ A single number
- 😐 Transparently derived
  - Averaging isn't hard – but settling on the budget and plan for distributed measurement is!
- ☹ Universally accepted by industry
  - Naïve approximations are recognized as such by some; others are uncomfortable with overly complicated or proprietary solutions
  - It's hard to find one party everyone can trust
- ☺ Tracks well with human panel judgment
- ☹ Tells you what's going wrong, so you can fix it
  - Not by a long shot!

Part II: Common challenges in building metrics for streaming

1) "Building the bridge" between MOS and diagnostic, operationally meaningful metrics
2) One-sided vs. two-sided measurement
3) The tradeoff between **diagnostic** and **conceptual** metrics
4) Understandable derivability (computability) of metrics
   (Even if you do something proprietary, can you convince your audience that that it is reasonable?)

These points are absolutely critical to our approaches (Genimedia and Keynote), even if we address some in different ways

We'll revisit this topic!

Part III: Metrics for Audio

A standard for comparison: speech quality over POTS

- A single MOS makes sense:
  - Expectations are straightforward
  - Extensive research exists on related psychology of perception
  - Connection-based packet communication is fairly consistent
  - Variability of experience is not that sensitive to end-points
- The main thing missing in MOS is latency
  - So (MOS & latency) tells you a lot

How MOS begins to fall apart for speech over phones

- Manageable differences:
  - VOIP introduces a different set of artifacts
  - Wireless extensions connected to POTS
- Mobile phones: a quantum leap in confusion:
  - Latencies may become more significant
  - Packet loss is higher even than in VOIP
  - Profound sensitivity to geography, which varies over time for a moving caller
  - Roaming can make responsibility difficult to assign

Why MOS until recently was even harder for streaming audio

1. Superposition of several kinds of variability:
   - Greater variety in audio streams than in speech
   - Encoded bandwidth
   - Other encoding features, such as sample rate and channel count
   - Different technologies and product versions
     - Sophisticated proprietary compression/decompression may emphasize differing features
   - Different behavior by Internet backbone and connectivity, “distance” from cache server
   - Variability over time

Why MOS until recently was even harder for streaming audio (II)

2. Several target qualities (say for different bandwidths), and so too many different sets of expectations to place neatly on a single MOS scale
3. New kinds of degradation:
   - Separation in time between packet arrival and packet playback through buffering, shielding yucky Internet statistical behavior
   - Codecs designed to allow greater packet loss
   - Changing encoding quality over time
4. Immature expectations, so that yesterday's MOS won't reflect tomorrow's users

State of the industry in measuring audio streaming quality

- At this point, marginally acceptable audio can fit over a phone line
  - Codecs are still improving slightly, and bandwidth is still growing for other connectivities
- Thus, one can develop MOS for a small number of genres (e.g. speech, acoustic music, electronic music) based on studio quality
  - Anything too low on this scale can then simply be written off a unacceptable

State of the industry in measuring audio streaming quality (II)

- Automatic calculation of MOS in real time is not so easy
- However, the impact of lost packets is fairly understandable, so the bridge between diagnosic and conceptual metrics is not huge
- Keynote's streaming service has made an attempt at such a bridge, which we will discuss more generally after covering video streaming
- Otherwise, this is a fairly mature subject that we will not survey

Part IV: Metrics for Video

Again, MOS metrics are harder for streaming than for TV

- Expectations have stabilized for analog television video, and similarly will stabilize for other formats such as HDTV; streaming is still evolving fast enough that expectations are changing
- Streaming video occurs at many different bandwidths, uses many different codecs, and introduces new kinds of degradations

So how do we build video streaming metrics to satisfy our five goals?

- The sheer visibility of artifacts makes this a compelling topic!
- Fundamentally, there are two possible kinds of approaches:
  - Building down from conceptual metrics closely associated with MOS (Genimedia)
  - Building up from discrete events such as lost packets, bandwidth allocations, or frame rendering behavior (Keynote)
- To contrast the two approaches, we'll consider each in turn

Genimedia

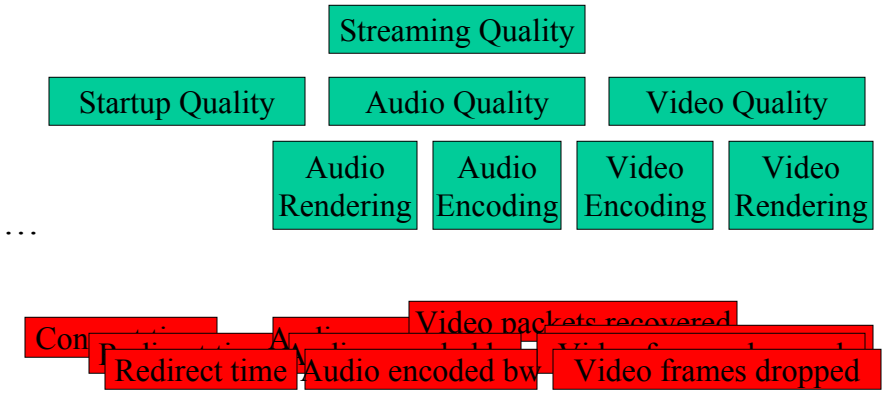
- Consider familiar kinds of degradations in video streams:
  - Blockiness
  - Blurriness
  - Jerkiness
  - Noise
- Genimedia has built a tool to estimate these from one-side measurement
- Observe how these track in the movie samples
- <show movies>

Genimedia (II)

- Since Genimedia's metrics are so high-level, they have intrinsic meaning and thus:
  - They serve as good predictors of MOS (within one fixed set of expectations)
  - The code is easily maintainable
- On the other hand:
  - If you are running a streaming service, you may be more interested to know that a particular cache server is dropping packets than in how this affects aggregate blockiness
  - You get some false positives, such as jerkiness at scene shifts

Keynote

- High-level metrics should have nearly consistent meaning across products and across product generations



- Low-level inputs vary across products and versions, and typically have crisper technical meaning

Keynote (II)

- Low-level metrics are diagnostic, and high-level metrics attempt to capture their impact (as derived by panel testing)
- This combination allows for broad comparison, together with more "actionable" data
- However:
  - Maintaining the "bridge" requires expertise and labor
  - Meaning of (sparser) proprietary higher levels is harder to understand

So how do we stack up to our wish list?

1. A single number …
2. … transparently derived…
3. … universally accepted by industry …
4. … that tracks well with human panel judgment …
5. … and tells you what is going wrong, so you can fix it

In an ideal world…

- One might want to squeeze a Genimedia-like into the Keynote hierarchy of metrics, and then calculate more rigorously how low-level metrics predict Genimedia ones
  - This would allow a "pixel-level gold standard"
- Even so, one will probably not see SLA's for "blurriness < 5%." – thus one still requires low-level data
  - Only after packet-level behavior is taken for granted will business contracts more more to higher-level conclusions

Market realities…

- Streaming is only beginning to become a viable business
- Thus, quality measurement and monitoring projects mostly have been put on hold
- Due to the limited use of third-party distributed testing services, the subject does not yet have enough inertia to develop widely accepted standards
  - Other than rudimentary ones, such as “packet loss rate”, “availability”, “average bandwidth”, …
- Even so, the way is already paved for more inclusive solutions!

Author contact info:

- Simon Robins:

  srobins@Genimedia.com

  www.simonrobins.com
- Chris Overton

  chris@crazytulip.com

QW2002 Paper 7T2

Mr. Srivatsa Srinivasan & Mr. Sandeep Prabhu
(Microsoft)

SPARTA: Architecture for Automated Testing of Network Protocol Stacks

Key Points

- Automated testing of network protocol implementations
- Case studies of TCP/IP and TCP/IPv6 testing
- Rapid test development

Presentation Abstract

This paper discusses SPARTA (Scripted Protocol- And Regression- Testing Architecture), a tool that is used to automate the testing of network protocol stacks. The first part of the paper discusses the motivation behind the SPARTA toolset. The second part discusses the SPARTA architecture. The final part of the paper presents case studies on how SPARTA was used to test the TCP/IP and TCP/IPv6 network protocol stack implementations on Microsoft Windows XP and Microsoft Windows CE.

About the Author

Srivatsa Srinivasan has been working in the Windows CE Platform Group at Microsoft for the past 3 years as a Software Design Engineer in Test. His work has involved developing tests and tools to test various networking features of Windows CE. Currently his focus is on developing tests for TCP/IPv4 and TCP/IPv6 networking protocols.

Sandeep Prabhu has been working in the Windows Networking Group at Microsoft for the past 3 years as a Software Design Engineer in Test. His work has involved developing components and tools to test the various networking protocols. Currently his focus is on developing tests for TCP/IPv4 and TCP/IPv6 networking protocols.

welcome

SPARTA : Architecture for Automated Testing of Network Protocol Stacks

Srivatsa Srinivasan
Sandeep Prabhu
Microsoft Corporation

Agenda

- Introduction
- Motivation
- Architecture
- Examples
- Conclusion

Introduction

- SPARTA stands for **S**cripted **P**rotocol- **A**nd **R**egression-**T**esting **A**rchitecture
- It is a tool that is used to automate the testing of network protocol stacks

Motivation

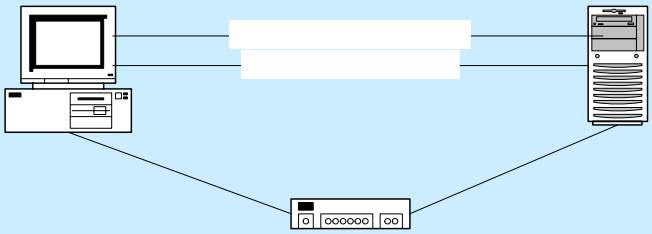
- A toolset that can run on a PC and perform generic network protocol stack testing with no requirements for any special hardware
- A toolset that can provide total flexibility in what the user can put on the wire given that such capability is often needed to do low-level protocol testing
- A toolset that provides a mechanism to remotely control the implementation under test at a very granular level providing very powerful primitives to the test developer to test complex network protocol stack features

Motivation

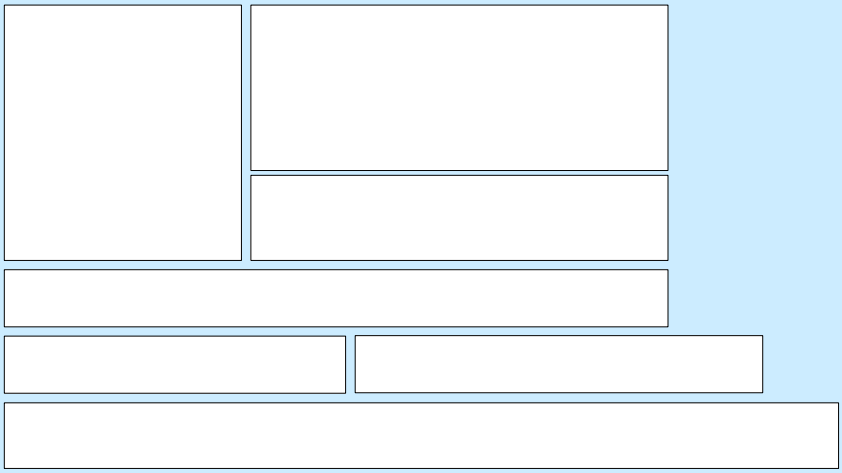
- A toolset that enables rapid test development, by providing a scriptable interface to test developers who can quickly write test scripts in common scripting languages like VBScript or Jscript, leading to greater testing efficiency
- A toolset that is easily extendable to add functionality for testing existing or yet to be designed network protocols
- A toolset that can be easily integrated with other tools and applications

Architecture

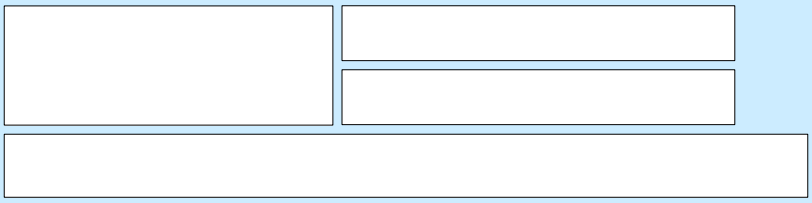
- SPARTA Host
- System Under Test
- SPARTA Control Channel
- SPARTA Data Channel



Architecture – SPARTA Host



Architecture – System Under Test



Architecture – SPARTA Control Channel

- A client-server mechanism that allows the SPARTA Host to remotely control the System Under Test by using a wide range of control primitives
- A client component runs on the SPARTA Host that communicates with a server component that runs on the remote System Under Test
- C++ and COM interfaces are provided on the SPARTA Host to invoke the control channel primitives. They provide a very powerful tool for the test developer to coordinate complex protocol interactions between the SPARTA Host and the System Under Test

Networ

Architecture – SPARTA Control Channel

- The following control primitives are supported:
  - System Primitives: Set the configuration parameters of the System Under Test, get system status/statistics
  - Stack Primitives: Set the configuration parameters of the network stack under test, get network stack status/statistics (e.g. TCP/IP parameters and statistics)
  - Socket Primitives: Create sockets (endpoints), make the System Under Test establish connections or accept connections on these sockets, send and receive data on these connections, get the socket status/statistics

SPARTA Host – SPARTA Driver

- It is a module that sits in between the SPARTA API at the top and the network driver at the bottom
- On the send path, it puts raw packets on the wire as handed down by the upper layer
- On the receive path, packet filtering is supported by the SPARTA driver. Other than filtered packets, the SPARTA driver passes all packets to the upper layer unaltered

SPARTA Host – SPARTA Driver

- Provides comprehensive queuing (important for slow script execution and bursty traffic)
- Packet filtering provides improved performance and reduces script complexity
- Multiple receive queues and filters are supported
- Following two levels of filtering are supported:
  - “Regular” filters such as for broadcast, directed, multicast etc.
  - Pattern based filtering - a program can specify a set of patterns for incoming traffic to be filtered on

SPARTA Host – SPARTA Protocol API

- A comprehensive set of objects describing packet structures found in the TCP/IP protocol suite
- Packet operations supported in objects (e.g. checksum and length calculation)
- Network driver functionality encapsulated in interface objects
- Automation of common packet exchanges (e.g. address resolution using ARP)
- Packet classes are designed in a layered fashion – mirroring packet header structures

SPARTA Host – SPARTA Protocol API

- Exposed as a C++ object model for use in high performance applications or for integrating SPARTA with other tools
- Exposed as a COM object model for scriptability using any popular scripting language like VBScript, JScript etc.
- Objects make the following available:
  - Network Interface commands, Pattern filters
  - Address Objects, Packet Objects (arranged in an hierarchical structure), Header Objects (correspond to protocol headers in the packets)
  - Exception handling

SPARTA Host – SPARTA Protocol API

- The following packet types are supported:
  - MacPacket – encapsulates the media type based frame (e.g. ethernet or token ring)
  - IpPacket – encapsulates the IPv4 packet
  - Ipv6Packet - encapsulates the IPv6 packet
  - ArpPacket- encapsulates the IPv4 address resolution packet

SPARTA Host – SPARTA Protocol API

- The following packet types are supported:
  - IcmpPacket – encapsulates the IPv4 ICMP packet
  - IgmpPacket (versions 1 and 2) – encapsulates the IPv4 IGMP packet
  - Icmpv6Packet- encapsulates the IPv6 ICMP packet
  - UdpPacket - encapsulates the UDP packet
  - TcpPacket - encapsulates the TCP packet

SPARTA Host – Utilities

- Apart from the Protocol and Control API sets, there are some utility APIs to perform the functions described below
  - Test results logging
  - Test variation tracking
  - Other utility functions

Examples –Testing TCP/IP

- Goal: Test the TCP SYN retry count feature (the TCP SYN retry count is the number of times a TCP/IP node retransmits the SYN packet before giving up)
- The test is implemented as a VBScript script that invokes the SPARTA Protocol and Control COM APIs
- Representative of several test cases that have been developed for the various features in the TCP/IP and TCP/IPv6 protocol suites
- Let us examine script as follows:
  - Script outline
  - Walk through each section of the script

Examples –Testing TCP/IP

- Script outline
  - Initialization
    - Create and initialize SPARTA Utility Object
    - Create and initialize SPARTA Protocol Object
    - Create and initialize SPARTA Controller Object
    - Create and initialize Network Interface Object
  - Test
    - Perform test setup tasks using Control channel
    - Run the test (create and send packets, receive packets, send commands etc.)
    - Analyze the results
  - Cleanup

Examples –Testing TCP/IP

```
Dim Core, Sparta, AutoSrv, IFace, pControllerObj

' Create the TestCore utility object
  set Core = CreateObject("Testcore.base.1")

' Create the Protocol object
  set Sparta = CreateObject("Spartacom.base.1")
  Sparta.InitLogging Core

' Create the Controller object
  set AutoSrv = CreateObject("AutoSrvcom.base.1")
 AutoSrv.InitLogging Core
```

Examples –Testing TCP/IP

```
' Create the AutoARP object
   Set AutoArpObject =
    Sparta.AutoArp(g_szLocalMacAddress,
    g_szLocalMacAddress, g_szLocalSpoofIp)
' Create the Network Interface object
  set IFace =
    Sparta.DriverInterface(g_szLocalMacAddress)
 ' Set the receive modes for the network interface
   IFace.EnableDirectedReceiveMode
   IFace.EnableBroadcastReceiveMode
```

Examples –Testing TCP/IP

```
' Enable filtering based on patterns
IFace.EnablePatternFiltering

' Create a pattern to filter by destination MAC address
Set pMacAddressObj =
  Sparta.MacAddress(MediaType,
  g_szLocalMacAddress)
Set pPattern =
  Sparta.CreateMacDestAddressPattern(MediaType,
  pMacAddressObj)
IFace.AddPatternFilter pPattern
```

Examples –Testing TCP/IP

```
' Create a pattern to filter by source IP address
Set pPattern =
  Sparta.CreateIpSrcAddressPattern(MediaType,
  g_szRemoteIpAddress)
IFace.AddPatternFilter pPattern

' Create a pattern to filter by SYN flag
Set pPattern =
  Sparta.CreateHeaderRelativePattern(MediaType,
  33, 1, Array(2))
IFace.AddPatternFilter pPattern
```

Examples –Testing TCP/IP

```
' Start listening
  IFace.StartListening

' Create the object used to specifically control System
    Under Test
Set pControllerObj =
    AutoSrv.InitController(g_szLocalIpAddress,
    g_usControllerPort, g_szRemoteIpAddress,
    g_usAutoSrvPort)

' Send commands over the control channel to the
    System Under Test to create a socket, bind to it and
    issue a connect socket call so we can test the TCP
    SYN retry count functionality
```

Examples –Testing TCP/IP

```
ulStatus =
    pControllerObj.CreateAndConnect(SOCK_STREA
    M,    g_szRemoteIpAddress, usRemotePort,
    g_szLocalSpoofIp, usLocalPort, ulTID,
    dwSessionId)

 if (ulStatus <> 0) then
     Core.FailSev1Variation "SYNRetryCountTest:
    CreateAndConnect - " & ulStatus
      exit Sub
 end if
```

Examples –Testing TCP/IP

```
' Make sure we receive 3 SYNs as that is the expected
  default TCP SYN retry behavior

  For ulCounter = 1 to 3
     Set pTCPPacket = IFace.ReceiveTimed(7000)
     if (pTCPPacket is Nothing) then
        Core.FailSev1Variation "SYNRetryCountTest:
 Failed to receive 3 SYNs"
        exit Sub
     end if
  Next
```

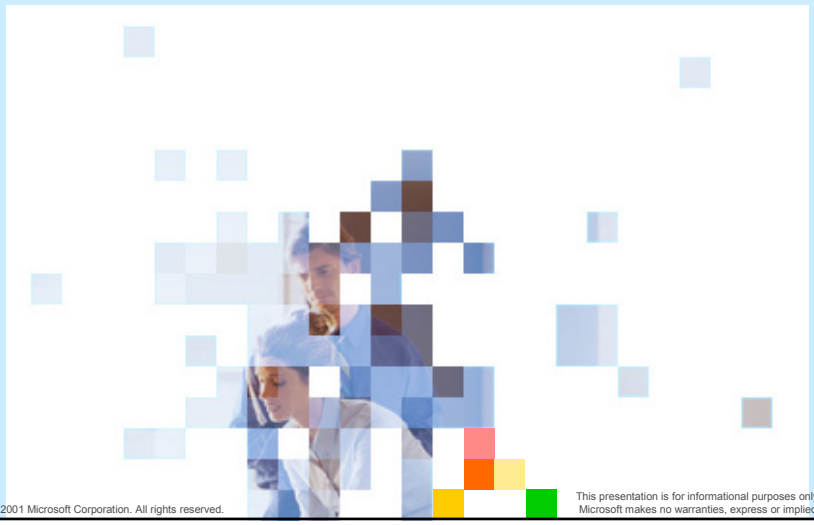
More examples

- Testing configurable network stack parameters (TCP SYN Retry Count test revisited)
- Testing timing related features (TCP Zero Window Probing test)
- Testing IPv6 (IPv6 Forwarding test)
- Testing network stack implementations on multiple OSes

Conclusion

- SPARTA is a test tool for automating network stack testing
- SPARTA does not require any special hardware, runs on a PC
- SPARTA provides a very easy scriptable interface for all its features
- SPARTA allows for granular control of the System Under Test during test execution
- TCP/IP and TCP/IPv6 tests have been developed to test the implementations on several Microsoft OSes

Microsoft



© 2001 Microsoft Corporation. All rights reserved.

This presentation is for informational purposes only
Microsoft makes no warranties, express or implied, in this summary.

QW2002 Paper 8T1

Dr. Ira Baxter
(Semantic Designs, Inc.)

Branch Coverage Tools For Arbitrary Languages made Easy!

Key Points

- How branch coverage is typically implemented
- How to implement branch coverage with a program transformation tool
- Can get coverage tools for any language this way

Presentation Abstract

Branch coverage tools are an important means for determing how well tested software is. Such tools are hard to obtain for nonstandard languages or dialects, or for unusual execution environments. This talk shows how a program transformation tool can be used to implement branch coverage easily for virtually any programming language or environment. Consequently, even software with unusual languages can obtain valuable test coverage information.

About the Author

Dr. Baxter has been building systems software for over 30 years. He is presently the Chief Technology Officer of Semantic Designs, a software-tools building company. He is also active in academic conferences, and is presently the Program CoChair for the International Conference on Software Maintence 2002 (Montreal).

Branch Coverage for Arbitrary Languages Made Easy!

Ira D. Baxter
Semantic Designs, Inc.
www.semdesigns.com

September 2002

Overview

- What is branch coverage?
- How branch coverage is typically implemented
- How to implement branch coverage with a program transformation tool
- Can get coverage tools for any language this way

When is Software Tested Enough?

- Ideal: Prove software meets formal specification
  - Oops: don't have formal specification
  - Oops: don't have theorem prover
  - Oops: theorem proving takes a long time
  - Oops: compiler doesn't implement source program
  - Oops: don't have skilled people to do this
- Approximation: Validate using test cases
  - Oops: may not have enough test cases
    - How can we possibly know?

Software Testing: How (Much)?

- Ideally test:
  - Product implements all specifications correctly
  - Product doesn't implement anything else
  - Need lots of test cases!
- Issue: When to *stop* testing?
  - Too little: product fails in field → company failure
  - Too much: → miss market; overtested → late
- Test Coverage: When enough code tested, stop!
  - *Code not tested is suspect; decide if more tests needed*
    - *How can we identify such code?*
  - Note: Other stopping criteria possible

Line Coverage

Tracks source lines executed

```
bool fibcached[1000];
int  fibvalue[1000];

int fib(int i)
{ int t;
  switch (i)
  { case 0:
    case 1: return 1;
    default:
      if fibcached(i)
         return fibvalue(i);
      else { t=fib(i-1);
             return t+fib(i-2);
           };
  };
};
```

Unexecuted. Should we ship?

5 lines executed
15 lines total
= 30% (line) coverage

} *Observation: Don't learn much from line coverage of sequential lines*

Executed

Branch Coverage

Track blocks of code controlled by conditionals

```
bool fibcached[1000];
int  fibvalue[1000];

int fib(int i)
{ int t;
  switch (i)
  { case 0:
    case 1: return 1;
    default:
      if fibcached(i)
         return fibvalue(i);
      else { t=fib(i-1);
             return t+fib(i-2);
           };
  };
};
```

Code Block

Executed

Better criteria:
Doesn't count <u>un</u>executable lines.

4 branch points executed
6 branch points total
= 64% (branch) coverage

} *Block executed if true conditional*

} *Block of code executed if conditional is false*

Test Coverage by Marking visited Blocks

```
bool fibcached[1000];
int  fibvalue[1000];

int fib(int i)
{ int t;
  switch (i)
  { case 0:
    case 1: return 1;
    default:
      if fibcached(i)
         return fibvalue(i);
      else { t=fib(i-1);
             return t+fib(i-2);
           };
  };
};
```

Original "C" program

```
bool fibcached[1000];
int  fibvalue[1000];

int fib(int i)
{ int t;
  visited[1]=true;
  switch (i)
  { case 0: visited[2]=true;
    case 1: visited[3]=true;
            return 1;
    default:
      visited[4]=true;
      if fibcached(i)
         { visited[5]=true;
           return fibvalue(i);}
      else { visited[6]=true;
             t=fib(i-1);
             return t+fib(i-2);
           };
  };
};
```

Marked program

Inserting Probes in Source Code

Why not do it this way?

- Requires sophisticated tools to modify source
  - Must understand structure of source language
- Until recently, such tools *not* available
  - Object code probe insertion is only remaining alternative
- Industrial-strength program transformation systems
  DMS, **TXL**, **XT**, **Refine**, …
  - Understand syntax/semantics of languages
  - Based on generalized compiler technology
  - Carry out language-sensitive modifications
  - Can be harnessed to insert probes in source code

More on these later…

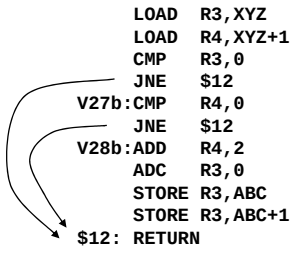
Putting Probes in Object code

Because everybody has it…(?)

- Can implement with compiler switch
  - Compiler generates probes woven into object
  - Requires compiler vendor to implement coverage
- Can instrument object code directly
  - Compiler vendor independent
  - Only need vendor to generate standard object
  - Harder to implement reliably

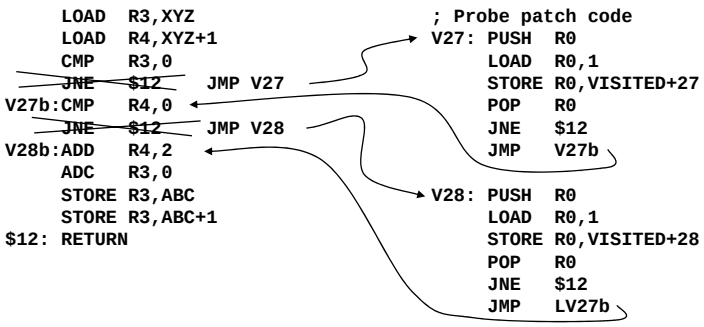
Inserting Probes in Object code

- Modify load image using linker symbol table information
  - Trace through object code looking for JMPs
  - Determine for each JMP the matching source code
  - Replace each JMP with a JMP to a probe-patch
- Replace exit logic with call to dump coverage vectors



Inserting Probes in Object code

- Modify load image using linker symbol table information
  - Trace through object code looking for JMPs
  - Determine for each JMP the matching source code
  - Replace each JMP with a JMP to a probe-patch
- Replace exit logic with call to dump coverage vectors



Inserting Probes in Object code:

Problems

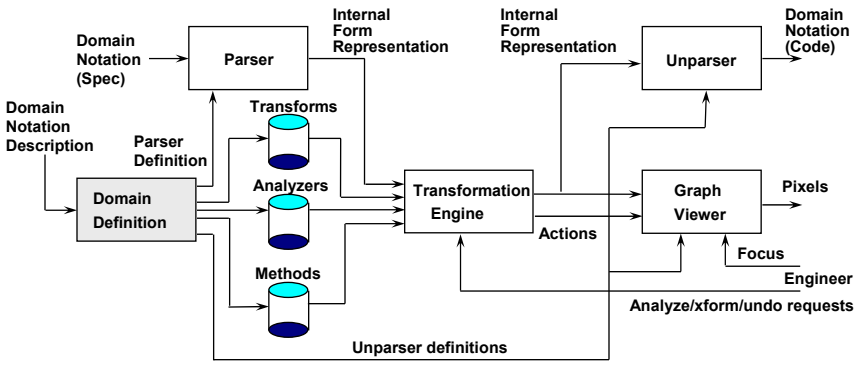
- Trace through object code looking for JMPs
  - Trouble: distinguish code from data → crash if wrong!
  - Trouble: discovering JMPs implicit in runtime conventions
  - Performance: instrument libraries whether desired or not
- Determine for each JMP the matching source code
  - Can't always uniquely determine source position matching JMP
    - Object code source information typically line-oriented
    - JMP in object code may not match any explicit source construct
- Replace each JMP with a JMP to a probe-patch
  - Performance: JMPs common
    - JMP out, save registers, do probe, restore, JMP back
  - Space: each JMP expands → program footprint grows
  - Trouble: Original JMP small/precedes entry point → almost impossible to patch correctly
- *Trouble: Not commodity platform/language: not available*
- *Trouble: interpretive language → No object code!*

DMS® Software Reengineering Toolkit

An industrial-strength Program Transformation System

- Enables wide variety of source-based SE tasks to be automated
  - Analysis and Testing
  - Code Generation
  - Reverse Enginering, Modification, Legacy System Porting
- For sources for large scale software systems
  - Scalable to millions of source lines, tens of thousands of files
  - Parallel processing foundations to support scale
- Handles many *and mixed* languages simultaneously
  - C, C++, Java, Ada, Fortran, SQL, XML, assembler, …
- Generalized compiler technology conveniently *integrated*
  - Parsing, Analyzing, *Transforming*, Prettyprinting
  - Enables practical customization for desired automation task
  - Predefined support for most standard computer languages
  - Huge infrastructure cost amortized over many tasks/customers

DMS Core



DMS Domain Definition for Java Parser + Pretty Printer

```
nested_class_declaration = nested_class_modifiers class_header class_body ;
  <<PrettyPrinter>>:  { V(H(nested_class_modifiers,class_header),class_body); }

class_header = 'class' IDENTIFIER ;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER); }
class_header = 'class' IDENTIFIER 'implements' name_list ;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER,'implements',name_list); }
class_header = 'class' IDENTIFIER 'extends' name;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER,'extends',name); }
class_header = 'class' IDENTIFIER 'extends' name 'implements' name_list ;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER,'extends',name,'implements',name_list);

class_body = '{' class_body_declarations '}' ;
  <<PrettyPrinter>>:  { V(H('{',STRING(" "),class_body_declarations),'}'); }

nested_class_modifiers = nested_class_modifiers nested_class_modifier ;
  <<PrettyPrinter>>:  { H(CH(nested_class_modifiers[1]),nested_class_modifier); }
```

… + 300 more rules…(COBOL is 3500!)

Java transform in Rule Spec Language

```
default base domain Java;

rule merge-ifs(\condition1,
               \condition2,
               \then-statements)
   "if \condition1
       if \condition2
          { \then-statements
          }
   "
   rewrites to
   "if \condition1 && \condition2
       { \then-statements }   ";
```

Domain

Domain Syntax

Transforms source pattern, whereever found, into target pattern

DMS transform(s) to mark program

```
default base domain C;

rule mark_function_entry(result:type, name:identifier,
                         decls:declaration_list, stmts:statement_sequence) =
        "\result \name { \decls \stmts };"
                rewrites to
        "\result \name { \decls { visited[\place\(\stmts\)]=true; \stmts };".

rule mark_if_then_else(condition:expression; tstmt:statement; estmt:statement) =
        "if (\condition)\tstmt else \estmt;"
                rewrites to
        "if (\condition)
            { visited[\place\(\tstmt\)]=true; \tstmt}
            else {visited[\place\(\estmt\)]=true; \estmt};".

rule mark_while_loop(condition:expression, stmt:statement) =
        "while (\condition) \stmt"
                rewrites to
        "while (\condition) { visited[\place\(\stmts\)=true; \stmt }".

rule mark_case_clause(e:expression, stmts:statements) =
        "case \e: \stmts"
                rewrites to
        "case \e: { visited[\place\(\stmts\)=true; \stmts }".
```

A few rules for each branching construct in the language…
~~ 50 rules or 300 lines total

Transforms: Source to Test Program

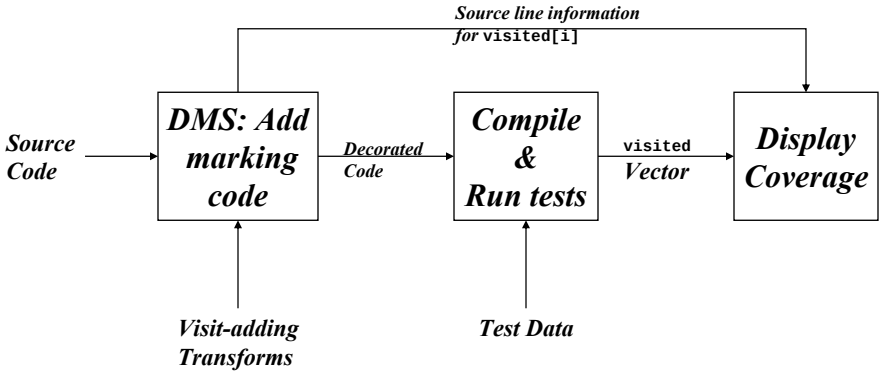
```
bool fibcached[1000];
int  fibvalue[1000];

int fib(int i)
{ int t;
  switch (i)
  { case 0:
    case 1: return 1;
    default:
      if fibcached(i)
         return fibvalue(i);
      else { t=fib(i-1);
             return t+fib(i-2);
           };
   };
};
```

Transformed To

```
bool fibcached[1000];
int  fibvalue[1000];

int fib(int i)
{ int t;
  visited[1]=true;
  switch (i)
  { case 0: visited[2]=true;
    case 1: visited[3]=true;
            return 1;
    default:
      visited[4]=true;
      if fibcached(i)
         { visited[5]=true;
           return fibvalue(i);}
      else { visited[6]=true;
             t=fib(i-1);
             return t+fib(i-2);
           };
   };
};
```

Original "C" program ***Decorated program***

Branch Test Coverage Tool Flow



Note: incrementing `visited` *rather then setting true changes this to profiler tool!*

Input to Coverage Tool

Source base directory

Target directory

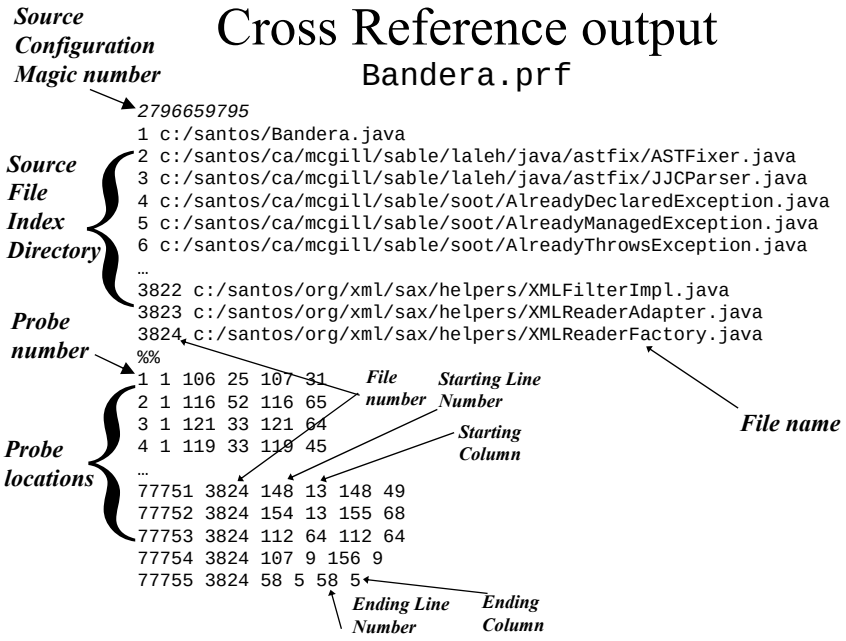
Probe cross reference result

Files to Probe

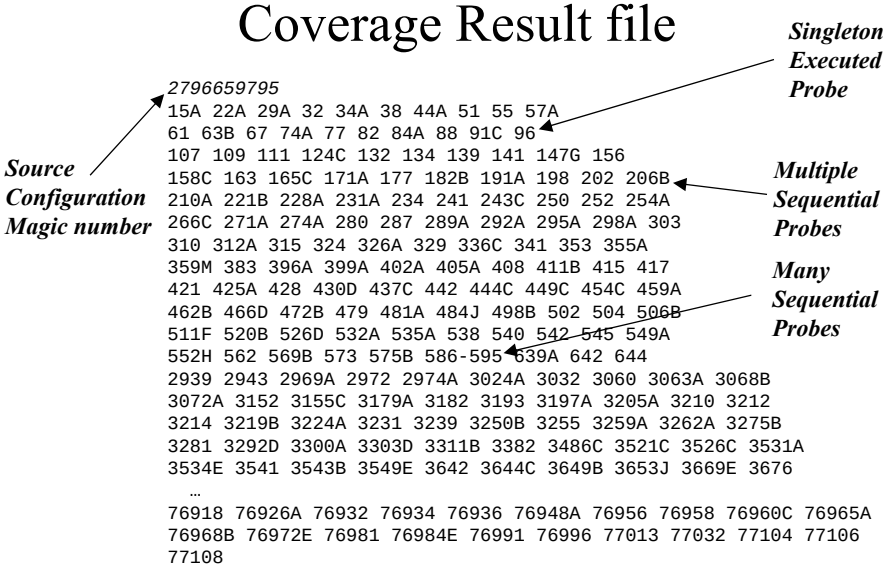
```
#     Total Files Listed:
#           3825 File(s)     31,485,751 bytes
#     Total packages 133.
C:\santos\workdir
Bandera.prf
C:\santos\workdir

Bandera.java
ca\mcgill\sable\laleh\java\astfix\ASTFixer.java
ca\mcgill\sable\laleh\java\astfix\JJCParser.java
ca\mcgill\sable\soot\AlreadyDeclaredException.java
ca\mcgill\sable\soot\AlreadyManagedException.java
ca\mcgill\sable\soot\AlreadyThrowsException.java
ca\mcgill\sable\soot\AmbiguousFieldException.java
ca\mcgill\sable\soot\AmbiguousMethodException.java
ca\mcgill\sable\soot\ArrayType.java
ca\mcgill\sable\soot\BaseType.java
ca\mcgill\sable\soot\Body.java
ca\mcgill\sable\soot\BodyExpr.java
ca\mcgill\sable\soot\BodyRepresentation.java
ca\mcgill\sable\soot\BooleanType.java
ca\mcgill\sable\soot\BuildAndStoreBody.java
ca\mcgill\sable\soot\BuildBody.java
        …
```

Cross Reference output

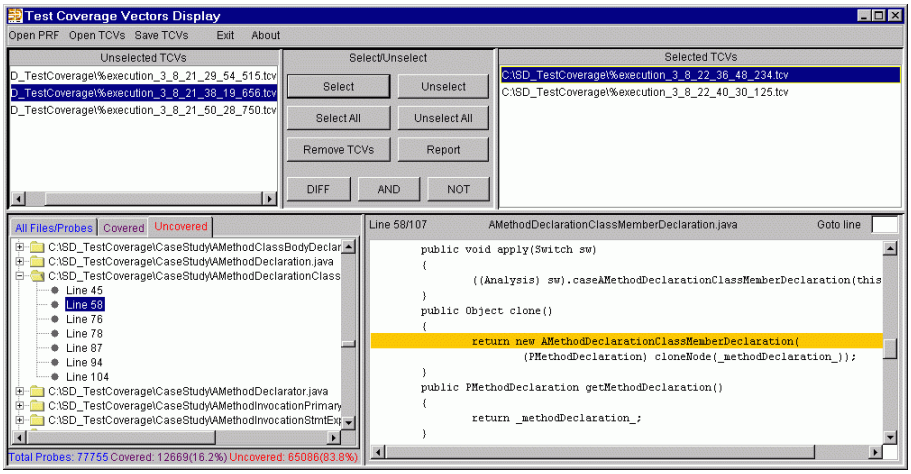
Bandera.prf



Coverage Result file



Coverage Display Tool



Additional Coverage Facilities

- Ability to do arithmetic on coverage vectors
  - OR: Combines test case runs
  - AND: Tells which tests cover same code
  - DIFF: Indicates tests that hit uncovered code
- Summary of coverage
  - Overall coverage totals
  - Per module
  - Per file

Practicality

- Implementations for
  - ANSI C 89, COBOL 85, Java
  - PARLANSE (parallel programming language)
  - Considering JavaScript, HTML, Perl, Python,…
- Performance scales well
  - Tested with Java system: 3800 *files*
  - 15% typical overhead; 50% in tight loops

Simple Extensions

- Multiple simultaneous languages
  - C + Assembler
  - Java + JavaScript
  - …
- Profiling
  - Uses counters instead of booleans
- Modified Condition/Decision Coverage
  - Verifies subconditions actually control branch
    - Required for DO178B Aviation Software Standards
  - Probable utility for Hardware Langauges
    - VHDL, Verilog

Branch Coverage: Summary

- Branch Coverage
  - An effective means for deciding "how tested"
- Object Code Instrumentation
  - Hard, not available for arbitrary languages/environments
- Industrial Strength Program Transformation Systems
  - Used to automate analysis and source modifications
- Source Code Instrumentation
  - Easily implemented as few hundred lines of program transforms
- *Practical Test Coverage Tools*
  - Small increment beyond program transforms: Display tool
  - Work for compiled *and* interpreted languages
  - Can handle large scale applications: many files and mixed languages
  - Can work for arbitrary execution environments

QW2002 Paper 8T2

Thomas Arnold
(Xtend Development, Inc.)

Automating Testing on ASP.NET Applications

Key Points

- Microsoft's Tools for Testing ASP.NET
- Creating an Automated Script with ACT 1.0
- Beyond Basics: Load Tests, Cookie Management and more

Presentation Abstract

With the release of Microsoft .NET comes new questions and challenges in testing. How does one go about automating the testing within the .NET Realm?

In this presentation Tom Arnold will look at Microsoft ACT 1.0, a new automation tool that was released with Visual Studio .NET in February 2002. This tool is full featured, allows for functional testing as well as load testing, and best of all, it's free with Visual Studio .NET.

About the Author

Tom Arnold has been programming, managing and consulting on software development & test automation projects since 1991. In 1993, Tom co-founded Software Testing Laboratories (later renamed to ST Labs, and eventually purchased by Data Dimensions / LionBridge / Veritest), one of the software industry's first outsourced software testing firms.

While at ST Labs, Tom added training & consulting to his repertoire when he began writing and teaching software test automation classes to Microsoft employees as he created ST Labs' training group. Two years later, Arnold published the book, Software Testing with Visual Test 4.0. Shortly after his book's publication, Tom managed the development team for Visual Test 4.0b for Microsoft Corporation; this version later became Rational Visual Test 4.0r.

Arnold continued to run software test automation projects, consult for companies looking to establish an approach to testing, and speak at industry conferences about effective uses and practical approaches to automated testing. Tom continued managing software development teams, including the programmers and test engineers that created Rational Visual Test 6.0 (released in November 1998). His 700-page book— Visual Test 6 Bible – and 10-tape (10-hour) training video series – VT6 InDepth—were both published in January 1999, winning a number of awards.

Tom's current focus is on software project management, programming, test automation and writing about software development topics. Tom maintains an active role in the software industry and presents at such conferences as STAR (Software Testing Analysis & Review), Internet World, and RUC (Rational Users Conference). His Bachelors Degree in Computer Science comes from Purdue University.

Automating Testing of ASP.NET Applications

(an introduction)

Thomas Arnold

Xtend Development

www.XtendDevelopment.com

Topics

- Microsoft .NET Framework (a brief introduction)
- Be aware of migration issues
- Inherent challenges for testing
- Testing deployed projects
- Our examples: ASP.NET
- More information

What You Walk Away With

- A general understanding of .NET
- Issues programmers face in moving to .NET (things that may result in bugs)
- Challenges you face as .NET attempts to hold our hands
- Tools/methods/tricks to help you test an ASP.NET deployed web application
- Where to continue learning about .NET

Why .NET?

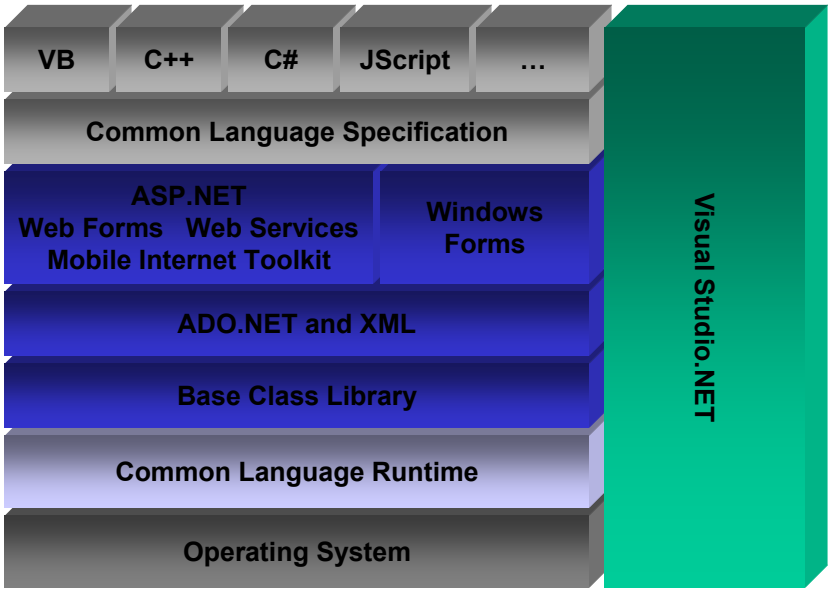
- Cements the relationship between Windows and Web developers
- Simplifies deployments of applications (xcopy)
- Better versioning support to avoid "DLL Hell"
- Provides a Framework that everything can work within

Why .NET? (cont'd)

- Security policies are more easily applied
- Makes it easy for other languages to be used in Windows/Web dev
- Allows sharing of classes/libraries

Let's look at the Framework…

.NET Framework



.NET Framework (cont'd)

- Any language can be plugged into the Framework by following the CLS (Common Language Specification)
- 80+ "namespaces" exist that are essentially class libraries used by these languages
- These "namespaces" allow for a common method of working with data bases (through ADO.NET and XML)

.NET Framework (cont'd)

- The program language is compiled down into a Common / Intermediate Language (IL) that works with the CLR (Common Language Runtime)
- The CLR works with the operating system. By providing that layer, a CLR can be written for Macintosh, Linux, and so on, without changing source code

Migration Issues

Many development organizations will want to take advantage of these benefits:

- Write new applications in .NET
- Migrate existing code to .NET

Migration Issues (cont'd)

With migration comes potential bugs:

- New types have been introduced. For VB 6 vs. VB.NET, for example:
  - No more fixed-length strings:
    ```
    dim Name As String * 30
    ```
  - VB 6's `LONG` equals VB.NET's `INTEGER`
  - VB 6's `INTEGER` equals VB.NET's `SHORT`
  - VB.NET's `LONG` is now a 64-bit integer
  - VB.NET does not support `VARIANT`

Migration Issues (cont'd)

- VB6 to VB.NET migration issues (continued)
  - Variable declarations have changed

```
dim x, y as long
'In VB6: x = variant, y = long
'In VB.NET: x and y are declared long
```

  - Keywords have been altered

    The `Empty` and `Null` keywords in VB6 are replaced by `Nothing` in VB.NET

Migration Issues (cont'd)

- VB6 to VB.NET migration issues (continued)

  Individual VB6 math functions have been implemented as members of the Math class in the System namespace

```
X = Cos(y) 'old method in VB6
X = System.Math.Cos(y) 'new
```

- Many others issues, not to mention the issue of re-opening code that is already working and the bugs that can result

Other Challenges

- One of the as-yet undefined challenges is how the Framework holds our hand
  - Benefit in that it avoids additional special-case coding for different browsers, versions, OSes, etc.
  - Potential issue in that its hand-holding may make it more difficult to work-around issues

Other Challenges (cont'd)

- ASP.NET and Visual Studio.NET have "WebForms"
  - Controls on a WebForm are not directly mapped to HTML
  - Execution of the WebForm and its ASP.NET code runs on the server
  - Now code is tailor made for each user

Other Challenges (cont'd)

- Compatibility testing has taken on a whole new meaning:
  - ASP.NET runs on the server-side allowing the .NET Framework to create HTML appropriate to the user
  - The HTML that's generated depends on the browser brand, version# and OS

Other Challenges (cont'd)

- That means that a text box on a WebForm could translate to one of two HTML equivalents:

```
<input type="text">
        -or-
   <textarea>
```

Other Benefits

- Additional benefits of .NET exist, however:
  - Better error handling (`try / catch / finally`)
  - Easier to track a user's session state (less programming)
  - Easy to globally configure an app

ASP.NET Example

We look now at a full-blown ASP.NET deployed application and consider methods of testing

Testing .NET

- It remains a "black box"
- Visual Studio.NET includes tools for testing a web app
- `Web.config` provides helpful information when testing

Still a Black Box

- Even with all of the changes to create this .NET Framework, the resulting applications can still be tested as a Black Box
- Don't feel that you must suddenly go get programming certifications to be able to do an effective job of testing .NET apps
- Simply be aware that you have tools available to you to help you navigate the maze

.NET Tools for Testing

There are a number of tools available to programmers for debugging in .NET, and some of them are useful to test engineers

- Microsoft Application Center Test
- `Web.config` (using `<trace>`)

Application Center Test (ACT)

- Designed to stress Web servers and analyze performance and scalability problems with Web applications, including ASP and components they use
- Simulates a large group of users by opening multiple connections to the server and rapidly sending HTTP requests
- Supports several different authentication schemes and the SSL protocol, making it ideal for testing personalized and secure sites

ACT (Continued)

- Although load testing is its main purpose, a programming component can also be useful for functional testing
- Compatible with all Web servers and Web apps that adhere to the HTTP protocol

`Web.config`

- `Web.config` is a file found at the root level of your web application
- This files gives developers easy access to application-wide settings
- The file is in XML format so it's easy to read and modify
- The setting we're interested in is `<trace>`

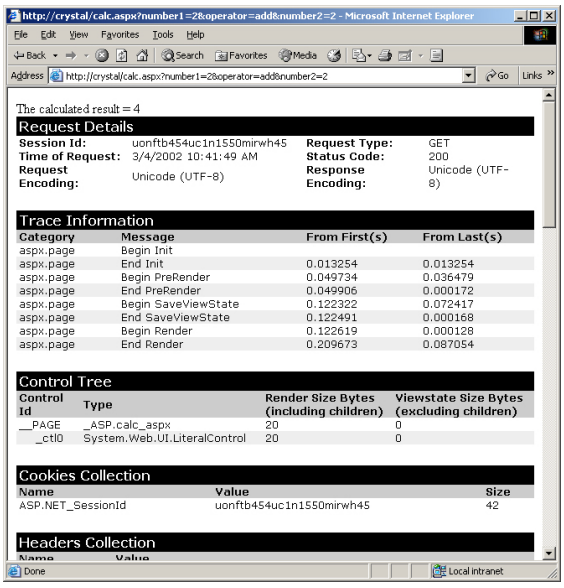
<trace>

- To use, `web.config` file should have:

```
<configuration>
  <system.web>
    <trace enabled="true"
     pageOutput="true"
     requestLimit="15"
     traceMode="SortByCategory" />
  </system.web>
</configuration>
```

Output of <trace>

- Request Details
- Trace Information
- Control Tree
- Cookies
- Headers
- Query String
- Server Variables



Other Tools

- ANTS – Load Testing
  "Advanced .NET Testing System"
  (Red-Gate.com)

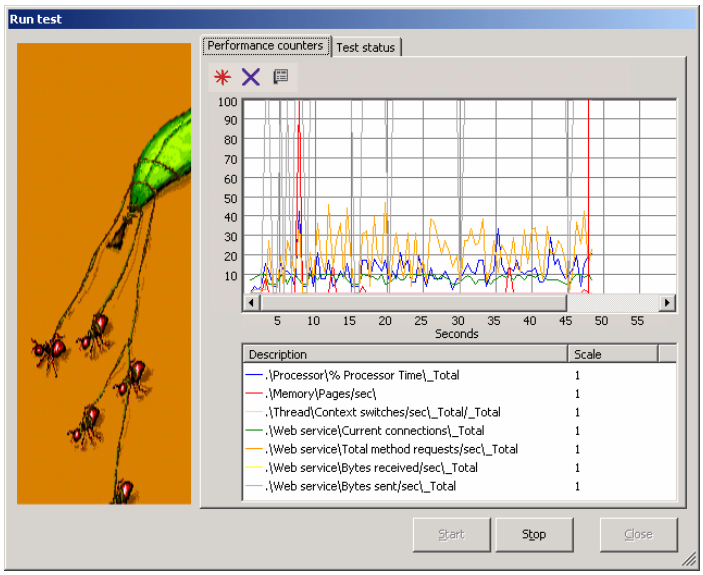
- PushToTest
  Free Open-Source Load Testing

PushToTest

- Rational RobotJ – Functional Testing

Rational
the e-development company

ANTS



Other Sources of Information

- My materials & articles:
  - "Xtend Information" newsletter (.NET related articles, links & news) www.xtenddev.com/newsletters/
  - "Automating Testing of ASP.NET Applications" white paper: www.xtenddev.com/qw2002/

Other Sources (cont'd)

- Books:
  - "Introducing Microsoft .NET" ISBN 073561377X, Microsoft Press
  - "Database Programming with VB.NET" ISBN 1-893115-29-1, Apress (Carsten Thomsen)
  - "VB.NET Language in a Nutshell" ISBN 0-596-00092-8, O'Reilly
  - "Programming VB.NET: A Guide for Experienced Programmers" ISBN 1-893115-99-2, Apress (Gary Cornell & Jonathan Morrison)

Other Sources (cont'd)

Web sites:

www.gotdotnet.com
www.dotnetjunkies.com
www.asp.net
www.AutomationJunkies.com

Summary

- Brief intro to .NET Framework
- Identified potential bug causes
  - Migrating from VB6 to .NET
  - Hand-holding (browser-specific code)
- Called out benefits of .NET
- Looked at an ASP.NET example
- Testing an .NET Deployed App
  - Black box testing still key
  - Application Center Test (ACT)
  - Web.config's <trace> tag & output
- Sources for learning more about .NET

Questions?

If we run out of time, contact me:

tom@xtenddevelopment.com

Automating Testing of ASP.NET Applications (an introduction)

Tom Arnold, tom@xtenddev.com
Presented at Quality Week 2002
Thursday, September 5th, 2002

Introduction

Microsoft has created a new environment that promises to ease the process of software development. As test engineers, it is up to us to figure out how the new .NET Framework applies to us, and our efforts, in testing the resulting ASP.NET applications.

Because of the enormity of the .NET Framework this paper moves from the large, high-level view of .NET down to specific examples in ASP.NET. That is, *WebForms* are explored instead of *WinForms*, two pieces of the puzzle that will become clearer as you read on.

This paper expands on my September 5th presentation at Quality Week 2002 conference. For more information about Quality Week visit www.qualityweek.com.

Topics

In this paper I will introduce you to a high-level view of what Microsoft .NET is, issues – such as migrating to the .NET solution – that could result in bugs, inherent challenges for software test engineers, approaches to testing deployed projects, and where to find more information to continue to learn about testing .NET applications.

Author/Speaker Background

My background is in software development and automated testing. I started my professional career in the software industry as a test engineer in the Seattle, Washington, area in 1990. Since that time I've continued to be involved in software testing (focusing mostly on test automation), development (C, C++, VB, and most recently Java), and managing software development projects.

I started using Microsoft .NET in August 2001 and have found it to bring some very exciting things to the table for developers. I was happy to see that Microsoft kept software test engineers in mind as they created this new solution, as you'll see.

Why .NET

Why Microsoft .NET? Sun Microsystems is one reason. Sun has been building on its Java solutions since 1995 when Java was first released, and they've been running hard for these past 7 years. Their solution is J2EE (Java 2 Enterprise Edition) that allows the Java language to work within multiple operating systems as well as with many databases. Sun has also come out with J2SE (Java 2 Standard Edition) and J2ME (Java 2 Micro Edition). These solutions allow Java users to work in simple web environments (J2SE), Enterprise (J2EE), and with handheld devices (J2ME). Very exciting, and all bundled up in a very nice package with many developers excited about the prospects.

Enter Microsoft, a company that has long dominated the software development industry, suddenly seeing some of its development supporters casting their gazes upon Sun Microsystems' solutions to Internet applications and multiple platform support. Sun, with its popular Java programming language that is secure and easy to use compared to C++.

Microsoft took the next logical step in the evolution of their development approaches and brought together many of its development solutions to be placed into a bucket named ".NET." Does that mean .NET is entirely new? No. Microsoft has taken all of its existing functionality, added in some additional bits (albeit some rather large and important bits), and pulled it into a solution that will compete (very well) with Sun Microsystems.

This is a good thing, why? Because Microsoft .NET brings a new focus on how to approach Windows and Internet development, an approach that not only opens up the architecture to allow a host of new languages to be supported in the .NET development environment, but operating systems as well. The .NET Framework is setting the scene to allow applications to be developed and deployed in many environments, and additional support for such deployment created by third party vendors.

.NET Framework

This framework, at first glance, seems more like a huge puzzle or maze. Just when we're getting things figured out, yet another enigma in the software industry presents itself, this time in the form of Microsoft .NET.

Fear not, .NET is not so overwhelming after all. Remember that it's an encapsulation of a number of pre-existing Microsoft technologies with a few new ones thrown in for good measure. This, as well as a common thread – or framework – that pulls it all together, is what makes up Microsoft .NET.

Common Language Specification

The top layer shown in **Figure 1** (on the following page) illustrates the default languages already supported by .NET: Visual Basic, C++, JavaScript (known at Microsoft as "JScript"), and Microsoft's new C# (pronounced, "C Sharp").

Microsoft Visual Basic .NET

Visual Basic now offers full object-oriented language features, including implementation inheritance. It also allows developers to create highly scalable code with explicit free threading and highly maintainable code with the addition of modernized language constructs like structured exception handling.

Microsoft Visual C# .NET

Microsoft also created its own Java-like language called C#. C# is very much like Java in that it handles all garbage collection, provides security, and is fairly easy to use compared to C++. It was built from the ground up with the .NET Framework in mind and is a modern, object-oriented, type-safe language. C# "is designed to bring rapid development to the C++ programmer without sacrificing the power and control that have been a hallmark of C and C++."

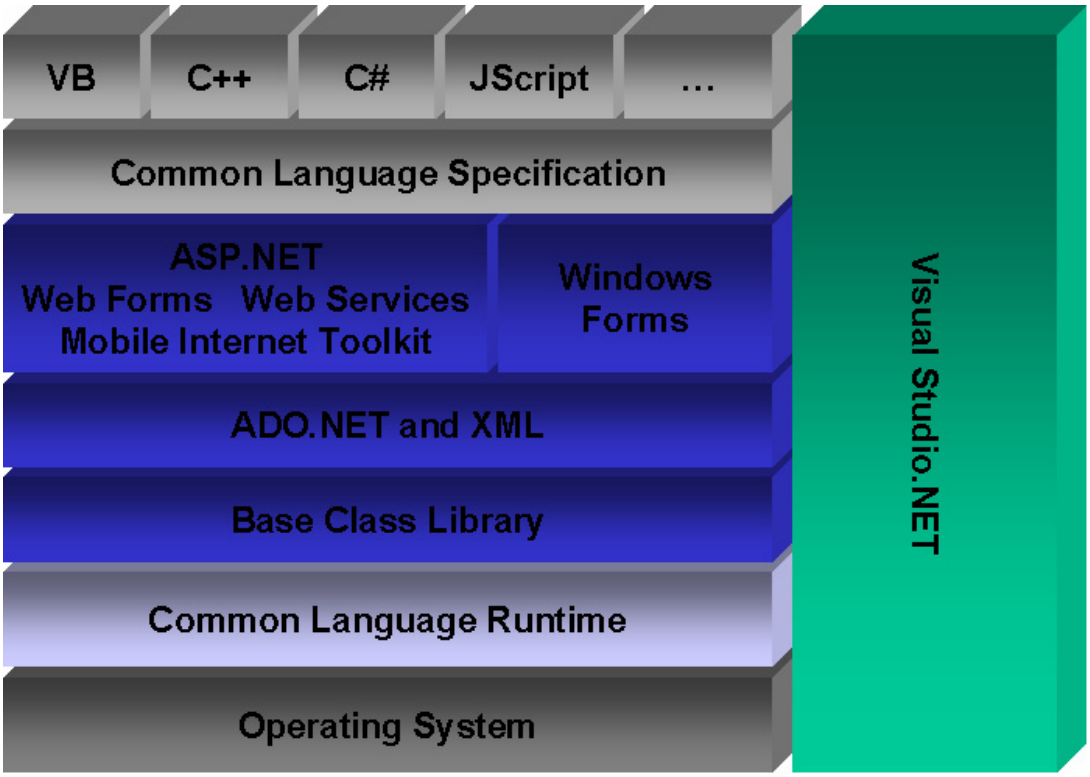


Figure 1 - *Microsoft .NET Framework*

Microsoft Visual C++ .NET

Traditional *unmanaged* (outside of .NET) C++ and new *managed* (within .NET's structure) C++ code can be mixed freely within the same application. Existing components can be wrapped as .NET components by using the managed extensions. Most importantly, providing support for C++ preserves investment in existing code while integrating with the .NET Framework.

JavaScript / JScript

JavaScript is the common language used for web development. This is because older versions of Netscape Navigator supported only JavaScript, while Microsoft Internet Explorer supported JavaScript and VBScript. If a website is being created with client-side scripting, maximum compatibility can be maintained by using JavaScript. It's no wonder the popular language is supported in the .NET Framework.

Other Languages

This is not a complete list of the languages supported by .NET, however, not by a long shot. By creating and publishing a *Common Language Specification* (CLS), third party vendors can take new or existing languages and fit them into the .NET puzzle. This means COBOL, FORTRAN, Java, and many other languages can now be used to program Windows (WinForms) and Web (ASP.NET) applications and services. Therefore the list will never be complete as vendors continue to add to the roster of supported languages. And, if these languages follow the rules laid down by the CLS, they can then access the libraries provided with .NET, as shown in Figure 1's middle layer.

The Middle Layer

The middle layer looks nice and neat the way it's divided up in Figure 1, but really it all exists in a grouping of over 90 collections, each one referred to as a *namespace*. All languages that follow the CLS can work with these namespaces and thereby use ADO.NET (database support) and the other libraries necessary to create Windows and Web applications. It is also possible to create namespaces outside of what Microsoft has already provided.

ASP.NET

The rewrite of ASP (Active Server Pages) – called ASPX for no other reason than they were focused more on creating something cool than trying to figure out a catchy name – became known as ASP+, and later renamed to ASP.NET. It addressed many shortcomings of ASP. While ASP pages (.asp) are still supported by the ASP.NET server, they must use the new file extension (.aspx). ASP.NET focuses more on separating the HTML from the code. By using a *code behind* approach, the HTML is kept in an .aspx file, and the new general practice is to place the blocks of code into separate files to be included by the .aspx files (such as pagename.aspx.vb for VB.NET code, or pagename.aspx.cs for C# code, for example).

In addition, ASP.NET works closely with Microsoft NT/Win2K servers' security settings. It works within those policies to make changes to permissions, rights, and more, much easier. It also allows a user session to be shared over multiple servers so that load balancing is easier and more effective. And, should one of the servers go down, the user remains blissfully unaware and is able to carry out his transaction because his session does not live on any one server.

And last, ASP.NET compiles its pages *just in time* so that execution is much faster than ASP. The first time an .aspx page is accessed after being saved to the web server, the page is compiled into a pseudo code form. This doesn't increase actual execution speed of the code (that is, it's not compiled into machine language), but it does allow the ASP.NET Server to avoid the compilation step for each and every user before spitting out the generated HTML. Execution speed seems faster to the end-user since the compilation step occurs only when the page is modified.

ADO.NET

Open Data Base Connectivity (ODBC) is an old tried and true standard for accessing data. It was designed to provide a common set of routines to programmers. These routines remained unchanged regardless of the type of database being accessed (e.g. Access, SQL, Oracle). The next step in the evolution of the anonymous data store was OLE-DB that not only supports ODBC, as well as its own methods for working with Access/SQL/Oracle, it also works with Exchange, Excel, and other applications (no, they don't have to be Microsoft applications, just support OLE-DB).

ADO.NET (ActiveX Data Objects) is a friendly interface to OLE-DB. It provides a set of objects to the languages working within the guidelines of the CLS. It's yet another level of abstraction to keep things simple and common to the programmer, and allows ADO.NET to deal with the bit twiddling behind the scenes.

Base Class Libraries

ASP.NET and ADO.NET are part of the base class libraries provided in the .NET framework. These namespaces are what provide common objects and methods used by all CLS-compliant languages in the .NET framework.

Common Language Runtime

The final layer is the Common Language Runtime, or CLR. This piece sits on top of the operating system and executes the compiled code. So here's where it gets really cool. Because all languages that want to work with .NET must conform to the CLS, and these *managed* languages all use the base class libraries (including ASP.NET and ADO.NET), everything can be compiled down to a common set of metadata or an *Intermediate Language*. This is the most basic level of data and at this point it doesn't matter what language the instructions were written in. VB, C#, C++, Java, COBOL, whatever, it all looks the same at this intermediate language level.

This is a wonderful thing because this means that all languages can (and do) share the same class definitions and objects defined further up the ladder. A namespace can be created and used by all of the languages because they all eventually end up at this very basic level so that the runtime engine can interpret them.

It gets better. Because the programming languages use the objects created via the .NET namespaces for file manipulation, and therefore the CLR separates the operating system from those languages, different versions of the CLR can be written for the Macintosh, Linux, and so on. When OS-specific versions of the CLR are rolled out, it will be possible to write your program once and have it deployed on multiple operating systems without any extra work. (In theory). Sound familiar? (Hint: Sun Microsystems' goal with Java).

Challenges

As with any new approach, there is always a price of entry, whether it's the learning curve or bringing your now-Legacy-code along into the new system. In the case of the .NET Framework there are two obvious challenges from the start: Migration of old code into the new environment and understanding how much control .NET wants to exercise in an attempt to hold our hands and make things easier.

Migration Issues

The Common Language Specification requires all languages to follow specific guidelines to be allowed to participate in the .NET Framework. This applied to creating a .NET version of Visual Basic as well. The result is changes that are easy to accept when creating new applications, but can be more involved when migrating code.

In the case of Visual Basic (VB.NET), for example, migration issues exist in regards to the introduction and handling of new data types, renaming/moving of functions, and the discontinuation of keywords.

What this has to do with software testing is that the code base that once worked "good enough" to share with the user-base gets touched, and in an invasive way. Opening a code base after Testing has blessed it is already a tricky business, but to modify code so that it can work within the new framework – replacing type declarations, using new functions, and more – will require a full test pass to verify nothing breaks in the process. (Software test automation that's already in place will come in handy).

New Types & Keywords

With the creation of a more generalized approach allowing many languages to work together comes the need to tighten and redefine past approaches. **Table 1** reflects just a few of the differences between VB6 and VB.NET.

Visual Basic 6	**VB.NET**
Fixed length strings were declared as: Dim Name as String * 30	Fixed-length strings are not allowed
An Integer type is 16-bits	A Short type is 16-bits
A Long type is 32-bits	An Integer type is 32-bits
No support for a 64-bits integer type	A Long type is 64-bits
Any data can be set to a Variant variable	The Variant type is unsupported
"Dim X, Y as Long" results in X declared as a Variant and Y as a Long	"Dim X, Y as Long" results in X and Y declared as a Long
Keyword "Empty" indicates an un-initialized Variant variable. "Null" indicates that a variable contains no valid data.	"Null" and "Empty" have been replaced by the keyword "Nothing"

Table 1 - *Visual Basic 6 & VB.NET differences (Types and Keywords)*

Moved Functions

To follow the new object-oriented approach that .NET utilizes through its libraries – known as *namespaces* – functions have been relocated. Let's take Visual Basic's Math functions for example. They have all been moved into the `System.Math` group, so now:

```
X = Cos(Y)
```

Becomes:

```
X = System.Math.Cos(Y)
```

A tool does exist for migrating Visual Basic 6 projects over to VB.NET. Before you breathe a sigh of relief, however, know that most people who have used this tool say that it is not that helpful on large conversion efforts. If you have a simple application to convert, it will provide you with some assistance. However, the changes between version 6 and VB.NET are great enough to make a re-write of the application worth considering, depending on the type of application and its features. For more details about upgrading your Visual Basic 6 applications to VB.NET, I recommend the following MSDN article as a very good starting point:

http://msdn.microsoft.com/vbasic/techinfo/articles/upgrade/vbupgrade.asp

No Direct Mapping of ASP.NET to HTML Controls

One of the exciting features of ASP.NET is its ability to spare the developer from the hassles of tracking which browser a visitor is using during a web session and providing different HTML based on the visitor's browser's capabilities. ASP.NET will issue the HTML it thinks best suits the client. This is a great concept, but in practice the results are not yet clearly known. It remains to be seen how well this type of handholding will work and if it results in workarounds being that much more challenging.

An example is the text box placed on a web page. In ASP.NET it looks like this:

```
<asp:TextBox id="SearchTextBox" runat="server"
MaxLength="25"></asp:textbox>
```

It generates the following HTML:

```
<input name="ModuleSearch:SearchTextBox" type="text"
maxlength="25" id="ModuleSearch_SearchTextBox" />
```

However, depending on the browser, it could also generate this HTML:

```
<textarea name="ModuleSearch:SearchTextBox" rows=1
maxlength="25" id="ModuleSearch_SearchTextBox"><textarea>
```

You will note that these are two different control types, yet they can resemble each other depending on the browser being used. The theory is that ASP.NET knows best, and this remains to be seen. To be sure, as feedback comes in ASP.NET will become much more robust as Microsoft builds on its goal of helping testers and developers alike worry less about browser compatibility.

Testing in .NET

Now that we have a high-level view of what Microsoft .NET is, some of the challenges that programmers face, and some of the issues testers need to be aware of, let's look at a sample ASP.NET application and some of the things we should consider in its testing.

The web application we'll use in this example comes with Microsoft Visual Studio .NET and is called "Duwamish 7.0." Its home page is shown in **Figure 2** on the following page. This application is for a fictional on-line bookseller and demonstrates the concepts of modular development, working with controls, searching, an e-commerce shopping cart, and working with a Microsoft SQL Server database. (Microsoft was kind enough to fill the database for us with sample data).

Black Box Testing Remains Crucial

Although .NET makes technical testing more accessible to software test engineers, the non-programming aspects for software testing remains key. Usability issues remain important, of course, as does verifying that an application behaves, as an end-user would expect. There is nothing new to be introduced to software test engineers in the realm of black box testing in regards to an ASP.NET web application. Browser compatibility testing remains an important part of the process, especially since ASP.NET generates HTML unique to a user's operating system, browser and browser version.

In regards to taking more technical approaches, there are many opportunities available to test engineers, which the remainder of this document will explore.

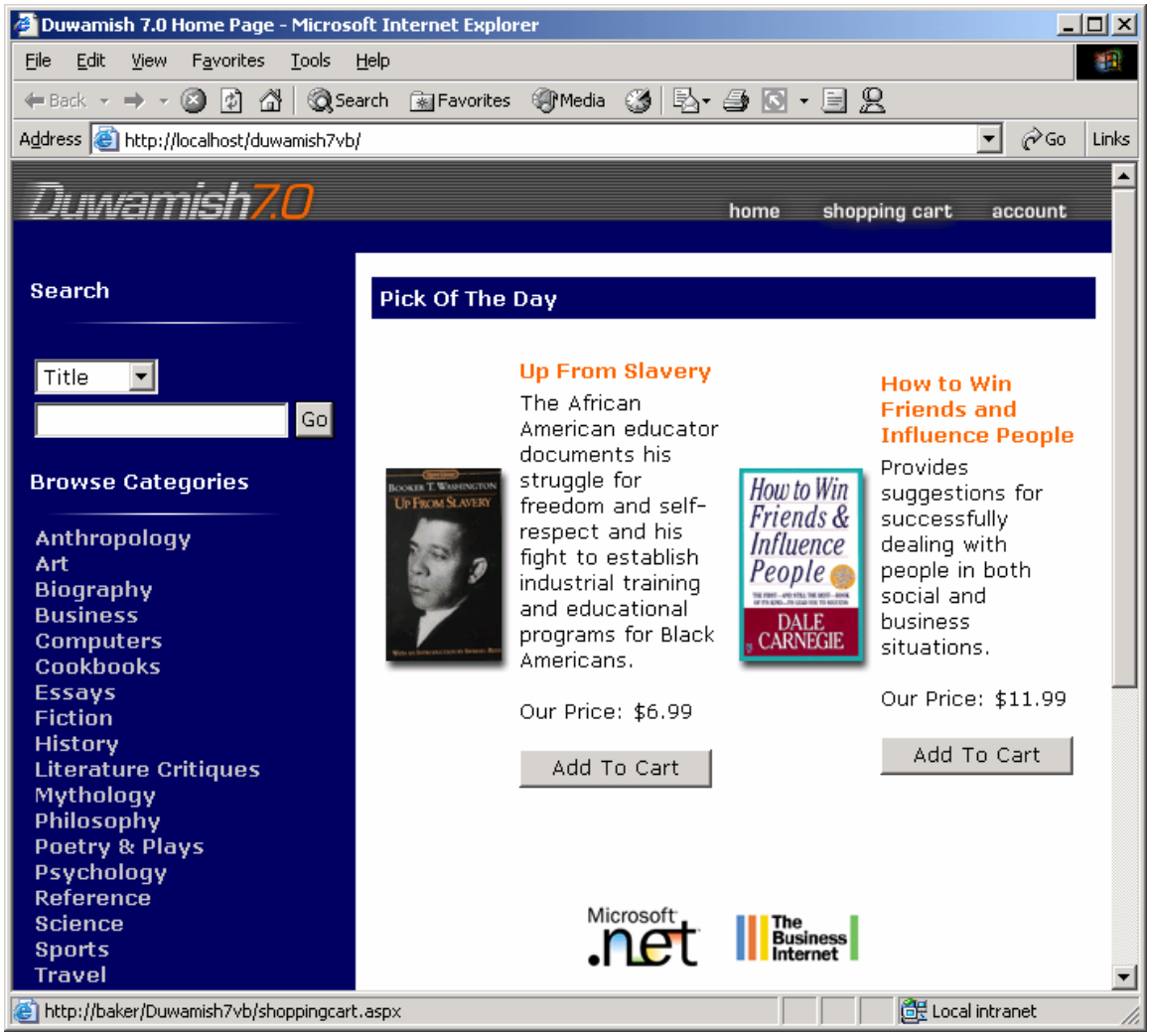


Figure 2 *– Sample application that invokes many of ASP.NET's features.*

Technical Testing

Microsoft has provided the tools necessary to test and debug ASP.NET applications that are not only useful for developers but testers as well. Some of these tools move into the realm of gray and white box testing, however, which some testing organizations are against. The concern of these organizations is that it steps too far away from what the end-user will experience. It also requires a more technical (and hence, typically more costly) test engineer. I am of the opinion that while black box testing is extremely important, the more technical a tester can be in their efforts, the more effective they can be in diagnosing and tracking down issues and bugs and communicating those problems to their programmer counterparts.

In this section we will introduce the `<trace>` setting that can be added to an application's `web.config` file and how it is used in debugging. We will also look at some of the tools available for automating the testing on ASP.NET applications, both in functionality and load/stress testing.

Web.config

ASP.NET has what Microsoft refers to as a *configuration system*. This system is an extensible infrastructure that enables all ASP.NET applications' configuration settings to be defined when an application is first deployed, and modified any time thereafter. The root configuration file is `machine.config` and configures the entire web server. Another file – `web.config` – can appear in multiple directories throughout the ASP.NET web application server. The `web.config` file affects the directory it is in, as well as its directory's sub-directories. In addition, a `web.config` file in a lower child directory can override or modify those settings of its parent.

Each `web.config` file contains a nested hierarchy of XML tags and sub-tags. These tags have attributes that specify the configuration settings. There are over 60 elements that make up the configuration schema that controls how ASP.NET web applications behave. You can even add your own, if you like. **Listing 1** shows an example of a `web.config` file.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
 <configSections>
  <section name="ApplicationConfiguration"
   type="Duwamish7.SystemFramework.ApplicationConfiguration,
   Duwamish7.SystemFramework" />
  <section name="DuwamishConfiguration"
   type="Duwamish7.Common.DuwamishConfiguration,
   Duwamish7.Common" />
  <section name="SourceViewer"
   type="System.Configuration.NameValueSectionHandler, System,
   Version=1.0.3300.0, Culture=neutral,
   PublicKeyToken=b77a5c561934e089" />
 </configSections>
 <system.web>
  <customErrors defaultRedirect="errorpage.aspx" mode="On" />
  <compilation debug="true" />
  <sessionState cookieless="false" timeout="20" mode="InProc"
   stateConnectionString="tcpip=127.0.0.1:42424"
   sqlConnectionString="data source=127.0.0.1;
   user id=sa;password=" />
  <globalization responseEncoding="utf-8"
   requestEncoding="utf-8" />
  <!-- security -->
  <authentication mode="Forms">
    <forms name=".ADUAUTH" loginUrl="secure\logon.aspx"
     protection="All">
    </forms>
  </authentication>
  <authorization>
    <allow users="*" />
  </authorization>
 </system.web>
</configuration>
```

***Listing 1**: Example of a web.config file.*

First and foremost, don't sweat it. This isn't as scary as it looks, and not only that, we only want to work with a very small section of the file. Specifically, we will insert a `<trace>` tag directly under the `<system.web>` tag and place it into the directory containing the pages we want to work with. In this case, to keep it simple, and to avoid multiple copies of `web.config` that could go forgotten, it will be the root version of `web.config` that we modify. The following line is added:

```
<trace enabled="true" pageOutput="true" requestLimit="15"
 traceMode="SortByCategory" />
```

Inserting this simple XML tag and its properties has dramatic effects on your ASP.NET web application. In standard ASP pages it was necessary to insert statements to print out the current status of variables or track what branches of code were executed. This was done by strategically placing `Response.Write()` functions throughout the .asp files. The problem was that these statements had to be removed later (and not forgotten). ASP.NET's solution to this is the *trace* functionality. With the above line added to `web.config` in the Duwamish 7.0 sample application, navigating to its home page (Figure 1) tacks on the additional information shown in **Figure 2**.

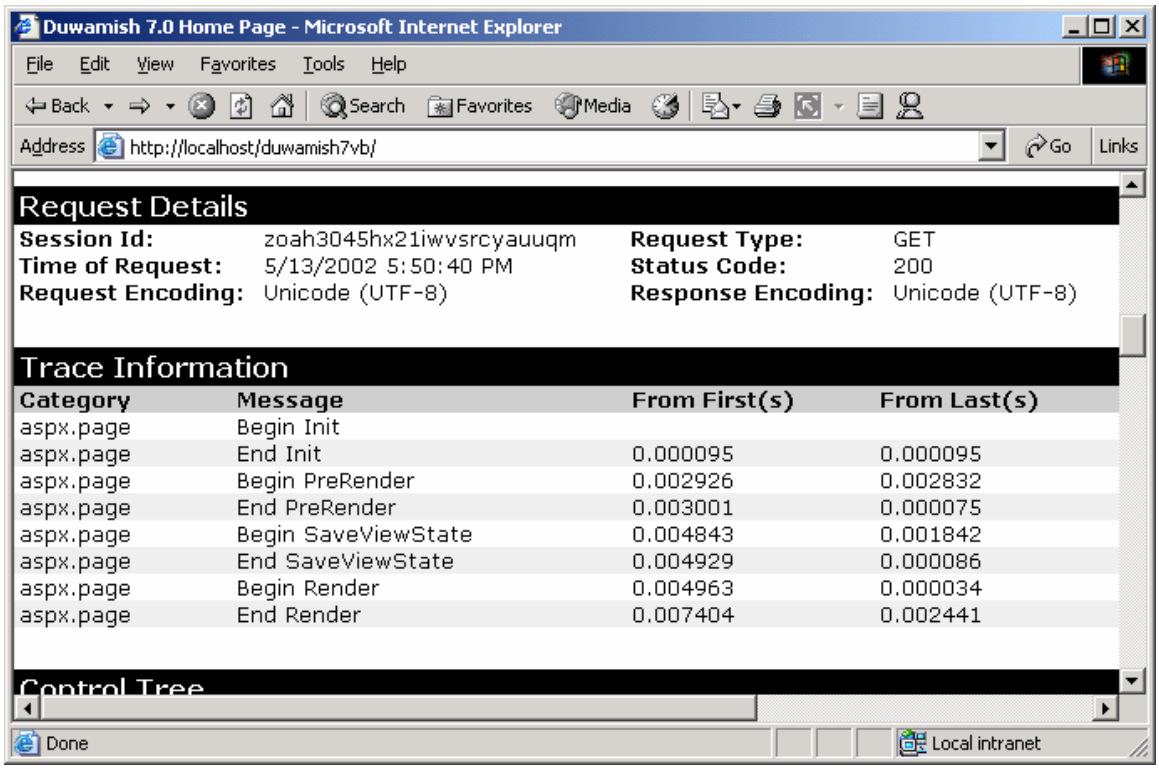


Figure 2 *– The <trace> tag is added to web.config in the Duwamish 7.0 web directory.*

As you can see, ASP.NET's new tracing functionality allows us to view verbose information about an application with minimal intrusiveness. In the past it was necessary to sprinkle `Response.Write()` routines throughout the code. Now, only one file needs to be modified lowering the likelihood this setting will be accidentally left enabled. In addition, `Trace.Write()` routines may be used throughout the code where `Response.Write()` methods might have been used for debugging purposes in

the past. Because `Trace.Write()` is only invoked when the `<trace>` tag is in place, those statements may remain in place without ill effect.

The output provided when using `<trace>` is provided in six sections:

- **Request Details:** This provides such basic information as the visitor's unique session ID, the time & date the request came into the server, HTTP request type and its status code.
- **Trace Information:** In addition to ASP.NET-generated information about the execution of the application, this is where `Trace.Write()` values are printed. The two parameters taken by `Trace.Write()` are displayed here as Category and Message values.
- **Control Tree:** This section provides information about the controls used within the page. ID, type, render size, and view state information is provided.
- **Cookies Collection:** Any cookies sent by the client in its request to the server are displayed.
- **Headers Collection:** HTTP header values sent by the client to the server are displayed here in a simple Name/Value pairing.
- **Querystring Collection:** This shows up only when the `GET` method is used to submit a form. It shows the variables and values sent with the request. (These same values can be found in their raw form in the `QUERY_STRING` entry of the Server variables section).
- **Server variables:** The Name and Value of server side variables are displayed in this table. This section includes a lot of the same information listed in the other sections, just not as nicely formatted.

This information is helpful to test engineers in a number of ways. The *Request Details* section, for example, is important in showing how the server responded back to the client. Specifically, showing a status code of 200 means that no errors were encountered. An error of 400 is a *Bad Request*, 401 is *Required Authorization*, 403 is *Forbidden Directory*, 404 is *Page Not Found*, and 500 is *Internal Server Error* (nothing new). Being able to print a page that shows the request string that resulted in an Internal Server Error can be very helpful to development. The *Trace Information* section can provide helpful information about which branches of code were executed if the `Trace.Write()` method was used by the programmers. The *Headers Collection* shows the type of browser and operating system used to access the page, which can be helpful in verifying the browsers your group identified as important are actually used in testing. The *QueryString Collection* makes it easy to see what values a form sent to the page currently being displayed.

Microsoft ACT 1.0

Purpose of Microsoft Application Center Test 1.0 – or ACT – is to stress test web servers and analyze performance and scalability problems with web applications. This includes ASP.NET applications and their components. This type of testing is accomplished by opening multiple connections to the server and rapidly sending HTTP requests, thereby simulating a large group of users. Although high-load stress testing over long periods of time is ACT's main purpose, it can also be used for functionality testing. Lastly, Application Center Test will work with any web server or web application that adheres to the HTTP protocol.

Putting Application Center Test into use on a testing project allows you to see how your web server reacts when several hundred users access your application at the same time. This simulates peak periods and not only provides performance and scalability information, but also tests databases in regards to such issues as concurrency, transactions, number of users supported, locks, pooling, and so on.

This tool comes with Visual Studio .NET Enterprise Developer and can be used within the Visual Studio .NET's Integrated Development Environment (IDE), but more options are available when the stand-alone ACT program is used.

```
Sub SendRequest1()
    Dim oConnection, oRequest, oResponse, oHeaders, strStatusCode
    If fEnableDelays = True then Test.Sleep (0)
    Set oConnection = Test.CreateConnection("localhost", 80, false)
    If (oConnection is Nothing) Then
        Test.Trace "Error: Unable to create connection to localhost"
    Else
        Set oRequest = Test.CreateRequest
        oRequest.Path = "/duwamish7vb"
        oRequest.Verb = "GET"
        oRequest.HTTPVersion = "HTTP/1.0"
        set oHeaders = oRequest.Headers
        oHeaders.RemoveAll
        oHeaders.Add "Accept", "image/gif, image/x-xbitmap, " + _
             "image/jpeg, image/pjpeg, application/msword, " + _
             "application/vnd.ms-powerpoint, application/vnd.ms-excel, */*"
        oHeaders.Add "Accept-Language", "en-us"
        oHeaders.Add "User-Agent", "Mozilla/4.0 (compatible; " + _
             "MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705)"
        'oHeaders.Add "Host", "localhost"
        oHeaders.Add "Host", "(automatic)"
        oHeaders.Add "Cookie", "(automatic)"
        Set oResponse = oConnection.Send(oRequest)
        If (oResponse is Nothing) Then
            Test.Trace "Error: Failed to receive response for URL to " + _
             "/duwamish7vb"
        Else
            strStatusCode = oResponse.ResultCode
        End If
        oConnection.Close
    End If
End Sub
```

Listing 2 *– Example of a single request generated by ACT 1.0's recorder.*

Listing 2 shows an example of a subroutine generated by ACT's recorder. When a script is generated, requests are broken up into individual subroutines that are then called one-by-one by a `Main()` subroutine.

Just like the rest of .NET, this application relies on an object-oriented approach and provides a number of objects that you can use. In Listing 2, note that we're looking at the `SendRequest1()` subroutine. This is the first routine generated when the webapp-under-test was navigated to. The line of interest is the one that says, `oRequest.Path = "/duwamish7vb"`. This is the root of the web directory or site we've selected for testing. The other pieces specify the type of request coming in (in this case "GET" instead of "POST" or "HEAD), the MIME types your browser declares

that it will accept and understand, your browser information, and so on. (You'll be happy to know that cookie handling is built in to ACT 1.0). After completing the `Request` information, it is dispatched to the web site with a call to the `Connection.Send()` method (`oConnection` is a Connection object, which was created when a call was made to `Test.CreateConnection` at the front part of the subroutine). The `oConnection.Send(oRequest)` call sends off the Request information that was filled out in the middle of the script. (Lots of stuff just to generate a single request, eh?)

Subsequent `SendRequest`*N*`()` routines are created (where *N* equals the next number in the sequence) based on what the displayed page relies upon. This includes a request for the cascading style sheet (/duwamish7vb/css/duwamish.css), images (bannerlogo.gif, bannerhome.gif, bannercart.gif, banneraccount.gif, line.gif, etc.), and the page itself (Default.aspx).

Finally, down around the 15th request, in the `SendRequest15()` routine, we get to the request that was generated when typing in a search string and clicking the submit button. The resulting request path looks something like this:

```
oRequest.Path = "/Duwamish7vb/searchresults.aspx" + _
"?type=0&fullType=Title&text=how+to+win+friends"
```

This is a direct call to the .aspx (ASP.NET) file using the `GET` method (the `GET` method causes the submitted form's values to be part of the URL, as opposed to `POST` which embeds the variables and their paired values into the HTTP header and goes unseen by the user). This call goes through the whole process previously described, sending off the above request to `searchresults.aspx` with the query string shown above, hoping for a reply in HTML by the server. The CSS file is downloaded as are the GIFs and JPEGs. Whew! Lot's of traffic going on just for a single request!

When all is said and done, 34 separate requests are generated, and all we did was:

1. Navigate to http://localhost/duwamish7vb
2. Type "how to win friends" into the search box
3. Clicked the "Go" button to submit the form
4. Clicked on the link of the book found by the query

The main routine that fires off each of these requests is simple enough, and shown (with some abbreviation) in **Listing 3**.

```
Sub Main()
    call SendRequest1()
    call SendRequest2()
    ': (3-33 removed for brevity)
    call SendRequest34()
End Sub
Main
```

Listing 3 *– Each Request is sent in turn by calls to their corresponding subroutines.*

Running the script is as simple as clicking a *Play* button found on the toolbar. As the test executes, the status can be viewed and looks similar to **Figure 3**. This status box communicates the current state (for example, "The test is now running"), time elapsed, time remaining, average requests per second, and total requests made so far. It also lists the number of errors currently encountered allowing you to decide if the test needs to be aborted or allowed to continue.

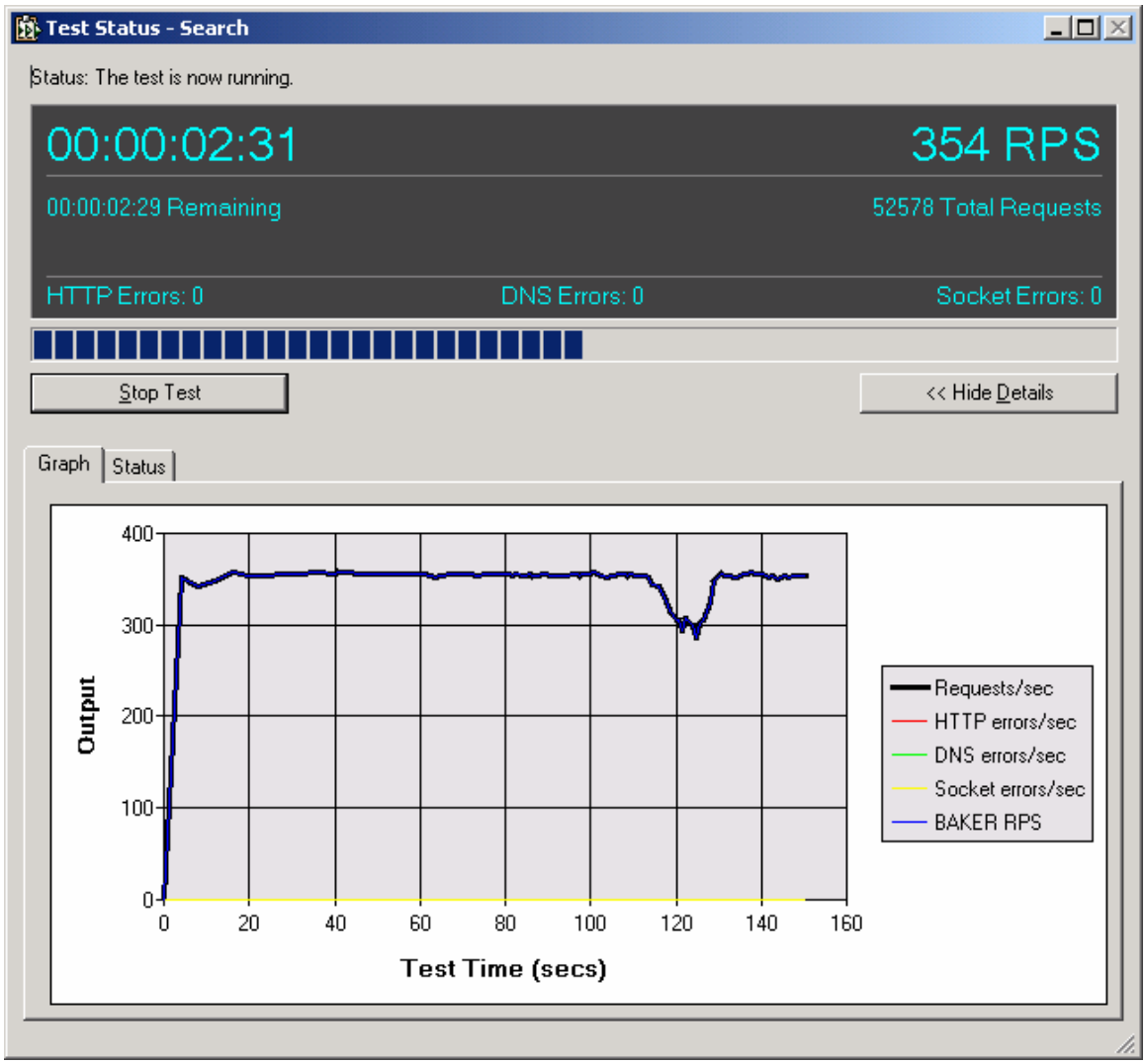


Figure 3 *– Execution of the ACT script halfway through its test run.*

At approximately 2 minutes and 5 seconds into the test run, I jumped to my browser and tried 3 searches, resulting in the dip in Requests Per Second (RPS) generated by the load test, shown in Figure 3. (Hey, the straight line was looking boring). This allowed me to see how peppy the site was even under load.

When a script completes, its results can be viewed by clicking on the *Results* object in the test project. The results are displayed in a list based on the name of the script being executed, and the time/date of its execution. Results show the total number of requests generated during the run (in this case, it was 5 minutes and generated 104,227 requests, an average of 347.42 requests per second). It also gives the test

engineer an idea of how responsive the system was for those requests. Responsiveness is measured by tracking the "Average Time to First Byte" (how long until the server started sending back a response) and "Average Time to Last Byte" (total time to send the requested item). Errors are also listed (HTTP, DNS and Socket), as are network statistics.

Considering my test example ran on the same computer as the web server, bandwidth was pretty good (1,227,864.16 bytes/second) on my 1.2ghz laptop. And last, response codes are tracked allowing us to understand the quality of the requests and responses. The return result of "200" communicates the request was received and a full response was returned. Using multiple computers on the network is one way to raise the number of requests per second experienced by the server. Getting a faster test machine with a big fat pipe direct to the server is another.

There is much more to discover about this program, including tracking server performance, test machine performance, and more. I've used it more in a recorder/playback capacity, but the language is based on VB Script and isn't difficult to master. There are only about 5 objects (with many methods each) that comprise the model: `Test`, `Request`, `Connection`, `User` and the `Response` object. Real-world use of this tool is likely to be: Record a few actions to generate a script, move the common bits into separate routines, and use constants based on the type of machine being used and browser being emulated. There is definitely room for structured programming to be put into practice to make it less bulky and more responsive to the evolution of a web site.

All in all, it's a great tool that provides a lot of capability, and it's "free" (as long as you have the full-blown version of Visual Studio .NET).

Other Considerations

There are other tools out on the market worth considering. These include *Red-Gate*'s new ANTS ("Advanced .NET Testing System") product, *Rational Software*'s new Java-based tool for testing Java and HTML applications, and *PushToTest*'s free open-source load & monitoring tools. These are new or less known, so I won't bother to list the other tool vendors and tools of which you are probably already aware.

- **ANTS** appears to be very similar to Microsoft Application Center Test 1.0 in what it provides. It can be used to generate HTTP requests as well as test Web Services through HTTP requests enriched with the SOAP protocol. They have a 14-day free trial that's worth a look (only allows 10 simultaneous connections in the demo, however). At present they're pricing their tool at around $2,000+.
- **Rational Software**'s new Java-based testing tool is very intriguing. Look for my white paper about it on the AutomationJunkies.com site in July when it releases. By taking advantage of an object-oriented language like Java, this new tool provides flexibility by treating each item on a web page or Java applet as an object. This allows the object to be modified in the script later if it is modified on the website or in the applet. Broken scripts due to UI changes are fixed more quickly (a common automation nightmare). In addition, this new tool has a level of intelligence built in allowing it to weigh the likelihood that a modified control is still the control it wants to interact with, further improving maintainability.

- **PushToTest** has an automation tool whose UI is in Java – allowing it to work anywhere – and the scripting language is based on *Python*, a language that's easier to learn than C or C++. Using the Java framework allows test engineers to deploy their Python test scripts and generate loads against their server under test. Best of all, it's open source, so you have the code base and are free to modify its functionality and capabilities to your heart's content.

Summary

This is a lot of information to absorb, to be sure. We started by looking at the .NET Framework and getting a 30,000-foot view of what it is and some of the things it has to offer. To continue learning about .NET I suggest visiting such sites as **GotDotNet.com** and **DotNetJunkies.com**. These are two great sites for tutorials and other information about working in .NET. Unfortunately, they're mostly programmer-centric, so you should also check out StickyMinds.com and **AutomationJunkies.com**. Be sure to also visit **QAForums.com** for a great list of discussion groups.

The next thing we looked at were some of the challenges development teams face – concerns to programmers and testers alike – when moving into the .NET realm. These include the modification of variable types between VB6 and VB.NET, the placement of functions within the .NET object model, and how .NET does some handholding, which could prove to be problematic when it comes to tweaking how a page is displayed in different browsers. (This could be especially problematic to automation tools that rely on specific controls to be in place and don't tie themselves to a single browser for testing).

We then looked at an ASP.NET example and a tool that comes with Microsoft Visual Studio .NET (Application Center Test 1.0), as well as intrinsic support for ASP.NET deployed applications (the `<trace>` tag in `web.config`). Let's not forget the other new tools that are coming to market to help support .NET testing, as well as new tools that weren't necessarily targeting .NET, but can be used regardless.

This paper scratches the surface, but still provides you with a strong starting point with some of the options available to you when automating ASP.NET testing.

# # #

If you have questions or comments about this paper, please address them to tom@xtenddev.com. If you find your team is undertaking automation approaches, be sure to also visit this new test automation site: www.AutomationJunkies.com

QW2002 Paper 9T1

Mr. Ed Adams (VeriTest)

Virtual Test Management: Rapid Testing Over Multiple Time Zones

Key Points

- The challenges of managing multiple test sites
- 5 steps to successful virtual test management
- Mistakes to avoid/lessons

Presentation Abstract

With the ever-changing challenges of testing, here comes the latest one: managing multiple test locations. More and more companies are spreading testing organizations throughout the country and the world. Based on real-life experiences of the speakers, learn the mistakes to avoid and lessons learned in managing multiple sites. Discover how the Virtual Test Manager can manage a dispersed test organization without having to always be physically present.

About the Author

As VeriTest's Vice President of Marketing, Worlwide Testing Services, Ed is responsible for evangelizing VeriTest's capabilities in outsourced testing, certification, and consulting. He has over 10 years experience in the software industry in Product Management, Testing, Development, and Marketing, and is a frequent speaker on quality and test automation topics. Prior to joining VeriTest, Ed spent 4 years with Rational Software's automated testing team. He is a member of the Order of the Engineer, ASME, and several university advisory boards. He holds a B.S. in Mechanical Engineering, a B.A. in English Literature, and an MBA from The Carroll School of Management at Boston College.

VeriTest.

Virtual Test Management

Rapid Testing Over Multiple Time Zones

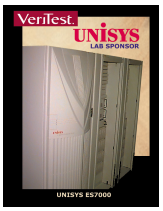
Ed Adams
Vice President, Worldwide Testing Services
VeriTest, a division of Lionbridge
Ed_Adams@lionbridge.com
www.lionbrige.com
www.veritest.com

Agenda

- Introduction
- Test Manager
- 7 Habits of Highly Effective Test Managers
- A Global Perspective & Challenges with Distributed Teams
- Lessons Learned
- Best Practices
- 5 Steps to Successful Virtual Test Management
- Conclusions & Recommended Reading

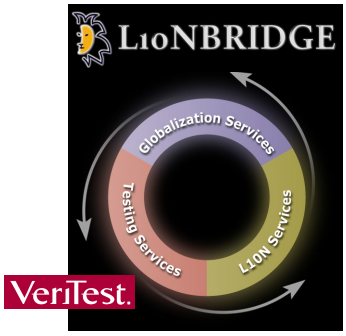
VeriTest: Introduction

- **Experience**
  - > First lab opened in 1987
- **Scalable Resources**
  - > 300+ career test professionals at ten labs worldwide
  - > Data center test beds with 'scale-up' 32-way servers plus 'scale-out' server farms
  - > Acquisition of Data Dimensions, Inc. completed in June, 2001
- **Leadership**
  - > Technology alliances with industry trend-setters Microsoft, Rational, Unisys
- **Global Footprint**
  - > Delivery capability in North America, Europe, Asia



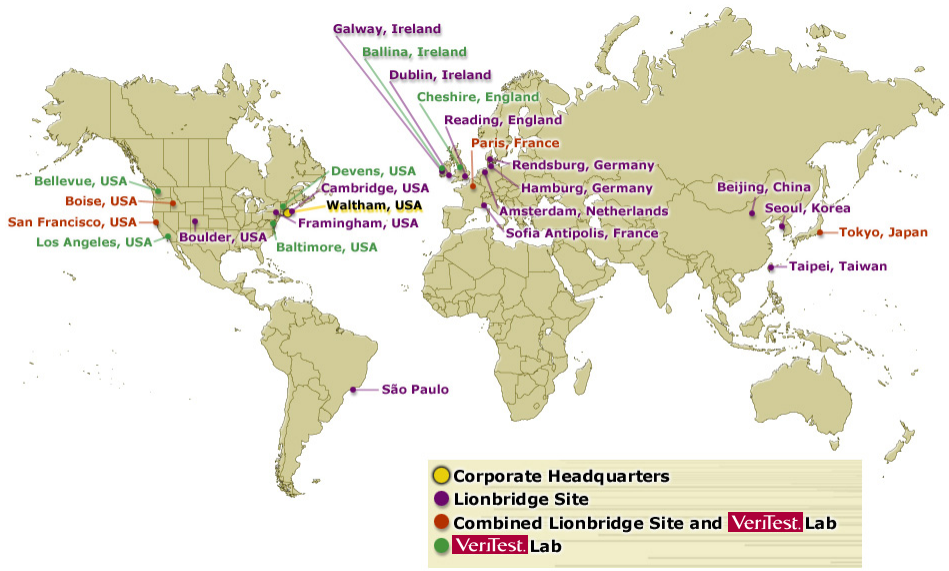
VeriTest: A Business Unit of Lionbridge

Lionbridge Thumbnails



- **Founded 1996**
  - > IPO 1999 (LIOX: Nasdaq)
- **Market Position in Globalization and Testing**
  - > No. 1 in revenue -- $125 million in 2002
  - > No. 1 in resources – 1,300 employees
- **Business**
  - > Globalization Solutions
  - > Testing Services
  - > Software Development
- **Industries**
  - > Fortune 1000 in IT, Life Sciences, Financial, Automotive

Global Infrastructure



Agenda

- Introduction
- Test Manager
- 7 Habits of Highly Effective Test Managers
- A Global Perspective & Challenges with Distributed Teams
- Lessons Learned
- Best Practices
- 5 Steps to Successful Virtual Test Management
- Conclusions & Recommended Reading

Test Manager

- Responsible for testing activities at one location
- Duties: managing, testing, employee and customer satisfaction and delivery
- Reports to the Virtual Test Manager
- No travel required

Agenda

- Introduction
- Test Manager
- 7 Habits of Highly Effective Test Managers
- A Global Perspective & Challenges with Distributed Teams
- Lessons Learned
- Best Practices
- 5 Steps to Successful Virtual Test Management
- Conclusions & Recommended Reading

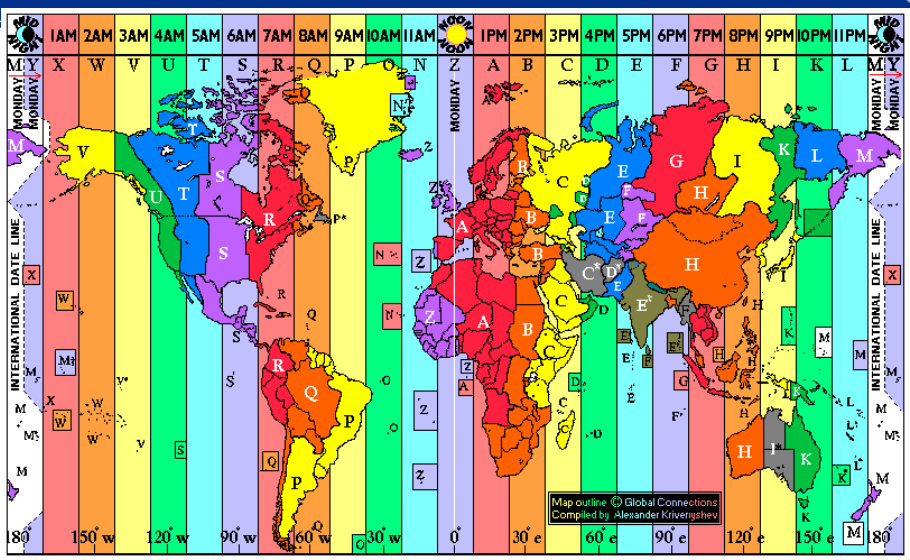
7 Habits of Highly Effective Test Managers

- Fulfill the leadership role within the test organization
- Act as a catalyst for positive results
- Facilitate and develop team members' capabilities
- Eliminate barriers to team effectiveness and efficiency
- Understand and communicate business & customer needs
- Constantly coach and mentor each individual & team
- Set a living example for all to follow

Agenda

- Introduction
- Test Manager
- 7 Habits of Highly Effective Test Managers
- A Global Perspective & Challenges with Distributed Teams
- Lessons Learned
- Best Practices
- 5 Steps to Successful Virtual Test Management
- Conclusions & Recommended Reading

A Global Perspective



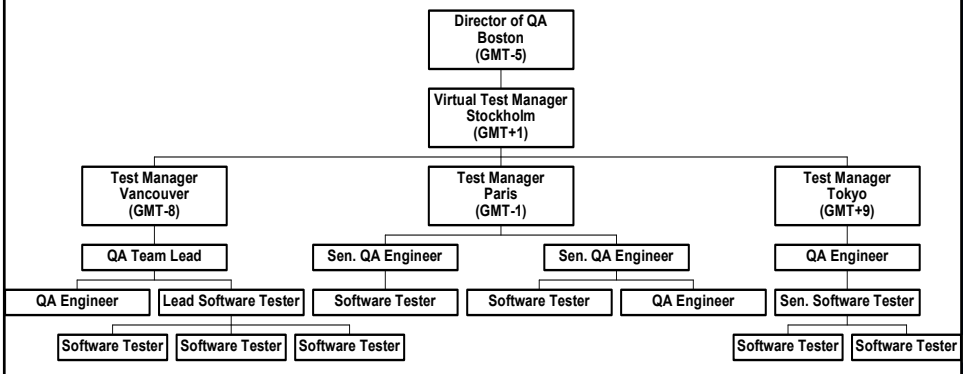
Complexities of Scale

- Geography
  - > Multiple test site locations
- Employees
  - > Multiple test teams and test managers
- Time management
  - > Attention allocated to each site
- Travel
  - > Can't be in more than one location at a time
- Resources
  - > Shared personnel and test labs
- Company cultures
  - > Multicultural environment
  - > Trying to transform multiple companies into one

Virtual Test Manager

- Virtual
  - > Mobile, informed, connected and accessible
  - > Use of tools (the mobile office)
- Test
  - > Metrics/measurements for all sites
- Manager
  - > Management of multiple test sites and teams
  - > Common goals and objectives
  - > Vertical and horizontal accountability

Example Virtual Test Organization



Challenges of Managing Multiple Sites

- Sense of identity for each site
- History of multiple site expansion
  - > Mergers and acquisitions, new office, existing site
- Time zone differences
  - > Constantly adjusting your "normal" work day
- Distribution and utilization of capabilities
- Project management
- Process, standards, information technology
- Internal vs. external competition
  - > People vs. product focus

Agenda

- Introduction
- Test Manager
- 7 Habits of Highly Effective Test Managers
- A Global Perspective & Challenges with Distributed Teams
- Lessons Learned
- Best Practices
- 5 Steps to Successful Virtual Test Management
- Conclusions & Recommended Reading

Lessons Learned

- There is no substitute for face-to-face interactions
- Travel across time zones will disrupt your lifestyle
- Get used to working "around the clock"
- Time zone "bias" will affect the quality of your relationship with each site
- There's more than just American culture out there
- Don't neglect the other sites when you're not there
- You will need many tools and techniques to help you do your job
- Sharing of test resources may not be that simple
- Make sure you have space and connectivity at each site when you visit

Rapid Testing Over Multiple Time Zones

- Collaborative effort among the test sites
  - > Shared resources
- Power of duplication
  - > Concurrently running programs
- "24/7 testing"
  - > Ongoing, multi-level test programs
- Test automation
  - > Maximize testing efficiency and save time
- Session-based test management
  - > Structured exploratory testing

Agenda

- Introduction
- Test Manager
- 7 Habits of Highly Effective Test Managers
- A Global Perspective & Challenges with Distributed Teams
- Lessons Learned
- Best Practices
- 5 Steps to Successful Virtual Test Management
- Conclusions & Recommended Reading

Best Practices

- Hire and empower well-qualified test managers
  - > Assist them with their recruiting efforts
  - > Give them as much empowerment as they can handle
- Be constantly reminded and respectful of time zone differences
  - > World time desk clock, dual time wristwatch
  - > Encourage acceptable real-time communication
- Manage and respect cultural and site differences
  - > Manage goals
  - > Cultural training

Best Practices (cont.)

- Rotate your time spent at each site
  - > Get involved with their day-to-day activities
  - > Be part of their team
- Keep each team connected and make each site feel like they are part of one team/one company
  - > Share news and information among the sites
  - > Ongoing team ("site building") exercises
  - > Group training and "face time" opportunities
  - > Shared use of tools

Agenda

- Introduction
- Test Manager
- 7 Habits of Highly Effective Test Managers
- A Global Perspective & Challenges with Distributed Teams
- Lessons Learned
- Best Practices
- 5 Steps to Successful Virtual Test Management
- Conclusions & Recommended Reading

5 Steps to Successful Virtual Test Management

1. 'Calibrate' each site to establish a baseline
   - Meet with test managers and all team members
   - Communicate all goals and objectives
   - Perform an assessment and form an improvement plan
   - Ensure needs are met and issues are resolved
2. Empower the test managers but be ready to intervene when necessary
   - Monitor conflicts among test sites and resource issues
   - Remember to constantly coach and mentor

5 Steps to Successful Virtual Test Management (cont.)

3. Continually manage each site
   - Find out what's really going on
   - Determine how you can assist them
   - Have ample "face time" with the test team and other groups
4. When you're not physically there, be there virtually
   - Keep communication lines open and touch base often
   - Participate in their activities on a weekly basis
5. Maintain harmony and synergy at each site and for the entire test organization
   - Ensure employee satisfaction
   - Ensure customer satisfaction

Agenda

- Introduction
- Test Manager
- 7 Habits of Highly Effective Test Managers
- A Global Perspective & Challenges with Distributed Teams
- Lessons Learned
- Best Practices
- 5 Steps to Successful Virtual Test Management
- Conclusions & Recommended Reading

Conclusion

- Virtual Test Manager =

  world traveler + test professional + technical manager
- Virtual Test Manager is the "glue" that keeps all the "pieces" (the test sites) together
- No matter where you are, you will always be managing virtually
- Manage to the organization with the complexities of scale
- Communication, travel, and "face time" are essential

VeriTest Representative Customers

Software/Web Developers

Hardware Developers

Networking Developers

Mobile Tech. Developers

Enterprise Developers

Case Study: AvantGo Software

- **Challenge**
  - AvantGo was behind schedule in launching a complex, multi-tier content synchronization system in Western Europe.
- **Action**
  - VeriTest's combination of technical and linguistic testing skills, ready-to-deploy test beds, and experience in mobile product technologies eliminated project ramp-up time.
- **Results**
  - AvantGo shipped on schedule.

AvantGo

"VeriTest was a critical factor in the successful launch our enterprise software. The VeriTest team uncovered a wide variety of localization defects. The result was a much more polished product. Working with VeriTest's professional staff was a pleasure."

- Kent Sorsky, Software Testing Manager
  AvantGo, Inc.

VeriTest.

Recommended Reading

- Bach, Jonathan. (2000). Session-Based Test Management. Software Testing & Quality Engineering, November/December 2000, 32-39.
- www.stickyminds.com
- www.professional-tester.com
- Duarte, Deborah and Tennant Synder, Nancy. (1999). Mastering Virtual Teams: Strategies, Tools, and Techniques That Succeed. Jossey-Bass. ISBN 0-78-794183-2.
- Fisher, Kimball and Duncan Fisher, Maureen. (2001). The Distance Manager: A Hands-On Guide to Managing Off-Site Employees and Virtual Teams. McGraw-Hill. ISBN 0-07-136065-4.
- Haywood, Martha. (1998). Managing Virtual Teams: Practical Techniques for High-Technology Project Managers. Artech House. ISBN 0-89-006913-1.
- Kostner, Jaclyn. (1996). Virtual Leadership: Secrets From the Round Table for the Multi-Site Manager. Warner Books. ISBN 0-446-67087-1.
- Lipnack, Jessica and Stamps, Jeffrey. (2000). Virtual Teams: People Working Across Boundaries with Technology. John Wiley & Sons. ISBN 0-471-38825-4.

QW2002 Paper 9T2

Dr. Nagesh Vempaty
(Aalayance Incorporated)

Cross Platform Framework For Integration Testing

Key Points

- CROSSFIT manages the integration testing of distributed applications
- It encourages automation of test suites
- Our customers report testing time reductions of 70-80%

Presentation Abstract

Aalayance developed a cross platform open source framework for integration testing (CROSSFIT) to manage the integration testing of distributed applications. The key design goal of CROSSFIT was to be capable of managing the integration testing of distributed applications. To meet this goal, we came up with a distributed agent based architecture. CROSSFIT was also designed to be simple, portable, usable and flexible. It encourages automation of test suites. We have been using it extensively for more than a year now with great success. Our customers report testing time reductions of 70-80%. In this document, we share some of the design principles, the implementation strategies and the practical experiences.

About the Author

Dr. Nagesh Vempaty is responsible for development and engineering activities associated with new customers at Aalayance. Dr. Vempaty's technical expertise is in large-scale software development and systems integration. As the CTO at Zoho, he was responsible for overall product development and for improving the availability, scalability and performance of Zoho's products. Prior to Zoho, Dr. Vempaty was the co-founder and CTO of hsupply.com. Prior to hsupply.com, Dr. Vempaty worked at Healtheon, Tenneco and NASA. Dr. Vempaty has managed several software development teams under extreme situations.

Dr. Vempaty received his Ph.D. in Computer Science from the University of Texas at Austin and is a recipient of the Microelectronics and Computer Corporation Award for outstanding graduate research and the Gordon Bell Prize in parallel computing. He received his Master's in Computer Science from the University of Texas at Austin and his Bachelor's in Compute Science from Indian Institute of Technology (IIT) at Madras.

A Cross-Platform Framework for Integration Testing

Dr. Nagesh Vempaty
VP of Engineering
Aalayance, Inc.

aalayance

Agenda

- Background
- Enterprise Application Integration Testing (EAIT)
- Objectives
- Case Study
- Summary

aalayance

Background

- Aalayance is an outsourced software development firm
- One of our core competencies is Enterprise Application Integration (EAI)
- We developed a Cross-Platform Framework for Integration Testing (CROSSFIT) to increase our efficiency and improve the quality of our deliverables
- CROSSFIT has been in use for over one year and has dramatically improved the EAI testing process

aalayance

EAI Testing Issues

- Software to be tested runs on multiple operating systems (Windows, Linux, Solaris, …)
- Utilizes a variety of messaging architectures and technologies (publish/subscribe, queuing, request/reply, DCOM, CORBA, EJBs, …)
- End applications built using a variety of languages (C, C++, Java, VB, PERL, …) and
- Flat files, directories, relational and object databases are used as data stores
- System Under Test (SUT) is distributed. Test logic is distributed and hence complex

Enterprise Application Integration Testing (EAIT) is a unique problem

aalayance

CROSSFIT Objectives

- Focus on EAI Testing (e.g. do not cater to front-end GUI testing or stand alone application testing)
- Support distributed, heterogeneous environments
- Minimize intrusion into the System Under Test (SUT)
- Support manual interactions with SUT
- Support multiple test scenarios and test cases with configurable target environments

aalayance

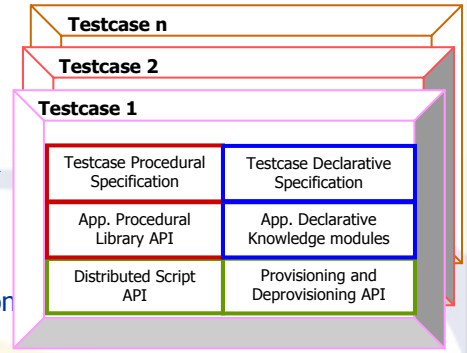
CROSSFIT Philosophy

- Test Cases defined using
  - Procedural specification (logic and flow)
  - Declarative specification (definition of test environment)
  - Note: separation of logic and flow from environment allows tests to be easily retargeted to new environments
- Test organization
  - A regression test library consists of a set of test scenarios
  - A test scenario consists of a set of test cases
- Support for test ware such as test data, versioning, etc.

aalayance

CROSSFIT Architecture

- Test Cases defined using
  - Procedural specification (logic and flow)
  - Declarative specification (definition of test environment)
  - Note, separation of logic & flow from environment allows tests to be easily retargeted to new environments)
- A test scenario is a collection of test cases
- A test library is a collection of test scenarios

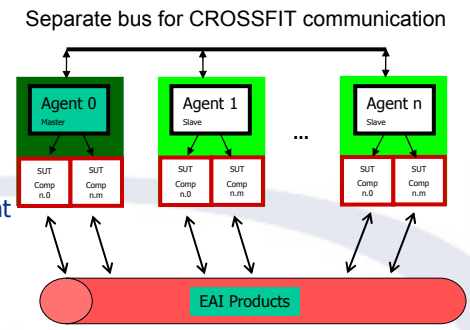


A test scenario

aalayance

CROSSFIT Architecture

- Distributed Agents
  - Organized in peer to peer network
  - A specific test case is orchestrated in master-slave mode
- Test Suite
  - Can be launched by any agent
  - Launching agent becomes "designated leader"
  - Remaining agents are subordinate to the leader
  - Any subordinate agent can launch a distributed sub-test



aalayance

CROSSFIT Implementation Issues

- Extensively used open source tools such as Tcl, Expect, BASH:
  - Used Tcl-DP for communication and synchronization
  - Very light weight
  - Very minimally intrusive
- Support for test ware management and versioning:
  - Test data sets
  - Configurations
- Data collection and reporting
  - Summary report
  - Drill down to issues

aalayance

CROSSFIT Case Study

- SUT Components
  - Messaging Platform
  - Enterprise Application
  - An Adapter
- Test Environments
  - Six target platforms (Solaris, AIX, Linux, HP-UX, Windows, DEC-UX)
- Test need to be re-run for every major and minor release of any component

	Item	Before CROSSFIT	After CROSSFIT
1	# of test scenarios	20	67
2	# of test cases	50	180
3	Time required to test	8-10 hrs	1.5 hrs (~0.5 hrs of work)
4	Time required to test on 6 target platforms	6-7 person/ days	3 hrs if parallel; 1 day if serial
5	Time to rerun the test in a new SUT configuration	1 person/ day	1.5 hrs (~0.5 hrs of work)
6	Coverage of features	Low	High

aalayance

Summary

- EAIT is a unique problem due to the distributed nature of the problem
- CROSSFIT addresses this problem via:
  - An agent based architecture
  - Separation of procedural logic and declarative logic
- We have demonstrated ROI in terms of reduction of manual work and ability to orchestrate distributed tests that cannot be run manually
- We have used it successfully for over a year
- For more details, visit http://www.aalayance.com

aalayance

Aalayance, Inc.
1250 Oakmead Parkway
Suite 312
Sunnyvale, CA 94085
O 408 992 0234
F 408 516 9718
www.aalayance.com

A Cross-platform Framework for Integration Testing (CROSSFIT)

Final Paper

Submitted By:	*Aalayance Inc.*
Submitted To:	*QualityWeek 2002 Conference*
Date Submitted:	*July 15, 2002*

Copyright © 2002. Aalayance Inc. All rights reserved. This document represents proprietary research on the part of Aalayance Inc.

A Cross-platform Framework for Integration Testing (CROSSFIT)

Prasad Chodavarapu, Rama Kanneganti, Nagesh Vempaty, Lakshmi Athreya,Venkatesh Subramanian, and Rohith T.S.

Aalayance Inc. (http://www.aalayance.com/)

Abstract

Aalayance developed CROSSFIT, a cross platform framework to manage and automate enterprise application integration testing. CROSSFIT was designed to be simple, portable, extensible, usable and flexible. CROSSFIT supports the development, orchestration, and maintenance of distributed test cases. It addresses the testing needs of distributed systems that are typical in enterprise application integration scenarios. We have been using it extensively for more than a year with great success. In this paper, we outline some of the design principles, the implementation strategies and the experiences.

1. Introduction

1.2 Motivation

Enterprise Application Integration Testing (EAIT) involves testing systems spanning diverse software applications and integration techniques. This complexity is more challenging if the system under test runs on different platforms, such as Linux, Solaris, HP-UX, MS Windows NT, MS Windows 2000, IBM AIX, etc., and uses different types of hardware. This is true most of the time in enterprise application integration (EAI). Application integration may be enabled via different transport protocols and techniques. EAI is often accomplished via messaging and some times via file transfer or request-response. EAIT is hence a complex task.

CROSSFIT addresses this EAIT problem. It helps in managing the complexity of distributed test cases. It can enable the automation of diverse and complex distributed testing scenarios that are needed to handle EAIT.

1.3 Overview

CROSSFIT aids EAIT. It does not cater to front-end GUI testing, for which a number of industrial strength tools exist. It does support traditional integration testing. However, its distinct utility is in addressing the distributed testing problems that are unique to enterprise application integration (EAI). We used a planned and disciplined approach to EAIT automation that drew upon the documented experiences of QA professionals who have practiced and analyzed the art. We also used basic principles of software engineering, such as re-use of both utilities and data sets, use of version control for test scripts and adherence to configuration management for test suites. In short, we treated test automation on par with software development and applied the same principles.

We used an agent-based architecture for CROSSFIT. An EAI test involving multiple applications running on multiple machines is carried out with the help of multiple distributed agents. A test case is built as a Tcl script [Ous94] using the CROSSFIT API. A test case is partitioned into a procedural

Copyright 2002. Aalayance, Inc. All rights reserved.

part and a declarative part. The procedural part concentrates on the test case's logic and flow. The declarative part deals with details such as command lines, environment specifications and host information.

1. The procedural API provides for running processes in the foreground/background, managing input/output in a time-bound fashion, filtering outputs, comparing against golden outputs, and cleaning up. Although the API makes it look almost like English, as the test case is an Expect script [Lib94], any legal Tcl/Expect construct can be used.
2. The declarative API deals with command lines, environment specifications, host information (for distributed testing) and the like. Distributed tests are treated just like single application tests; only the host information for an application needs to be specified declaratively. Making this declarative supports the usage of symbolic environment names and host names. Therefore, the customization of test scripts for each target test bed requires only the editing of a few configuration files. Thus, modifying a test case to deal with changes in the application environment and diverse systems is easy.

We used open source tools as much as possible in the implementation, to handle many of the distributed and cross-platform issues. We used Tcl with Expect, a language widely adopted by the testing communities worldwide. Distributed processing was handled with Tcl-DP. We have been able to use CROSSFIT on a heterogeneous network of platforms, including many flavors of UNIX, LINUX, AIX and Windows.

We have been using CROSSFIT for almost all of our test automation needs for more than a year now. We were able to reduce the time taken for testing and improve the number of integration tests that we could run in the reduced time. We are also able to conduct the tests on multiple platforms and test beds with minimal effort.

2. The Integration Testing Problem

2.1 Problem Definition

Consider a company dealing with application integration testing problem. The applications may be running on various platforms. There may be numerous operating systems, hardware, applications and integration tools. These applications are written in a host of languages like C++, VB, Java, PERL, and so on. The integration solutions may use technologies like DCOM, CORBA, various types of messaging, and so on. The applications utilize DBMSs, directories, file servers, and so on. The applications communicate with products in partner companies using established protocols. The components of the application under test will be distributed across various servers in the network. All these need to be tested and certified for product releases as well as changes in the configuration of any of the component systems. Such distributed test scenarios need to be rerun over multiple environments such as QA, Staging, Production, and so on.

The certification of a system or a change needs to be under taken on a host of operating systems. The testing time is very limited due to tight deadlines. Future changes to any aspect of the system must pass a regression test suite. The testing should orchestrate a complex tango of interactions to catch any potential bugs that could jeopardize the functioning. The testing should be repeatable as many times as possible, within the deadline, to assure consistency. Due to the asynchronous nature of

Copyright 2002. Aalayance, Inc. All rights reserved.

messaging, race conditions are a common source of bugs. Foregoing a day of testing activity early in the project is likely to cost the firm 3 to 10 days of activity downstream [Jon94]. This inefficiency will undermine the development speed; increase the cost, and the time to market.

The degree of success and repeatability of the above complex scenario will ultimately affect the bottom-line of the company as well as the trust placed in the IT system integration.

2.2 Nuances of the Problem

Interactive automation complicates the testing problem, as it requires different actions depending on the intermittent run-time output. In some cases, a temporary transfer of control to the user executing the automation will be required. Yet, in some other cases where a process outputs a lot of data, automating the analysis of the data will be required.

It is also important to elegantly control the configuration of a process, its provisioning and de-provisioning. It is important to restore the original environment after the completion of testing. Different sub-operations within the testing process should isolate their specific environments from each other. Testing a fully distributed system involves development of test cases that involve distributed logic. The manual orchestration of certain scenarios is not practical due to the near real time synchronization requirements of the test cases. Any delays in manual orchestration intrusively change the system behavior and cannot catch race conditions spanning multiple applications.

While analyzing test data from a single system itself is a complex task, analyzing test data from multiple applications and the integration tools in near real time is more complex. Therefore, it is important to facilitate filtering the reams of output to the specific portions that determine the outcome of the test and provide clues on the bugs that may exist. It is also desirable to automate matching actual outputs to the expected or gold output, so that the tester can focus on specific problems and their diagnosis.

The EAIT tool itself should be lightweight, so that it does not affect the actual performance of the system under test (SUT). The tool should also minimize intrusion into the behavior of the SUT, so that it does not alter or mask the defects in the SUT during testing.
Due to the distributed nature of the EAI domain, the testing tool should also be distributed. It should support its own synchronization and communication mechanisms to support the test programs. These need to be out of band with respect to the SUT to be non-intrusive.

3. CROSSFIT Solution

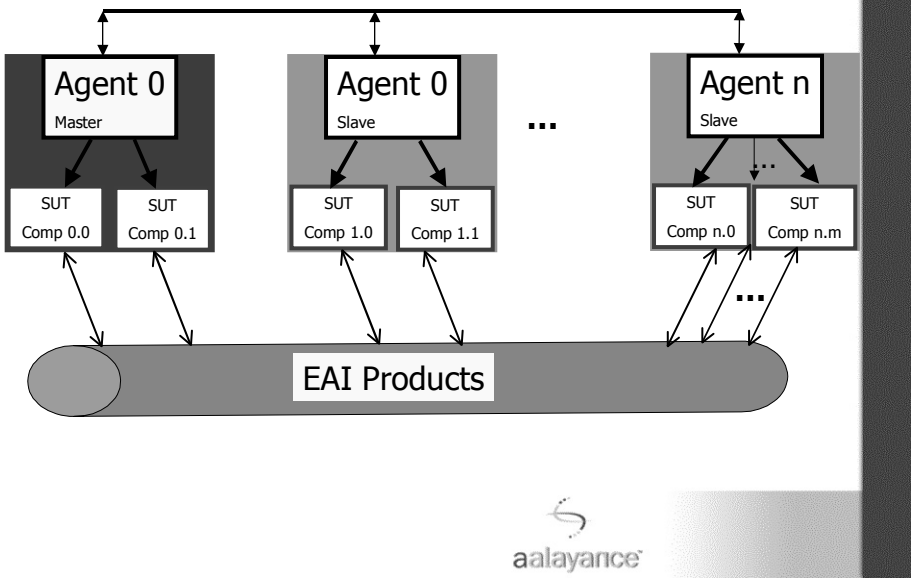
3.1 Architectural Details

CROSSFIT uses a framework of distributed agents. The agents are organized in a peer-to-peer network. A test suite can be launched from any of the agents. The agent network is then managed in a master-slave mode. The orchestration comes from a designated leader. The rest of the participating agents are subordinate to the leader. During the execution of a test suite or a test case, it is possible

Copyright 2002. Aalayance, Inc. All rights reserved.

that one of the slave agents may start a distributed sub-test. It that case, the specific agent becomes the master of orchestration of the sub-test, and the other agents participating in that sub-test are slaves in the context of that sub-test. The overall architecture relies on a peer-to-peer agent framework. This is illustrated in Fig. 1.

Fig. 1: CROSSFIT agent architecture



The agents launch their portions of the SUT in a controlled sandbox. This is to isolate the testing tools behavior from the defects in the SUT. A crash in the SUT does not bring down the test harness.

The test framework itself is lightweight, so that it does not affect the actual performance of the product being tested. It supports its own synchronization and communication mechanisms that are utilized exclusively by the test programs and do not interfere with the SUT. The test programs can create distributed events across the test framework to test the SUT.

At a very high level, a test run consists of three steps:

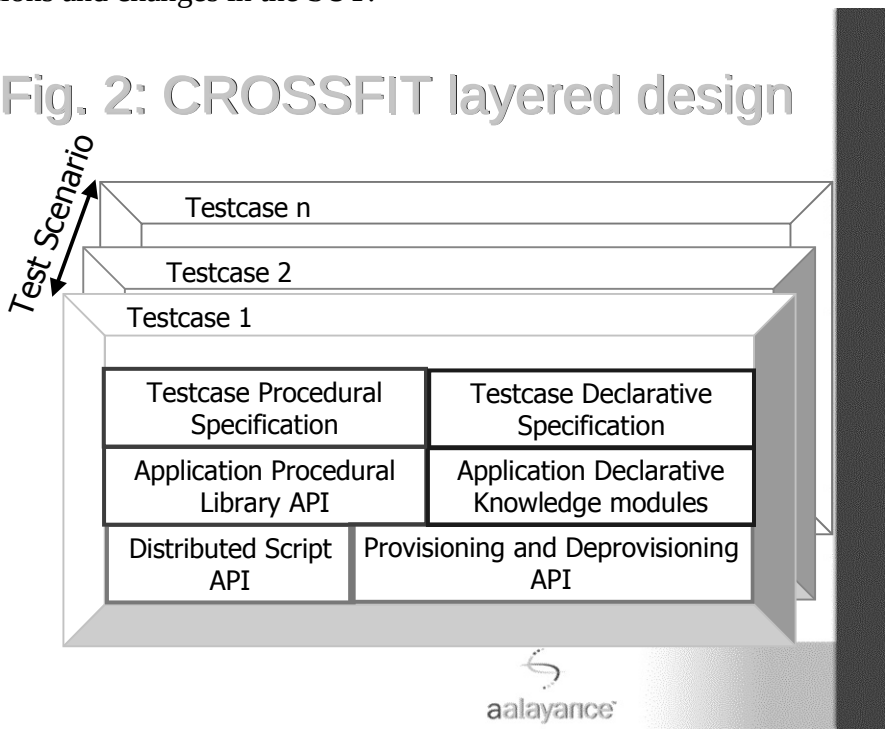
1. Provisioning: Setting up the SUT, activating necessary agents and launching various components.
2. Execution: The test program executes in a distributed fashion, starting with the master agent and percolating across the agent framework.
3. De-provisioning: The test result data are collected in individual agents and collated into a database by the master agent. The participating systems are restored to the appropriate state. It is important to support de-provisioning as a large number of participating systems need to be cleaned up for other tasks after the conclusion of a test run.

CROSSFIT encourages test developers to separate the test code into a declarative piece and a procedural piece. This philosophy was followed ground up during the design and implementation of

Copyright 2002. Aalayance, Inc. All rights reserved.

CROSSFIT. Fig. 2 illustrates the layered design of CROSSFIT. A test scenario is a collection of test cases. Each test case has a declarative portion and a procedural portion. These are in turn built from procedural and declarative modules provided by CROSSFIT. The separation of a test case into a declarative piece and a procedural pieces eases maintenance effort and retargeting the tests to different configurations and changes in the SUT.

Fig. 2: CROSSFIT layered design



3.2 Implementation Details

We implemented CROSSFIT using open-source software such as Tcl, Expect and Bash [New98]. Expect, a superset of Tcl, is a very unique language as it meets the requirements of interactive automation. Bash is used as the glue to bootstrap the driving of test cases. Using Tcl makes the framework cross-platform. CROSSFIT supports the option to flush output and re-start buffering at anytime during the test run. This is made possible by use of Expect's internal mechanisms. By tapping the power of scripting languages like Expect, TCL and Bash, the CROSSFIT tool is very lightweight.

While test outputs are collected by individual agents and analyzed, summary data and problems detected are collated by the master agent in charge of the test run, and written to a database. CROSSFIT supports an API and data model to help the collection of such data. The data model is suitable for summary reporting and drilling down to specific defects.

3.3 Distributed Testing Capabilities

Application components of the SUT can reside anywhere on the network. The test driver on the master agent just needs the IP address of the machines hosting the components. Tcl has a module, Tcl-DP, which we used for distributed testing. Internally, Tcl-DP uses RPC for remote execution. A dedicated port is used for the communication between the agents. This keeps the communication and

Copyright 2002. Aalayance, Inc. All rights reserved.

synchronization in CROSSFIT clear of the EAI tools used in the SUT. While the agents themselves are peer-to-peer, there is a designated master agent in charge of running a test case. This agent drives all the other agents, leading the orchestration of the test case being run over multiple agents running on multiple machines. During the test case orchestration, the lead agent assumes the designation of a master role and the other agents take slave roles. The designation of roles is dynamic to suit the need of a test run.

3.4 Testware Management

Testware is the product of test development. It consists primarily of test data and utility sets. Testware development is a continuous process and it needs as much discipline as software product development. Version control and configuration management are two aspects of that discipline. CROSSFIT supports a version control repository for testware. The details of that repository are beyond the scope of this paper.

3.5 Benefits

CROSSFIT provides the following benefits in the context of EAIT:

1. Multi-platform support including various flavors of UNIX, LINUX, AIX and Windows.
2. Lightweight and minimally intrusive due to the use of a few simple script files. There are no executables or DLLs.
3. Easy to learn, code, deploy, version, and change test suites.
4. Ability to develop, run and maintain automated regression test libraries for complex EAI scenarios.
5. Savings of several days per QA cycle in a complex setup.
6. Ability to run a test case on multiple SUT configurations with minimal effort.

4. Advanced Features

The following advanced features are supported in CROSSFIT:

1. Reporting: A web/HTML report allows a drill down of the results of test execution from the top. A drill down from the summary graph of results to detailed log files is possible.
2. Test result analysis: CROSSFIT enables a drill-down to failures, so that testers can focus on identifying and diagnosing defects.
3. Output filtering: A filter is a text conversion program that will read the source from standard input and write the converted text to standard output. A date filter should be provided to replace commonly occurring date formats with placeholders like <DATE> and <TIMESTAMP>. This filter may optionally be configured to replace all/only some of the date formats.
4. Output comparison: The following comparisons are supported:
    a. Exact comparison with a golden output file
    b. Searching for the chunk of text specified by a golden output file

Copyright 2002. Aalayance, Inc. All rights reserved.

5. Case Study

5.1 Project Alpha

Project Alpha involved testing a simple EAI system consisting of a messaging platform, an application, and an adapter. At a very high level, the business processes consisted of sending data to the application via the messaging platform and the adapter, and receiving data from the application via the adapter into the messaging platform. The goal was to test the builds and releases of the adapter. There were six target platforms (Solaris, AIX, Linux, HPUX, Windows and DEC-UX) to be tested and certified, for every major release and minor release each of the messaging platform, the application and the adapter.

5.2 Cost benefit analysis

Prior to CROSSFIT, this was all done manually. The configuration of the system under test (SUT) was itself a significant task. Testing was limited to a few scenarios that can be managed manually. Manual testing on each platform took one complete person day of effort. The product was certified on six different platforms. So it took six person days of effort to test the product manually.

Table 1 Comparison of effort and complexity of testing before and after CROSSFIT.

#	Item	Before CROSSFIT (Manual)	After CROSSFIT (Automated)
1	Number of test scenarios	20	67
2	Number of test cases	50	180
3	Time required to test	8-10 hrs	1.5 hrs (about 0.5 hrs of person work)
4	Time required to test on six target platforms (Solaris, AIX, Linux, HP-UX, Microsoft Windows and DEC-UX)	6-7 person days	3 hrs if done in parallel; one day if serialized.
5	Time to rerun the test in a new SUT configuration on a given target platform	1 person day	1.5 hrs (about 0.5 hrs of work
6	Coverage of features	Low	High

Using CROSSFIT, it was possible to build and manage a regression test suite spanning sixty-seven test scenarios and a total of one hundred and eighty test cases. These test scenarios were desirable in QA, but hitherto infeasible. It was also possible to declaratively specify the configuration of the SUT. The separation of declarative and procedural portions of a test scenario enabled testing on several target platforms of the product with minimal effort. With the automation provided by CROSSFIT, executing the whole regression test library took 90 minutes on each platform. Several platforms could be tested in parallel. It was possible to cover a larger and more comprehensive set of test scenarios with fewer people and less time. With CROSSFIT, it was possible to complete the entire testing effort on the six target platforms in one-person day instead of six days, thus reducing the time by nearly 85%.

Copyright 2002. Aalayance, Inc. All rights reserved.

Another noticeable benefit was that CROSSFIT enabled building regression test libraries in a reasonable time frame. These libraries can be reused over subsequent product releases and multiple platforms. This was not practical before.

5.3 Future Directions

The next version of CROSSFIT will support advanced code generation templates for accomplishing a lot of common tasks. This will aid in developing new test scenarios by reusing templates as building blocks.
We are also building a GUI for CROSSFIT to ease the building and managing of test scenarios. Integration with project schedule management software and bug tracking systems is also in design. This leverages more value already available in other software development and testing tools in an enterprise. Support will be added for integration with GUI testing tools, for example with tools that test drive web interfaces.
Another interesting extension is the ability to orchestrate tests across the Internet. An example business case for this is the ROSETTANET initiative that involves collaboration across a value chain network of partners. Testing such a scenario would require extending CROSSFIT with a security protocol to span multiple enterprises.

6. Conclusions

EAIT is an interesting and challenging domain for QA due to the nuances of the problem. The system under test often spans multiple applications and multiple platforms. The test scenarios involve distributed techniques that are hard to code, and impossible to orchestrate manually.
CROSSFIT uses an agent-based architecture and a simple distributed programming model that separates the procedural portion of a test from the declarative portion. It builds on a number of best practices in software development and testing. The benefit of CROSSFIT is the ease of building, running and maintaining a large set of test scenarios in a regression suite the context of EAIT. Some of the scenarios enabled are not executable manually due to the distributed nature of the problem.
CROSSFIT was successfully used by us and our customers to manage EAIT in a number of scenarios. We see measurable ROI in reduction of costs and improved coverage of test scenarios. This results in better quality in the end products that are being tested.

References

[Jon94] C. Jones, *Assessment and Control of Software Risks,* Yourdon Press Computing, ISBN 0-13-746401-4, 1994.
[Lib94] D. Libes, *Exploring Expect: A Tcl-based Toolkit for Automating Interactive Programs,* O'Reilly & Associates, ISBN 1-56592-090-2, 1994.
[New98] C. Newham and B. Rosenblatt, *Learning the bash Shell, 2nd Edition,* O'Reilly & Associates, ISBN 1-56592-347-2, 1998.
[Ous94] J. Ousterhout, *Tcl and the Tk Toolkit,* Addison-Wesley, ISBN 0-201-63337-X, 1994.

Copyright 2002. Aalayance, Inc. All rights reserved.

QW2002 Paper 2A1

Mr. Michael Weider
(Watchfire Corporation)

Testing and Managing Risky Content in the Online World

Key Points

- Risk problems associated with privacy
- Risk problems associated with accessibility
- Risk problems associated with third party content

Presentation Abstract

Many organizations still lack insight into what users are experiencing on their website and have no means of finding the risk issues that drive users away. In addition to quality issues that have the potential to damage a company's brand, new problems have arisen. These issues include privacy glitches, accessibility issues and managing third-party links. Since these issues are often buried within tens of thousands of pages, they are cumbersome and difficult to find and resolve. Until now, this has not been an area well addressed by vendors or internal quality assurance teams.

Understanding the extent of the problems, where they're located, and where the responsibility lies for creating and addressing the problems is critical. Current manual testing methods are expensive and ineffective. This lack of a centralized, automated solution often means customers will find the problems first.

The consequences of these problems can be severe. They can affect your ROI with impacts such as lost customers, brand erosion, and reduced conversion and customer retention rates. They can also affect the bottom line through decreased productivity, increased litigation risk from lack of compliance to corporate or government standards, and a lack of e-metrics to manage the website effectively.

Manually auditing and monitoring corporate websites is not a one-time event and can be costly, inefficient and virtually impossible. Mr. Weider will share his insight on how to better understand potential risk management issues and how businesses can use the latest technologies to help proactively manage these issues.

When you leave this session, you will learn:
• How to better understand potential online risk management issues
• How to identify common online risk problems
• What can be done to prevent them
• How to bridge the communications gaps between your legal and technical teams
• How new technologies can automate the testing and analysis of sites for risk issues
• How to integrate risk management into the Web publishing process
• What are the best practices that the world's leading sites are using to test and manage their sites for risk

About the Author

Michael Weider is the Founder and CTO of Watchfire Corporation. Founding Watchfire in 1996, Mr. Weider has spearheaded the company into a leadership position in the Website Management software market. As CTO, Michael is responsible for product strategy, engineering, technical support and customer service. Prior to Watchfire, Mr. Weider was a co-founder of a software startup, Quadrillion Corporation. He holds a B.S. in engineering from Queens University.

Mr. Weider has spoken at numerous high profile industry and business events, including Internet World UK and the annual Privacy by Design conference. He has been frequently quoted in the business and trade press including the Wall Street Journal and New York Times, Investor's Business Daily and USA Today. Mr. Weider has also been a guest on CNNfn on several occasions to discuss various website management issues including the topics of privacy and risky content. Mr. Weider was voted the 1999 Young Entrepreneur of the Year by Profit Magazine and was nominated for the young entrepreneur of year by Ernst and Young.

Testing and Managing Risky Content Online

Michael Weider, Founder and CTO, mikew@watchfire.com

OVERVIEW

- What are the issues and risks with Web content?
- Why is this problem not being addressed well?
- What are some solutions to the problem?

watchfire

WATCHFIRE CORPORATION

- Private software company (www.watchfire.com)
- Focused on content, testing, analysis and reporting
- Scalable, enterprise website management solutions: WebXM
- Executive headquarters in Boston, Ottawa
- Customers include over 50% of F500
- Major Partners include: Inktomi, Interwoven, Vignette, SPSS, PricewaterhouseCoopers and Deloitte and Touché

watchfire

EMERGING MARKET

watchfire

CONTENT ISSUES AND RISK EXPOSURE

- Web Content Issues:
  - Broken links
  - Application failures
  - Browser compatibility
  - Usability issues
  - Outdated or stale content
  - Search and navigation issues
  - Accessibility
  - Privacy
  - Security
  - Third party exposure
- Potential Risks:
  - Lost business
  - Brand damage
  - Litigation
  - Regulatory problems

watchfire

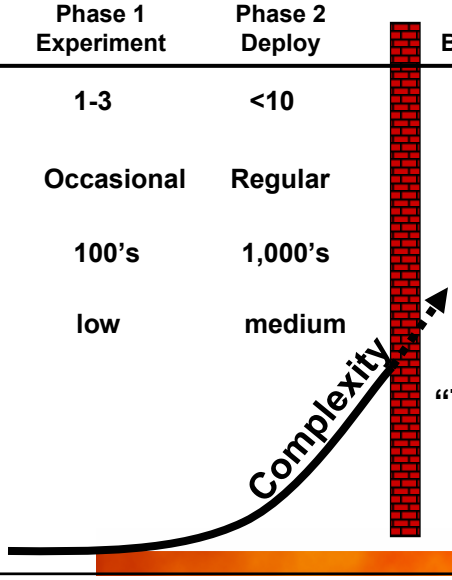
WHY ARE SITES SO BAD?

1. Lack of standards for content
2. Lack of process
3. Unclear accountability and ownership
4. Lack of automated tools
5. Amount of content is growing too fast

watchfire

SITES GETTING LARGER AND MORE COMPLICATED

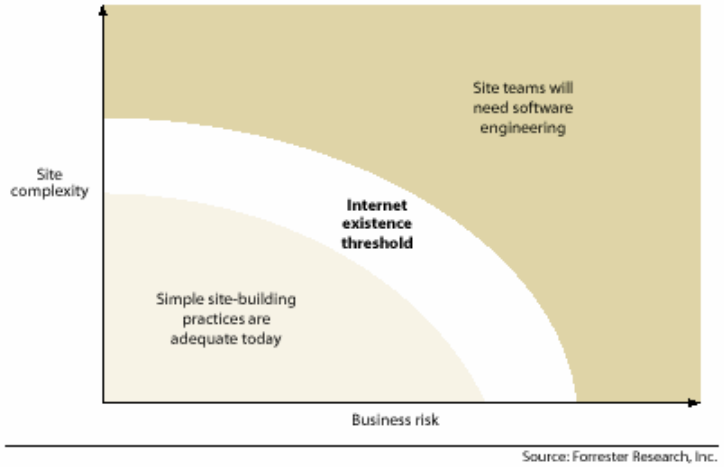
	Phase 1 Experiment	Phase 2 Deploy	Phase 3 Business Critical
1 Number of Contributors	1-3	<10	10-10,000+
2 Frequency of Changes	Occasional	Regular	Daily or hourly
3 Number of Pages	100's	1,000's	10,000's – 10's mil
4 Importance and risk to business	low	medium	high



"The Web Wall"

watchfire

NEW SYSTEMS AND PROCESSES REQUIRED



Source: Forrester Research, Inc.

watchfire

BEST PRACTICES FOR TESTING CONTENT

1. Establish standards for your content
2. Establish ownership, accountability and enforcement
3. Assess site for gaps
4. Fix problems found
5. Integrate prevention system into publishing process
6. Establish ongoing monitoring, reporting and management

watchfire

SAMPLE CONTENT STANDARDS

- Common look and feel
- Minimum quality (broken links, spelling, etc)
- Page weight
- Metadata
- Accessibility
- Legal statements
- Privacy
- Security

watchfire

ACCOUNTABILITY AND ENFORCEMENT

- Many different stakeholders:
  - Business units, marketing, development, QA, legal, IT
- Who is accountable for issues?
  - Quality, accessibility, privacy, security, etc
- Who is doing the testing?
- Who is going to fix problems?
- Who will enforce the problem if left unresolved?

watchfire

ASSESSING CONTENT FOR GAPS

- How does the site stack up against the standards?
- Where are the gaps?
- Some things to consider:
  - What site(s) do you need to check?
  - Do you even know what sites you own?
  - Do you have an automated solution to scan the content?

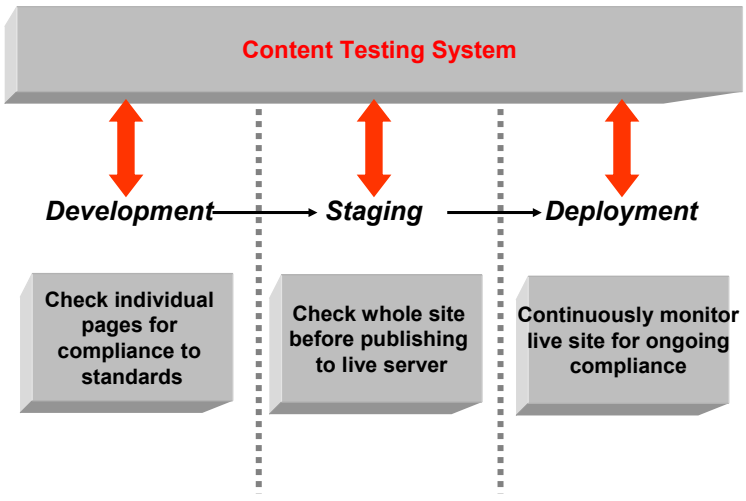
watchfire

FIXING PROBLEMS

- Each content owner needs a custom report detailing what the issues are.
- Provide guidance how to resolve issues where not obvious
- Warning, this can be a lot of work.

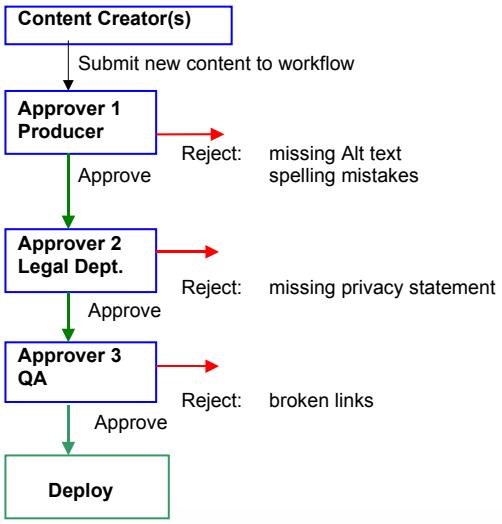
watchfire

PREVENTION - INTEGRATE INTO PUBLISHING PROCESS



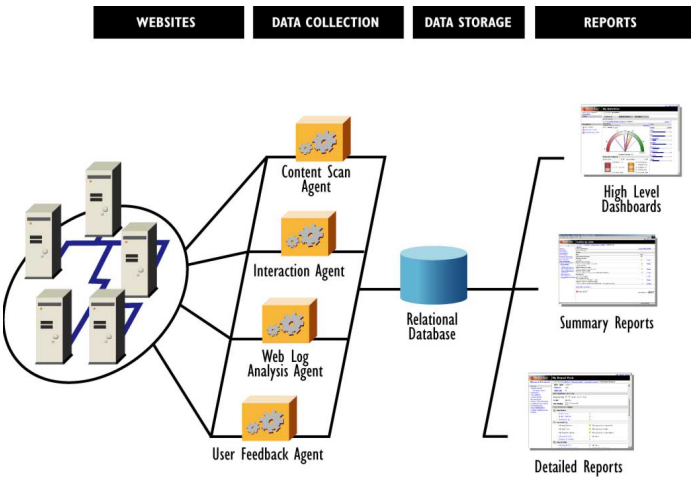
watchfire

Approve/Reject Work Flow Example



watchfire

MONITOR AND MANAGE



watchfire

CONCLUSION

- Content is one of your organizations most important IT assets
- Needs management to avoid to risks
- New approach to content testing required
  - Standards
  - Accountability
  - Assessment
  - Remediation
  - Prevention
  - Monitoring

watchfire

Testing and Managing Risky Content Online
Michael Weider, Founder and CTO, mikew@watchfire.com

QW2002 Paper 2A2

Mr. Peter Lafleur
(Public Works & Government Services Canada)

Testing a large scale application

Key Points

- Start or improve the process of testing a large scale web application.
- Adapt some of the techniques presented to your specific project needs.
- Apply cost saving and practical approaches to your own projects.
- Understand the areas of your organization impacted by Web conversion.

Presentation Abstract

This article describes a strategic and systemic approach to testing a large scale application conversion to the Web based on the experience gained from the first phase of conversion.

Public Works and Government Services Canada is a department of the Government of Canada currently participating in major infrastructure software development projects that will enable existing applications and facilities to be accessible as multi-tiered web-based applications. The Common Departmental Financial System(CDFS) is an internally developed back-office application offered to other government departments which is being transitioned from a client server architecture to a multi-tiered web architecture. CDFS is the financial management system of choice for 20 government departments with in excess of 1000 full time users.

The transition is planned to tale place over a three year-period. This paper describes the process followed, including the development and application of standards and checklists to facilitate effective and efficient testing of a large, scale web-enabled application. It will focus on the impact on 30+ experienced testers and developers as well as the problems faced in transforming current facilities with limited resources. Special testing concerns are introduced by the fact the current data server will remain static while the client server will be replaced by a web server hosted on a different mainframe.

Although, the team of software development and testing professionals working with the Financial and Reporting Products Directorate has considerable experience in testing traditional mainframe and client-server applications. The management challenge is to leverage this experience and practically apply the knowledge to new and emerging web based technologies.

This article will describe the approach taken, some of the problems encountered in this transition including developing new approaches to knowledge transfer supplemented with real examples and lessons learned which will be applied to subsequent phases.

About the Author

Peter Lafleur is a manager and business analyst with the Government of Canada with 25+ years experience in the areas of quality assurance, development and testing of government systems.

Testing a Large Scale Web Application

Presented By:
Peter Lafleur, Manager Product Development
Financial & Reporting Products

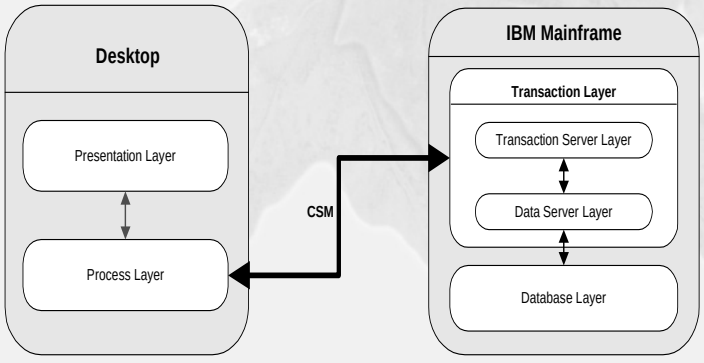
Outline

- **Introduction**
- **Web Enabling CDFS**
- **Web Testing**
- **The Human Factor**
- **Conclusion**

Background

- Public Works and Government Services Canada
- Common Departmental Financial System (CDFS)

Background: Common Departmental Financial System (CDFS) Architecture



Web-enabling CDFS

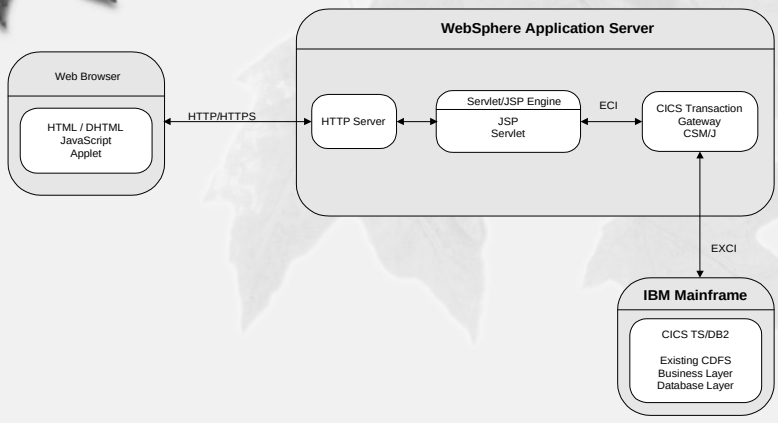
Why Web-enable:

- Current user interface has WinDOS screens
- Presentation layer barrier to new clients
- Conversion to web simplifies distribution of client component
- Simultaneous installation better coordinated

CDFS Standard Accounts Payable Screen

CDFS Web Accounts Payable Screen

CDFS Web Architecture



Web-enabling CDFS

Scope:

- Testing responsibility split between 3 partnering organizations
- Focus of this presentation on functional testing and testing on multiple platforms

Web Testing: Strategic Approach

Two Key Elements:

1. Reduce testing requirement by strategic decision-making in web design
2. Review and revise test strategy to identify impact of web conversion on existing testing approach

Web Testing: Strategic Approach

Strategic decision-making in web design:

- Adopted thin client implementation
- Phased conversion
- Identify policies, architectural constraints & page design standards

Web Testing: Strategic Approach

Review and revise test strategy:

- Testing approach validated
- Changes in scope imposed by web conversion
- Challenge to get testers ready for transition

Public Works and Government Services Canada | Travaux publics et Services gouvernementaux Canada | Canada

Basic Objectives of Testing CDFS

- Object and testable units function in accordance with specifications
- Business rules operate as intended including message handling
- Interfaces operate correctly & effectively integrated with other components
- Product conforms to system standards and methodologies

Basic Objectives of Testing CDFS – cont'd

- Design documentation adequately reflects user requirements
- Problems identified before release rolled out
- Application functions properly on all supported platforms

Web Testing: Impact on Objectives

- Platform verification biggest change to scope of testing
- Lab environment established to support multiple platforms

Web Testing: Quality Factors

- Adaptability
- Accessibility
- Auditability
- Dependability
- Functionality
- Integrity
- Interoperability
- Operability
- Performance
- Reliability
- Security
- Usability

Web Testing: Impact on Quality Factors

- Expanded meaning for assessment of certain software quality factors:
  - Accessibility
  - Reliability
  - Usability

Web Testing: Impact on Quality Factors

Accessibility - all users can access the application

- Common Look and Feel standard which defines what a web page should look like
- World Wide Web Consortium definition of accessibility

Web Testing: Impact on Quality Factors

Reliability - processes perform without failure

- Number of supported browser/operating system combinations

Web Testing: Impact on Quality Factors

Usability - easy to operate, prepare input & interpret output

- Page rendering dependent on browser version & workstation settings
- Verified recommended workstation browser settings in test lab

Web Testing:
Problem Management

Traditional Process:

- Problem Management Report (PMR) raised for all problems and tracked through automated system
- Raised by anyone who interacts with system
- Tracked through 5 level problem analysis process

CDFS Problem Management Report

```
===>

SSC08010                  Summary Display                 Problem: 01307322

Reported by..... JIBB/NANCY
Reporter group..... GOSFRPG   Phone... 819-956-6567       Dept.... PWGSC
Assignee name..... JIBB/NANCY
Assignee group..... GOSFRPG   Phone... 819-956-6567
Date occurred..... 2002/04/03              Problem type..... APL
Problem status..... CLOSED                 Foundation..... DB2
Business impact..... 4                     Application..... CDFS
Plant impact..... 4                        Environment..... DTST
Current priority.... 4                     Plant outage..... NO
System name..... MVS1                      Tracked by..... NASMF
Location code..... PDP/1LA2                Y2K related?.....
Description..... WRONG ERROR MSG FOR WEB ACCRUAL & COMMITMENT
Impact.....

     Select one of the following, or reply END or CANCEL to exit display
 1. Reporter data.     5. Freeform text.          9. Similar PMRs.
 2. Status data.       6. Additional data.       10. PMRs by: JIBB/NANCY
 3. Resolution data.   7. Administration data.   11. Record utilities.
 4. Display history.   8. Resource tracking.     12. Process Tracking

                                                                    01/07
Connected.   MVSZ  TN3270 mvsz.pwgsc.gc.ca  23  TCP00360  A01A
```

Web Testing:
Problem Management

New process:

- Internal ‘Bug’ report to avoid unnecessary duplication of reporting
- Ensure ‘bugs’ tested across multiple platforms
- Test team discusses with technical analysts to establish if real application problem
- PMR raised subsequently, if necessary

Bug Report

Tested by/User I.D.:	Description of Script:
Date of Testing:	Tested in Standard screen:
Database:	Configuration using:
System:	- Browser & Version
Environment:	- Operating System
Object being Tested:	- Browser language
Language in Application:	Description of problem:
Test Case/Script Number:	Action/PMR Number:

Web Testing: CDFS Web Testing Approach

Four levels of testing supported by separate detail test plans:

i. Object Testing

ii. Integration Testing

iii. Interface Testing

iv. Regression Testing

Test Script

Test Script number & description	Additional criteria	Expected result	Test Result
11.1.1.1 Client options-EAA options-Section 32-N; User EAA ind. - N	No EAA applet download request nor pop up for EAA - PAN screen	04084 Commitment @1 has been approved	
11.1.1.2 Client option-EAA options-Section 32-Y; User EAA ind. - N	No EAA applet download request nor pop up for EAA - PAN screen	04084 Commitment @1 has been approved	

Web Checklist

Presentation Checklist		Check Mark
1. Aesthetic Conditions:		
1	Are the section headings the correct consistent colour?	
2	Are the menu bar and session info field the correct consistent colour?	
3	Are all the section headings in the consistent font?	
4	Is the text in all fields consistent with the font?	
5	Are all the section headings aligned on the screen?	
6	Are all the input boxes and displayed values aligned on the screen?	

Lab Testing Schedule

Platform Schedule

Browser	Time slot	Machine	Day 49 Jul 2 Tester	Objects
Netscape	TS01	M001	Man Fong	
Netscape	TS01	M002	Simon	
Netscape	TS01	M003	Kumar/Nagma	
Netscape	TS01	M004	Ivo	
Netscape	TS01	M005	Nancy	
Netscape	TS01	M006	Jean-Yves	
Netscape	TS01	M007	Long/Nagma	
Netscape	TS01	M008	Loubna	
Internet Exp.	TSAM1	M000	Leo	
Internet Exp.	TSAM1	M000	Xiao	
Internet Exp.	TSAM1	M000	Sameera	
Internet Exp.	TSAM1	M000	Vo Jing	

The Human Factor

Greatest Challenge:

– Having the right people
– Ensuring necessary skills
– Testing with confidence

The Human Factor

Management Response:

- Finding appropriate training
- Empowering staff – participate in finding solutions

Testing a Large Scale

Web Application

Prepared By: Peter Lafleur
Manager, CDFS Development Division
Financial and Reporting Products Directorate
Central Accounting and Reporting Sector
Government Operational Services
Public Works and Government Services Canada

Background

Public Works and Government Services Canada

Public Works and Government Services Canada (PWGSC) is the Government of Canada's major common service organization. It is our job to help keep the wheels of government turning as smoothly as possible. We are builders and buyers, architects, engineers, auditors, interpreters, translators, management consultants, environmental scientists, real estate professionals and telecommunications specialists.

The department is divided into branches that offer services both to the department itself and to other Government departments. One of the common services offered by PWGSC is the Common Departmental Financial System (CDFS) which is an endorsed, shared financial system of the Canadian federal government and is used by twenty departments and agencies.

The Common Departmental Financial System (CDFS)

CDFS is a comprehensive on-line departmental financial management and reporting system designed to simplify and streamline financial administration.

A team of government financial specialists developed CDFS jointly with a private sector consortium of systems and accounting experts - *specifically* for government use. The system was developed in the early 90's following Object Oriented Design techniques as a multi-layered Client/Server Application. The current design implementation supports two layers in the client domain primarily running on client LANs and two layers in the server domain running on a PWGSC mainframe. The client layers include the presentation layer that contains the code for all screens and the process layer that handles message routing, formatting and routine data editing.

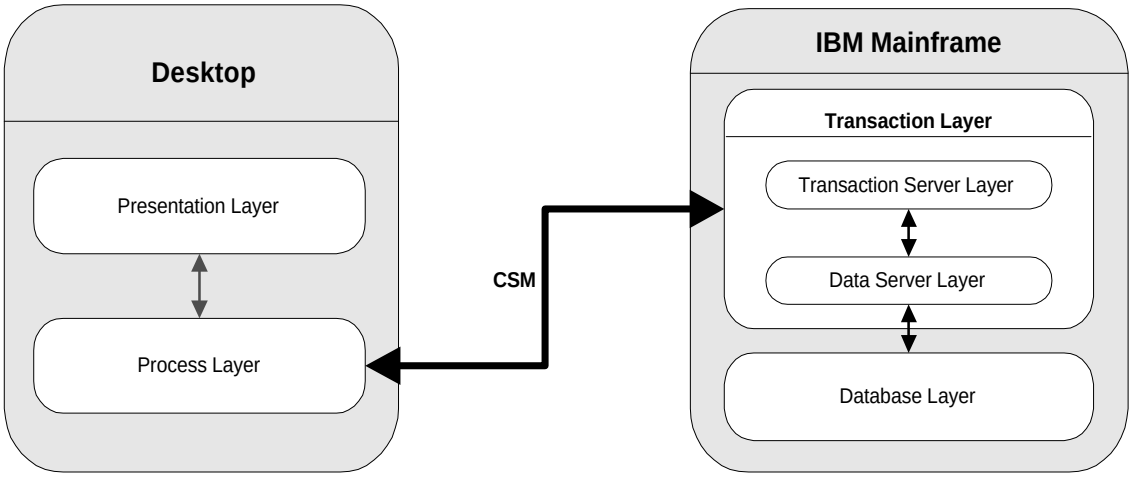


Figure 1: CDFS Standard Architecture

CDFS is one of the largest applications run on PWGSC infrastructure with over four thousand programs, one hundred and thirty objects and nine hundred English screens with an equal number of French screens

Web-enabling CDFS

Scope

The web enabling of the Common Departmental Financial System (CDFS) is a joint effort involving three PWGSC organizations – Financial and Reporting Products Directorate (FRPD), Application Management Services (AMS) and Network and Computer Services (NCS). FRPD has product management responsibility for CDFS and is responsible for the day-to-day operation of the system, operational planning, client support, system requirement definition and application quality assurance. CDFS is managed by FRPD with all changes fully documented and compiled in one of the two full system releases that are done each year. NCS provides the technical infrastructure, electronic communications infrastructure and support, data center support and web support services, AMS supplies architecture development, design analysis and programming services.

The responsibility for testing CDFS is split across the three partnering organizations. AMS is responsible for program code testing, unit testing and platform certification. NCS is responsible for operational testing and certification. FRPD is responsible for functional testing, including functional testing on multiple platforms. This paper focuses on the FRPD testing experience.

Overview

The current user interface for CDFS employs WinDOS based screens that simulate a windows application. When CDFS was originally developed Windows was in its infancy and the decision was made to go with DOS based screens. The presentation layer was later modified to accommodate the Windows operating systems. Although the application has very robust functionality the presentation layer remains a major sore point with our current clients and is the number one obstacle cited by prospective new clients.

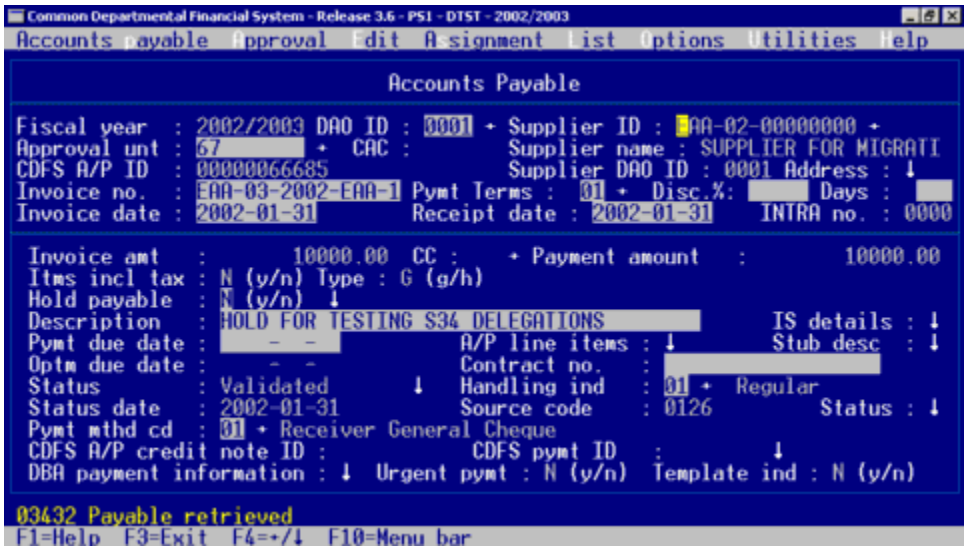


Figure 2: CDFS Standard Accounts Payable Screen

The decision to convert the Client component to the web allows us to redesign the user interface and take advantage of the increased layout and formatting capabilities of web pages. This also lessens the burden of finding qualified programmers to support the application.

Figure 3: CDFS Web Accounts Payable Screen

Conversion to the web also addresses a number of problems with the current distribution of the client component within the client environment. The client component is distributed to clients by CD or by NAL object to client LANs with each release of CDFS for simultaneous installation. Simultaneous installation is required since all clients use the

same version of CDFS. Rather than having to co-ordinate installation across the country users will access the same address when a new version is released.

The decision to web-enable CDFS was made two years ago. Faced with a number of implementation options we decided to establish a multi-year project and treat this as a technological conversion rather than a re-design of the application. Although moving to the web has an obvious impact on architecture the same basic architectural approach is being implemented for web as was implemented for CDFS Standard (non-web). The presentation and process layers will reside on the web server while the transaction and database layers reside unchanged on the host server.

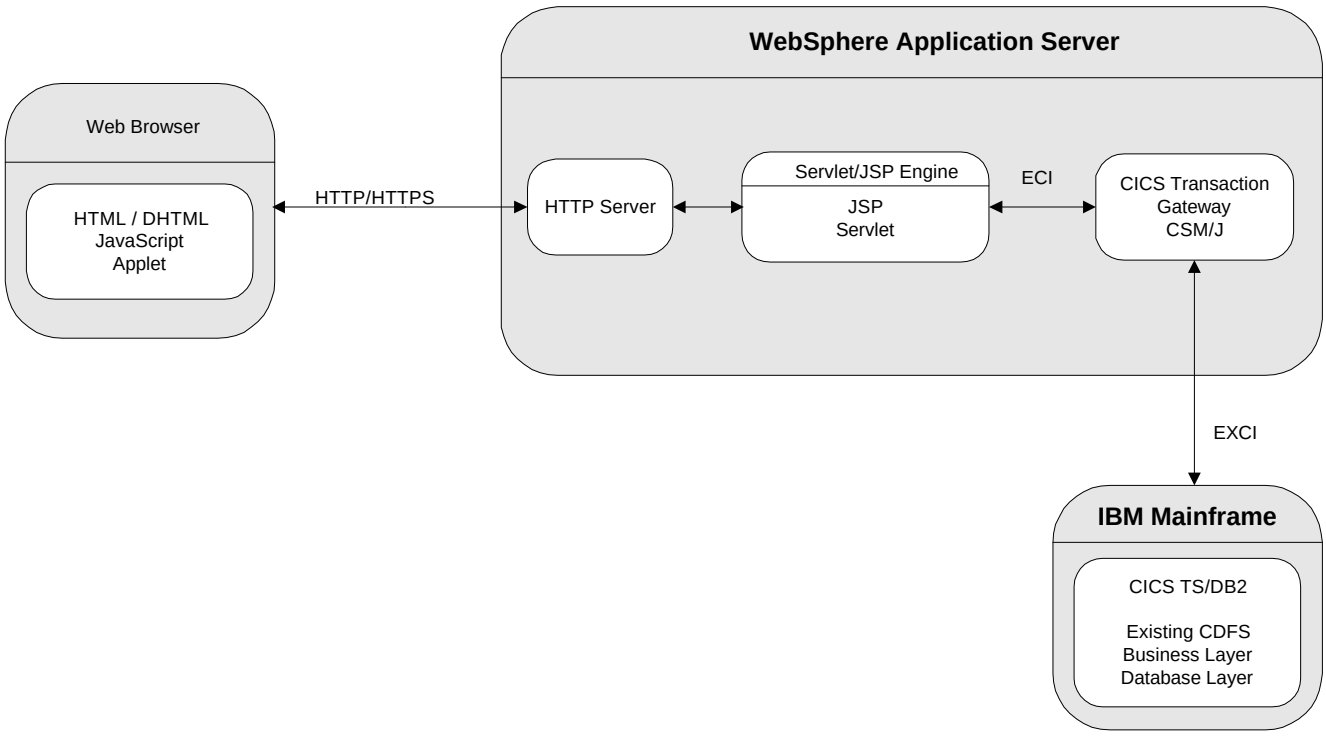


Figure 4: CDFS Web Architecture

Web Testing

Strategic Approach to Testing

Collectively the management and staff of FRPD have extensive experience managing large systems and development projects. Since testing of CDFS is based on both the defined change requirements and the revised design, we recognized the importance of reducing our testing domain through strategic decision-making in our web design. Based on industry experience and recommendations from external consultants we chose a thin client implementation. The client layers reside on a PWGSC web server connected to the host server. The presentation layer and process layer will be rewritten for the web with all data edits removed but will use the same messages for communicating with the server as the standard version. This limits the degree of change

and minimizes the risk and the scope of testing since two of our four layers will remain unchanged.

We also chose to implement this conversion on a phased basis by converting linked objects in groups allowing for a systematic conversion geared towards the majority of end-users. Conversion of the first twenty-five objects should satisfy the daily, operational needs of ninety percent of our users. A phased conversion also allows us to limit the extent of testing in each phase to manageable volumes.

During the design phase we identified policies, architectural constraints and the need for page design standards, all in support of limiting test scope. This includes adherence to corporate standards such as the corporate Common Look and Feel Policy as well as the development of a Functional Standards Guide and a Web Application Architecture document specific to CDFS Web. These documents provide the basis for designing our test cases and standard test checklists.

Recognizing that moving to the web would have serious impacts on both our resources and our testing approach we undertook the review and revision of our testing strategy to identify the impact of web conversion on our existing testing approach and highlight the changes in scope imposed by web conversion. We found that our basic test objectives had not changed but our role in testing against certain objectives had expanded.

Finally, we took a strategic approach to preparing our business analysts and testers to begin testing the converted objects. We identified a core group of business analysts to participate in the conversion process from the design phase through to testing. We also identified and provided the necessary training to ensure they would be prepared before testing began.

Objectives and Quality Factors Affecting Testing CDFS

The basic objectives of testing CDFS, established during the development of CDFS Standard, are:

> object and testable units function in accordance with specifications;
> business rules operate as intended including meaningful message handling;
> interfaces operate correctly and are effectively integrated with other components;
> design documentation adequately reflects user requirements;
> the product conforms to system standards and methodologies;
> any problems are identified before the release is rolled out; and
> the application functions properly on all supported platforms.

The greatest impact resulting from web conversion was on platform verification. We had traditionally viewed platform testing as strictly a technical item with platform testing as the responsibility of AMS and NCS. All FRPD testing was done on our regular workstations on a common platform.

The decision to support both Netscape Communicator and Internet Explorer as browsers and versions of Windows from 95 up to, but not including XP, necessitated a change in

this view. This was further complicated by the requirements of a bilingual system – CDFS supports both English and French implementations. The responsibility for final product quality resides with FRPD so we now conduct platform testing and have established a lab environment that supports multiple platforms .

We also identified the impact of web conversion on the software quality factors we currently assess for CDFS Standard:

Adaptability	- functions can be enhanced without major design rework
Accessibility	- all users can access the application
Auditability	- tables present all access and transactions
Dependability	- users can trust results provided
Functionality	- software performs as per specifications
Integrity	- data is stored and returned unaltered
Interoperability	- data can be imported or exported properly
Operability	- software can easily be put into operation- not complex
Performance	- intended functions are performed within predefined time limits
Reliability	- processes perform without failure
Security	- the system and data can only be accessed by authorized users
Usability	- easy to operate, prepare input and interpret output

The factors of accessibility, reliability and usability take on expanded meaning under a web conversion. Accessibility is impacted by the Common Look and Feel standard which covers both what a web page should look like and defines accessibility in line with the World Wide Web Consortium definition . Reliability and usability are impacted by the number of operating system/browser combinations that are supported and by the page rendering control that is given up to the browsers. The way in which a page is rendered is dependant not only on the user's browser version but also on individual workstation settings which are at the total control of the end-user. In order to minimize the impact of this loss of control we developed recommended workstation browser settings for our clients which we verified in our test lab.

Problem Management

Problem management is also affected by web conversion. Traditionally we identify problems with CDFS in testing or in production via a problem management report (PMR) in INFOMAN which is our departmental, automated change and problem management system.. A PMR can be raised by anyone who interacts with the system as soon as it is discovered. The problem is then tracked and reported on through a structured five level problem resolution process.

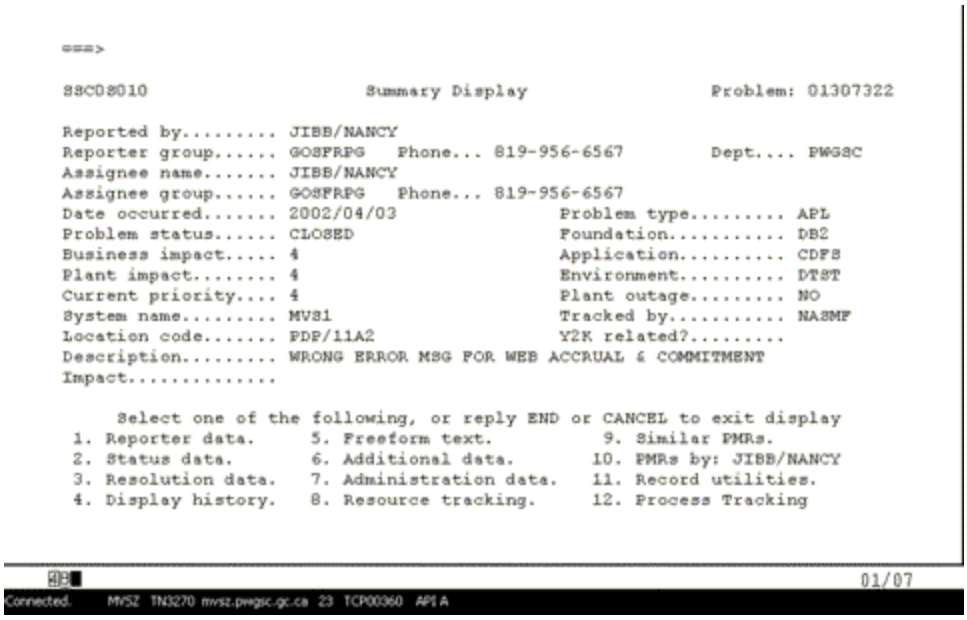


Figure 5: Sample Problem Management Report

We supplemented this process with internal "Bug" reporting to avoid unnecessary duplication of problem reporting and to ensure bugs are tested across multiple platforms. Initially problems are identified as bugs by our business analysts (testers) which are reviewed by senior business analysts who subsequently review the bugs and determine if they are really different or if multiple reports identify the same problem. The senior analysts also identify any supplemental testing that may be needed. On a weekly basis the entire test team meets with our technical analysts to review and discuss these bugs to determine if a real application problem exists or if we are dealing with browser limitations. Once the bugs have been reviewed and real problems are identified we raise PMRs which go through the normal problem support process.

Tested by/User ID:	
Date of testing:	
Database:	
System:	
Environment:	
Object being tested:	
Language used in application:	
Test case/script number:	
Description of script:	
Tested in Standard Screen:	
Configuration using:	
Operating System:	
Browser & Version:	
Browser Language:	
Description of Problem:	

Severity:	
Reviewed by:	
Action/PMR Number:	
Date:	

Figure 6: Sample Bug Report

CDFS Web Testing Approach

Our testing strategy identifies four distinct levels of testing each supported by a project plan and detail test plans. The testing project plan is maintained by a testing co-ordinator and is used to control testing progress and feeds into our high level release plan. The detail test plans are developed by senior business analysts based on system design documentation and, in the case of web conversion, on the existing CDFS Standard screens.

Although the four levels of testing are generally carried out at different times in our test window they can be done simultaneously. Each level – Object, Integration, Interface and Regression – has a separate focus and separate test plans.

Object Testing

Object validation is the first level of acceptance testing. It is the software quality assurance validation of basic object functionality to ensure that the object is operating as the functional design documentation specified. Object testing is done as much as possible in a stand alone environment. Object testing validates all four layers of each object on both web and non-web versions.

Examples of object testing include:

- verification that screens and pull down menus appear as per specifications;
- verification of action availability;
- verification that lists called are appropriate;
- verification of the format/update of date fields;
- verification of maximum field lengths; and
- verification that messages are valid and correct.

Web page format and content are tested for compliance to standards. Additionally, all business edits have been moved from the process layer to the transaction layer so object tests are used to confirm that the edits have been successfully moved to the transaction layer.

Changes to the database are initiated by messages from the transaction layer. The updates to the database layer are verified by comparing the updates made using the web screen

for input and updates made using the standard screen for input. The object tests performed on the standard screens and the same object test performed on the web page must update the tables in the same fashion.

Presentation Checklist		~~Check Mark~~Checkmark
1 Aesthetic Conditions:		
1	Are the section headings the correct consistent colour?	
2	Are the menu bar and session info field the correct consistent colour?	
3	Are all the section headings in the consistent font?	
4	Is the text in all fields consistent with the font?	
5	Are all the section headings aligned perfectly on the screen?	
6	Are all the input boxes and displayed values aligned perfectly on the screen?	
7	Are all field literals aligned correctly on the screen?	
8	Can the screen be minimisable?	
9	Can the screen be resizable?	
10	Are all the field literals spelt correctly?	

Figure 7: Presentation Layer Object Checklist

Integration Testing

Integration testing includes testable unit and system wide testing. Testable unit or functional testing is the testing of an integration of CDFS objects to form business functions and sub-functions. A testable unit is made up of several objects which combine to form a discrete business process, such as raising an accounts payable and issuing a payment, and verifies that the process performs in accordance with the functional design specifications and responds correctly to all conditions presented by data flows.

System wide testing is designed to test the integration of the transactions from the beginning of a processing cycle to the end of the processing cycle. In the case of web conversion this validates the integrity of data entered using web objects and processed further by non-web objects.

Test Script number & description	Additional criteria	Expected result	Test Result
11.1.1.1 Client options-EAA options-Section 32-N; User EAA ind. - N	No EAA applet download request nor pop up for EAA - PAN screen	04084 Commitment @1 has been approved	
11.1.1.2 Client option-EAA options-Section 32-Y; User EAA ind. - N	No EAA applet download request nor pop up for EAA - PAN screen	04084 Commitment @1 has been approved	

11.1.2.1 Client options-EAA options-Section 32-N; User EAA ind. - N	No EAA applet download request nor pop up for EAA - PAN screen	04084 Commitment @1 has been approved	

Figure 8: Sample Test Script

Interface Testing

Interface testing validates the inter-operability of CDFS with its partnering systems. CDFS provides multiple interface options including batch, on-line and message file transfers. The message file transfer capability is included in the first phase of web conversion and is accomplished using a signed applet. The testing of signed applets adds a new dimension to our existing tests since they represent true web functionality for distributing small programs. We now have to assess both the functionality provided by the applet and its impact on user workstations.

Regression Testing

Regression testing is testing of previously verified functionality ,to ensure that the process has not been impacted by other system changes. Regression testing is primarily conducted using using automated testing tools. FRPD uses two different automated test tools -AutoTester and Rational Robot. Both automated systems support web testing but new web shells and scripts are required. The benefits of these automated tools will be seen in subsequent releases of CDFS.

Platform Coverage

In testing CDFS we consider a "Platform" to be the "Hardware" and "Software" environment on which an application runs. The CDFS Web application must operate on all supported platforms including operating system/browser combinations in both English and French. Platform testing is carried out in a lab environment on a scheduled basis.

There are no individual testplans for platform coverage but a testing schedule is developed to ensure that sufficient coverage of each supported platform takes place. Platform testing is conducted continuously throughout our test window with target coverage determined by a survey of client platforms.

	Platform schedule			
Browser	**Time slot**	**Machine**	**Day 49**	
			Jul 2	
			Tester	Objects
Netscape	TS01	M001	Man Fong	
Netscape	TS01	M002	Simon	
Netscape	TS01	M003	Kumar/Nagma	
Netscape	TS01	M004	Ivo	
Netscape	TS01	M005	Nancy	
Netscape	TS01	M006	Jean-Yves	
Netscape	TS01	M007	Long/Nagma	
Netscape	TS01	M008	Loubna	
Internet Exp	TSAM1	M000	Leo	

Figure 9: Sample Platform Testing Schedule

The Human Factor

Conversion to the web presents a number of testing challenges to large scale applications. The need to review testing strategies, limit scope and assign responsibility for testing addresses some of these challenges however, the greatest challenge is ensuring that you have the right people with the necessary skills who can test with confidence. A major technological change can be intimidating even for highly experienced testers. In FRPD we have a great deal of experience testing applications and our techniques and approaches are equally valid in a web environment but moving to the web brings with it a whole new vocabulary, new technology and new perspectives on testing.

We knew our senior analysts were the right people to lead testing, but we needed to address the vocabulary, technology and testing perspectives issues. We had to ensure our analysts had access to the training and information they needed to master these new concepts. This was by far the most difficult aspect of conversion because when we started the process all the training we could find related to writing HTML or managing small web sites. We sent all of our staff, including management, on web familiarization courses that alleviated some, but not all ,of our concerns.

Fortunately, we found a series of workshops offered by AMIBUG Inc. that covered web testing based on real life experiences. We worked with Robert Sabourin of AMIBUG to adapt the workshop material and provided this training to all of our managers and business analysts. The result has been successful completion of our first round of testing with confident testers who understood what had to be tested and how to perform the testing.

Providing proper training and the opportunity to participate and learn fr om the design phase through to testing empowered our people to adapt as they

encountered new issues and problems and share their knowledge in finding solutions.

Conclusion

On July 02, 2002 we successfully completed testing of our first web objects. We will be running a production pilot in October with three of our clients and plan to go to full production mid-November. Unfortunately, at the time of writing this paper we have not conducted a lessons learned session but are planning one for late July. Based on this round of testing I expect there will be some interesting comments from our business analysts that will enable us to streamline and refine our testing approach in time for the next round of testing.

QW2002 Paper 3A1

Dr. Erik Simmons
(Intel Corporation)

Software Defect Arrival Modeling With the Weibull Distribution

Presentation Abstract

One of the most common yet vexing questions asked of Software Quality Assurance managers and testers is "When will we be done testing?" Product engineering and marketing groups have a vested interest in knowing when the software under test will be at an acceptable level of quality. While this question is not at all easy to answer, modeling the arrival of defects during testing can provide clues such as predictions of when a given percentage of the estimated total defects will be found, or the time at which the rate of newly arriving defects will be below a given threshold. The Weibull distribution serves as an excellent model for software defect arrival. Three case studies based on actual projects are provided.

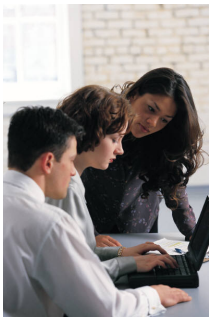
About the Author

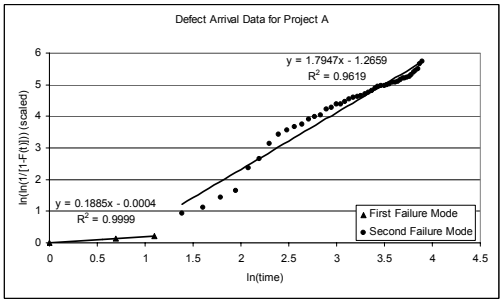
Erik Simmons has 15 years experience in multiple aspects of software and quality engineering. Erik currently works as Platform Quality Engineer in the Platform Quality Methods group, part of the Corporate Quality Network at Intel Corporation. He is responsible for Requirements Engineering practices at Intel, and lends support to several other corporate software and product quality initiatives. Erik is a member of the Pacific Northwest Software Quality Conference Board of Directors and the Steering Committee of the Rose City SPIN. He holds a Masters degree in mathematical modeling and a Bachelors degree in applied mathematics from Humboldt State University in California, and was appointed to the Clinical Faculty of Oregon Health Sciences University in 1991.

When Will We be Done Testing?

Software Defect Arrival Modeling With the Weibull Distribution

Erik Simmons, Intel Corporation





Version 1.1
7/02

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

For information, contact:
Erik.Simmons@Intel.com

Contents

1. The Weibull Distribution
2. Software Defect Arrival Modeling – the Basics
3. Fitting the Weibull Distribution to Defect Arrival Data
4. Case Studies
5. Sources for More Information

Version 1.1
7/02

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

The Weibull Distribution

Discovered in 1937 by Waloddi Weibull

Used since the 1950s to model diverse things:

- •Hardware failures
- •Radar clutter
- •Warranty & support costs
- •Spare parts levels
- •And many more…

The two-parameter Weibull:

$$F(t) = 1 - e^{-(t/\eta)^{\beta}}$$

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

The Weibull Distribution

The parameters of the Weibull distribution:

β - the Shape parameter

η - the Characteristic Life

$\beta < 1$ indicates 'infant mortality', where the longer the system runs the less likely failure becomes

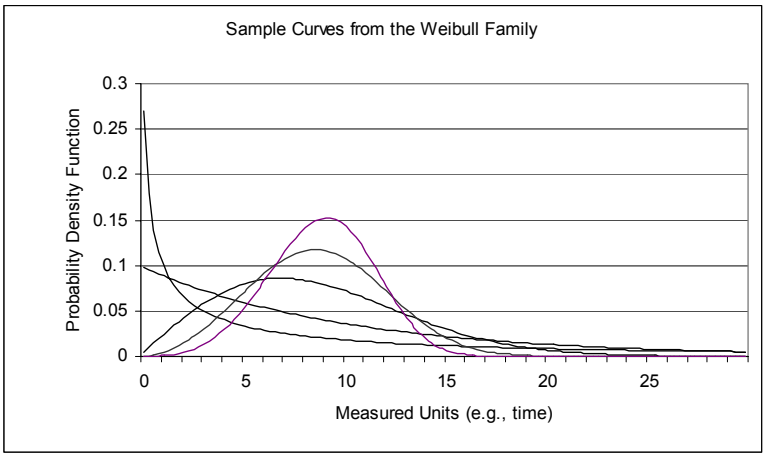
When $\beta = 1$, the Weibull reduces to the Exponential distribution, implying random failure

$\beta > 1$ indicates wear out, where the longer the system runs the more likely failure becomes

η is the point at which 63.2% of the failures have occurred

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

The Weibull Distribution



Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Software Defect Arrival Modeling

Failure model classifications [Lyu95]:

- Failures per time period
- Time between failures

Time domains [Musa87]:

- Execution time
- Calendar time

The execution time domain is generally recognized as superior to calendar time, but can be harder to measure

This paper uses the number of failures per calendar week

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Weibull Model Assumptions

1. Testing occurs in a way that is similar to the way the software will be operated
2. All defects are equally likely to be encountered
3. All defects are independent
4. There is a fixed, finite number of defects in the software at the start of testing
5. The time to arrival of a defect follows a Weibull distribution
6. The number of defects detected in a testing interval is independent of the number detected in other testing intervals for any finite collection of intervals

Luckily, the Weibull is robust to most violations…

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Fitting the Weibull Distribution

The steps to fit the two-parameter Weibull distribution to defect arrival data are:

1. Obtain an estimate of the number of defects in the software
2. Calculate the cumulative proportion of total defects arriving each period
3. Transform the data to obtain a linear form
4. Fit a least-squares line to the data
5. If the fit is acceptable, use the line to obtain estimates of β and η.
6. Plot the Weibull distribution versus the actual data

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Sample Data

Week	New Defects
1	5
2	10
3	27
4	82
5	94
6	61
7	77
8	111

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Estimate the Total Number of Defects

An estimate can be derived in several ways:

- Historical data
- Commercial software
- LOC → Function Points → Defects
- Etc…

If the estimate is off significantly, you will see it in the plots as time goes on

The estimate can be revised during the process

You can also use a high and low estimate and plot more than one model

Sample data estimate: 1200 defects

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Calculate the Cumulative Proportions

Week t	New Defects	Cumulative Proportion F(t)
1	5	0.004
2	10	0.013
3	27	0.035
4	82	0.103
5	94	0.182
6	61	0.233
7	77	0.297
8	111	0.389

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Transform the Data to a Linear Form

The two-parameter Weibull can be re-expressed in a linear form:

$$F(t) = 1 - e^{-(t/\eta)^{\beta}}$$

$$\ln\left(\ln\left(\frac{1}{1-F(t)}\right)\right) = \beta\ln(t) - \beta\ln(\eta)$$

Y = mX + B

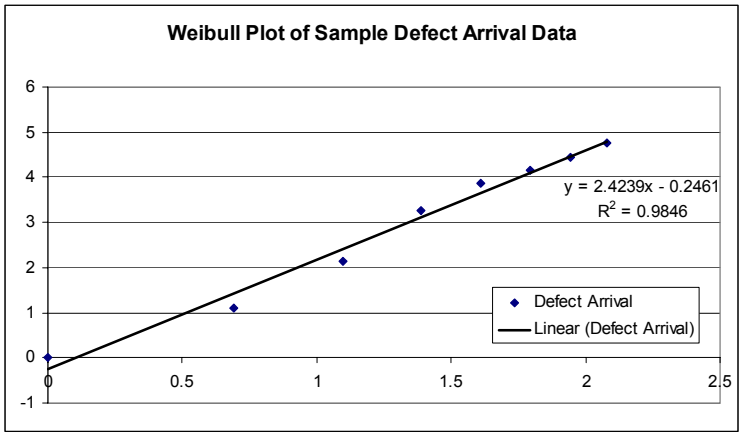
Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Transform the Data to a Linear Form

ln(Week)	ln(ln(1/(1-F(t)))	Rescaled
0	-5.47855	0
0.693147	-4.37574	1.102808
1.098612	-3.33465	2.143905
1.386294	-2.21576	3.262797
.1609438	-1.60701	3.871539
1.791759	-1.32947	4.149070
1.94591	-1.04434	4.434213
2.079442	-0.70739	4.771166

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Fit a Least Squares Line to the Data



Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

If the Fit is Acceptable, Estimate β and η

The R^2 for the line is .9846, so about 98% of the variation in the data is explained by the line

β can be read from the regression equation as the slope: 2.4329

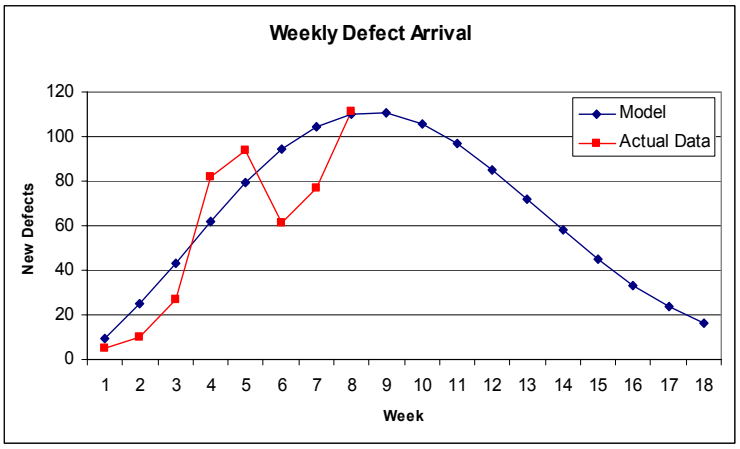
η can be calculated by solving the following equation (remember that we rescaled the data):

$$\ln\left(\ln\left(\frac{1}{1-.632}\right)\right)+5.47855=2.4239\ln(\eta)-.2461$$

Solving, we get η = 10.6 weeks

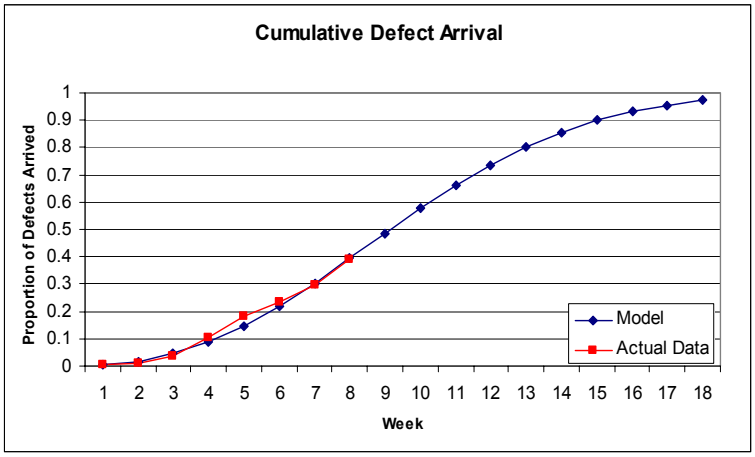
Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Plot the Model vs. the Actual Data



Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Plot the Model vs. the Actual Data



Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Case Studies

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Project A

Two-tiered client/server application

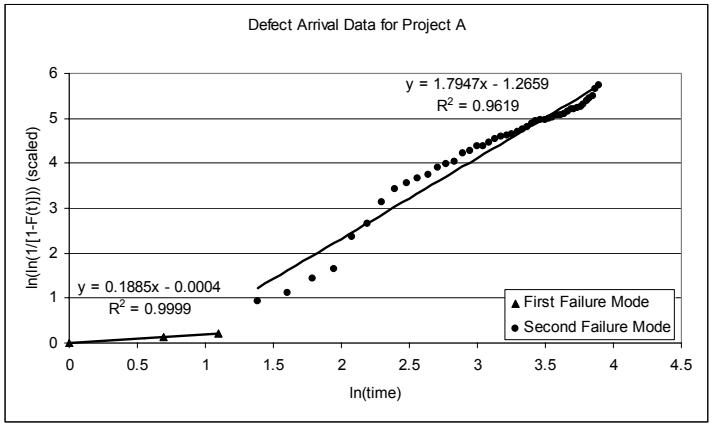
4 developers, 1 tester

Manual testing according to a written plan

Weibull fit using 1008 located defects

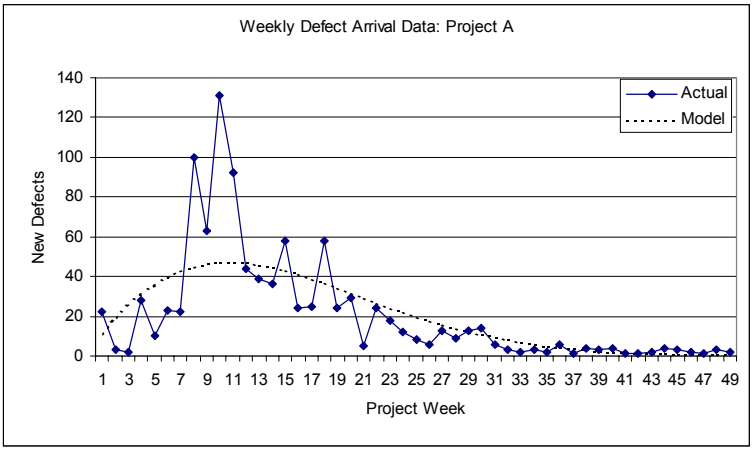
Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Project A: Weibull Plot



Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Project A: Weekly Arrival



Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Project B

Web-enabled three-tiered client/server application

8 developers, 1-2.5 testers

Manual testing according to a extensive, inspected plan

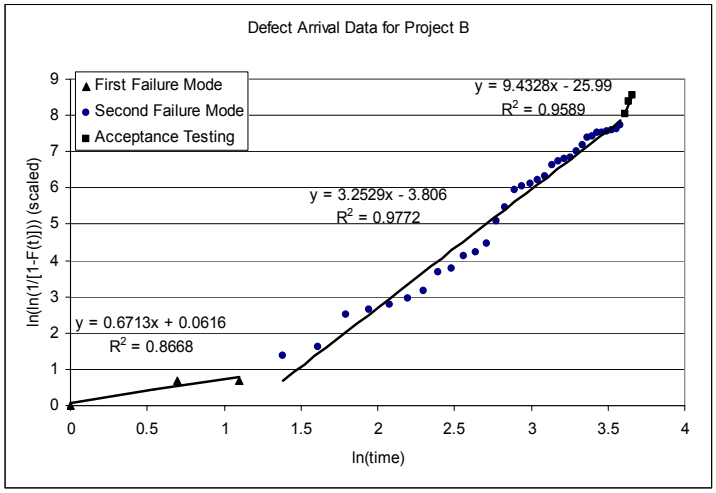
First Weibull fit to 1000 estimated defects

Total defect estimate trimmed to 800 near the end

Final fit to 739 defects

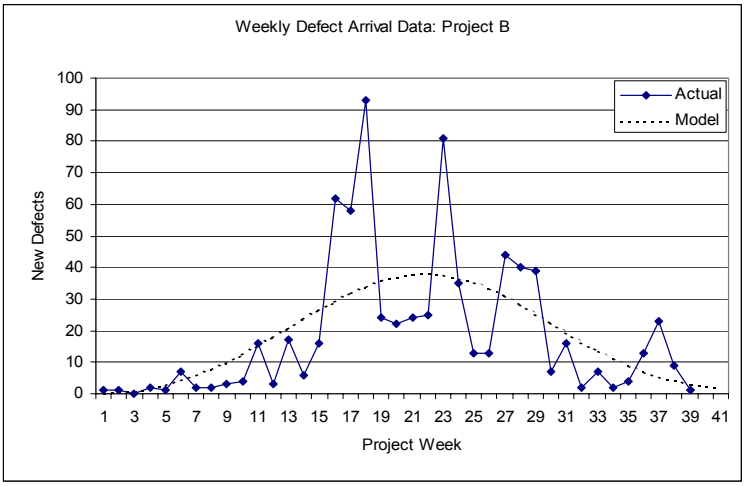
Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Project B: Weibull Plot



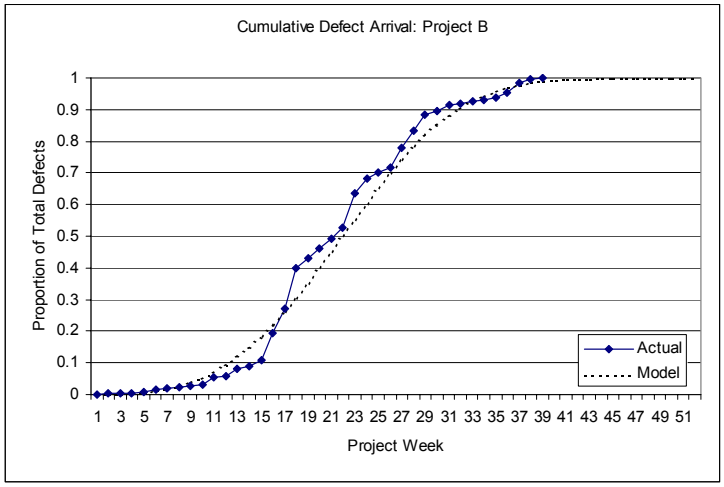
Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Project B: Weekly Arrival



Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Project B: Cumulative Arrival



Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Project C

Traditional two-tiered client/server application

12 developers, 4-7 testers

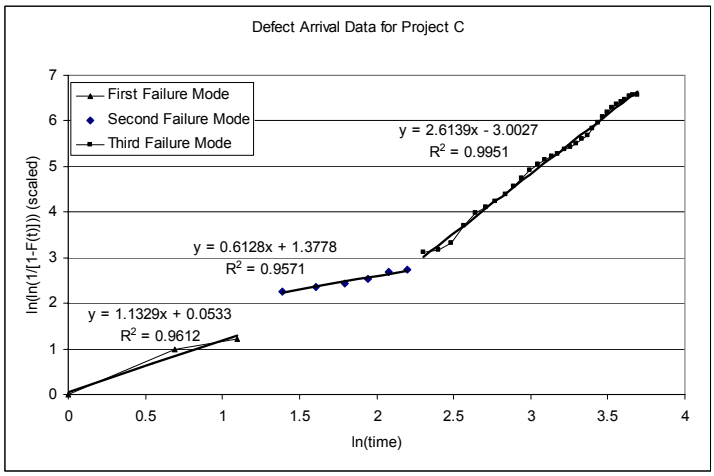
Mix of mostly manual and some automated testing according to extensive plans and specifications

First Weibull fit to 2000 estimated defects

Total defect estimate raised to 2500 during testing

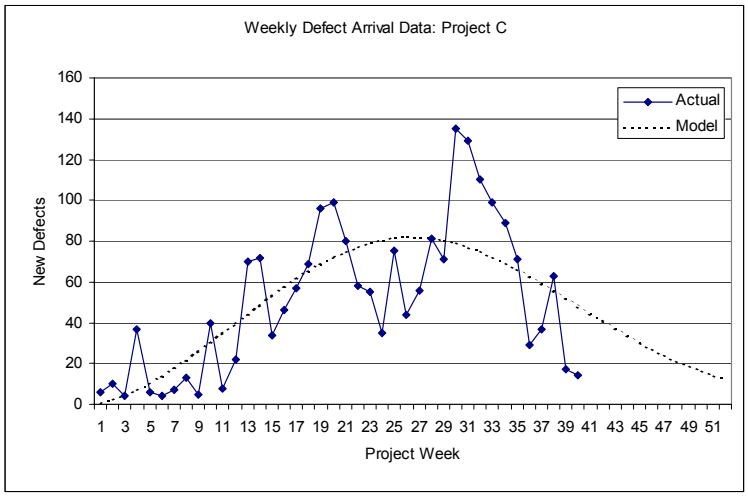
Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Project C: Weibull Plot



Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Project C: Weekly Arrival



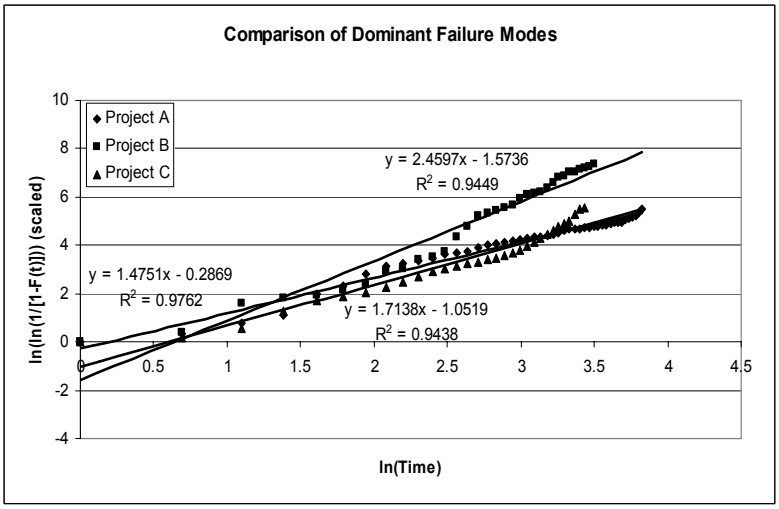
Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Discussion

- Fits appear to be better when non-cosmetic defects are used
- 'Infant mortality' failure is common at the beginning of testing efforts
- Use care when comparing β parameters between projects because of nonlinear axes
- Estimates appear to be stable enough to be useful only a few weeks after the main failure mode appears
- The project's lifecycle will influence defect arrival patterns

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Comparison of Dominant Failure Modes



Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Possible Future Work

There are three main areas for more work:

1. What factors influence β?
2. Can the need for an estimate of total defects be removed?
3. Are results significantly better with execution time domain and/or time between failures data?

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

References

Abernethy98 — Abernethy, Dr. Robert B., *The New Weibull Handbook, 3rd ed.*, Self-published 1998

Jones97 — Jones, Capers, *Software Quality – Analysis and Guidelines for Success*, Thompson Computer Press 1997

Kan95 — Kan, Stephen H., *Metrics and Models in Software Quality Engineering,* Addison Wesley 1995

Lyu95 — Lyu, Michael (ed.), *Handbook of Software Reliability Engineering,* McGraw-Hill/IEEE Computer Press 1995

Musa87 — Musa, John, *et al., Software Reliability – Measurement, Prediction, Application,* McGraw-Hill 1987

Putnam92 — Putnam, Lawrence, and Myers, Ware, *Measures for Excellence – Reliable Software On Time, Within Budget*, Yourdon Press 1992

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Backup

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Relationship Between MTTF and η

For the Weibull distribution, the Mean Time to Failure is related to the Characteristic Life η by the following formula:

$$MTTF = \eta\Gamma(1+1/\beta)$$

where:

Γ is the gamma function, and
β Is the Weibull shape parameter

If $\beta = 1$, then the MTTF = η (the Exponential distribution)

If $\beta > 1$, then the MTTF < η

If $\beta < 1$, then the MTTF > η

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Derivation of the Linear Weibull Form

$$F(t) = 1 - e^{-(t/\eta)^{\beta}}$$

$$1/(1 - F(t)) = e^{(t/\eta)^{\beta}}$$

$$\ln(1/(1 - F(t))) = (t/\eta)^{\beta}$$

$$\ln[\ln(1/(1 - F(t)))] = \beta \ln(t) - \beta \ln(\eta)$$

Y = mX + B

Copyright © 2000, 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

QW2002 Paper 3A2

Dr. Kelly A. Shaw
(TeamShare, Inc)

Using Defect Tracking Integrations to Improve your QA Process

Key Points

- Integrate defect-tracking with customer support to collect accurate field quality info.
- Integrate defect tracking with version control to predict when you have tested enough.
- Integrate defect tracking with build tool to eliminate unnecessary last-minute changes.

Presentation Abstract

Defect-tracking tools contain a wealth of information that can provide valuable insights into your QA Process. This presentation will show case studies from two companies who used integrations to their defect-tracking tool to measure, evaluate and improve their processes, both within QA, and in other organizations throughout the enterprise.

About the Author

Dr. Shaw has been working in the software development field since 1983, when she was commissioned as a Lt. In the US Air Force. In the Air Force she held positions as a systems programmer, an operations manager, a database programmer and a configurations manager. Upon leaving the Air Force, she completed her Ph.D. in Computer Science before joining Autometric, Inc., first as a senior developer, then as the Technical Director for the Space Applications Division. She joined TeamShare, Inc. as Director of Strategic Alliances in 1999, and has been responsible for proposing and developing integrations between TeamShare's defect tracking and change management tool, and other 3rd party tools.

Using Defect-tracking Integrations to Improve Your QA Process

Kelly A. Shaw
Director of Strategic Alliances
TeamShare, Inc.

What You Will Learn in This Presentation

- How one company improved its entire QA process through an aggressive defect-tracking integration strategy
  - Integrated defect-tracking and customer support tools
  - Integrated defect-tracking and version control tools
  - Integrated defect-tracking and release management tools

What You Will Learn in This Presentation

- How another company put its testers back to work testing by integrating defect-tracking into their company intranet site

Case 1
Managed Objects

Managed Objects is a Business Service Management company helping customers determine how their technology infrastructure affects business productivity.

Case 1
Managed Objects

- Many high profile customers fueled rapid company growth
- Software product formula® central to their business strategy
- Challenge: continue to deliver formula as a quality product to their customers as managed objects grew and changed

Case 1
Managed Objects

- QA process improvement goals
  - Find 95% of all severe software defects in the lab rather than in the field
  - Stop last-minute unnecessary software changes
  - Use benchmarks to determine when the software is ready for release

Case I
Managed Objects

- QA process improvement goals
  - Continue to provide high quality customer support while moving many support functions from engineering to a dedicated support center
  - *Do all this without overburdening existing QA resources*

Find Defects in the Lab, Not in the Field

- IEEE standards state that 95% of all defects in top quality software should be found in the lab rather than in the field
- *Very few* software products meet this rigorous standard, in many cases because the defects are out of the control of the software vendor
- Integrated Customer support and defect-tracking tools helped Managed Objects drive toward this goal

Find Defects in the Lab, Not in the Field

- Integrated Customer support and defect-tracking tools helped Managed Objects drive toward this goal.
  - Gather reliable and accurate information about defects found in the field.
  - Perform a root cause analysis of all defects found in the field.
  - Determine where in the lifecycle the defect should have been found. (Requirements? Design? Code? Documentation?)

Find Defects in the Lab, Not in the Field

- Integrated Customer support and defect-tracking tools helped Managed Objects drive toward this goal.
  - Calculate the ratio of defects found in the lab vs. Found in the field to determine if quality goals have been met.
  - If the goals have not been met, take actions to improve the software quality in the next release.

Stop Last-minute Unnecessary Software Changes

- Many defects found in the field are a result of last-minute and unnecessary changes
- Managed Objects QA wanted tight controls over all software changes at the end of the development cycle
- Defect-tracking integration with a release management tool helped them meet this goal

Stop Last-minute Unnecessary Software Changes

- Defect-tracking integration with a release management tool helped them meet this goal
  - At the end of the release cycle, QA identifies a set of defects to be included in the release candidate.
  - During source code check-in and check-out, developers identify the defect associated with each code change.
  - At build time, only code changes associated with “approved” defects will be included in the release candidate.

Decide When the Software Is Ready for Release

- Over-testing wastes valuable QA resources
- Under-testing enables the release of software that does not meet quality goals
- Managed Objects wanted to use measurable statistics to find the “sweet spot” between under and over testing
- Integrations to their customer support and version control tools helped them achieve this goal

Decide When the Software Is Ready for Release

- Integrations to customer support and version control tools helped them achieve this goal
  - Integrated customer support and defect-tracking tools provide reliable and accurate information about bugs found in the field.
  - The defect-tracking tool provides reliable and accurate information about bugs found in the lab.
  - Integrated source code control and defect-tracking tools provides reliable and accurate information about how much code has been added, modified or deleted.

Decide When the Software Is Ready for Release

- Defect density predicts the number of defects that can be expected in a software release. See “Gauging software readiness with defect tracking” *IEEE Software*, vol. 14, no. 3, May/June 1997.
  - Historical defect data can provide information on the expected defect rate per KLOC. (Thousand lines of code)
  - Use the historical data to predict the defect density in new releases.
  - Stop testing and release when you find the expected number of defects

Deploy a Customer Support Center

- Initially, engineering provided customer support
- As the company grew, they needed a department dedicated to supporting the customer
  - Customer support agents needed accurate status on problems escalated to engineering
  - QA needed to see all verifiable problems reported from the field
  - Engineering needed accurate statistics from both customer support and QA

Deploy a Customer Support Center

- Integration between Customer Support and defect-tracking tools made this possible
  - All defects reported to customer support could be automatically promoted to engineering.
  - Customer support could provide accurate and timely status about outstanding tickets
  - Customers could use web reporting to get status about their own open tickets.
  - Customer support receives updates when a ticket is closed by engineering.
  - Engineering could get reliable and accurate defect statistics from a single tool.

Impact on Existing QA Resources

- QA spends less time providing customer support directly to customers
- QA spends less time tracking down KLOC information from Engineering
- QA knows exactly what defects have been changed in each build, and can test accordingly
- QA knows when it needs to test more, and when the software is ready for release

Conclusion

Integrations between defect tracking tools and other software development tools can help you improve your QA process.

However, not all integrations need to be extensive to be effective.

Case 2
Transmeta

Transmeta develops and sells software-based microprocessors, and develops additional hardware and software technologies that enable computer manufacturers to build computers that simultaneously offer long battery life, high performance and x86 compatibility.

Case 2
Transmeta

- Sales and customer care needed defect rate information
- QA needed to spend less time reporting status and more time testing

Case 2
Transmeta

- Business Improvement Goals
  - Provide sales accurate and timely defect rate information as an offensive and defensive sales weapon.
  - Provide customer care with accurate and timely customer-centric defect information.
- QA Process Improvement Goal
  - Get QA back into the QA business.

Case 2
Transmeta

- Solution to achieve all three goals: integrate defect-tracking into the company intranet site.
  - Sales can run real-time reports to determine return rates, or to see returns for a specific customer.
  - Customer care can run real-time reports to determine return rates by customer.
  - QA spends less time helping sales and customer care, and spends more time testing.

Case 2
Transmeta

- An interesting success story
  - Because this information was available to customer care, Transmeta was able to help a customer find a defective piece of equipment in the *customer's* factory that was manifesting itself as defective deliveries from Transmeta.

Conclusion

- Even small integrations can have profound effects on your business.

Other Defect-tracking Integration Ideas

- Project Management
- Test Management
- Requirements Management
- IDE
- Billing

QW2002 Paper 4A1

Mr. Michael Hackett
(LogiGear Corporation)

The mysteries of Unit Testing Explained for Testers

Key Points

- Get a deeper understanding of the Unit, Integration, System lifecycle including activities
- Learn about the use and role of Unit testing
- Move out of UI based testing to test earlier and more focused.

Presentation Abstract

The job of Senior Tester is one that has been somewhat forgotten in the mad rush to get any Tester with experience into a Test Lead role. In the same rush to get software out the door test teams are asked to do more testing in less time. Test automation is usually the first solution proposed. However, test automation of black box test cases is still black box testing. There are other solutions. Testers can test earlier during development, when errors are cheaper to fix, but their skill level and domain of expertise must grow. The goal of this talk is to develop a new approach to Unit testing from the perspective of learning what tests a tester take on, what tests a developer ought to be running, though they may not be, and some tools for testers to accomplish these new testing tasks.

About the Author

Michael Hackett, Vice President, is a founding partner of LogiGear Corporation. He has over a decade of experience in software engineering and the testing of shrink-wrap and Internet based applications. Michael has helped well-known companies including Palm Computing, Oracle, CNET, Electronics for Imaging, Adobe Systems, The Learning Company, Power Up Software, PC World, and The Well produce, test, and release applications ranging from business productivity to educational multimedia titles — in English as well as a multitude of other languages. Michael also teaches software testing for the University of California at Berkeley Extension, the Software Productivity Center in Vancouver, the Hong Kong Productivity Centre and LogiGear University. Michael is a co-author of the soon to be published second edition of Testing Applications on the Web (Wiley 2002). Michael holds a Bachelor of Science in Engineering from Carnegie-Mellon University.

LogiGear™

The Mysteries of Unit Testing Explained for Black Box Testers

Michael Hackett

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Overview

What is the Problem?

I have been a Black box tester for a few years but now I need to:

– Test earlier.
– Test at a lower level
– Test functionality of non-UI code.
– Work closer with my developers.

Or perhaps…I hear we are moving to XP development- what does that mean for me?

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Overview

Overview of a tester's work
Putting Unit Test in that context
Examples of Unit Testing
Tools and a Tool Example
Recommendations

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Overview

Let's start with the basics of what we, as testers do?

- We design test cases,
- execute tests and
- analyze or validate results.

We most commonly think of executing tests by entering input to the system or application under test through the User Interface and getting some kind of output or behavior to evaluate.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Overview

Let's focus on the test execution. We exercise the application or system through some interface. Whether the user interface, line command interface, API (application program interface), registry, data file or some other interface- we need an interface or method to access the program.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 5

Overview

The User Interface built for your application provides the easiest and least technical access to most, but not all, of the program's function.

The UI may not allow you complete access to the application or it may be built late in the development process. Testing a non-trivial application through the UI may be so high-level that when you find a defect isolation may be tedious and very time consuming.

If your test project calls for more low-level, earlier testing or testing the program's abilities that are not intended to accept input through the UI you need to find another method to access the application other than the UI.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 6

Overview

Today we will focus specifically on test execution not using the user interface. More specifically we will focus on executing tests early in product development, often called coding or Unit phase.

The ideas we will talk about today are effective for early development phases whether you call these Unit, Component or Module Test phase.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 7

Unit Testing

What is the earliest or lowest level access I can get to the code of my application?

The unit of code itself.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 8

A survey of definitions

What is a Unit? Is it a module, an object, a dll, a routine, a function?

- The smallest building block. A unit is a coherent set of instructions with identifiable inputs and output
- Any logically distinct part of the program.
- A unit is typically a function, a small collection of functions, a function library, or a class.

IEEE Glossary

www.Whatis.com

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Unit Testing

"Unit testing" is about pieces-individual classes, functions and methods.

"Integration testing," on the other hand, is designed to ensure that the pieces fit together as designed and that the system as a whole behaves correctly under all reasonable conditions.

Reigning in C++ Test Harnesses

SD Magazine

http://www.sdmagazine.com/documents/s=738/sdm0011c/0011c.htm

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Unit Testing

What is Unit testing?

Brian Marick calls it Developer Testing since it requires knowledge of the internal program design and the code itself.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Is Unit testing effective?

- Find errors early
- Easier to isolate defects
- Highly efficient
- Focused concentration on specific functions, methods or modules
- Isolate genuine boundaries
- Isolate all branches
- Isolate transitions between algorithms
- Finer grain measurement (e.g. coverage)
- Able to assess code coverage
- There is no UI so you have to use unit test techniques.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Development Phases

- Unit Test – test one unit of code at a time. *Usually* done by developers.
- Integration Test- checking the units of code work together incrementally.
- System Test- the entire product as it is intended to be used.
- User Acceptance Test- requirements based testing by the project sponsor.
- Release – testing the production or release system
- Maintenance- continued testing of the product maintenance releases or patches.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 13

Unit Tests

Static:

- Tested by default
  - Compilers
  - Compile-time semantic error checkers
- Tool specific (sometimes expensive)
  - Complexity metrics
  - Code path tracing
  - Coverage analyzers
  - Data flow testing (based on data flow tracing)
  - Debuggers
  - Assertion checking
- Tools, Methods (less expensive)
  - Syntax checkers
  - Style checkers
  - Inspections and code walk-through (these are not “testing” because you don’t *execute* the code, but they are important ways to find bugs).

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 14

Unit Tests

Dynamic:

- Unit testing, using stubs & drivers, especially for driving modules through boundaries and error conditions.
- Functional- verify functionality
- Structural - code path and data structure testing.
- Run time testing- looking for memory leaks and more
- Code mutation- inject defects into the code to see if the test group catches it.
- Fault injection

These are the tests for us!

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Bugs you will find Unit Testing

Dynamic unit tests:
run-time errors,
data flow errors,
exception handling and errors,
logic errors.

For Integration Testing:
Problems in the services and behaviors of the constituent system parts.
Incompatibilities and bottlenecks between the dependant components.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Problems that Unit Testing Often Misses

- Timing-related bugs
- Side effects of interrupts
- Unexpected error conditions
- Special data conditions
- Interaction with background tasks
- Invalid onscreen information
- User interface inconsistency
- User interface everything else
- Failure to comply with contract or regulation
- Configuration/compatibility failures
- Volume, load, performance
- Hardware faults

Testing Computer Software

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 17

Unit Testing

Let's take a look at some useful definitions to help us frame this discussion.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 18

Unit Testing Terms

Test Scaffold or Test Framework- A Scaffold or Framework provides an environment or interface to build your harness, drivers, stubs and tests.

Test Harness-A test harness builds, executes and reports the results of tests. Harness simulates the outer context.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

A survey of definitions

Test Driver- An empty function that calls functions under test. It contains just enough code to set up parameters and globals prior calling.

Test Driver is software which executes software in order to test it, providing a framework for setting input parameters, executing the unit, and reading the output parameters

Test Driver-software modules used to invoke a module(s) under test and, often, provide test inputs, control and monitor execution, and report test results.

Software Development Glossary

http://webster.cs.ucr.edu/Page_softeng/softDevGuide_contents.html

www.whatis.com

IEEE Glossary

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Unit Testing

Stub: An empty to-be-called function that replaces a function that is yet to be written. It has the same interface as a module, and does minimal manipulation. A stub usually:

- Tests (validates) the input data
- Prints message “test successful” with parameters to use return as a test.
- Get return values from interactive input
- Return a standard answer regardless of input

Unit Test Environment

http://cs7132.comp.hkbu.edu.hk/~jiming/

Stubs can be hard to write and maintain as well as taking as much time as writing the unit itself!

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Unit Test some definitions

Mock objects are similar to, not the same as, a stub. It simulates an object that is not there.

Mock Object

Reasons to mock:

1. real object has non-deterministic behavior
2. real object is difficult to set up
3. real object has behavior that is hard to cause (e.g., network error) (similar to 1)
4. real object is slow
5. real object has (or is) a UI
6. test needs to query the object, but the queries are not available in the real object (e.g., "was this callback called?")
7. real object does not yet exist

www.mockobjects.com

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

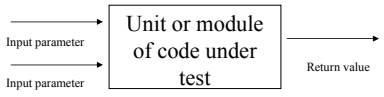
Unit Testing

There are 3 common approaches to unit testing:

- Top Down
- Bottom Up
- Isolation

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

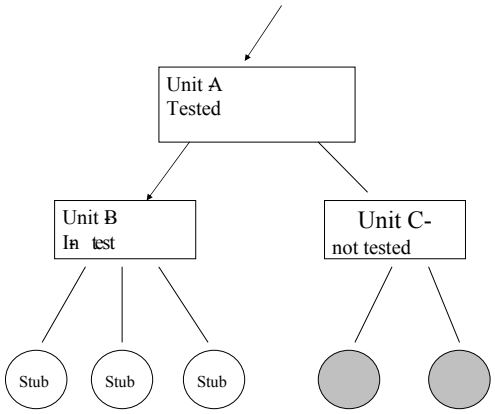
Unit Testing



What we are testing here is the specific action, calculation or behavior of the unit of code. The unit has input parameters, and output or return values.
The input and output can be a variable, constant, or another object of type string, number, pointer, object or many other data types.
The return values will also be a variable, a string, another object or a state change.

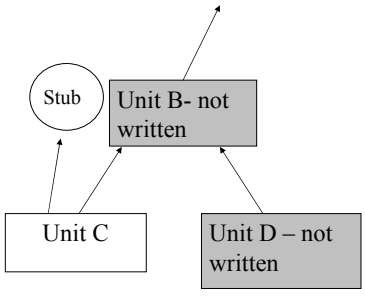
Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Unit Testing
Top Down



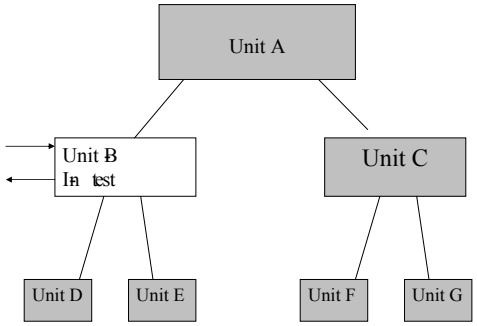
Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 25

Unit Testing
Bottom Up



Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 26

Unit Testing



Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 27

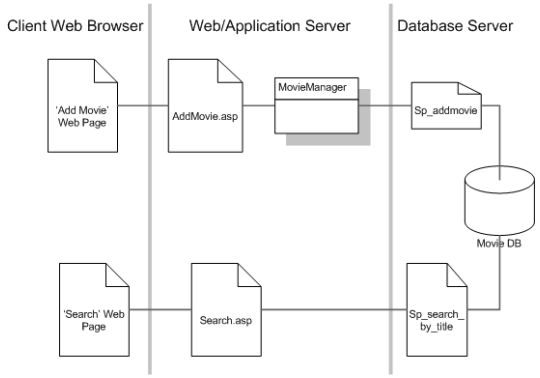
Unit Testing

The next series of examples use an Online Movie Application as a basis.

The application has a UI running in a browser, ASPs, a database and some stored procedures.

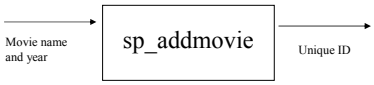
Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 28

System Under Test



Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 29

Unit Test Example



The Addmovie stored procedure inserts a record in the AllMovie Table. There is a constraint on the Title column of the table to not add a movie with an existing title and year.
The input, under normal use, will come from user input through the UI in a browser. The input parameters are movie title and year. The value returned from this action is a unique ID for this record.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 30

Unit Test Example

The Stored Procedure under test:

```
CREATE Procedure sp_addmovie
@NameAndYear nvarchar(65)

As
declare @MovieID int
insert into AllMovies(Title) values
  (@NameAndYear)
--select the unique ID of the row just
  inserted
select @@IDENTITY as 'MovieID'

Go
```

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 31

Unit Test Example

The SQL code for the test:

```
/* enter a new movie */
exec sp_addmovie 'My Movie (2001)'

exec sp_addmovie 'My Movie (2002)'

exec sp_addmovie 'My Movie (2001)'
```

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 32

SQL Unit Test Output

Two rows were added. The third set of parameters failed, as expected.

```
MovieID
---------------------------------------
363
(1 row(s) affected)

MovieID
---------------------------------------
364
(1 row(s) affected)

Server: Msg 2627, Level 14, State 2,
  Procedure sp_addmovie, Line 6
Violation of UNIQUE KEY constraint
  'IX_AllMovies'. Cannot insert duplicate
  key in object 'AllMovies'.
The statement has been terminated.
```

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Unit Test- Stub Example

In this example we will run a test of the MovieManager object before the stored procedure it accesses is created. We can use a Stub in place of the stored procedure. In the fully functioning application the stored procedure will return the value of the row added to the table. This stub of the stored procedure will return a value of “1” as the row number for all tests.

The Stub in place of the stored procedure:

```
create Procedure sp_addmovie
@NameAndYear nvarchar(65)
As
select 1 as 'MovieID'
```

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

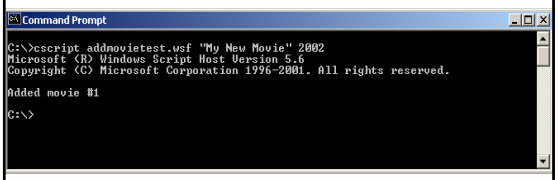
Unit Test- Script for Stub Example

```
<job>
<script language="vbscript">
option explicit

dim strMovieName'variable to hold movie name
dim intYear     'variable to hold the year
dim objMovieMgr 'variable to hold the movie manager
dim intMovieID  'variable to hold movie Id returned by
   AddMovie()
                'get the command line parameters
strMovieName = cstr(WScript.Arguments(0))
intYear = cint(WScript.Arguments(1))

set objMovieMgr =
   CreateObject("MovieDatabase.MovieManager")

objMovieMgr.DBConnection = "DSN=MovieDb;uid=sa;pwd=sa;"

'now call AddMovie
intMovieID = objMovieMgr.AddMovie(strMovieName,intYear)

'now print the movie ID to the screen
WScript.Echo "Added movie #" & intMovieID

</script>
</job>
```

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 35

Unit Test- Execution of Stub Example

The results:



Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 36

Tools for Testers during Unit Test

www.qacity.com

Other Resources: Tools / Utilities: Source Code Analyzers

These tools generally check for bad syntax, logic, and other language-specific programming errors at the source level. This level of testing is often referred to as unit testing and component testing. The developer normally executes this testing.

Unit Test Frameworks

parasoft.com

http://www.parasoft.com/jsp/home.jsp

vbunit.org

junit.org

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Quick Look at one Tool C++Test from Parasoft

C++Test says it excels at catching certain kinds of errors:

- GIGO (Garbage In, Garbage Out) for functions. This is C++Test's bread and butter.
- Functions and methods will be exercised with a rich set of parameter combinations.
- Uninitialized member variables.
- Code that doesn't get run. By inspecting the results of the "Coverage" window, you can tell at a glance if there are parts of your code that aren't getting executed.
- Heap overruns and wild pointers-but only if you're lucky enough to cause an exception with them,
- Newly-introduced errors. (Building a Regression Suite)

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

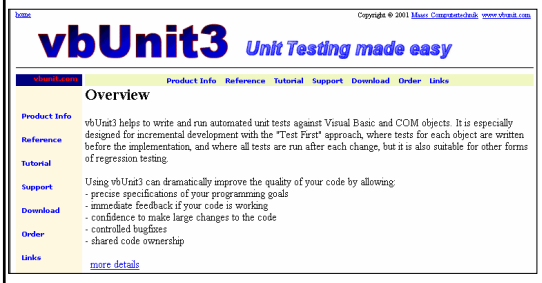
Using a Test Harness

Two harness frameworks becoming more popular are vbunit and junit.

- www.vbunit.org
- www.junit.org

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 39

Using a Test Harness



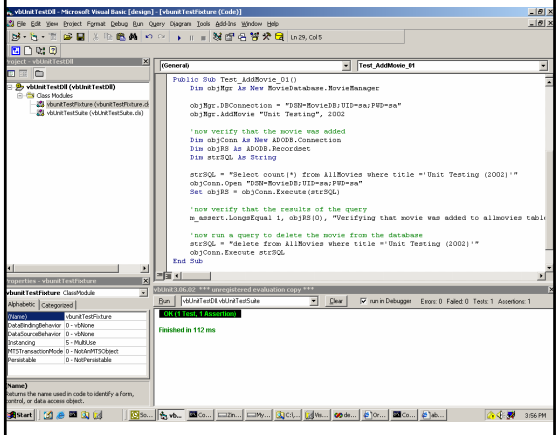
Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 40

vbunit Example

- If you don't know a programming language and the specific function of the code enough to write the unit tests, a developer friend will have to write some tests for you.
- The good thing is with some simple input manipulation you can create many tests from a few well written, simple tests.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Using a Test Harness vbunit



Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Using a Test Harness vbunit

The interesting part for us are the parameters passed:

```
Public Sub Test_AddMovie_01()
    Dim objMgr As New MovieDatabase.MovieManager

    objMgr.DBConnection = "DSN=MovieDB;UID=sa;PWD=sa"
    objMgr.AddMovie "Unit Testing", 2002

    'now verify that the movie was added
    Dim objConn As New ADODB.Connection
    Dim objRS As ADODB.Recordset
    Dim strSQL As String

    strSQL = "Select count(*) from AllMovies where title ='Unit Testing (2002)'"
    objConn.Open "DSN=MovieDB;UID=sa;PWD=sa"
    Set objRS = objConn.Execute(strSQL)

    'now verify that the results of the query
    m_assert.LongsEqual 1, objRS(0), "Verifying that movie was added to allmovies tabl

    'now run a query to delete the movie from the database
    strSQL = "delete from AllMovies where title ='Unit Testing (2002)'"
    objConn.Execute strSQL
End Sub
```

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 43

Using a Test Harness vbunit

You can learn the test harness interface and create and run many tests of your own. You can manipulate the input parameters and/or run the tests on many different environments.

The example shown passes:
"Unit Testing" and "2002"
as the input parameters into the Movie Database.
You can easily edit the test to:

1- " " "0000"
2- "##$%" "abcd"
3- "string way too long…"
"2002"
4- "¤W¥«§K¶" "'98"
5- "à l'arrivée" "1999"
6- "a l'arrivee" "1999"
…etc.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved. 44

eXtreme Programming

How does this type of testing fit in with Test-Driven/Test-First or eXtreme Programming?

One of the main premises of XP is you write tests before you code. Unit testing is the foundation of eXtreme Programming. Tests direct the coding.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

eXtreme Programming

XP, as defined by some, defines two levels of testing:

- Unit Testing (Developer)
  - Each class implemented must have programmer-developed unit tests, for everything that "could possibly break".
  - These tests are to be written during coding of the class, preferably right before implementing a given feature.
  - Tests are run as frequently as possible during development, and all unit tests in the entire system must be running at 100% before any developer releases his code.*
  - *By release, we mean transferring code to the integration area.

http://www.xprogramming.com/qa/xp_q_and_a_QA.htm

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

eXtreme Programming

XP defines two levels of testing:

- Functional Testing (QA)
  - Each feature of the system, which is defined by something we call a User Story, rather like a Use Case, must have one or more functional tests that test it.
  - The functional tests are the responsibility of what we call the "customer“ (also responsible for defining the requirements).
  - The implementation and running of functional tests should be performed by the Software QA group.

http://www.xprogramming.com/qa/xp_q_and_a_QA.htm

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Unit Testing
Solutions for Black box Testers

Now that we have this information, what part of this can traditional black box testers do?

The first recommendation:

> Focus on excellent Test Case design and Test Cases development skill to give the developer so they can build better unit tests.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

Unit Testing
Solutions for Black box Testers

The second recommendation:

Learn some interface for you to execute the unit tests the developers write. Windows Scripting Host, vbunit, SQL, etc.

The third recommendation is-

If it is important to the project team- Learn how to develop Unit Tests yourself.

Learn a language- some scripting language, VB, SQL, Java, C++ …

Talk with the developers and build your knowledge of your application's code.

Build your technology skill set.

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

References

Testing Computer Software Edition 2., Kaner, Falk, Nguyen Wiley 1986

Testing Applications on the Web, Hung Nguyen, Wiley, 2000

Integrated Test Design and Automation: Using the Testframe Method, Hans Buwalda, et. Al. Addison-Wesley Pub Co. 2001

IEEE Glossary

Software Development Glossary

http://webster.cs.ucr.edu/Page_softeng/softDevGuide_contents.html

www.whatis.com

Parasoft. White Papers

http://www.parasoft.com/jsp/pr/tech_papers.jsp

Creating a Test Harness, Frans Coenen, University of Liverpool

Structural Programming and Data Structures Osmar Zaiane University of Alberta

Unit, Integration and System Testing Kurt Keutzer, EECS University of California Berkeley

Copyright (c) 2002 *LogiGear Corporation* All Rights Reserved.

QW2002 Paper 4A2

Jan Grinnell
(GovConnect)

Reduce Overall Project Development Costs with Structured Testing

Key Points

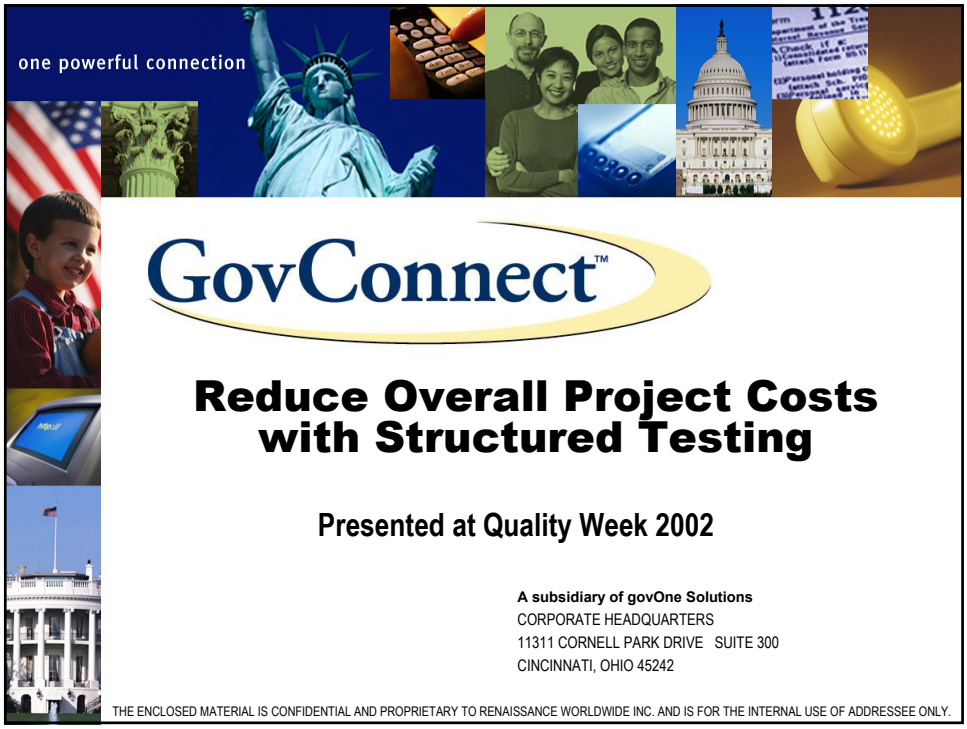
- how to reduce project costs by using desktop software for testing processes
- Providing accountability for covering all requirements during testing
- Providing management with metrics for making informed decisions regarding implementation

Presentation Abstract

The focus on new technology today causes organizations to overlook critical basic functions that affect the quality of software, as well as impact the overall project costs. A simple approach to making a difference in these areas is structured testing - beginning with requirements definition, and ending with a postimplementation assessment. This presentation details a structured testing methodology that focuses on concepts and techniques that are valuable for both legacy systems and new development efforts.

About the Author

Jan Grinnell has an MBA from Fairleigh Dickinson University and over 30 years of application software development life cycle management experience. Her experience spans both Federal and State Government projects as an employee and a consultant. Her current position of Testing Practice Manager for GovConnect has given her the opportunity to gain valuable insight into the current situation with State Government testing environments. Ms. Grinnell has presented her Structured Testing Methodology™ at several national conferences including Government Technology Conference (GTC) West, the National Child Support Enforcement conference, and previous GovConnect E-Gov conferences. She offers practical and cost effective solutions to complex systems testing.



GovConnect™

Reduce Overall Project Costs with Structured Testing

Presented at Quality Week 2002

A subsidiary of govOne Solutions
CORPORATE HEADQUARTERS
11311 CORNELL PARK DRIVE SUITE 300
CINCINNATI, OHIO 45242

THE ENCLOSED MATERIAL IS CONFIDENTIAL AND PROPRIETARY TO RENAISSANCE WORLDWIDE INC. AND IS FOR THE INTERNAL USE OF ADDRESSEE ONLY.

Overview

- Industry Statistics
- Background Information
- Software Development Life Cycle and Testing
- Structured Testing Methodology™ Phases
- Structured Testing Methodology™ Tasks
- Key Components of STM
- Sample Documentation

@2000 GOVCONNECT ALL RIGHTS RESERVED

Industry Statistics

- Costs of defective software can be as high as 50% of software development costs
- Origin of software errors:
  - 64% Design and Analysis Phase
  - 36% Coding Phase
- Studies at IBM demonstrate that compared to catching defects before or during coding, it is 10 times more costly to correct an error after coding and 100 times more costly to correct a production error.

Bala Subramaniam, ISSRe Systems, Inc.
Effective Software Defect Tracking, 1999

@2000 GOVCONNECT ALL RIGHTS RESERVED

How to reduce costs?

- Use structured methodology for testing process
- Use desktop applications instead of automation until testing process is solidified
- Build for the future if using automation by using the desktop applications first
- Use standardized templates to enhance process

@2000 GOVCONNECT ALL RIGHTS RESERVED

Birth of a Methodology

- 30 years experience in software development industry
- experience in private and public sectors
- resource restrictions for staff and budget

@2000 GOVCONNECT ALL RIGHTS RESERVED

Background Information

Based upon the Software Quality Engineering (SQE) STEP methodology and ANSI/IEEE 829 Standard for Software Test Documentation and ANSI/IEEE 1008 Standard for Software Unit Testing

@2000 GOVCONNECT ALL RIGHTS RESERVED

Software Development Life Cycle and Testing

Based upon concept of incorporating the testing process into the life cycle development; beginning with the requirements definition, and participating throughout the entire project until implementation

@2000 GOVCONNECT ALL RIGHTS RESERVED

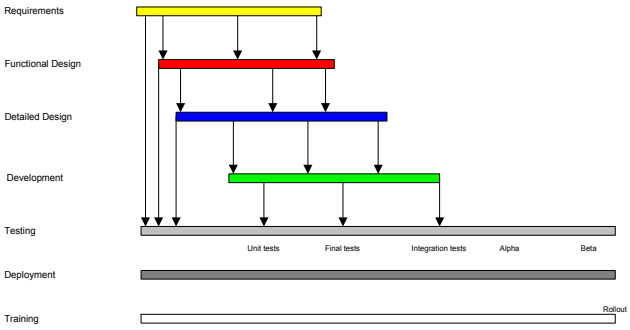
Basic STM™ Philosophy

Simple and Cost Effective process and procedures that can be easily incorporated into Government agencies software development life cycles.

@2000 GOVCONNECT ALL RIGHTS RESERVED

Software Development Life Cycle

Figure 1. Software Development Life Cycle



@2000 GOVCONNECT ALL RIGHTS RESERVED

STM™ Phases and Key Documentation

Requirements Phase
Project Plan
Master Test Plan
Requirements Traceability Matrix

Functional Design Phase
Testware Design

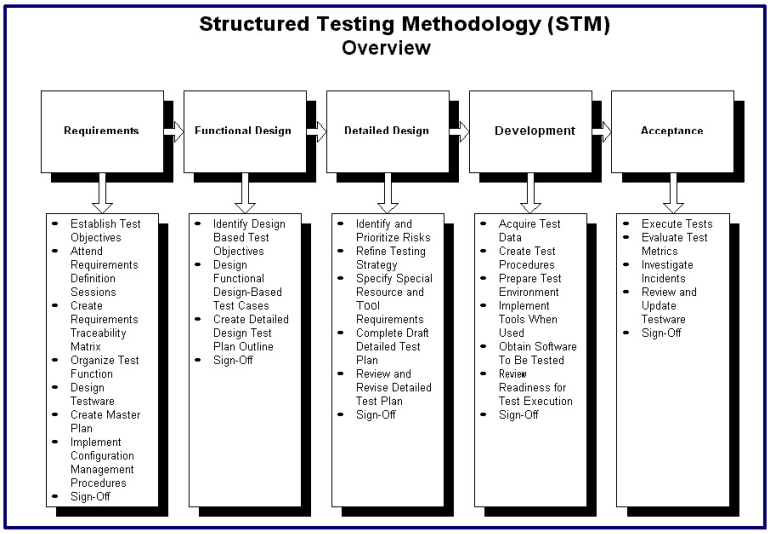
Detailed Design Phase
Prioritize Risks in Matrix

Development Phase
Create Test Scripts

Acceptance Phase
Execute Tests
Test Report

@2000 GOVCONNECT ALL RIGHTS RESERVED

STM™ Tasks - Total Methodology



@2000 GOVCONNECT ALL RIGHTS RESERVED

Project Plan

- Created from Project Plan template using pre-defined tasks
- Tasks are taken from STM™
- Maintained by Test Manager
- Used as input to overall Project Plan that is maintained by Project Manager

@2000 GOVCONNECT ALL RIGHTS RESERVED

Project Plan Template

ID	Task #	Task Name	Start	Finish
1	**1.0**	**Requirements**	**04/06/01**	**04/06/01**
2	1.1	Establish Test Objectives	04/06/01	04/06/01
3	1.2	Attend requirements sessions	04/06/01	04/06/01
4	1.3	Create requirements traceability matrix	04/06/01	04/06/01
5	1.4	Organize test function	04/06/01	04/06/01
6	1.5	Design testware	04/06/01	04/06/01
7	1.6	Create Master Test Plan	04/06/01	04/06/01
8	1.7	Implement configuration mgt procedures	04/06/01	04/06/01
9	1.8	Sign-off	04/06/01	04/06/01
10	**2.0**	**Functional Design**	**04/06/01**	**04/06/01**
11	2.1	Identify Design Based Test Objectives	04/06/01	04/06/01
12	2.2	Design Functional Design-Based Test Cases	04/06/01	04/06/01
13	2.3	Create detailed design test plan outline	04/06/01	04/06/01
14	2.4	Sign-off	04/06/01	04/06/01
15	**3.0**	**Detailed Design**	**04/06/01**	**04/06/01**
16	3.1	Identify and Prioritize Risks	04/06/01	04/06/01
17	3.2	Refine Testing Strategy	04/06/01	04/06/01
18	3.3	Specify Resource and Tool Requirements	04/06/01	04/06/01

@2000 GOVCONNECT ALL RIGHTS RESERVED

Document: Master Test Plan

Purpose: This document summarizes the organization of the overall testing approach

- The primary focus of this document is to provide information that will be used by management to assess the overall strategy being used for this testing.
- It is created following Joint Application Development (JAD) sessions

@2000 GOVCONNECT ALL RIGHTS RESERVED

Master Test Plan

EXECUTIVE SUMMARY 1
SECTION 1 1
INTRODUCTION 1
SECTION 2 2
TEST ITEMS 2
SECTION 3 4
FEATURES TO BE TESTED 4
SECTION 4 9
FEATURES NOT TO BE TESTED 9
SECTION 5 10
APPROACH 10
SECTION 6 13
ITEM PASS/FAIL CRITERIA 13
SECTION 7 14
SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS 14
SECTION 8 15
TEST DELIVERABLES 15
SECTION 9 16
TESTING TASKS 16
SECTION 10 20
ENVIRONMENTAL NEEDS 20
SECTION 11 21
RESPONSIBILITIES AND ROLES 21
SECTION 12 22
STAFFING AND TRAINING NEEDS 22
SECTION 13 23
SCHEDULE 23
SECTION 14 24
RISKS AND CONTINGENCIES 24
SECTION 15 25
APPROVALS 25

@2000 GOVCONNECT ALL RIGHTS RESERVED

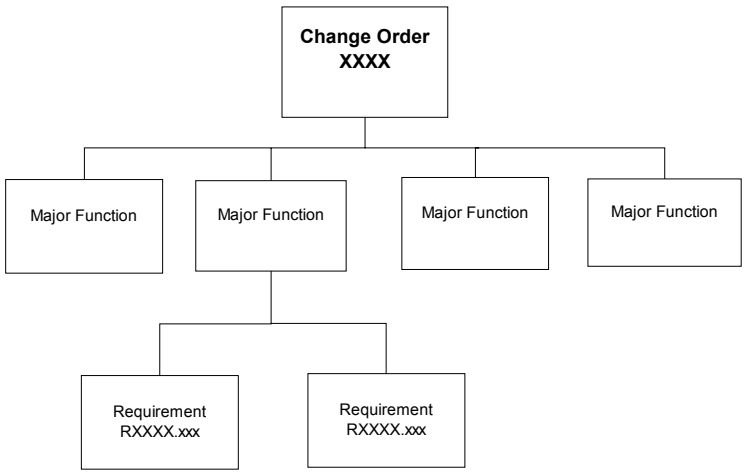
Document: Testware Design

Purpose: To design the flow of the requirements during testing, and to assure that all requirements will be tested.

- **Testware - arrangement of test components to meet test objectives and requirements in a cost-effective way**
- **Test high risk 'deep' and low risk 'thin'**

@2000 GOVCONNECT ALL RIGHTS RESERVED

Testware Architecture



@2000 GOVCONNECT ALL RIGHTS RESERVED

Document:
Requirements Traceability Matrix

Purpose: assures that all requirements are included in coding and testing

- Core document to entire methodology
- Coordinates basic project documentation
- Forces accountability of inclusion of requirements in each phase
- Provides information to make decisions regarding scope of testing via the Risk Prioritiztion

@2000 GOVCONNECT ALL RIGHTS RESERVED

Requirements Traceability Matrix Template

	Change Order [#]					
Req #	**Requirement Description**	**Business Req.**	**Technical Spec**	**Test Script #**	**Training**	**Risk**
Requirement # from requirements document	Cut and paste narrative description from the requirements document and the technical specification to assure that it was covered in both documents	Place and 'X' when the requirement is copied to the matrix	Place and 'X' when the requirement is copied to the matrix	Insert Test Script number when completed to test requirement; more than one requirement might be covered with one script, or more than one script might be required for one requirement.	Place and 'X' when the requirement is verified as part of the training material	High, medium or low rating for each requirement in relation to this project.
Sample completion of matrix:						
1380.2	**Bus Req -** The first generation of the report should include all cases that meet the criteria from 10/1/2000 to the date of the report. **Tech Specs -** For the initial run of this program, those currently off aid are those as of the day the program is run.	X	X	TS-1380.3	X	High

@2000 GOVCONNECT ALL RIGHTS RESERVED

Test Requirements

- Bug prevention is far more cost effective than bug detection
- Test requirements early for understandability and accuracy
- This can occur prior to code completion

@2000 GOVCONNECT ALL RIGHTS RESERVED

Sample RTM Test Results Template

Change Order [#]								
Req #	Requirement Description	Understandable		Accurate		Dependency	Risk	Input/Comments	Output
		Pass	Fail	Pass	Fail				
Enter # from Req. document	Detailed narrative of requirement which is cut and pasted from requirements document	These columns are used to record the test results for testing accuracy and understandablity of written requirement				What cross functional dependencies exist	Evaluate requirement as high, medium, or low in terms of this change request	List any inputs that would be used for testing purposes	List expected outputs
Sample requirement follows below:									
R1234.01	The onlines should have a new updatable field for adding the particpant's address	X		X		Case initiation function should be reviewed for impact	High	Particpant address	Screen should be viewed with updated address after hitting enter button

@2000 GOVCONNECT ALL RIGHTS RESERVED

Document: Test Script

Purpose: to provide detailed procedures for testing all requirements and include expected results prior to test execution

- **Accelerates test results analysis**
- **Provides documentation of results**
- **Can be used for future regression testing**
- **Can use Word template of Access database**

@2000 GOVCONNECT ALL RIGHTS RESERVED

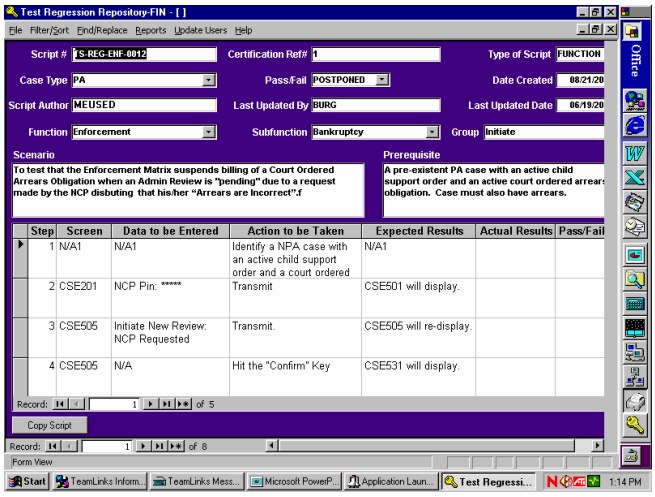
Sample Test Script - MS Word

Test Case#: TR-PRWORA-01
Scenario:
This will test a collection on a former TANF case with two NCPs. NCP #1 has an order for $250.00 per month child support with $3,000 arrears. The order is for one child. There is no judgment for the arrears and no periodic payment. NCP #2 has an order for $100.00 per month child support for one child. There is a judgment for arrears in the amount of $7,500.00 with a periodic payment of $50.00 per month on arrears.
NCP #1 pays $3,000 via personal check.
Prerequisite:
None

Step	Screen	Data to be Entered	Action to be taken	Expected Results	Actual Results	Pass/ Fail
01.	101	CP name: Jane Doe CP address: 1 Main Boston, MA 02025 Hit enter	Review CP INTAKE screen	Automatically navigates to NCP INTAKE screen. Entry is maintained by system. Data is correct.	Data disappeared.	F
02.	102	NCP name: Joe Doe Hit enter	Review NPC INTAKE screen	Entry is maintained by system. Data is correct	NCP name was maintained by the system. Data is correct.	p

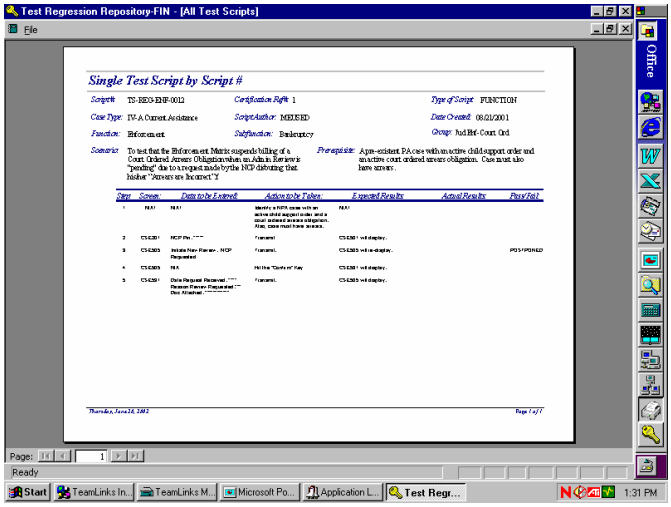
@2000 GOVCONNECT ALL RIGHTS RESERVED

Sample Test Script - MS Access



@2000 GOVCONNECT ALL RIGHTS RESERVED

Sample Test Script - MS Access



@2000 GOVCONNECT ALL RIGHTS RESERVED

Sample Test Script Spreadsheet using MS Excel

Run #: 1379.02
Date: 10/18/2000
Environ.: 01

TS-PR0 1379.2 Review Letters (Actual)

	CP PIN		CP Name		Office		Address		SSN		Expected Outcome Letter Yes	Expected Outcome Letter No	Letter YES	Letter NO	Pass	Fail	P.L. #
Req. #	1379.5		1379.3				1379.4		1379.6								
	1051.1631	X	Tyhitia Moore	X			19B Ellsworth Ave, Brockton, MA 02401	X	012-58-0726	X	X		X		X		
	1068.0786	X	Cary Castillo	X			7 Pidulski Way A587, So. Boston, MA 02127	X	013-58-6443	X	X		X		X		
	1069.0919	X	Rosa Martinez	X			42 Beechland St. A187, Roslindale, MA 02131	X	013-60-1255	X	X		X		X		
	1086.0954	X	Lori Turner	X			916 Rodman St., Fall River, MA 02721	X	014-62-9244	X	X		X		X		
	1100.5918	X	Alicia Buffalo	X			262 Green St.#3, Brockton, MA 02301	X	015-60-2439	X	X		X		X		
	1423.0874	X	Paula Miller	X			586 Haverhill St., Lawrence, MA 01840	X	034-56-1234	X	X		X		X		
	1155.0835	X	Ratond Davis	X			43 Islandview Pl #201, Dorchester, MA 01253	X	018-58-7347	X	X		X		X		
	1165.5475	X	Laurie Voss	X			P.O. Box 278, Sagamore, Ma 02561	X	019-50-4634	X	X		X		X		
	1260.2308	X	Marisol Borrero	X			103 Robert Dyer Cir., Springfield, MA 01109	X	024-56-9415	X	X		X		X		
	1189.4844	X	Patricia Devlin	X			19 1/2 Everard St., Worcester, MA 01605	X	020-56-2883	X	X		X		X		
	1189.6073	X	Tammy Barboza	X			229 North St, New Bedford, MA 02740	X	020-56-3132	X	X		X		X		

@2000 GOVCONNECT ALL RIGHTS RESERVED

Document:
System Test Execution Log

Purpose: To plan for individual test runs, and record results

- Used to coordinate test team activities
- Used for day start meetings to plan testing
- Record set up procedures
- Can tailor format to project needs

@2000 GOVCONNECT ALL RIGHTS RESERVED

Sample System Test Execution Log

PRWORA Letters and Reports	SYSTEM TEST EXECUTION LOG		
Test Run Number:	**TR-1379-11**		
Test Date:	October 26, 2000	**Test Time:**	8:50 am
Technical Analyst:	Sam Smith	**Environment:**	DevPrd 01
Business Analyst:	John Green, Nancy White		
Weekly #3	This test was executed several times due to the tasks being copied from production as scheduled on a weekly basis. Additional modifications also had to be made to the test team code due to needing to clear the profile codes when restoring from the local save.		
Test Plan/Scenario/Case	**Comments**		
CP PIN # - 8732.1234 (PA with arrears) NCP PIN # - 7654.3210 DP PIN # - 0123.4567 Case # - 1234567 Move from PA to NPA, arrears stay as PA arrears	Correctly moved		
Run IVA-IVD schedule CFT25 Effective date of 10/25	Ran successfully		
Verify CFT25 TS-PRO-1379.11	All results were verified as correct, including the out of state FIPS code not getting a letter.		

@2000 GOVCONNECT ALL RIGHTS RESERVED

Document: Test Report

Purpose: Summarizes testing and test results, and makes recommendation for approval of software

Test Report Outline

- Summary
- Variances (from Test Plan, test designs or test procedures)
- Comprehensive Assessment
- Summary of Results
- Evaluation
- Summary of Activities
- Approvals

@2000 GOVCONNECT ALL RIGHTS RESERVED

Additional Documentation Available

- Organization of Test Function
- Readiness Report
- Organization of Test Function
- Issues Log
- Post Implementation Review Report

@2000 GOVCONNECT ALL RIGHTS RESERVED

QW2002 Paper 6A1

Michael R. Waller & Timothy D. Nelson
(STAMP Technologies)

Reducing Test Automation Maintenance Costs

Key Points

- reusable modular code with a consistent use of automation artifacts
- Test automation is software development
- Outline the benefits of the automatic (dynamic) creation of test documents

Presentation Abstract

Many organizations attempt to implement test automation, but end up with shelfware instead of testware. This can represent a substantial loss in costs associated with tool evaluation, licenses, and training. Additionally, this can adversely impact team productivity & morale, which creates a lost opportunity to improve the effectiveness of the testing process. Testers can spend many hours creating automated tests only to find the same scripts take much more time to maintain and enhance. There are many reasons why the maintenance cost of automation tests are high.

As maintenance activities climb, the focus and creativity of the testers quickly turns to software maintenance instead of their real job… testing the application. Project deadlines approach, the testing schedule falls further behind, and the decision is made to cut your losses and proceed manually. Management deems the automation effort a failure and the automated tests & tool are relegated to shelfware. Why did the test automation process fail? What is the real problem?

About the Author

Michael R. Waller is a Senior QA Consultant. He has over 20 years of IT experience including software QA, test automation, software development, and operations. His business sector experience spans manufacturing, financial, and software development companies. He has an MS in Management from Florida Institute of Technology and a BS in Math/Computer Science from Boise State University. Mike can be reached at mwaller@stamptech.net.

Timothy D. Nelson is a Senior QA Test Automation Specialist. He has 25 years of IT and programming experience within the test and QA industry. He has a BS in EE from the University of Minnesota, with minors in both computer science and business. His experience has extended across several different engineering and business development sectors. He has engineered diagnostic and automated test systems for hardware gate and block model simulators to designing custom enterprise wide test automation frameworks for client/server and Web based technologies. He is now a principle owner in a Software Test and QA consulting firm (STAMP Technologies, LLC) that is developing a new generation of highly qualified software test and automation specialists. Tim can be reached at tdnelson@stamptech.net.

An Enterprise Framework for Test Automation

Reducing Test Automation Maintenance Costs

Copyright © 2002 STAMP Technologies LLC, All Rights Reserved

Agenda

- **Test Automation**
- Test Documentation
- Logging Test Results
- Keeping Automation Useable
- Conclusions

The state of test automation

- Many organizations attempt to implement test automation, but end up with shelfware instead of testware.
- This can represent a substantial loss in costs associated with tool evaluation, licenses, and training and creates a lost opportunity to improve the effectiveness of the testing process
- Additionally, this can adversely effect team productivity & morale.

Why does test automation fail?

- On of the primary reasons is maintenance overload.
  - Testers can spend many hours creating automated tests only to find the same scripts take much more time to maintain and enhance.
  - More time is spent in test maintenance than test execution.

Why does maintenance overload occur?

- Test automation not implemented as a software development discipline.
- Maintenance overload is often the result of insufficient planning, brittle tests, bad coding practices and testability issues.
  - Insufficient planning: Not defining enterprise testing goals and the methods used to attain those goals.
  - Brittle Tests: The simplest changes to a user interface or data content breaks the test script.
  - Bad Coding Practices: Many testers do not have a development background and don't know about good coding practices .
  - Testability issues: A person may be able to see the GUI control, however an automation tool may not.

COTS Automation Tools

- COTS = Commercial Off The Shelf
- All these tools use record-and-playback technology as a key component.
- This feature is just a tool in the test automation development process; it is not the "end all" solution to test script creation.
- Reliance solely on record-and-playback will result in brittle tests.

COTS Automation Tools

- Many COTS offerings also provide test management tools for the purpose of managing your test development process from test planning to test development, execution, defect logging and status reporting. Again, these are tools that may, or may not, be applicable to your automation requirements.
- The real power of COTS tools comes from their internal programming language and their openness to external languages (VB, C, C++, C#, etc.). This openness allows you the flexibility to create your own custom solutions that address your particular needs and processes.

Look before you leap
(define your automation requirements)

- What do you want to achieve with automated testing?
- How will you measure your progress and success?
- How will you measure your test coverage?
- What is your test automation development process?
- Who maintains the scripts, executes them, and analyzes the results?
- What kind of version/source control process will be used?
- What documentation is required and how is it maintained?
- How will you share techniques, methods and code?
- How will you handle a mixture of manual and automated tests?
- What is your budget and what tools are required?
- What existing development tools could be used or shared?
- What terms and definitions will be used?
- How will you continue to improve the process?

What is the solution?

This rest of this presentation describes a test automation framework methodology that provides a solution to three focus areas:

1. Creating and maintaining test documentation
2. Logging test results so they are easy to analyze & report on
3. Keeping automation useable in a high-change environment.

This helps reduce the cost of maintaining your tests, allows for customization as your process improves, and promotes maximum code reuse.

Agenda

- ✓ Test Automation
- ➢ **Test Documentation**
- Logging Test Results
- Keeping Automation Useable
- Conclusions

Test Documentation

- Identifying what is to be tested (test requirements), the test coverage required to verify the test requirements, and the method of reporting the status is part of any test process. This applies to test automation as well.
- If not done correctly, automation can actually increase the workload needed to perform these tasks.

Documentation – Manual tests

- Test case documents (TCDs) are used to plan and execute tests.
  - Each test has a unique identifier, which is especially useful when correlating to test requirements and results. This allows you to measure planned coverage, actual coverage, and test progress (test metrics).
  - TCDs can also improve communication and knowledge transfer to other team members (testers and non-testers).

Documentation – Automated tests

- TCDs become reference-only documents.
  - The desire is to have TCDs reflect the current state of the test scripts, but this rarely happens because it simply takes too long.
- One of two things happen
  - The documents are deleted and any benefits they could provide are lost.
  - Documentation overload occurs when detailed step-by-step documentation is expected.
- Or you can use…

Dynamic Document Creation

- Use existing automation artifacts as the source for the documents and create them only when needed.
  - We utilize a **FileParser** utility to dynamically create test case documents. This is done by searching existing automation artifacts for tags and information.
  - This allows you to create up-to-date documents when required. If a documents are never required, then you haven't wasted any time manually writing and updating them.
- There are many artifacts that can be utilized for document creation, but for the sake of simplicity we will focus on test requirements and test scripts.

Test Requirements

- Identifies the requirements for testing activities.
- Created by gathering information from:
  - Software requirement/design documents
  - Use cases
  - Test Plan documents
  - Hallway conversations
  - Status meetings
  - The application
  - Any other source that is available

Test Requirements (cont.)

- This information is traditionally contained in a Test Plan.
- Instead of creating a document for documentation's sake, we recommend storing the information in a format where it can be used as part of an automation framework.
  - Can be a spreadsheet or a data table.

Test Requirement File (TRF)

The TRF includes the following columns:

- Test Group: Identifies a test group identification number.
- Requirement ID: Identifies the test requirement identification number.
- Requirement Description: Contains a description of the test requirement.
- Test Script: Location & name of the test script used to test the requirement.

Group	Req ID	Description	Test Script
1	**0**	**Field Verification**	
1	1	Screen 1 verification	Scriptname1
1	2	Screen 2 verification	Scriptname2
1	3	Screen 3 verification	Scriptname3
2	**0**	**Workflow Tests**	
3	**0**	**Load Tests**	

Test Requirement Groups

Organizing test requirements into groups helps modularize your tests and can help identify testing gaps.

- **Navigation**: Verify expected UI navigation.
- **GUI Consistency**: Verifies the integrity of the GUI screen by screen.
- **Required Field Verification**: Verifies required fields for all screens.
- **Special Business Rules Verification**: Verifies special business rules.
- **Data Integrity Verification**: Verifies data entry characteristics (field length, types, boundary conditions, etc.).
- **Workflow Tests**: Verifies data entry throughout a business workflow.
- **Performance Tests**: Load & Stress tests.
- Others…

Test Scripts

- Inserting meaningful comments into test scripts should be done as part of a good coding practice.
  - Increases the readability of the code
  - Helps keep maintenance costs down
- We've expanded on this practice by utilizing tags for documentation. Two examples could be:
  - #-Scenario:
  - #-Test Case:

Dynamic Document Creation

Documentation is created by combining the "what" in the TRF...

Group	Req ID	Description	Test Script
1	**0**	**Field Verification**	
1	1	Screen 1 verification	Scriptname1
1	2	Screen 2 verification	Scriptname2
1	3	Screen 3 verification	Scriptname3
2	**0**	**Workflow Tests**	
3	**0**	**Load Tests**	

With the "how" in the test scripts (Scriptname1 contains the following):

#-Scenario: Whatever 1
  #-Test Case: Field1 verification
  #-Test Case: Field2 verification
#-Scenario: Whatever 2
  #-Test Case: Field3 verification

Dynamic Document Creation (cont.)

The result would show the following:

1.0 Group: Field Verification
- 1.1 Requirement: Screen 1 verification [Scriptname1]
  - 1.1.1 Scenario: Whatever 1
    - 1.1.1.1 Test Case: Field1 verification
    - 1.1.1.2 Test Case: Field2 verification
  - 1.1.2 Scenario: Whatever 2
    - 1.1.2.1 Test Case: Field3 verification
- 1.2 Requirement: Screen 2 verification [Scriptname2]
- 1.3 Requirement: Screen 3 verification [Scriptname3]

2.0 Group: Workflow Tests

2.0 Group: Load Tests

➔ **Numbers are created dynamically**

Agenda

✓ Test Automation

✓ Test Documentation

➢ **Logging Test Results**

- Keeping Automation Useable
- Conclusions

Test Execution Logs

- Using a consistent execution logging mechanism across all projects can make analysis & reporting simpler.
- STAMP uses functions contained in a ***ResultLib*** library to document the results of a test.
- Results are tied to the same dynamic numbers that are used for documentation. This ensures that errors are easy to track back to test requirements, the test scripts, and the location within each test script.

Test Execution Logs (cont.)

- Test log
  - Each test case: passed/failed, failure description, script name, test description, date/time of execution
- Summary log
  - Total number of scripts run, number not completed & number completed
  - Total number of scenarios run, number failed & number passed
  - Total number of test cases run, number failed & number passed
  - Total number of test conditions checked, number failed & number passed
  - Total run time
  - Start available resources, end available resources & available resource delta

Agenda

- ✓ Test Automation
- ✓ Test Documentation
- ✓ Logging Test Results
- ➢ **Keeping Automation Usable**
- Conclusions

Automation in a high-change environment

- How do you keep test automation going when applications are continually being updated?
- Reusable modular code & consistency are the keys.

Framework Components

- **Entry Process**: Performs setup activities – establishes the report location; clears counts & timers; logs memory/resource information; Etc. This process calls the Initialization Process and UtilityLib functions.
- **Exit Process**: Performs cleanup activities – logs counts, timers, and resource information; unloads dll's & GUI files.
- **FileParser**: see previous slides
- **FuncLib**: This is a library of general and application specific functions. These functions are geared towards the manipulation of screen objects (push button, entry fields, etc.).
- **GUI Map**: A physical to logical mapping of GUI objects.
- **INI File**: Contains values for global variables. Used by the Initialization Process.

Framework Components (cont.)

- **Initialization Process**: Sets up the test automation development & run environment. Variable & path assignments are made here. This is always called from an Entry Process, but is separate to allow execution of individual Test Scripts during development & debugging.
- **ResultLib**: see previous slides
- **Test Requirement File**: see previous slides
- **Test Data Table**: Contains input & verification data.
- **Test Script**: Performs user & verification actions. Also contains additional business rules.
- **Test Suite**: Controls the batch execution of multiple tests. Calls an Entry Script, one or many Test Scripts, and an Exit Script.

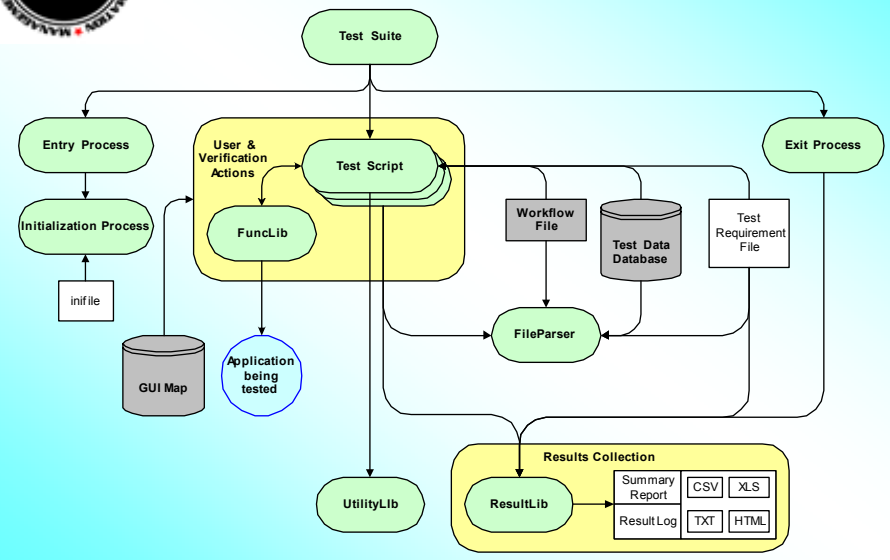
Framework Components (cont.)

- **UtilityLib**: A library of functions that logs system information such as OS, memory utilization, etc.
- **Workflow File**: A sequential list of navigation steps thru a workflow process. Contains information about what windows to use, which buttons to press, what fields to enter data into, and where to get that data.

Note: All framework components have templates to aid in ease of use and consistent development.

For more information on GUI Map, Test Script, and Workflow File see the Workflow Driven Tests whitepaper.

Component Relationships

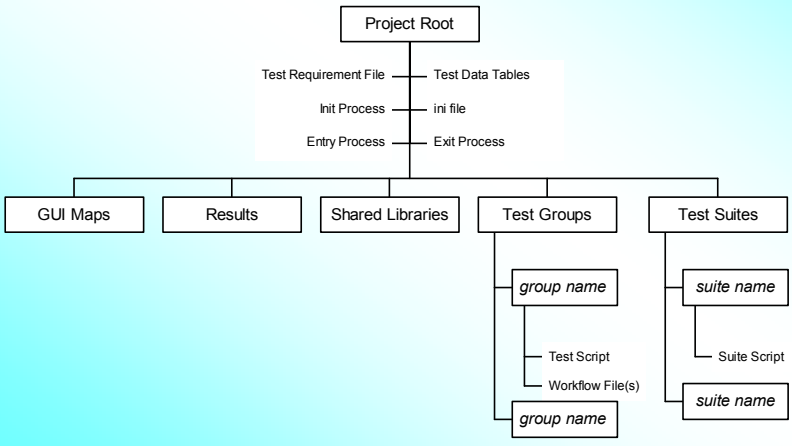


Consistency

- Provides a common look and feel to your automation efforts. This aids both maintenance and knowledge transfer.
- Consistency needs to be implemented for the location of your files, the structure of your code, and common utilities.

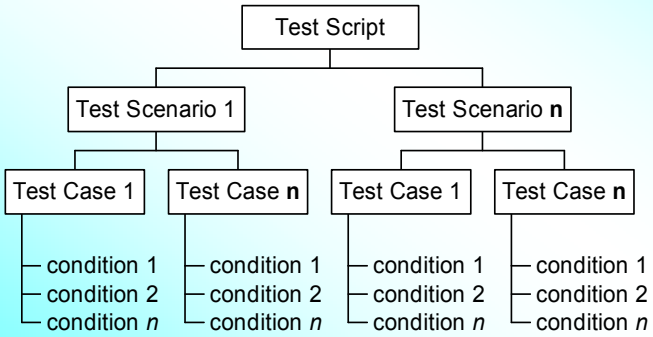
Consistency (cont.)

STAMP recommends the following directory structure.



Consistency (cont.)

Test scripts should also have a common look and feel. We use a script template that supports the ***FileParser*** documentation & dynamic numbering process.



Consistency (cont.)

- We have established templates for test scripts to promoting consistency of use and aid in their creation. Our templates also include calls to UtilityLib & ResultLib functions to ensure consistent logging.

- A "high-change" environment can also apply to your staff as well. Employee turnover can be painful if key knowledge walks out the door. Having a consistent practice will mitigate this impact and allow you to continue to meet your testing needs with the minimum stress.

Agenda

- ✓ Test Automation
- ✓ Test Documentation
- ✓ Logging Test Results
- ✓ Keeping Automation Usable
- ➢ **Conclusions**

Conclusions

- A test automation framework has two key characteristics:
  - **Reusable modular code** reduces the number of places maintenance is needed when something changes.
  - **Consistency** reduces the impact that personnel & organizational changes can have on development, maintenance and training.

Conclusions
(more framework characteristics)

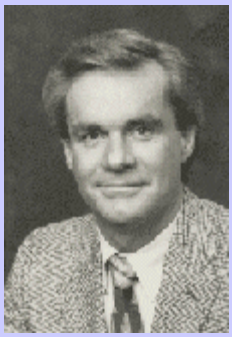
- Incorporates common processes and practices that support your enterprise goals for test automation.
- Is customizable to your enterprise automation needs.
- Is not strictly related to coding and scripting.
- Is open to incorporating 3rd party or custom tools.
- Reduces the impact of organization or staffing changes.
- Provides time savings in long-term test development.
- Minimizes maintenance and lowers cost of knowledge transfer.
- Tool independent.
- Establishes a base level of knowledge about test automation.

Conclusions
(a framework provides)

- A file structure for test artifacts (files, scripts, & libraries)
- A means to develop, track, manage, and maintain the test requirements for each project
- Script templates
- Custom test execution reporting
- Test requirements tracking
- Automated generation of the test case documentation
- Traceability between test requirements, test cases, and test results
- Implementing an automation framework requires some up-front investment, but more than pays for itself over time.

SOFTWARE
TEST
AUTOMATION
MANAGEMENT
PROCESS
STAMP
TECHNOLOGIES

Questions?



QW2002 Paper 6A2

Mr. Gregory Pope
(University of California LLNL)

Risk Based Regression Testing

Key Points

- The regresion testing dilemma
- Risk can be used to solve the delimma
- Full and Partial Regresion using sampling

Presentation Abstract

How to set up a risk based regression testing policy to solve the dilemma of when and how much regression testing to do whan under tight schedule deadlines.

About the Author

Gregory M. Pope is a Computer Scientist at the University of California Lawrence Livermore Laboratory. Over the past quarter century, Mr. Pope has worked in a variety of capacities at virtually all levels of software development and testing. He began his career developing software used to test jet engines and helicopters. He later worked in the defense industry, testing missioncritical software for military and space applications. Working in the private sector, he has developed and patented techniques for computer-aided testing. Among his inventions is the Ferret, a highly regarded software-testing tool manufactured and marketed by his company.

As a consultant and teacher, Mr. Pope has conducted hundreds of seminars for software development professionals throughout the United States, Asia, Canada, Mexico, South America, and Europe. In his thriving consulting, training, and testing practice, he has worked with many Fortune 500 companies, including Microsoft, IBM, Apple Computer, Sun Microsystems, AT&T, Eastman Kodak, DHL Airways, and Knight- Ridder, as well as NASA, the Pentagon, U.S. defense contractors, the Internal Revenue Service, and numerous foreign companies.

Mr. Pope is sought out to write articles on the subject of software testing for a number of industry publications, including CIO Magazine, Computer Design, Industry Week, Computer World, Signal, Electronic Defense News, San Jose Mercury, and Software Maintenance News. He holds a BS degree from Connecticut State University, an MBA from University of Phoenix, and is a member of IEEE.

Risk Based Regression Testing

By:
Gregory M. Pope
University of California
Lawrence Livermore National Laboratory

Quality Week
San Francisco, CA
September, 2002

1

Purpose of a Regression Test

- To reduce the risk that changes made to the software, most commonly to repair defects or add enhancements, have not introduced new defects or undesired or unintentional side effects.
- A standard technique for reducing this risk is to periodically rerun all test cases to make sure they still pass.

2

The Regression Test Dilemma

- There is seldom time to rerun all test cases every time the software changes.
- Some neophyte project managers still think that testing is a one shot activity, i.e. test it and we are done.

3

So What Can Be Done?

- Skip regression testing
- Don't change the software
- Faith Healing
- Do partial regression testing
- Do at least one full regression test just prior to deployment
- Automate regression tests
- Risk based regression testing

4

Reasonable Options

- The software is going to change
- Doing no regression testing is a bad idea
- Can not do full regression testing every time the software changes
- Probably not be able to automate regression testing until the product matures
- Do full and partial regression testing based on risk

5

Regression Test

- Full regression test requires running all test cases
- Partial regression test focuses on repaired area and requires rerunning failed cases, verifications of failed cases, and sampling of other areas.

6

Partial Regression Test

- Re-run test cases that failed
- Re-run test cases related to the one that failed
- Re-run test cases that the developer suggests
- Run a random set of test cases for the remainder of software under test
- Run ad hoc tests in areas of concern

7

Random Numbers Using Excel

- Using a random number generator to pick N test case numbers.
- For instance to RANDBETWEEN (1,100) function in Excel
- N could be 10% or 5%
- So for 100 test cases @ 5% sampling do →

Random Regression Test Numbers		
17		
7		
68		
89		
29		

8

Assigning Random Numbers

Enumerated Number	Test Case Name
1	GUI 1
2	GUI 2
3	GUI 3
4	Shut Down 1
5	Shut Down 2
6	Initialize 1
7	Initialize 2
Etc.	Initialize 3

9

Risk Based Regression Testing

- To determine whether to run full or partial regression testing
- Do a risk assessment of the safety (consequences of failure)
- Do a developer interview to determine the risk based on the nature of the changes

10

Example Safety Criteria

- RQ1 – Consequences of failure are extreme, for instance death or injury to public, violation of laws and regulations, large financial losses.
- RQ2 – Consequences of failure are moderate, for instance injury/illness to public, downtime greater than a week, moderate financial loss.
- RQ3 – Consequences of failure are not so great, for instance down time less than an hour, annoying to customers, small financial loss.

11

Risk Categories

- Develop safety risk categories appropriate for the product and industry
- Develop risk categories based on development environment

12

Developer Interview Risk

- What was the nature of the changes to the software:
  - DR1 - A foundation class piece of code used throughout the software system.
  - DR 2 - A piece of code or data used by other software.
  - DR 3 - An isolated piece of code or data not used by in any other software.
  - These are examples - Depends on process and product.

13

Suggested Levels of Regression Testing

RQ Level	Points	DR Level	Points
RQ 1	50	DR 1	50
RQ 2	25	DR 2	25
RQ 3	15	DR 3	15

Score	65-100	Full	
	40-64	Partial N = 10%	
	30-39	Partial N = 5 %	

14

When To Perform Regression Tests

- After every modification?
- After every patch?
- Wait until all repairs and enhancements completed?
- After clusters of repairs?
- Full regression on each major deployment?
- Judgment required

15

Configuration Management and Regression Testing

- Code being regression tested needs to be under Configuration Management control.
- Do not want code to be modified after testing and before it is checked in.
- Regression tested code needs to have a separate build or view immune to developer changes.
- Regression testing needs to have a separate and controlled database.

16

Buy In

- Regression testing policy should be adopted by the people who develop the code and do the testing.
- Adopted by test management.
- Adopted by development management.
- Adopted by project management.
- There will be resistance.
- You may need to negotiate and wordsmith.

17

Waivers

- Should have an escape valve.
- Ability to reduce or eliminate regression testing in a special case.
- Needs to be at a high level (Program Manager or Executive) sign off.
- Should have a risk assessment prepared for a waiver so management makes an “informed” decision.
- Somebody (hopefully higher up in management) signs the waiver!!

18

Regression Testing Form

- Brief
- Tester, Date, View, Data Base, Instance, Byte Size, etc.
- Safety Criteria Stated. (RQ Level)
- Developer Risk Criteria Stated (DR Level)
- Number and Names of Test Cases Run
- Findings
- Anything else deemed important to know later
- Comments

19

Follow Up To Improve

- Measure the number of defects not found by risk based regression testing.
- Do root cause analysis, find the error source.
- Continue to tune the scoring system and risk criteria.
- Review the test cases, reverse engineer the test design, measure coverage, etc.

20

Actual Experience

- Sampling's uncanny ability to find problems
- Comfort level of testers following a policy
- Educational for Management
- Documentation trail for postmortems
- Low false alarm rate on PLC software
- Have made minor changes to form and risk criteria

21

Summary

- Regression Testing is Important
- Regression Testing is Challenging
- Multiple Levels of Risk Can Be Used to Determine How Much Regression Testing is Appropriate
- Consider Adopting a Risk Based Regression Testing Policy
- Follow Up To Improve Results

22

Risk Based Regression Testing

by Gregory M. Pope

1.0 Purpose

The purpose of this presentation and paper is to present a Risk Based Regression Testing strategy which can help determine the amount of regression testing to perform and the frequency of performing regression testing.

2.0 Scope

Would apply to System (Black Box) Testers, Integration Testers, Project Managers, and Software Developers. Medium to Large scale software projects.

3.0 Regression Testing

The purpose of regression testing is to reduce the risk that changes made to the software under test, most commonly to repair defects or enhancements, do not cause new defects or undesired or unintentional side effects. A standard technique to reduce the risk of repairs (or enhancements) made to the software causing new defects, is to periodically rerun all test cases to make sure they still pass.

3.1 Regression Testing Dilemma

The classic regression testing dilemma on a medium to large scale software project is that if all test cases were run after each and every change to the software, the length and the expense of all but the smallest software project would be prohibitive. So the dilemma is how to reduce the risk of undesired side effects of defect repairs from going undetected without the necessity to run all test cases after every change.

3.2 Regression Testing Options

The best way to reduce risk of undesired side effects from repaired code is to follow good design principles in building the code in the first place. These include techniques such as modularity, coupling and cohesion, code inspection and review, coding practices and compliers that promote structured programming practices, debugging techniques, static code analyzers that identify high risk areas, and configuration management tools that keep track of and manage all changes to the code. Many of these development principles are being followed on NIF and are constantly being improved upon.

The secondary way to reduce the risk of undesired side effects from repaired code is to have and enforce an effective regression testing policy. While this technique does not prevent errors, it will increase the probability of detecting them should they occur prior to release to the operational environment. An effective and reasonable regression testing policy must be tailored to the size and complexity of the project, the consequences of failure, available resources, and budget and schedule constraints.

Automation of regression tests is an ideal strategy for mature products that undergo periodic upgrades. Automating regression testing while developing a new product is very difficult because the test scripts will need to constantly be changing along with the new code. The software staff size would need to almost double to support the developers needed to write the test code. The technical risk then becomes using new test code to test new software code. Also, who tests the test code? While regression test automation is clearly a long range solution, it is probably not the best short term solution other than “smoke tests,” isolated emulators, and performance-related testing.

It goes without saying that a bad option is to skip regression testing altogether because of schedule pressure. This technique almost always leads to major problems after release to operations.

3.3 Risk Based Regression Testing

The risk based regression testing strategy will address the regression testing dilemma by using a combination of full and partial regression test strategies, sampling theory, and risk-based regression testing.

3.4 Full and Partial Regression Testing

A **full regression test** requires running all test cases for the software under test again, as well as some ad hoc tests. To pass the regression test, all test cases must pass. (Note: ad hoc tests are tests which do not have to be written down a priori, but are based on experiential, spontaneous, or intuitive sources. If an ad hoc test does uncover a defect, the procedure followed should then be fully documented, including observed results, and added to the set of test cases.

A **partial regression test** requires:

1. Re-running the test case(s) that previously failed to assure they now pass.
2. Running any test cases closely related to the area that previously failed.
3. Running any test cases that the developer suggests or feels is important.
4. Running a random sample set of test cases for the remainder of the software under test.
5. Running ad hoc tests in areas of concern.

3.5 Sampling Technique for Partial Regression Tests

In partial regression testing, it is important to conduct a random sample of test cases over the software under test, in addition to testing the effected area, areas related to the effected area, and areas suggested by the developer. The random sample technique requires use of a random number generator to pick N test case numbers, where N is the number of test cases to run. A function such as RANDBETWEEN in Excel can be used to generate random numbers.

Note: If function RANDBETWEEN generates an ?name error on your spreadsheet, you may have to add it to Excel using the Analysis Tool Pack. This can be done doing the following:

1. On the **Tools** menu, click **Add-Ins**.
2. In the **Add-Ins available** list, select the **Analysis ToolPak** box, and then click **OK**.
3. If necessary, follow the instructions in the setup program

For instance, if there are a total of 100 test cases for the software under test, and it is desired to run 5% of them on a random basis (N=5), then use RANDBETWEEN(1,100) in five cells of a spreadsheet to generate the five test case numbers. See the example below:

26
53
75
54
6

In this case, test cases 26, 53, 75, 54, and 6 would be run in addition to the specific test cases related to the problem that was fixed. If all the sample test cases pass, the regression test is complete, if any of the sample test cases fail, a complete regression test should be done.

4.0 Risk Based Regression Testing

To determine when full versus partial regression testing is appropriate, an evaluation of risks should be made. The two major risk categories are safety and developer risk. However, other risks may enter into the decision, such as historical or experiential factors. If there is high risk involved with the software

under test, complete regression testing should be done. For moderate risk, either complete regression testing or partially regression testing with a large (10%) random sample size should be done. For lesser risk, a partial regression test with a smaller (5%) random sample size should be done.

4.1 Example Safety Criteria Risk

Concern for safety is a major concern on most software projects, and the most important factor to evaluate when considering regression testing risk. In this discussion we shall assume 3 levels of safety criteria:

1. SR1.- A software induced failure could result in the release of hazardous materials to the environment, or be in violation of a Federal, State, or Local law, or cause death or injury to the general public or a financial loss over one million dollars or downtime greater than 30 days.

2. SR2. – A failure could result in the release of hazardous materials in a confined space, or cause injury or illness to workers, or cause critical system downtime of a week or greater or financial losses between one hundred thousand and one million dollars or downtime from 7 to 30 days.

3. SR3.- A failure could cause damage to the system with downtime of a week or less, or would annoy workers, or cause financial loses under one hundred thousand dollars.

To determine the risk involved with the software to be regression tested, pick the S level that most closely fits the description of the consequences of failure. In some cases, consultation with developers or other technical leads may be required to pick the closest answer. When in doubt, pick the higher risk level.

4.2 Developer Interview Risk

The second major risk factor concerns the nature of the repair from the perspective of the developers. The nature of the repair can be determined by interviewing the developer who made the repair, as well as their supervisor. Working with the developer and their supervisor, see which one of these three levels of repair risk best describes the repair:

1. DR1. – The repair was made on software or data that is in the frameworks, within a foundation class, or the repaired software is used numerous places, or the repair was made to software that has been very troublesome in the past when repaired, or the repair was made to software that was written by someone who is no longer working on the project.

2. DR2. – The repair was made to software or data that is used in more than one place and the other places it is used are well understood and have not been troublesome in the past.

3. DR3. – The repair was made on software that is not used anywhere else, is an isolated piece of code or data not used by anyone else and has not been troublesome in the past.

5.0 Determining Level of Regression Test

After determining the two risk ratings based on safety considerations and the developer interview, consult the table below to determine the score:

RQ Level	Points	DR Level	Points

SR 1	50	DR 1	50
SR 2	25	DR 2	25
SR 3	15	DR 3	15

For instance, an SR2 safety risk and DR2 developer risk would be a score of 50.

After determining the risk score, consult the regression testing table below to determine the regression testing technique to use:

Score	65-100	Full	
	40-64	Partial N = 10%	
	30-39	Partial N = 5 %	

For example, for a risk score of 50, partial regression testing must be performed with a random sample size of 10%

6.0 Determining Frequency of Regression Testing

After determining the type of regression testing that is appropriate based on risk, the next step is to determine how frequently to conduct the regression testing. One extreme would be to conduct a regression test after every repair or patch to the software. The other extreme would be to wait until all repairs and patches are completed before regression testing. The first extreme, regression testing after every change, would be very time consuming. The second extreme, waiting until all changes and patches are complete, would be most time efficient, but leave little or no time to repair defects found during regression testing.

The best choice for regression testing frequency would be a strategy between the extremes. Repairs and patches usually do not occur evenly throughout the build cycle. Normally they come in clusters of activity. For instance, the test engineer might run 100 test cases for three days and generate 50 Test Incident reports. The following week the developers might issue a new release or a set of patches to fix the defective code. It would be at this point that a regression test might be most efficient. Notice that there is a relationship between regression testing risk and the number of repairs. The more numerous the repairs, the greater the likelihood of more than an isolated piece of code being effected, so the developer risk factor goes up. Testing fewer repairs at a time would tend to lessen the developer risk.

There are exceptions, however. Especially close to the software release date to on-line or operational environments, where a critical defect is found, fixed, and requires immediate regression testing.

The best point in time to do the regression testing requires judgment on the part of the test engineer and developer. As a general rule, it should be after a cluster of repairs rather than on a fix by fix or patch by patch basis, unless of course there is compelling reason to do otherwise. It is common to regression test the code numerous times during its development. Before final release is another obvious opportunity to run a regression test.

7.0 Configuration Management and Regression Testing

Before beginning any regression test, assure that the software to be tested is under configuration control. If the regression testing is performed on code that is not under configuration control the possibility exists that the developer may make changes to the code after it is tested and before checking it in. These

changes, no matter how small, will invalidate the regression test and require the regression test to be repeated.

8.0 Summary

Hopefully the Risk Based Regression test approach can help solve the dilemma of how much regression testing to do on software projects that are under schedule pressure to deliver. The number and levels and definitions of risk can be tailored to best suit the process and product being developed. The sampling size and scoring can also be tuned for a particular environment. The examples given in this paper are by no means universally appropriate for all projects, but the under lying principle of varying the amount of regression testing based on risk might be worth adding to your arsenal of testing tools.

THE AUTHOR:

Gregory M. Pope is a Computer Scientist at the University of California Lawrence Livermore Laboratory. Over the past quarter century, Mr. Pope has worked in a variety of capacities at virtually all levels of software development and testing. He began his career developing software used to test jet engines and helicopters. He later worked in the defense industry, testing mission-critical software for military and space applications. Working in the private sector, he has developed and patented techniques for computer-aided testing. Among his inventions is the Ferret, a highly regarded software-testing tool manufactured and marketed by his company.

As a consultant and teacher, Mr. Pope has conducted hundreds of seminars for software development professionals throughout the United States, Asia, Canada, Mexico, South America, and Europe. In his thriving consulting, training, and testing practice, he has worked with many Fortune 500 companies, including Microsoft, IBM, Apple Computer, Sun Microsystems, AT&T, Eastman Kodak, DHL Airways, and Knight-Ridder, as well as NASA, the Pentagon, U.S. defense contractors, the Internal Revenue Service, and numerous foreign companies.

Mr. Pope is sought out to write articles on the subject of software testing for a number of industry publications, including CIO Magazine, *Computer Design, Industry Week, Computer World, Signal, Electronic Defense News, San Jose Mercury,* and *Software Maintenance News.* He holds a BS degree from Connecticut State University, an MBA from University of Phoenix, and is a member of IEEE.

QW2002 Paper 7A1

Mr. James Lyndsay (Workroom Productions Ltd.) & Niel vanEeden (Jobpartners Ltd.)

Adventures in Session-based Testing

Key Points

- Session-based testing can control an unscripted test process
- A useful coverage metric for a session-based test process
- Session-based testing provides a foundation for improvement

Presentation Abstract

This presentation describes the way that a UK company controlled and improved ad-hoc testing, and was able to use the knowledge gained as a basis for ongoing, sustained improvement. We detail the methods we used to define test scope, to control the activities of the team and to measure coverage. The story of how these methods were implemented will include details of problems, solutions, and the ways that the methods were improved. We will also cover the ways that their results helped put the case for improvements throughout development, and ways in which the team built on them to arrive at a better overall test process.

About the Author

James Lyndsay is an independent test consultant with more than ten years experience. Specialising in test strategy, he has worked in a range of businesses from banking and telecoms to the web, and pays keen attention to the way that his clients' focus is shifting away from functional testing.

Niel vanEeden is the Test/QA Manager at the UK company in question, and has implemented a process of ongoing improvement to the methods detailed in the presentation.

Adventures in Session-based Testing

James Lyndsay
Workroom Productions

jdl@workroom-productions.com
www.workroom-productions.com

© Workroom Productions 2002
www.workroom-productions.com

What?

© Workroom Productions 2002
www.workroom-productions.com

Session-Based Testing

Manages and controls unscripted testing

- Limited Duration
- Directed Exploration
- Recorded Activities

Supports 'Agile' Process

Allows quick metrics

Has wider uses

© Workroom Productions 2002
www.workroom-productions.com

Adventures?

Product

- Internet application, commercial and in use

Team

- Small, inexperienced

Business context

- Rapid change
- Low trust in test team
- Commercial constraints
  - Must not find fewer / less significant bugs
  - Must not stop or slow down
  - No increase in team size / budget

© Workroom Productions 2002
www.workroom-productions.com

Strategy

Stay within constraints

Deal with change rather than enforce stasis

- Use lightweight methods
- Encourage learning process
- Generate dynamic, up-to-date metrics

Three goals

- Control scope
- Control work
- Measure risk and coverage

© Workroom Productions 2002
www.workroom-productions.com

Why Session-Based Testing?

Fitted existing methods

- Ad-hoc testing fits well into sessions

Improvement of test techniques

- Helps learning through feedback and review

Improvement of test management

- Control and measurement helped from day 1
- No step-change
- Still hit existing deadlines with good bugs

© Workroom Productions 2002
www.workroom-productions.com

Methods

© Workroom Productions 2002
www.workroom-productions.com

Methods: Overview

Control and recording

- Control Scope
- Control Work
- Recording a Session

Assessment and Measurement

- Assessing Risk
- Assessing Coverage

Infrastructure

- Real-time Metrics
- Social techniques

© Workroom Productions 2002
www.workroom-productions.com

Methods: Control

Scope – made of Test Points

- Test Point - piece of work
- Estimated cost
- Risk

Work done – made of Test Sessions

- Test Session - piece of time
- 1 - 4 Test Points
- Actual cost
- Coverage

© Workroom Productions 2002
www.workroom-productions.com

Example: Test Points

Test Point ID	Title / Description (Comments in brackets)	Risks	Estimated Time (mins)	Time spent (mins)	% Tested / Complete	Tester Name	Date	Total time	Time left
TP0100815-0001	Candidate Search Module: Candidates Search in Options now accessible through postings and requests (DP3.1 PP3.8	3	240		0%	Peter		240	240
TP0100815-0001a	*Add additional test point if needed*				0%			0	0
TP0100815-0002	Candidate Search Module: Speculative application (DP3.2 PP3.7 Akash)	2	240		0%	Peter		240	240
TP0100815-0002a	*Add additional test point if needed*				0%			0	0
TP0100815-0003	Integration Module: Monster integration (DP4.1, PP8.5.1-2-3, James)	2	180		0%	Peter		180	180
TP0100815-0003a	*Add additional test point if needed*				0%			0	0
TP0100815-0004	Integration Module: CV Online (DP4.2, PP8.1.1-2, Sanjay)	2	180		0%	Peter		180	180
TP0100815-0004a	*Add additional test point if needed*				0%			0	0
TP0100815-0005	Integration Module: Totaljobs (DP4.3, PP8.3, Sebastien)	2	180		0%	Peter		180	180
TP0100815-0005a	*Add additional test point if needed*				0%			0	0
TP0100815-0006	Integration Module: Medlapp (DP4.4, PP8.2, Sebastien)	2	180		0%	Peter		180	180
TP0100815-0006a	*Add additional test point if needed*				0%			0	0
TP0100815-0007	Requests Module: Open-ended interview questions (DP5.1, PP6.3, Lloyd)	2	180		0%	Pinal		180	180
TP0100815-0007a	*Add additional test point if needed*				0%			0	0
TP0100815-0008	Requests Module: Additional Candidate Fields: Salary scale or grade as well as Salary value (DP5.2.1, PP6.4, Lloyd)	1	180		0%	Pinal		180	180
TP0100815-0008a	*Add additional test point if needed*				0%			0	0
TP0100815-0009	~~Requests Module: Additional Candidate Fields: Job type classification or category (DP5.2.2, PP6.2, Sumsun)~~	1	120		0%	Pinal		120	120
TP0100815-0009a	*Add additional test point if needed*				0%			0	0
TP0100815-0010	Requests Module: Additional Candidate Fields: Description of Bank/branch location (DP5.2.3, PP6.5, Anand)	1	120		0%	Pinal		120	120
TP0100815-0010a	*Add additional test point if needed*				0%			0	0
TP0100815-0011	Requests Module: Additional Candidate Fields: Location "Region" for travel (DP5.2.4, PP6.6, Lloyd)	1	120		0%	Pinal		120	120
TP0100815-0011a	*Add additional test point if needed*				0%			0	0
TP0100815-0012	JobsatJP: Anonymous/confidential applications (DP6.1.1, PP7.1.3, Sumsun)	3	60		0%	Selena		60	60

© Workroom Productions 2002
www.workroom-productions.com

Methods: Control

Recording a Test Session

- Plan, Expectations, Actions, Observations
- Bug details
- Actual Cost
- How done are we? Ì Coverage

Allows

- Review
- Forgetting
- Retrospectives

© Workroom Productions 2002
www.workroom-productions.com

Example: A Test Session

Test Description		
Test ID	Risk	Date + time done
Your name	IDs, Machines, Files and other resources used	
Estimated time Actual time	How much more testing does this need? % complete?	

Plan

Notes

© Workroom Productions 2002
www.workroom-productions.com

Methods: Assessment

Risk

		Likelihood of failure	
	Risk	High	Low
Cost of failure	High	3	2
	Low	2	1

Coverage

- Ask the Experts - the testers!
- How done are you? How much is left?
- Collect many subjective Ì few objective assessments
- Overall figures can go down as well as up
- Ongoing learning - estimates improved

© Workroom Productions 2002
www.workroom-productions.com

Methods: Support

Real-time metrics

- Immediate feedback
- State of *testing*, not of system
- Fast, frequent, informed re-planning

Soft skills/Social Techniques

- Experimental Approach
- Ongoing learning
- Communication
- Empowerment

© Workroom Productions 2002
www.workroom-productions.com

Example: Metrics

We are 95% complete
38% of our tests have been successful
We think we have 7 hours left

No. failed tests	55
~~No. still not fixed~~	~~#NAME?~~
No. fixed not tested	1

% planned tests complete	95%
Time taken so far	125.15
Est. time for these	100.7

	total	left
Pre-test estimate of time	105.7	-19.45
Post-test new guess	132	7

		All tests		Planned tests		Completed Tests		Successful
					Est.		Actual	
		Number	Hours	Number	Hours	Number	Hours spent	Number
	Total	**139**	**182.5**	**94**	**92.6**	**89**	**106.1**	**34**
H	High risk	26	20.1	23	17.1	20	18.9	6
M	Med risk	68	80.4	48	46	46	54.5	18
L	Low risk	45	82	23	29.5	23	32.8	10

Completion	
%Done	%Successful
Number	Number
95%	**38%**
87%	30%
96%	39%
100%	43%

	All retests		Planned retests		Fix tested		Successfully fixed
						Actual	
	Number	Hours	Number	Hours	Number	Hours spent	Number
Total	**64**	**13.3**	**63**	**13.1**	**63**	**19.05**	**49**

Completion	
%Done	%Successful
Number	Number
100%	**78%**

The spreadsheet looks consistent

© Workroom Productions 2002
www.workroom-productions.com

Did it work?

© Workroom Productions 2002
www.workroom-productions.com

Results

Short-term

- Stayed within business constraints
- Improved accountability, reporting, trust
- Manager and team 'in control'

Long-term

- Still in use a year later
- Metrics used within business as a benchmark
- Initial processes enhanced and expanded
- Improved motivation, reduced 'churn'
- Used for all testing - included scripted and automated

© Workroom Productions 2002
www.workroom-productions.com

Lessons Learned

© Workroom Productions 2002
www.workroom-productions.com

Lessons Learned

Improvements

- Four Groups
  - New - driven by changes
  - Retests - driven by fixes
  - Regression
  - Investigatory
- Document tracking
- Earlier involvement
  - Design
  - Just before delivery to test - late unit test

© Workroom Productions 2002
www.workroom-productions.com

Lessons Learned

Useful, visible, up-to-date metrics

- Increase openness, reduce surprise, blame
- Popular and increasingly well-used

Actively encourage learning process

- Feedback has improved estimating skills
- Session review has improved test techniques
- Test Point ownership has improved planning and scoping skills
- Weekly 2-hour self-training session as a team
- Greater interest in testing, greater motivation

© Workroom Productions 2002
www.workroom-productions.com

Conclusion

Communication

- Sessions gave us the tools to communicate
- Better communication improved trust and effectiveness

Empowerment

- Improved analysis and estimation skills
- Increased morale

Openness

- All figures available, all of the time, always current
- Encouraged trust and interest

© Workroom Productions 2002
www.workroom-productions.com

Slide 21

Conclusion

Test Sessions are an effective tool to bring control to unscripted testing

Overall lessons

- Simple measures are the best
- Favour effective communication over knee-jerk documentation
- Unobtrusive, immediate metrics allow real-time control

© Workroom Productions 2002
www.workroom-productions.com

Slide 22

More

Further details:

- Paper, updates, references etc.
  - www.workroom-productions.com
- Contact:
  - James Lyndsay
    - jdl@workroom-productions.com

© Workroom Productions 2002
www.workroom-productions.com

Adventures in Session-Based Testing

Authors James Lyndsay, Workroom Productions Ltd, Niel van Eeden, Jobpartners Ltd.

Biographies James Lyndsay is an independent test consultant with ten years experience. Specialising in test strategy, he has worked in a range of businesses from banking and telecoms to the web, and pays keen attention to the way that his clients' focus is shifting away from functional testing.

Niel vanEeden has a background in mechanical engineering, IT hardware, software and retail. Since relocating to the UK from his native South Africa, he has worked in customer facing roles and in software quality, and has been involved in testing at JobPartners since May 2000. In January 2001 he became the test manager, and is currently responsible for product quality.

Abstract This paper describes the way that a UK company controlled and improved ad-hoc testing, and was able to use the knowledge gained as a basis for ongoing, product sustained improvement. It details the session-based methods initially proposed, and notes problems, solutions and improvements found in their implementation. It also covers the ways that the improved test results helped put the case for change throughout development, and ways in which the team has since built on the initial processes to arrive at a better testing overall.

Session-based testing can be used to introduce measurement and control to an immature test process, and can form a foundation for significant improvements in productivity and error detection.

Keywords Ad-hoc testing, Exploratory testing, Session-based testing, Functional testing

Session-based Testing

Session-based testing is a technique for managing and controlling unscripted tests. It is not a test generation strategy, and while it sets a framework around unscripted testing, it is not a systematic approach whose goal is precise control and scope. Rather, it is a technique that builds on the strengths of unscripted testing - speed, flexibility and range - and by allowing it to be controlled, enables it to become a powerful part of an overall test strategy.

At the heart of the technique is the idea of effective limits. A Test Session has a well-defined start and end time, limiting its duration. During a Test Session, a tester engages in a directed exploration of a limited part of the thing being tested - it should be obvious to the tester that an action or test is inside or outside these limits. Within these limits, moment-to-moment activities are not controlled, but left to the tester's judgement. The tester records his or her activity - and includes whatever other information seems relevant; the reactions of the system, data used, conditions, diagnosis or ideas.

Session-based testing mirrors the activities of experienced testers, but is not the subject of a great many papers or books. This paper describes one situation in which session-based testing was successfully implemented.

Context

The product to be tested was an application delivered over the internet, and had been commercially live for just under a year. The application had a few hundred active users at a few dozen firms, and dealt with a large amount of incoming data submitted by many thousands of internet users,.

This application had been developed in-house by a medium size team (30-40 people total). The team continued to develop the application, and released a new version of the application every two weeks or so. Work was driven by a semi-formal change request process.

Although customers were satisfied with the overall service, there was a perception within the company that the quality of the product had to improve. Something in the development process (requirements, analysis, design, coding, testing, infrastructure, release) had to change - and unsurprisingly, attention initially concentrated on the test process.

The existing test process was immature, and the five-member team had little experience. None of the team had experience of a well-run test process. The precise nature of the problems cannot be detailed in this paper, but the process exhibited the following common characteristics.

- Reactive - and therefore uncontrolled, and not necessarily focussed on important areas
- Could miss important bugs which had an immediate effect on customers
- Could not produce reliable information about the readiness of a release, and was not trusted.

The test team were active users of the mature bug tracker Bugzilla. This tool was central to the team’s processes, and drove much of the fix/retest work in the coding and testing teams. They had been using the tool for the life of the product, and had a well-established bug list.

To help initiate the changes, the company bought in experience, and engaged one of the authors of this paper (James Lyndsay) for forty days spread over ten weeks.

Constraints

With a mandate for change, management were supportive of changes within the test team and their test processes. However, the changes made had to stay within existing budget and resource. The test effort needed rapid improvement, yet at the same time, the coders were to increase the rate of introduction of new features, and many known bugs would be fixed for retest.

Existing testing found good bugs, but in a haphazard way. The most significant issues - particularly those caused by data problems and often characterised by intermittent symptoms - were most regularly found during ad-hoc testing, and the team were reluctant to move away from a proven approach. Scripted tests would be ineffective, and would be resisted by management and by the test team.

The team were already stretched. Any time spent on training or setting up new procedures would have to be saved elsewhere. However, as the existing process was inefficient, time savings were not hard to find.

Basic strategy for test improvement

We needed a simple process which would improve on the existing ad-hoc methods. To allow us to do this, we had to introduce some element of measurement, so that we could see which parts of the process were working, and which parts were not. We also needed to introduce control, so that we could define our tasks, record our actions, and so repeat or improve them.

Canter/Derr had done work at e-greetings.com which inspired the team. Their paper (*A Case Study in Extreme Quality Assurance*, referenced below) helped the team to believe that they were not alone, and that similar problems and pressures had been overcome in other organisations. However, Canter/Derr's approach introduced change well outside the test team, and while desirable, could not be implemented under the existing mandate.

James Bach's exploratory test methods (referenced below) meshed well with James Lyndsay's existing 'Empirical Test' techniques, and Jonathan Bach's practical implementation of session-based test methods (described in the paper *Session-Based Test Management*) seemed to offer a useful and practical starting point. Bearing in mind the need for a simple, lightweight process that would form the basis for ongoing improvement, we set out to do the following things:

- Control the scope of testing
- Control the work
- Measure risk and coverage

Session-based methods

Session-based testing parallels the way that many experienced testers approach unscripted testing. While not a new technique, it has not been formalised - and there are no hard and fast rules to its execution. However, session-based testing is characterised by (at least) the following:

- A test session is a unit of time, generally a couple of hours long. It is uninterrupted, as far as possible, and its limits are well-defined.
- During a test session, testers test something specific. They may test a feature, a characteristic, a business scenario - they may hunt bugs or introduce failures. These limits are less well-defined, but they are defined *before* the start of the test.

By introducing these limits, session-based testing seeks to focus tester attention, allowing control, increasing the reliability of metrics and the repeatability of tests, and limiting the cost of poor exploration.

The methods below allowed the team to:

- Control the scope of testing
- Control their work
- Assess coverage
- Assess risk and set priority

Controlling the scope - introducing Test Points

The team had no existing test list, and the project as a whole did not have uniquely-identified requirements. Each release introduced a wide range of new tests, and although test scope was driven in part by bug fixes, there was no list of new tests or record of tests done.

We needed some sort of a list of tests, to enable us to:

- select tests and so drive work
- consciously omit certain tests
- make easy and repeatable assessments of the state of testing
- avoid duplicates
- preserve important information, allowing members of the team to move on
- simplify communication within the team, and extend communication outside the team
- generate reliable statistics

The items in this list might be tests, but the concept of a test in unscripted testing is very different from a scripted test. We tried to avoid some of the characteristics of unscripted testing by carefully choosing the things we would control. The items on the list:

- Would not be single tests, but aspects of the application that needed to be explored. A tester would typically conduct many tests in their exploration.
- Needed to be independent units of work - not steps along a path. A tester could start or finish with any item.
- Needed to be based on a wide range of sources. These included change requests, functional requirements, development information, release notes , regression test requirements - and on conversations over lunch, inference from unspoken topics, eavesdropping, wiretapping and covert midnight operations.

To avoid confusion with 'tests', we call these items Test Points. Our test points have the following broad characteristics:

- A Test Point is a unit of work and typically takes between 20 minutes and 4 hours. This estimate of duration is first made at the point when the Test Point is defined, and can be used as a simple metric for the *cost* of the test. It is refined during testing.
- Each Test Point has a simple *risk* assessment. This assessment is also done as part of the process of defining the Test Point. If a Test Point has a range of risks, it is split.
- Test Points are retained from one release to the next. Some Test Points may only be explored rarely, some become part of a set of regression Test Points, some crop up each release but their exploration changes as the functionality changes.
- Every piece of test work has its associated Test Point - including test work from more formal methods and work for non-functional testing.

Example Test Points:

- Is a field for 'Salary' offered as an optional input at all appropriate points?
- Examine User Access Control, using usertype *xxx* and usergroup *yyy*
- Does the 'Forgotten my password' option ever fail to send an email?
- Check navigation in 'Options' part of application, paying particular attention to 'back' button functionality within application, and within browser.
- Check button text within 'Options' part of application for each language offered

The list of Test Points is dynamic - additions are made frequently, based on bugs found, new understanding, unanticipated functionality and fixes delivered. We currently hold the list in a spreadsheet that can be accessed by the team at all times, but the same job might be done as well, or better, by a database accessible over the internet. The list can only be changed directly by the test team, but can be (and is) accessed by many others. Regular users outside the immediate test team include the Development project manager, Professional Services and senior management.

Each Test Point is associated with the following information:

Test point ID	Unique ID. We found it useful to include the release number in this unique ID. Test Points related to bugs also had the bug ID as part of their unique ID.
Title/Description	Enough to set the boundaries of the test
Risk	Simple, repeatable risk assessment. See 'Risk and Prioritisation' below for details
Estimated time to complete	If the estimated time to complete is >4 hours, it may be an indication that the TP should be split into two simpler parts. It's hard to do useful exploration in less than 15 minutes, so a very short time might indicate that something needs to be re-assessed. This estimate is updated after test execution to be time spent ÷ % complete Also called 'Cost'
Time spent so far (this release)	Updated as test sessions are completed. Reset after release - so it records the time spent to far this release.
% necessary testing completed	Basic estimate. Will be at 100% if testers feel that the TP has had enough attention to be passed to the customer. May start at 100% if TP has no need to be tested. Will generally increase with work done, but may go down if the TP looks like it needs more testing.
Time remaining	Calculated as Estimated time to complete * (1 - %complete)
Tester	Named Tester, responsible for this Test Point
Documentation	Cross-references to relevant documentation

We needed to have the best assessment of risk and cost, and felt that this would be made most reliably by those closest to the Test Points - so individual testers were made responsible for Test Points. This had the added effect of motivating the team and neatly defining their test tasks. Responsibilities include:

- identifying the documentation – and raising notice of its absence
- prioritisation of Test Points (with team and Test Manager)
- exploratory testing around these Test Points
- raising bugs found
- filling in the Test Session Report
- updating Test Point information – risk, time spent, necessary testing completed
- adding new Test Points for the release if they feel it is necessary
- talking about the Test Points at group meetings

Controlling the work with Test Sessions

While a Test Point might be described as a unit of work, a Test Session is a unit of time. A Test Point may be repeated - a Test Session is planned, happens, and is recorded. Each is unique. By setting the scope of individual tasks, controlling the time taken to do them, and requiring deliverables on completion, we controlled the work, and were able to dynamically adjust the plan.

A Test Session is an uninterrupted period, generally half a day or less. During a Test Session, testers will investigate one or more Test Points - with the minimum size of a Test Point at around 20 minutes, most Test Sessions look at no more than four. The choice of the Test Points to include in a session can be made on a number of criteria, and for planning, works well when related Test Points are chosen together. However, in action, it tends to be a very dynamic process, reflecting the team's need to react to the fast-changing priorities that are characteristic of a rapid-release environment. A plan that cannot adapt to fit circumstance is worse than useless in a changing environment.

Typically, the Test Manager controls the initial choice of Test Points for a release. The Test Manager also acts as the driver for change when necessary - and can substantially change test priority and scope during the process, and at short notice. His/her decision is based on:

- Availability of software
- Availability of test resources
- Time needed to test vs. time available
- Tests done so far, and their coverage of functional areas and risk
- Existing Plan

Session Timeline and Reporting

Before starting a Test Session, Testers print off a blank test session report and fill in the administrative details - including the Test Points they plan to explore. Test Session Reports have space for the following information:

Test Session ID	Unique ID. It was helpful to include information about the release, and about the tester doing the test session - although note that including the release ID could both cause and resolve confusion when testing (particularly fixes) in a release other than the one being used by the rest of the team.
Title/Description	Enough to set the boundaries of the session
Contents	Test Points covered in the session (often 1, rarely more than 4)
Administrative details	Tester name, release, date + time etc.
Notes	To be filled in while testing, although system conditions, available data etc. is often noted before the start of the session.

While exploring those Test Points, the testers use the Test Session report to record data, impressions, problems, queries, possible bugs, bugs raised, diagnosis and so on. The reports aren't formal, or neat, but they are good records - and got better as the testers became more experienced.

A Test Session is a timed activity. At an appropriate point around the end of the allocated time, the session ends. Testers are not encouraged to spend more time than planned on any one Test Session - although the Test Points may be returned to in a later session.

The deliverables at the end of the Test Session are:

- completed session report filed appropriately
- updated list of Test Points

It is important to remember that although the Test Session may be finished, testing may not be over. Although exploratory testing can be controlled by imposing an end-time, problems found during testing can result in a greatly increased - or decreased - estimate of the time necessary for adequate testing. Dealing with this was an important factor in deciding what to measure.

Review of Test Session Reports

Although simply writing a Test Session Report helps the testers (see below), recording the events allows Test Sessions to be reviewed after the event. This helps different people in a wide range of ways.

- Helps testers and the Test Manager when setting the severity of a bug, looking for duplicates etc. It is particularly helpful when the severity is queried
- Helps decide how to approach testing if the time needed is more than the time taken so far.
- Helps the Test Manager and individual testers to control and improve the quality of testing. The Test Session Report is a useful coaching tool; the coach does not have to sit with the tester for the duration of the test, and more than one session can be reviewed and compared. The Test Manager also gets a good idea of the approach to testing of each tester, and can re-direct as appropriate.
- Helps the test team to look back on a session, to be reminded of their actions and results, to examine the data used in the light of new information etc. Reviews also allow testers to re-interpret their conclusions, or to use multiple session reports for diagnosis or examples.
- Helps testers to share information - testers can swap Test Session Reports when handing over areas of responsibility, or can compare their different approaches to the same Test Point.
- Helps coders and designers get over the 'if I haven't seen it, it isn't a bug' problem. Illustrating the problem by showing the data used can be very helpful, and noting times helps to match problems with known system events - network failure, batch job etc.
- Having a tangible and discussible record of test work available for review helps the business have confidence in the test process, and the reviews themselves help the testers and the business understand each others priorities and desired approaches.

Assessing test coverage

Coverage - a brief explanation

Coverage is a measurement of what has been done as a proportion of what could be done, and is an important test metric. It plays a key role when assessing whether the product has had enough testing to give the business confidence that it can be released. When broken down by area, it can indicate those aspects of the product that have had enough, or too little testing, and so becomes a vital input in adjusting the plan as circumstances change.

There are many ways of measuring coverage - Cem Kaner lists 101 coverage metrics in his paper Software Negligence and Testing Coverage, some serious, some not so serious. Although they are each measures of 'testedness', they do not all measure the same thing - so 'good' coverage by one metric may be inadequate when measured by another (i.e. Statement coverage vs. Service Level Requirements coverage). It also can be hard to measure some coverage metrics without instrumentation set up prior to testing (i.e. branch coverage), and others can be impossible to measure in some projects (i.e. requirements coverage in a project without formal requirements). Indeed, while comprehensive testing results in good coverage measured by most methods, testing driven by a single formal technique can result in excellent coverage when measured by one method, and poor coverage when measured by another.

Formal methods of coverage measurement do not work well with unscripted testing, and can introduce complexities to test execution that work against many of unscripted testing's better qualities. We introduced a measurement of coverage that was simple to assess yet gave a good indication of the testing that had been done compared to what needed to be done, the readiness of the system for live operation, and the parts of the system that needed more testing.

Our coverage metric

We based our coverage metric on a subjective assessment of 'testedness'. By using Test Sessions to focus and control the work and take many small-grained assessments, we hoped to be able to make an objective measurement from the combination of subjective estimates.

At the end of a test session, the testers recorded two figures;

- the amount of time they had spent testing each Test Point.
- an estimate for how ‘tested’ the test point was, as a percentage.

As these figures were recorded, other figures used in planning future work were calculated (new estimated test cost, time remaining, overall coverage). Calculating these figures on the fly gave the testers immediate feedback and a more concrete perspective on their estimate, helping them to make their estimates more consistent.

Example: *A Test Point was estimated at 3 hours. The tester completed 3 hours exploration, but felt she had not yet tested all the things she wanted to test, and estimated she was 75% done.*

On recording these figures, two figures were calculated automatically. The Test Cost was raised to (3÷75%=) 4 hours. The time remaining was set to the 1 hour that remained.

	Test Point	Cost (Est. Time)	Completed	% complete	Cost/time remaining
Before	assess email functionality triggered by …	3 hours	0	0%	3 hours
After	assess email functionality triggered by …	4 hours *calc*	**3** *input*	**75%** *input*	1 hour *calc*

We hoped that the tester’s expertise would enable them to make a fair judgement – but recognising that the testers were not yet expert, we encouraged the testers to spend time looking back over past estimates and discussing their accuracy – and, if necessary to re-assess their current estimates. The team found this useful, and were also helped by the immediate feedback from their estimates into figures used for planning.

Special cases -

- Test Points that needed no testing were at 100% coverage from the start. This was achieved by setting their test cost to 0 and including a special condition in the number crunching functions,

as keeping the cost and setting work done to match it introduced confusion. Note that such Test Points are excluded from some summaries.

- Tests where the tester had done three hours - but kept the estimate at two - were not listed as 150% done, but were fixed at 100%. However, they were highlighted - to highlight input mistakes, and to allow the Test Manager to review the test with the tester and avoid the issue next time.
- A 1-hour test that the tester estimated was 10% complete needed 9 more hours. Some tests genuinely needed this much more testing, but others could be more accurately re-estimated, or given special attention by experienced team members or the business to see if the testing could be made more efficient. Giving feedback to the testers by exposing the planning figures helped avoid this problem.
- At the start of testing for each release, the amount of testing completed for each Test Point was reset to zero, and estimated cost and risk was re-assessed for existing Test Points.

Combining estimates for more accurate overall figure

Although each individual estimate was rough and subjective, their worth improved when they were combined. Note that a less blunt combination, split by risk, would be used for most decision making. Estimates were combined as follows:

- Overall test time remaining, derived from the sum of (estimated time - completed time). This was useful throughout testing - it gave an estimate of the effort needed at the start of the process, and tracking it in real time allowed the Test Manager to see how well the team was staying on target. It is worth noting that there was always more testing to do – we never reduced this figure to zero.
- Coverage for the whole release, derived from the (sum of time completed)/(sum of estimated time to complete). This gave a one-figure summary of the progress of testing. Just as the % complete of individual test points could go down as well as up, so could the coverage. This was entirely appropriate - and was usually the result of the addition of new test points from bugs, or as a result of underestimation of test cost / complexity.

Conclusion

Testers are notoriously bad at informal estimates of how much testing is needed. We aimed to improve this ability – and, in part, we achieved this by placing the skill at the centre of the planning process.

These metrics assessed not the state of the system, but the state of testing. While necessarily subjective, the metrics turned out to be repeatable - different testers came up with broadly the same estimates of completion, particularly as they worked together, and grew more experienced in their methods, in testing, and in the system. Because the system was assessed in small grains, individual errors in estimation at a Test Point level were small compared to the overall figures. The team updated their figures regularly and often, and the metrics became an important feedback to the team, helping them perceive a common goal and giving good feedback into their process of ongoing improvement.

Management understood that the metrics gave a reliable indication of how the testers felt about how well they had tested the system. This, in combination with the number, type and severity of problems found and fixed, was soon an important part of the go-live decision. Rather than wait until the end of testing to find out how good the system was, the decision could be assessed earlier, allowing warning of problems and re-prioritisation of effort. The ‘coverage’ figure was both useful and effective.

Risk and Prioritisation

The test team needed to be reactive. Their reactions needed to be fast - but could not be allowed to be uncontrolled. They needed a means of assessing new circumstances against existing tasks quickly and accurately. In making decisions about importance and urgency, it was good to have information about risk and cost.

Risk is a combination of cost of failure and likelihood of failure. To allow assessments would be consistent and repeatable, we needed a simple method of assessing risk. We started with possibly the simplest:

		Likelihood of failure	
	Risk	High	Low
Cost of failure	High	3	2
	Low	2	1

Each Test Point was assessed for risk. Work was prioritised by risk, and metrics were split by risk. We found that the risk associated with a Test Point was easily communicated to the business, and broadly matched their expectations.

Note that test work did not concentrate simply on the highest risk test points. It was important to spread the test effort in case the risk assessment was wrong, and we typically spread the test effort so that all major functionality had some testing. However, at the end of testing, high risk Test Points generally had better coverage than low and medium risk Test Points.

In some releases, coverage of high-risk test points never matched the coverage of low- and medium-risk test points. This was due to two factors

- high-risk elements released to testers / fixed close to live release deadline
- the more testing that was done in high risk areas, the more the testers felt they needed to dig deeper

We found that the proportions of Test Points over time matched the following profile:

Risk	% TP by number	% time required	% time spent
3 (highest)	15%	20%	25%
2	35%	40%	40%
1	50%	40%	35%

This profile indicates that individual high risk test points generally required more time than low-risk ones. The difference between %time required and %time spent indicates that when the team were running out of time, high-risk test points were given priority over low-risk test points.

Note that these figures include a substantial proportion of regression-test-related Test Points, which are generally judged low- and medium- risk. This lowers the *proportion* of high-risk tests. If these figures were for a single release concentrating more on new functionality and fixes than regression testing, it would have a greater proportion of high-risk tests.

Building on the Process

The new methods were not adopted immediately, but good initial results encouraged the team to persevere. The methods have been in place since June 2001, and have been supplemented with a number of improvements and refinements.

Process and Techniques

- Some Test Points now include scripted and automated elements if appropriate and effective.
- Test Points are grouped not only by risk, but also by family. The four families are:
  - Test Points for new functionality
  - Test Points for fixes
  - Test Points for regression testing
  - Test Points for QA investigation
- The Test Team has a process of ongoing learning:
  - A ten-minute daily meeting keeps the team focussed and together, and highlights common issues
  - A two-hour training session, each Friday afternoon, helps the team take a longer-term look at the application, at test techniques, tools and at process improvement. It allows them to share their experience, and encourages them to think of testing as a skilled job.
  - The figures and feedback have helped the team to improve their estimating skills, and have encouraged their planning skills
  - The practice of recording their test sessions allowed review and improved their ability to test without scripts. Sessions were reviewed by peers, by senior testers, and by the testers themselves after the fact.
  - The responsibility for individual test points encouraged ownership and interest, improving test analysis and planning skills
  - Communication improvements driven in part by test sessions encouraged skill sharing and greater interest in the general process of testing and its literature

Metrics

- Refined figures improve the focus for individual testers and Test Point families
- Automated metrics provide a 'Test Dashboard', giving instant feedback of overall summary and planning figures as test are completed
- Improved metrics allow more complete views and more reliable assessments
  - Identification of bugs found in production has allowed the team to start measuring test effectiveness
  - Measurements of estimation accuracy help improve the estimation skills of the team

Early involvement

- The Test Team is now involved earlier in the process, and is able to discuss design and assess code before delivery to test
  - Testers are involved in document review – and find design problems before coding
  - Testers spend a short period testing code with a coder, at the coder's machine, after unit testing but before promotion to the generally available code. Not only do the testers increase their familiarity with the deliverable (and the coder), but they also spot simple problems that have not been apparent from the unit tests.

Changes outside the Test Team

- Improved documentation references allow testers to link each test point to each identified document and track dates for its completion and availability. Documentation availability has made work more intensive, but more focussed, and the team is able to be more productive

- Better documentation has improved the linkage between tests and business requirements
- Visible improvement in the Test Team and their processes encouraged process improvement elsewhere, particularly in the generation of inception, design and implementation documentation, and in the processing of urgent requirements for live fixes.

Results

Perhaps the most significant result is that testing is now seen as a vital aspect of the development process, rather than incompetent, obscure, and a hindrance to productivity.

'Tangible results' listed below are those characterised by a deliverable or directly measurable quantity. 'Intangible results' are those characterised by a change in perception.

Immediate, Tangible results

- The team produced a useful coverage metric from the first pass through testing, and showed that riskier parts of the system had received more attention. This has subsequently become a central metric.
- The Test Manager was able to review testers' work off-line i.e. without being with the tester while the work was done.
- Test Session reports were a useful record of unscripted test activities, where previously there had been nothing but a bug report.
- Because few measurements had been taken before the introduction of these methods, it was hard to get real results in the early stages. However, the rate at which significant bugs were found stayed the same on the introduction of these methods, and increased for the next five months – although this reflects the increasing complexity and size of the code (as so the number of bugs to be found) as well as process improvements.

Longer term Tangible results

- The product is more stable and has fewer outstanding bugs
- In the last few months, the rate at which significant bugs are found has fallen, although new functionality is still being introduced as fast as ever. This reduction is thought to be due to the increasing quality of the code, rather than test failure. The test team's skills and procedures have been instrumental in helping the designers and coders achieve this improvement.
- The test team's metrics are used as a basis for improvement by non-test teams.
- Problems outside the test team were no longer obscured by test team problems, and could be identified and addressed. This applied particularly to documentation, which was refined to fit a useful purpose rather than simply generated as part of a deliverable, and to the way that live problems were handled.

Immediate, Intangible results

- The test team felt in control of their work. They could see the size of it, see how much they had done, and what was left. They could decide what to do next, and back up those decisions.
- The Test Manager felt more confident in controlling and planning testing.

Longer term Intangible results

- The coders felt that problem logging and diagnosis had improved
- Visibility of test process and progress allowed other teams to trust the Test Team's information, and the communication that the trust enabled resulted in a 'virtuous circle'.
- The introduction of more formal, scripted testing was easier as the test sessions helped the testers to think more rigorously, and to work in a systematic and analytical way.
- The test team take a much greater interest in their jobs, and morale has improved. The team generate three or more good ideas a week, of which at least one is implemented.

Lessons Learned

The team needed a process that enabled learning and encouraged improvement. We recognised that while we might not start out well, the right attitude and the right tools would allow us to develop an effective and efficient process.

Three overall factors

While we learnt a number of useful lessons, three key factors stood out. These factors underlay many key parts of the approach - without any one of them, the approach would have failed

- Communication. The methods above gave us the tools to communicate within and outside the team. By improving communication, we felt that we reduced the number of misunderstandings. Communication also helped to increase trust, which both improved personal relations, but also helped facilitate solutions.
- Empowerment. Testers were individually responsible for Test Points. They were encouraged to measure their own progress and their estimates were trusted. Morale improved, and the test team was seen as an interesting and valuable place to work.
- Openness. The list of Test Points, the work done and the work needed were available to the coders and designers at all times. Although initially attracting little interest, the fact that the information was always available, and always up-to-date, encouraged the other teams to work with the test team, take an interest in their activities, and trust their work.

Cost estimation

Test cost - in terms of the time a test would take - was a vital metric. By comparing the actual cost with the initial estimation, we hoped to improve estimation skills.

Within the first couple of cycles, it became obvious that the whole team were not only underestimating the cost of risky tests, but their estimates got worse for longer tests. This was bought to the team's attention, and estimates improved. Analysis of a recent release indicated that 5% of tests done would have required more than twice the estimate to be fully tested, and that estimates were within 35% of the required time for 70% of the tests. Over the whole release, the time the testers felt was needed to fully test the release was 25% more than their original estimate – but note that this straight average is deceptive – tests that need less time than estimated cancel out those which need more.

We believe that the testers are accurately estimating the time needed to explore a Test Point to an acceptable level. This is supported by the improvement in test effectiveness (bugs found in live / all bugs found). This is an important skill, and allows the Test Manager to plan and react with confidence. It also allows the rest of the business to trust the testers estimates.

Test Points - analysis

The analysis needed to define a list of Test Points was not trivial, but the process of generating Test Points gave form and repeatability to a necessary analytical task that was not otherwise addressed. Although this analysis was unfamiliar, it was easy for the test team to see when they had finished - they had a definition, a cost and a risk. As releases and test cycles passed, the team got better and faster at doing the analysis.

Test Points are likely to overlap - particularly when defined by someone who is not familiar with the list. The Test Manager plays an important role in identifying duplicates and overlaps, but his/her job is made easier because the list is public. Resolution is made simpler because each Test Point has a Tester assigned.

The task of defining Test Points has been made easier by improved documentation and tester involvement in design and implementation meetings. The team finds that not only does the increased familiarity with the requirements help, but that the extra time to think improves the scope of their analysis.

Test Points - writing descriptions

One of the difficult parts of writing Test Points is to define a non-trivial exploratory area that is well-defined enough for the testers to know what is in the area, and what is not. Consideration of three aspects helped:

- time limit - areas that might take a couple of hours to explore were easier to define than larger areas.
- risk - if an area of exploration had risky parts, and not-so-risky parts, it was probably two Test Points.
- wide range of different approaches that could be taken - one Test Point might examine the functionality of 'Back' buttons and return navigation throughout the application, while another might look at ways that email-sending functionality could be broken.

Test Sessions

Testers may find it productive to collaborate on a test session, particularly when the area is unfamiliar. Some test sessions, for functionality hidden from the users, may be performed in collaboration with the coders. Collaboration with coders is also used in short test sessions (called 'splash testing' on-site) that are performed immediately before the coders incorporate new functionality into the main body of code.

Test Session Reports

We found that the act of writing stuff down encouraged better testing, as testers could refer back to what they had done, and leave distractions for later without losing track of them entirely. They could draw diagrams, annotate previous notes and use colours and sticky labels – and, under pressure, most testers found it faster to write than to type (this may because the testers use the same PC to test as to run the word-processor). Testers using paper documentation did not have to worry about a PC failure causing the loss of their session log. We also found that when the sessions were reviewed, a hand-written log was a better visual mnemonic than a typed or on-line document.

It is worth noting that each member of the team has a different style of testing, and each produces a slightly different style of test session. One of the team feels that the advantage of hand-written documents are outweighed by the ability to use copy/paste – particularly given the legibility of his handwriting – and prefers to use a word-processor and other PC tools to record his Test Session report.

Testers got better at writing session reports - partly as a result of reviews, and partly as they started to use the reports as a tool in themselves

Maps

We found that a map of the navigation of the product helped the testers with aspects of the system that manifested in many places, acting both as a breadcrumb trail and as a checklist. It also helped them plan their testing and estimate completion more accurately. However, the map cannot be constructed automatically and the team has had problems with obsolescence.

Documents needed

An important change in the development process was the introduction, enforcement and tracking of standardised documentation. These documents helped the testers explore areas more effectively, and the tracking helped them plan their activities to match the design and coding teams schedules and events.

- Inception Document: contains the original idea. Describes the way the new feature needs to work, contains the requirements and the design logic.
- Design Document: Analysis of changes needed to database, classes, modules and pages
- Implementation Document: Details changes actually made. Signed off on delivery of code to test.

Rapid reactions and real-time results

Once the tracking spreadsheet had been set up to include a real-time test dashboard, the Test Manager always had an up-to-date picture of the tests that had been done, the tests yet to be done, current issues, coverage and risk. This knowledge allowed the team to react more quickly to changing circumstance, without losing track of the overall aims of testing. The improved response had a direct effect on the way that other teams and staff interacted with the test team, and increased trust and communication.

Fixes, retests and regression tests

The team quickly adopted test sessions to drive and record retests and regression tests. Test sessions allowed a faster response to the arrival of a fix, and served as effective proof that the fix had been well implemented and tested. The team found that looking at the session for the test when the problem had been found helped plan the retest.

Test Points are now classified into four families (see "Building on the Process" above), one being 'Regression Test Points'. This important improvement has resulted in the development of a comprehensive regression test set. A selection of Test Points for 'new functionality' is added to the set of Regression Test Points at the end of each release, keeping the tests current. The team can isolate coverage figures by family, allowing ongoing assessment of the depth of regression test coverage for the release.

Outside the team

Other parts of the organisation can understand and read test points – and the figures summarising test progress are available at all times. The business may add to the test points, but in practice will always ask a tester to add any that may be required. This allows control, and directly assigns ownership of the test point to an individual.

We found that having improved the test process and the visibility of its results, other teams started to change their processes, as thin spots were revealed. Changes in coding practice and in the preparation of the design have been initiated partly because testing could reveal and measure the points where existing practices were not working well. Embarrassment and peer pressure can be an important motivating factor in the improvement of code quality!

Live bugs

With session-based testing, we were able to get real value from analysis of live bugs. We could look back over sessions for the current release or previous releases, and could analyse the tests done to discover how the bug had been missed. This approach, impossible with poorly-recorded ad-hoc testing, drove a multitude of small process improvements.

Mistakes and problems

We drove the testing from a single, complex spreadsheet. While this allowed good flexibility and quick improvements, it caused a number of problems:

- Corruption and data loss: The spreadsheet was shared – and sharing did not work perfectly. Summary test metrics were helpful in identifying corruption, and a few 'sanity checks' were built in. The spreadsheet was backed up regularly. These problems have become less frequent as the team have become more familiar with the spreadsheet application.
- One line per test point: A spreadsheet is not database. In particular, it does not allow a simple method of recording many actions to one item, as each item is recorded on a single row. In this case, the restriction made it hard to input and extract good information for test points which were performed more than once in a release (inclusion in more than one session, re-delivery of software after a fix, poor approach the first time etc.). It also caused problems when a test point was performed by two testers working together, or when adding multiple, dated comments. There was no easy solution to this, but the numbers involved did not unacceptably compromise the accuracy of the metrics.

There was no explicit link between Tester, Test Session and Test Point – each Test Point had a Tester, and each Test session had Tester and Test Point, but there was no linkage to allow the extraction of all the Test Sessions that had involved a particular Test Point. While this seemed important in planning, in practice the close ownership of Test Points by individual Testers meant that the information was easy to reach.

Close ownership of test points meant that testers were unfamiliar with some aspects of the application – which could lead to poorer testing if one of the testers was unavailable. Once a family of regression tests was developed, ownership of the regression tests was rotated each release to give each tester exposure to the full application.

Including the release in the ID of the test point and the test session made good sense during the first few releases. However, it could become confusing when re-doing test points that had been generated

for previous releases, or test points which were to retest an area following a fix. We now include the release ID in some, but not all of the Test Point IDs and Test Session IDs.

Naturally, Test Sessions were rarely uninterrupted. We also found it difficult to separate out time spent setting up / clearing down the test, time spent doing the test, and time spent logging bugs. Some activities that might be considered part of exploratory testing - such as talking to the coders and designers about the system and its problems - were not generally part of the test session. Work is in progress to address this.

Testers dealt with large amounts of information – and the volume of the information means that sessions can contain errors and omissions. Ideally, the testers would have liked to copy/paste directly into their handwritten session logs – but without a budget to create an impossible tool, they printed out information and stapled the printout to the session. This needed a nearby printer to avoid upsetting flow of work.

The simple risk assessment worked well in the initial stages. However, the business and the testers soon demanded a more refined scale, although no method has been decided which allows consist assessment by different people. A priority field is currently being used in conjunction with the risk assessment to plan testing.

The team are very happy with session-based testing, but this has led to some resistance to systematic methods and automated tools. The team are, however, finding success in fitting systematic methods, automated tools and scripted test cases into their familiar system of Test Points and Test sessions.

Failures

It seems obvious to perform related Test Points in the same session, and the testers found that this was an efficient approach. However, it may bias the testers assessment of the length of time a test needs – two related Test Points performed together will need less time than if performed separately.

Any testing driven by a single coverage metric is flawed - and the methods described above are indeed driven by a single coverage metric. Splitting the Test Points by risk and into families helps with this, but it would be good to see (for instance) requirements coverage being assessed simultaneously.

Each TP is a slice through the system – many overlap, and this can be seen as inefficient testing. However, inefficiencies tend to be concentrated on commonly-used parts of the application and would still exist in other approaches.

Test sessions, as hand-written records, cannot be parsed electronically, and any statistics gathered are based on information logged by the testers in addition to the test sessions.

We are disappointed that our test effectiveness metrics are historically supported by only anecdotes, as the data has been lost. We hope to be able to extract test effectiveness metrics from current data.

Conclusion

Session-based test techniques worked well on the occasion described in this paper; while staying within budget and using existing resources, they allowed unscripted testing to be controlled, refined and to add real value. They may be less effective in a more sophisticated environment, and they are not appropriate in environments that require systematic and complete approaches to test definition. However, by bringing control to unscripted testing, session-based techniques are a useful addition to the test arsenal.

In implementing this approach, we used a number of project-specific measures - as described in this paper, the methods may not fit other projects. However, they shared the following principles:

- Simple measures are the best
- Favour effective communication over knee-jerk documentation
- Unobtrusive, immediate metrics allow real-time control

Our experience has shown that, when given appropriate feedback, testers can learn to improve both the effectiveness of their unscripted testing, and the accuracy of their estimates. Central to this process is a repeatable and trusted coverage metric which allows many *subjective* assessments to be gathered into an *objective* view of the degree to which the product has been tested. Session-based testing allows the subjective assessment to be controlled so that it can be drawn together in this way.

Appendix 1: References

Session-Based Test Management
Jonathan Bach
http://www.satisfice.com/articles/sbtm.pdf
also STQE magazine V2, I6 - 11/2000 and STARWest 2000 conference notes.
also see James Bach's wide range of articles on context-based testing and other practical techniques at http://www.satisfice.com

A Case Study in Extreme Quality Assurance (XQA)
Authors: Jim Canter/Liz Derr
http://www.stickyminds.com/docs_index/XDD2561filelistfilename2.zip

Near Zero Undiscovered Defects and Shorter Time-to-Market!
Authors: Jim Canter/Liz Derr
http://www.stickyminds.com/docs_index/XUS202021file1.doc

Software Negligence and Testing Coverage
Cem Kaner
http://www.kaner.com/coverage.htm for 101 coverage

A Guerrilla Guide to Empirical Testing
James Lyndsay
http://www.workroom-productions.com/papers.html (may not be available for Spring 2002)

Appendix 2: Acknowledgements

The authors would like to thank the designers, coders and testers involved in the project for putting our ideas into practice and for supporting us with so many fine ideas. We would also like to thank the reviewers for their patience and suggestions.

QW2002 Paper 7A2

Mr. Flavio Kubagawa (QAD Inc.) & Mr. Hans Buwalda (LogiGear Corp.)

Experiences with Action Based Testing, a Case Study

Key Points

- Challenges in testing enterprise software
- Introduction to action based testing
- Practical experiences and solutions

Presentation Abstract

In 2001, QAD started a project to implement Action Based Testing (ABT), an approach for testing and test automation. In this talk Flavio Kubagawa of QAD will present how the ABT process was applied, the problems encountered along the way, the challenges due to the specific environment at QAD, and how they were resolved. The main focus of the case study shared by Flavio is the practical experiences with Action Based Testing. As part of the discussion, he will show the specifics of the architecture developed by him and his colleagues at QAD. Hans Buwalda of LogiGear will provide a background introduction to the concepts behind ABT.

About the Author

Flavio Kubagawa is a Test Automation Engineer at QAD Inc and has been working in the testing arena for the past 5 years both as a tester and a test automation engineer. Currently he is implementing the use of Action Based Testing methodology as QAD's test automation tool. Prior to joining QAD he has worked in Brazil as a developer and systems analyst for major banks (Itau, Banespa), as well as in projects for the Brazilian Navy and Brazilian Air Force, where he obtained his degree in Computer Science from Instituto Tecnológico de Aeronáutica, in the city of São José dos Campos.

Hans Buwalda, ABT Chief Architect, leads LogiGear's Action Based Testing™ research and development, including ABT Toolset™ operations, and oversees the practice of ABT methodology. Prior to joining LogiGear, Hans served as Project Director at CMG The Netherlands where he was the original architect behind the Action Words approach — an integrated method for planning, managing, and deploying software testing and test automation. Hans is an internationally recognized expert specializing in action-based test automation, test development, and test-technology management. He speaks and presents workshops at international conferences on testing concepts such as Action Based Testing, The Three Holy Grails of Test Development, Soap Opera Testing, and Testing in the Cold. Hans authored (along with Dennis Janssen and Iris Pinkster) Integrated Test Design and Automation (Addison Wesley 2001). He holds a Master of Science degree in Computer Science from Free University, Amsterdam.

LogiGear Expert Testing... Real World Solutions.

QualityWeek 2002

Experiences with Action Based Testing™ a Case Study

Flavio Kubagawa, QAD Inc
Hans Buwalda, LogiGear Corporation

QAD Connect • Collaborate • Compete

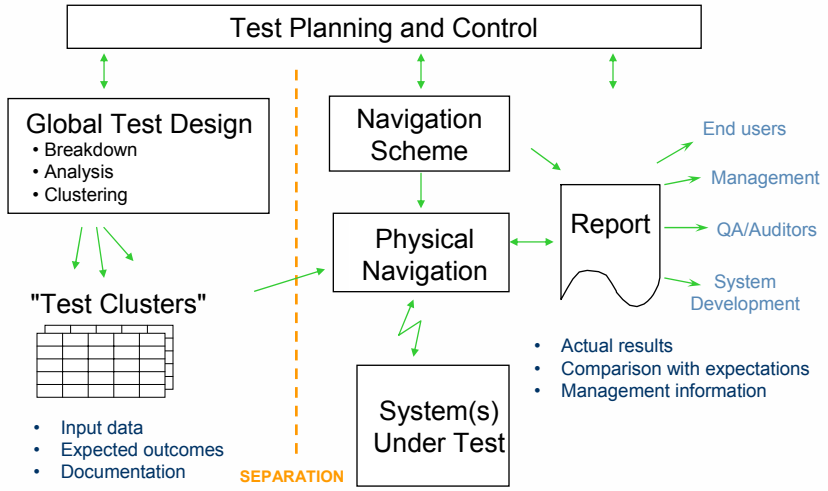
LogiGear QAD

Agenda

- Introduction to Action Based Testing (ABT)
- QAD and the MFG/PRO test project
- Application of ABT on MFG/PRO
- Experiences

(c) 2002, QAD, LogiGear, all rights reserved

Action Based Testing (ABT)



(c) 2002, QAD, LogiGear, all rights reserved

Test Lines

cluster	**EXAMPLE OF A TEST CLUSTER**			
version	1.0			
author	Hans Buwalda			
section	**1. Entering clients and balances**			
	last name	first name	account nr	balance
enter client	Green	John	458473948	1500
enter client	Wood	Anna	422087596	2100
section	**2. Money Transfers**			
	from	to	sum	
transfer	458473948	422087596	500	
transfer	422087596	785793025	1201	
section	**3. Checking names and numbers**			
	account nr	last name	first name	
check name	458473948	Green	John	
check name	422087596	Wood	Anna	
	account nr	sum		
check balance	458473948	1000		
check balance	422087596	1399		

(c) 2002, QAD, LogiGear, all rights reserved

LogiGear QAD

Test Objectives

...
TO-3.51 The exit date must be after the entry date
...

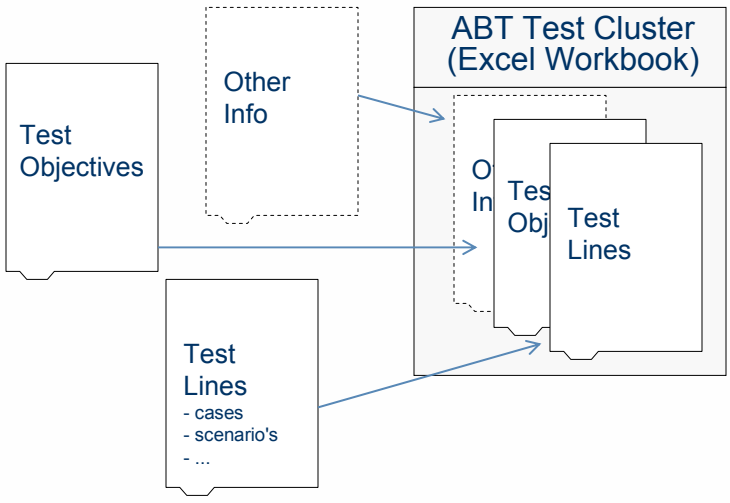
test objective TO-3.51

	name	entry date	exit date
enter employment	Bill Goodfellow	2002-10-02	2002-10-01
check error message	The exit date must be after the entry date.		

(c) 2002, QAD, LogiGear, all rights reserved

LogiGear QAD

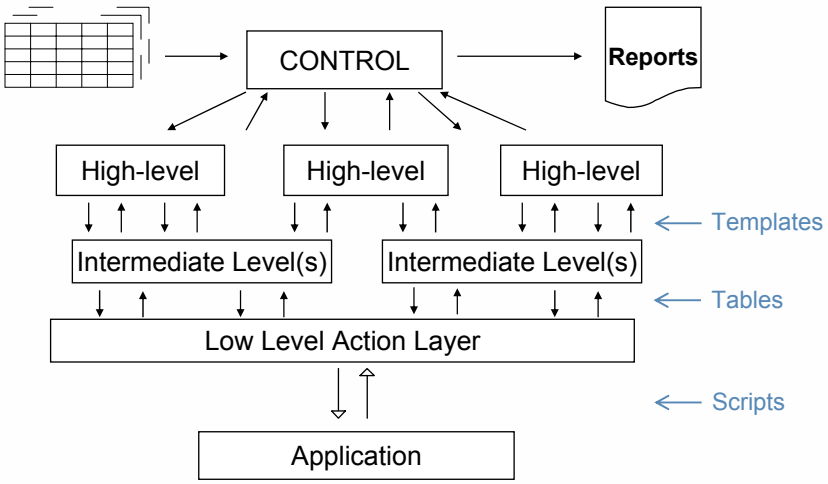
Test Cluster



(c) 2002, QAD, LogiGear, all rights reserved

LogiGear QAD

Navigation Scheme



(c) 2002, QAD, LogiGear, all rights reserved

LogiGear QAD

About QAD

- Producer of advanced enterprise software for manufacturers
  - "collaborative commerce" concept, focusing on integration between the diverse business processes
- Manufacturers of automotive, food and beverage, consumer, electronics, industrial and medical products use QAD applications
- 5,200 licensed sites in more than 80 countries and in as many as 26 languages
- for more information, please see: www.qad.com

(c) 2002, QAD, LogiGear, all rights reserved

About MFG/PRO

- Enterprise software
- Menu with over 2,700 options ("programs")
- For the test automation project, used version with a textual interface using a terminal emulator
  - there is also a newer, web based, version, but the text version had priority for the test automation

(c) 2002, QAD, LogiGear, all rights reserved

Project Objectives

- Create an automated regression test that supports:
  - Integration testing against MFG/PRO baseline for:
    - "Projects staged"
    - "Maintenance fixes staged"
  - Service pack testing
  - Possible localization testing
- Absorbs project changes
- Represents long term payoff for QAD

(c) 2002, QAD, LogiGear, all rights reserved

Criteria for Automation Priorities

- First priorities:
  - Modules most impacted by new projects
  - Legacy maintenance against baseline
- Prioritize next modules with highest defect-density
  - Based on previous release
- Remain responsive
  - Be flexible enough to reorder priorities as needed

(c) 2002, QAD, LogiGear, all rights reserved

Action Based Testing

- The navigation scheme is basically an interpreter
  - Core is the "ABT Engine", with a library of functions that is loaded from a testing tool
  - Reads the test lines from the Excel file
  - Calls functions created by the user to execute the actions and checks
  - Processes the results into easy to read reports
- Test step ☒ action line
- Test script ☒ set of action lines ☒ cluster
- MFG/PRO interface ☒ interface layer

(c) 2002, QAD, LogiGear, all rights reserved

Test Automation: Development Process

- Test case descriptions are created and streamlined from system test scripts
  - by a test specialist (Robin Kronk)
  - 700 pages = 1,600 test steps
- On basis of descriptions ABT test clusters are made and the actions are implemented with an ABT navigation scheme plus generic Winrunner functions
  - by two test engineers (Flavio Kubagawa and Edward Duran)
  - 600 Action Words, 65 scripts in Winrunner
- Tools used to support test automation:
  - Winrunner
  - ABT Engine
  - MS Excel

(c) 2002, QAD, LogiGear, all rights reserved

Test Automation: Execution Process

- Environment
  - Dedicated machine to run the tests
- Test run
  - Create a “master” cluster to string all clusters together

(c) 2002, QAD, LogiGear, all rights reserved

Test Automation: Challenges

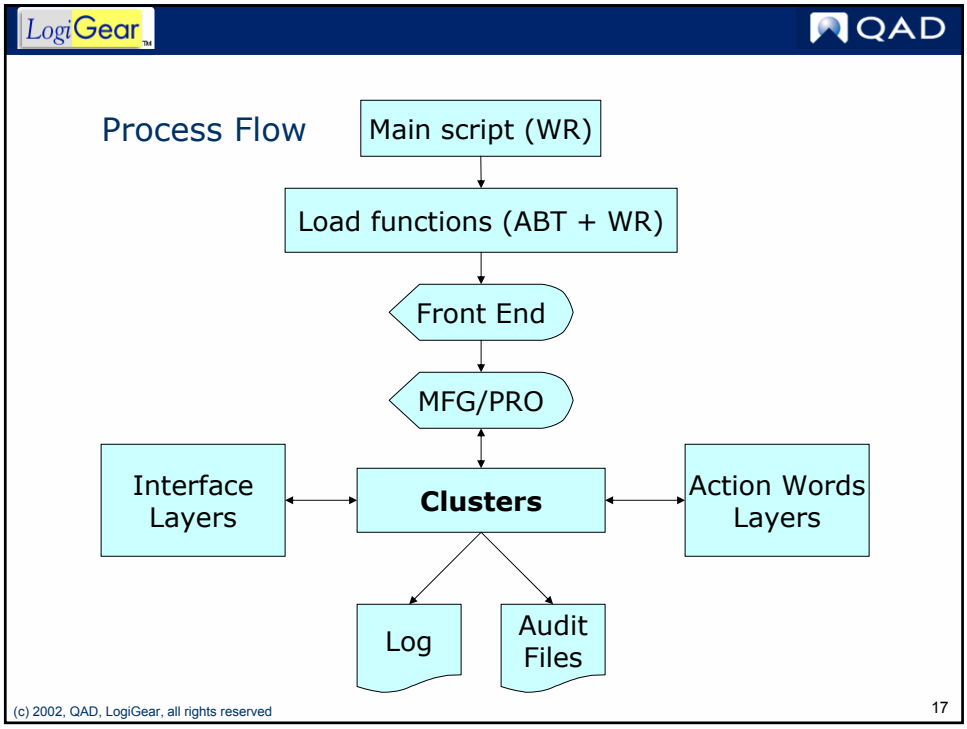
- MFG/PRO ChUI (Character User Interface) interface
  - Created WinRunner functions to work with ChUI
- Maximize reusability
  - Developed methodology to create generic action words to increase reusability
- Minimize need for additional scripts
  - Created generic functions to decrease need of coding

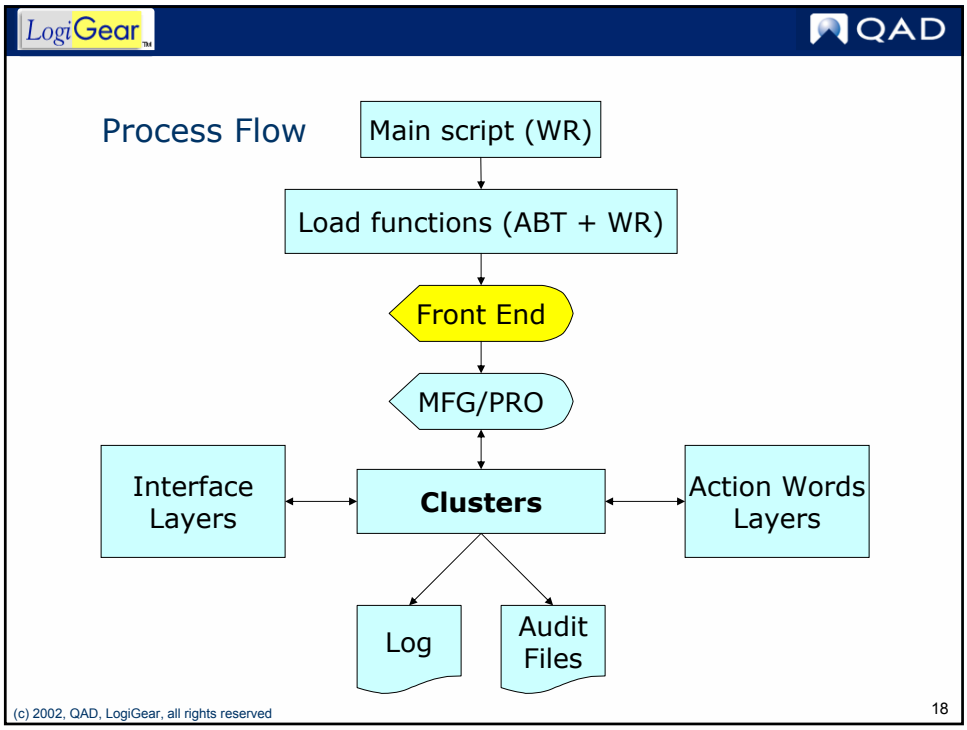
(c) 2002, QAD, LogiGear, all rights reserved

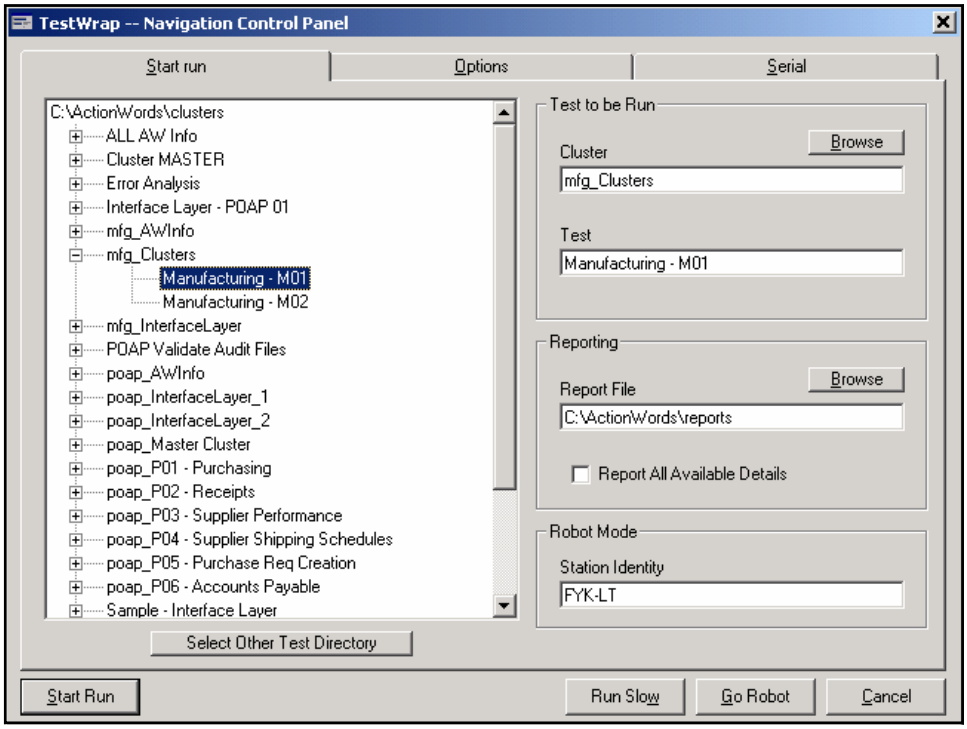
Test Automation: Generic Functions Layer

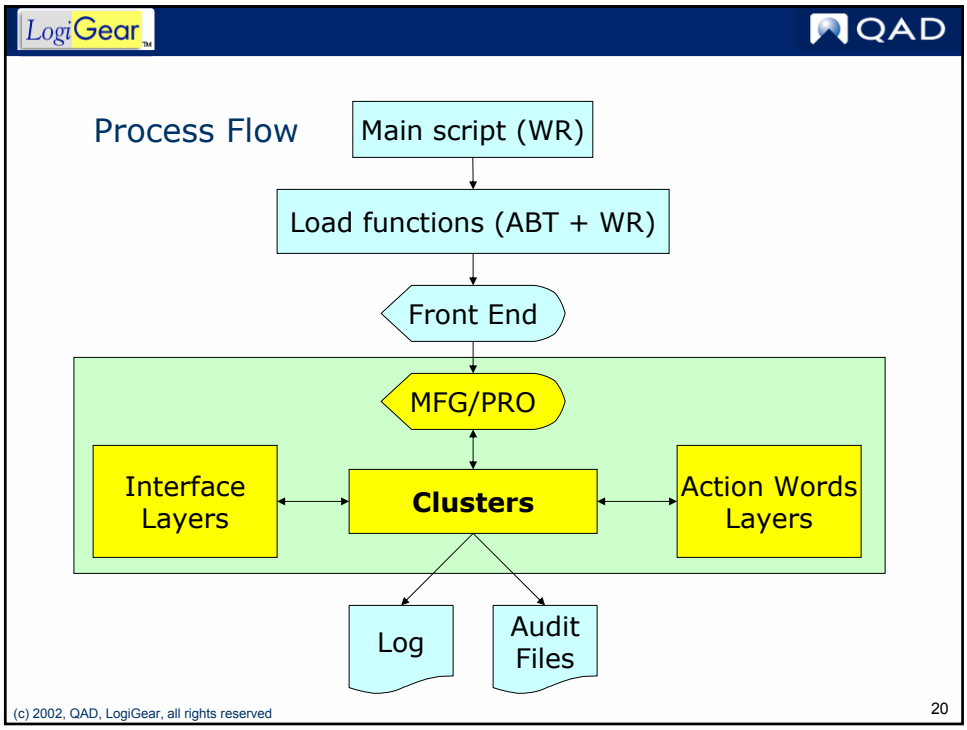
- Implements the middle layers of the ABT Navigation Scheme (without the use of templates)
- Virtually eliminates the need to create new WinRunner scripts/functions to implement a new action word
- Speeds up the process of automating test scripts
- Eliminates need for large number of automation engineers

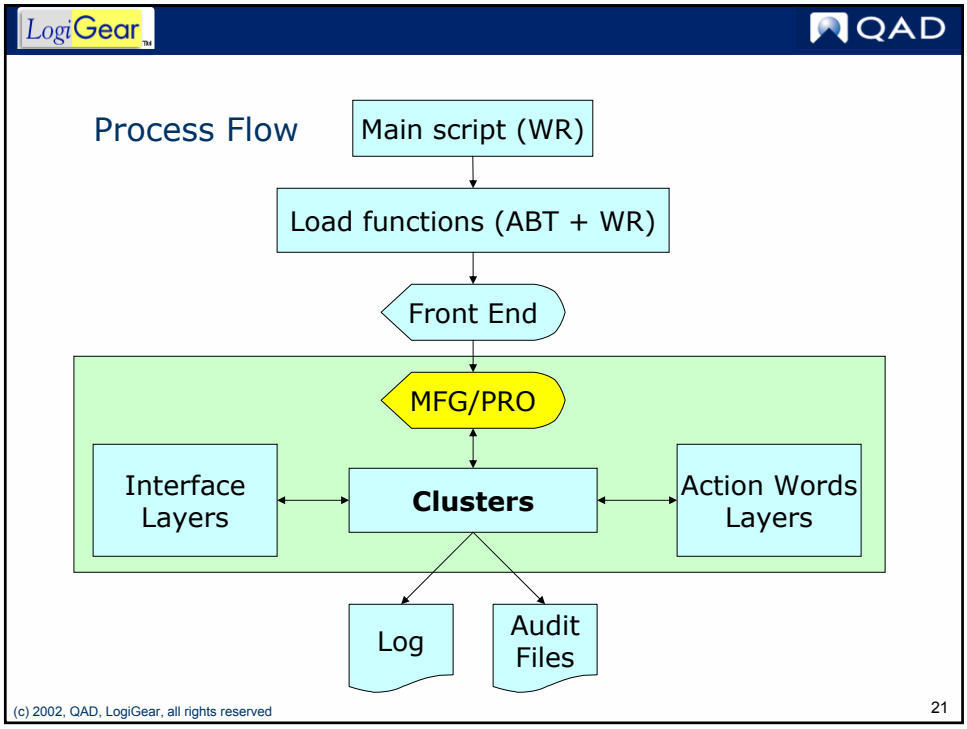
(c) 2002, QAD, LogiGear, all rights reserved

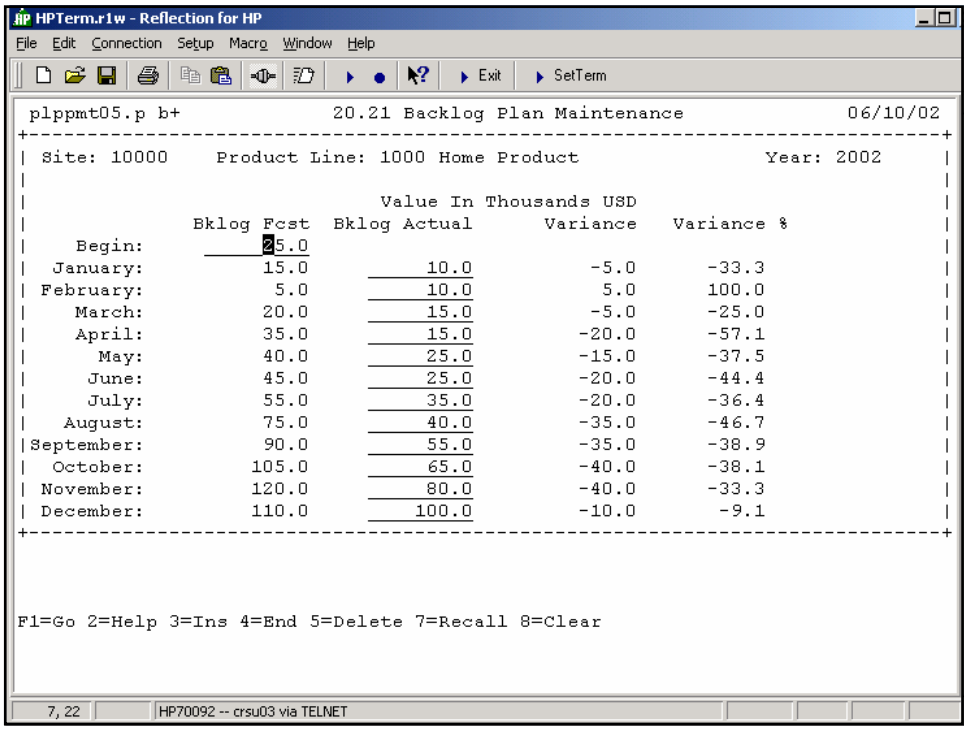

LogiGear
QAD
Process Flow
Main script (WR)
Load functions (ABT + WR)
Front End
MFG/PRO
Interface Layers
Clusters
Action Words Layers
Log
Audit Files
(c) 2002, QAD, LogiGear, all rights reserved
17

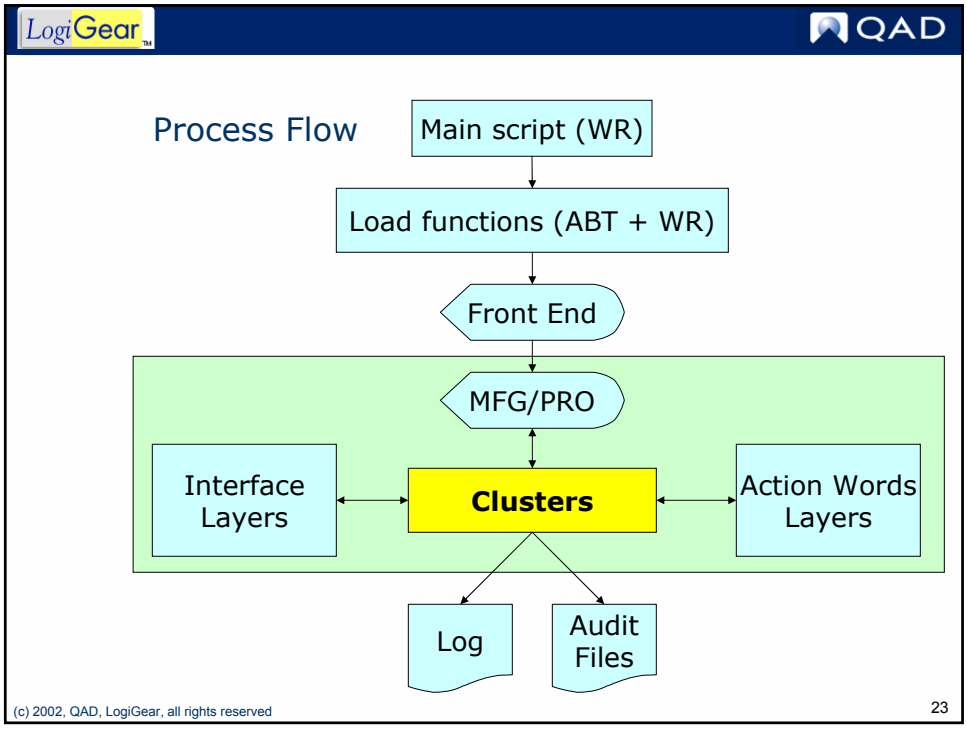

LogiGear
QAD
Process Flow
Main script (WR)
Load functions (ABT + WR)
Front End
MFG/PRO
Interface Layers
Clusters
Action Words Layers
Log
Audit Files
(c) 2002, QAD, LogiGear, all rights reserved
18


TestWrap -- Navigation Control Panel
Start run
Options
Serial
C:\ActionWords\clusters
ALL AW Info
Cluster MASTER
Error Analysis
Interface Layer - POAP 01
mfg_AWInfo
mfg_Clusters
Manufacturing - M01
Manufacturing - M02
mfg_InterfaceLayer
POAP Validate Audit Files
poap_AWInfo
poap_InterfaceLayer_1
poap_InterfaceLayer_2
poap_Master Cluster
poap_P01 - Purchasing
poap_P02 - Receipts
poap_P03 - Supplier Performance
poap_P04 - Supplier Shipping Schedules
poap_P05 - Purchase Req Creation
poap_P06 - Accounts Payable
Sample - Interface Layer
Select Other Test Directory
Test to be Run
Cluster
Browse
mfg_Clusters
Test
Manufacturing - M01
Reporting
Report File
Browse
C:\ActionWords\reports
Report All Available Details
Robot Mode
Station Identity
FYK-LT
Start Run
Run Slow
Go Robot
Cancel


LogiGear
QAD
Process Flow
Main script (WR)
Load functions (ABT + WR)
Front End
MFG/PRO
Interface Layers
Clusters
Action Words Layers
Log
Audit Files
(c) 2002, QAD, LogiGear, all rights reserved
20


LogiGear
QAD
Process Flow
Main script (WR)
Load functions (ABT + WR)
Front End
MFG/PRO
Interface Layers
Clusters
Action Words Layers
Log
Audit Files
(c) 2002, QAD, LogiGear, all rights reserved
21

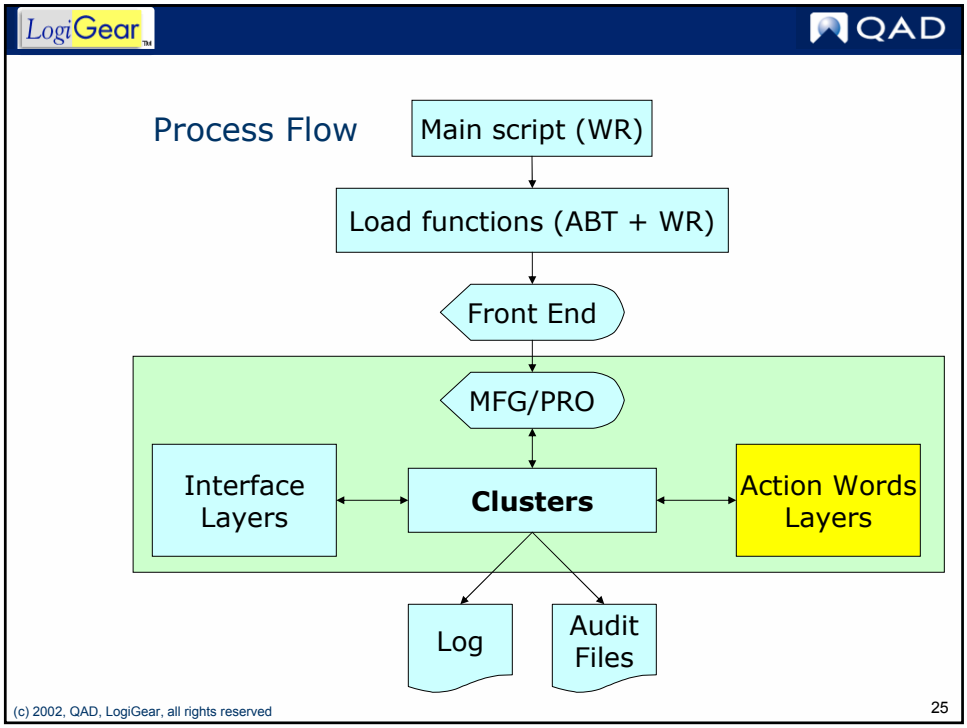

HPTerm.r1w - Reflection for HP
File Edit Connection Setup Macro Window Help
Exit
SetTerm
plppmt05.p b+ 20.21 Backlog Plan Maintenance 06/10/02
Site: 10000 Product Line: 1000 Home Product Year: 2002
Value In Thousands USD
Bklog Fcst Bklog Actual Variance Variance %
Begin: 25.0
January: 15.0 10.0 -5.0 -33.3
February: 5.0 10.0 5.0 100.0
March: 20.0 15.0 -5.0 -25.0
April: 35.0 15.0 -20.0 -57.1
May: 40.0 25.0 -15.0 -37.5
June: 45.0 25.0 -20.0 -44.4
July: 55.0 35.0 -20.0 -36.4
August: 75.0 40.0 -35.0 -46.7
September: 90.0 55.0 -35.0 -38.9
October: 105.0 65.0 -40.0 -38.1
November: 120.0 80.0 -40.0 -33.3
December: 110.0 100.0 -10.0 -9.1
F1=Go 2=Help 3=Ins 4=End 5=Delete 7=Recall 8=Clear
7, 22
HP70092 -- crsu03 via TELNET

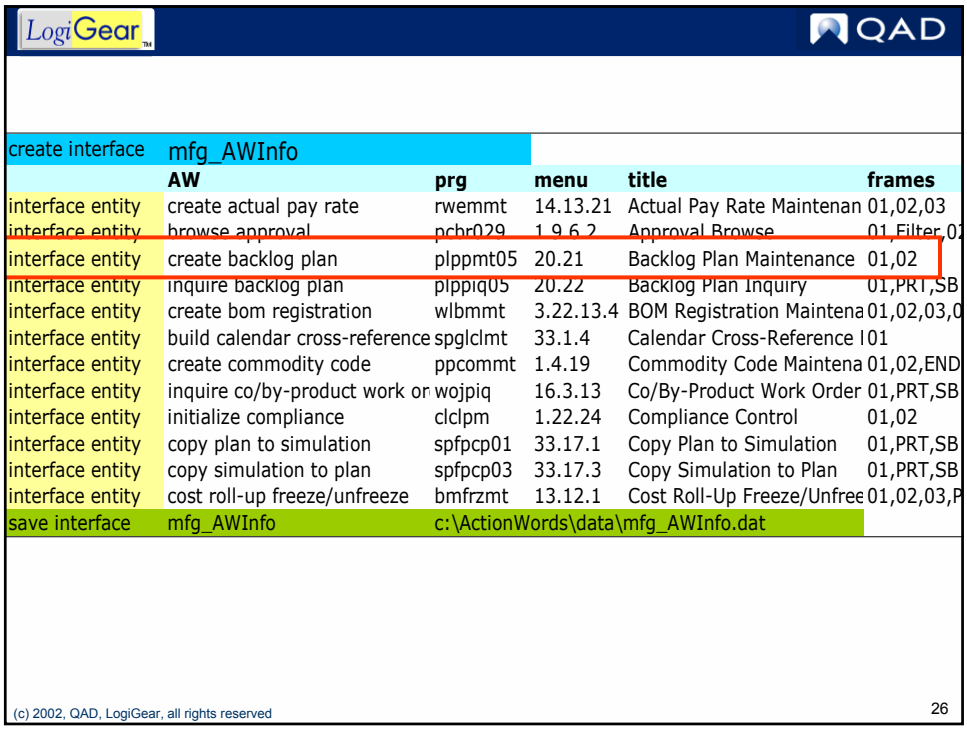


(c) 2002, QAD, LogiGear, all rights reserved

cluster	**Manufacturing**			
sheet	scenarios			
version	1.0			
author	Flavio Y. Kubagawa			
date	Jun/2002			
...				
	Enter actuals that are lower than the plan.			
section	20.21 Backlog Plan Maintenance			
	Site	**Product Line**	**Year**	
create backlog plan	10000	1000	D>Year+1	
	January	**February**	**March**	**April**
>>>	10	10	15	15
	May	**June**	**July**	**August**
>>>	25	25	35	40
	September	**October**	**November**	**December**
>>>	55	65	80	100

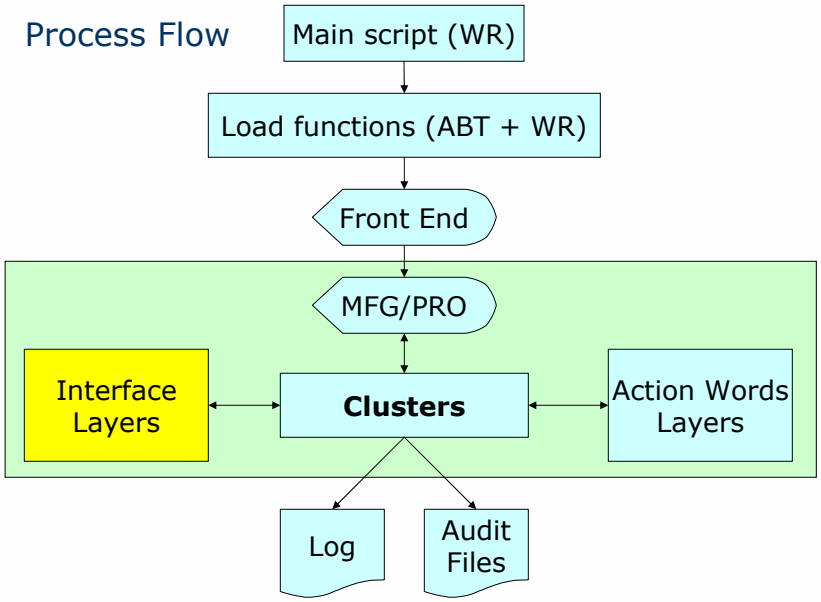
(c) 2002, QAD, LogiGear, all rights reserved


LogiGear
QAD
Process Flow
Main script (WR)
Load functions (ABT + WR)
Front End
MFG/PRO
Interface Layers
Clusters
Action Words Layers
Log
Audit Files
(c) 2002, QAD, LogiGear, all rights reserved
25


LogiGear
QAD
create interface mfg_AWInfo
AW prg menu title frames
interface entity create actual pay rate rwemmt 14.13.21 Actual Pay Rate Maintenan 01,02,03
interface entity browse approval pcbr029 1.9.6.2 Approval Browse 01,Filter,01
interface entity create backlog plan plppmt05 20.21 Backlog Plan Maintenance 01,02
interface entity inquire backlog plan plppiq05 20.22 Backlog Plan Inquiry 01,PRT,SB
interface entity create bom registration wlbmmt 3.22.13.4 BOM Registration Maintena 01,02,03,0
interface entity build calendar cross-reference spglclmt 33.1.4 Calendar Cross-Reference I 01
interface entity create commodity code ppcommt 1.4.19 Commodity Code Maintena 01,02,END
interface entity inquire co/by-product work or wojpiq 16.3.13 Co/By-Product Work Order 01,PRT,SB
interface entity initialize compliance clclpm 1.22.24 Compliance Control 01,02
interface entity copy plan to simulation spfpcp01 33.17.1 Copy Plan to Simulation 01,PRT,SB
interface entity copy simulation to plan spfpcp03 33.17.3 Copy Simulation to Plan 01,PRT,SB
interface entity cost roll-up freeze/unfreeze bmfrzmt 13.12.1 Cost Roll-Up Freeze/Unfree 01,02,03,P
save interface mfg_AWInfo c:\ActionWords\data\mfg_AWInfo.dat
(c) 2002, QAD, LogiGear, all rights reserved
26

LogiGear QAD

Process Flow

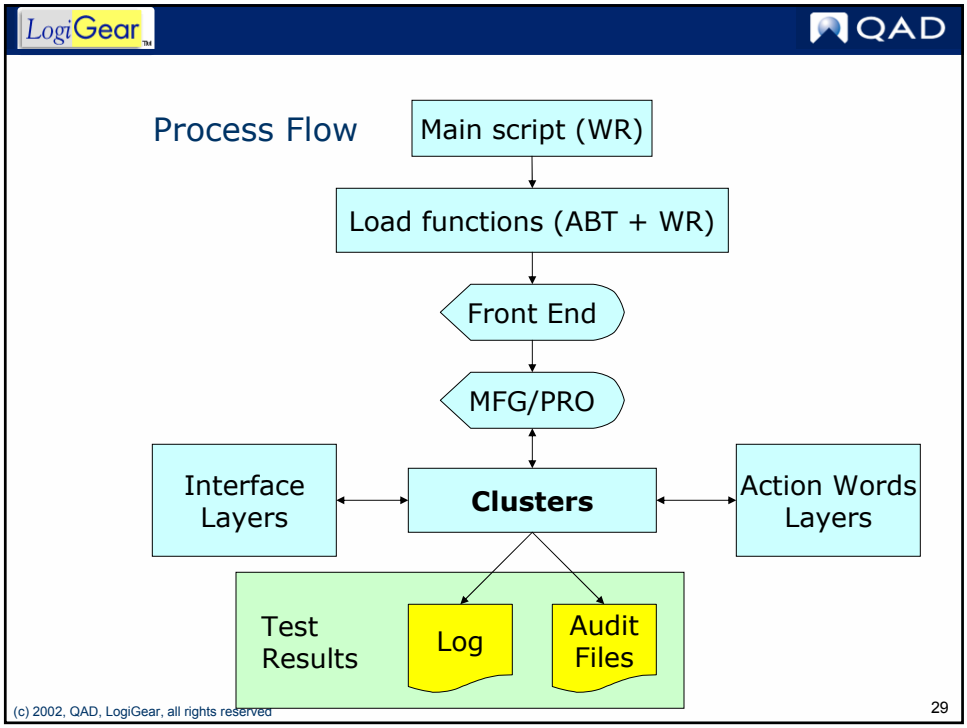


(c) 2002, QAD, LogiGear, all rights reserved

LogiGear QAD

create interface	**plppmt05**						
interface entity	fields_plppmt05						
interface entity setting	title	20.21 Backlog Plan Maintenance					
	name	class	nbr	init	endkey	condition	value
interface element	frame01	frame	01	3,9			
	name	class	tab	row	col	len	frame
interface element	Site	edit	0	3	9	8	1
interface element	Product Line	edit	1	3	32	4	1
interface element	Year	edit	2	3	71-74	4	1>
	name	class	nbr	init	endkey	condition	value
interface element	frame02	frame	02	7,17-23			
	name	class	tab	row	col	len	frame
interface element	Begin	edit	0	7	17-23	9	2
interface element	January	edit	1	8	31-37	9	2
interface element	February	edit	2	9	31-37	9	2
interface element	March	edit	3	10	31-37	9	2
interface element	April	edit	4	11	31-37	9	2
interface element	May	edit	5	12	31-37	9	2
interface element	June	edit	6	13	31-37	9	2
interface element	July	edit	7	14	31-37	9	2
interface element	August	edit	8	15	31-37	9	2
interface element	September	edit	9	16	31-37	9	2
interface element	October	edit	10	17	31-37	9	2
interface element	November	edit	11	18	31-37	9	2
interface element	December	edit	12	19	31-37	9	2>
save interface	plppmt05	C:\ActionWords\data\fields_plppmt05.dat					

(c) 2002, QAD, LogiGear, all rights reserved



(c) 2002, QAD, LogiGear, all rights reserved
29

Very Good Reusability

- Interface layer and action words: "one size fits all"
  - Once created, can be reused as often as needed
- Clusters: as reusable as the related manual test
- New clusters can be created reusing existing action words
  - Different business cycles

(c) 2002, QAD, LogiGear, all rights reserved
30

Reusability, Some Numbers

- Top 5% most used Action Words
  - Suite 1 = 27% of Total Steps: 7 Action Words = 86 Steps
  - Suite 2 = 27% of Total Steps: 10 Action Words = 113 Steps
- Top 10% most used Action Words
  - Suite 1 = 39% of Total Steps: 14 Action Words = 123 Steps
  - Suite 2 = 39% of Total Steps: 19 Action Words = 168 Steps
- 50% of the Test Suites were completed with:
  - Suite 1: 23 Action Words (17%) = 159 Steps (50%)
  - Suite 2: 30 Action Words (16%) = 214 Steps (50%)

(c) 2002, QAD, LogiGear, all rights reserved

Reusability Example

- Suite 1 – Incorporating Project Enhancements
  - added 2 new sections to the manual tests
  - 22 pages
  - 44 steps
  - 12 new Action Words
  - Reused 73% of existing Action Words
  - Quickly implemented new Action Words using the layered structure

(c) 2002, QAD, LogiGear, all rights reserved

Overall Experiences

- ABT process is a major improvement
  - maintainability
  - re-usability
  - manageability (clear tests and test results)
- LogiGear's support is very effective
  - training and coaching
  - quick and appropriate answers to all questions
- QAD expects to extend the use of ABT to other systems and other tests

(c) 2002, QAD, LogiGear, all rights reserved

Questions ?

(c) 2002, QAD, LogiGear, all rights reserved

QW2002 Paper 8A1

Dr. Fabrizio Fabbrini & Mr. Mario Fusani
(IEI-CNR)

Using Sw Process Assessment to Manage Quality of Suppliers: an Experience in Automotive

Key Points

- Goals of the project (capability determination & process improvement of suppliers)
- Project implementation
- Results and lessons learnt

Presentation Abstract

The presentation concerns an activity carried out by Fiat Auto in cooperation with the Center for Software Certification (CCS - an independent organism of the Italian National Research Council that performs evaluation and certification activity in Information Technology) to set up a methodology supporting the management of software projects and suppliers. The goals that Fiat Auto has set for this activity are:
- To improve its own process to select suppliers, by taking into account criteria to evaluate confidence in the capability of suppliers.
To improve the software development process of suppliers, helping them to detect possible weaknesses and risks in specific processes, to define improvement paths and to provide tools for verifying the results of improvement actions.
- To achieve a better control on the software development project and on the quality of the resulting product. To fulfill the above objectives it was established to adopt the ISO 15504 (SPICE) approach to evaluate a set of software suppliers, selected on the basis of possible participation to Fiat projects for new vehicles. During the presentation, the details of every phase of the activities will be given and the results of the performed assessments will be discussed and analyzed more deeply.

About the Author

Fabrizio Fabbrini obtained his degree in Computer Science from the University of Pisa, Italy, in 1974. Since 1975 he has served as a scientific researcher at the Institute for Information Processing (IEI) of the Italian National Research Council (CNR), where now he is Senior Researcher and coordinates the Software Laboratory of the Center for Software Certification. Fabrizio Fabbrini's present activity is focused on Software Quality, and more precisely on the development of methodologies and standards for the assessment and the evaluation of software products and processes, with particular attention to Software Engineering Standards and Software Certification. Software Process Assessment & Improvement, Software Verification & Validation, Computer Security & Data Privacy represent the main fields of application of such research activities.

Mario Fusani obtained his degree in Electrical engineering from the University of Pisa, Italy, in 1971. Since 1973 he has served as a scientific researcher at the Institute for Information Processing (IEI) of the National Research Council (CNR), where now he is Senior Research. His present activity is focused on Software Quality, including the development of methodologies and standards for the assessment and the evaluation of software products and processes. Since 1999 he has been the Scientific Coordinator of the Center for Software Certification of the Italian National Research Council.

Using Software Process Assessment to Manage the Quality of Suppliers: an Experience in Automotive

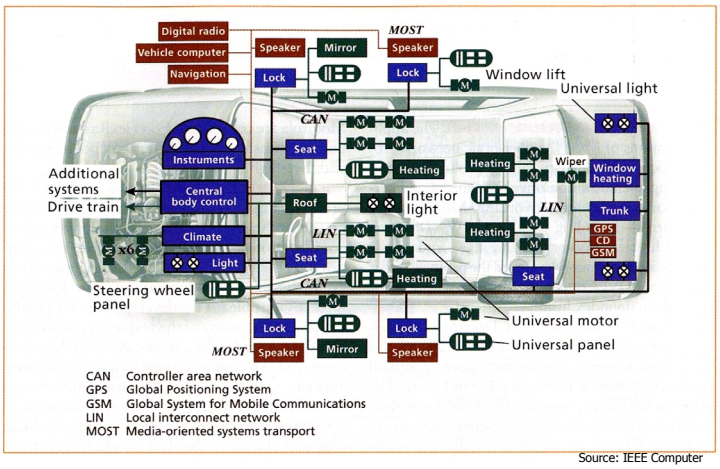
F. Fabbrini, M. Fusani*, G. Lami*, E. Sivera***

*ISTI-CNR, Pisa, Italy
**FIAT Auto, Torino, Italy

Vehicle electronic networks

- Comfort electronics
  - ✓ Seat and window movement
- Real-time critical functions
  - ✓ Cruise control
  - ✓ Antilock brakes
  - ✓ Engine management
- Multimedia applications
  - ✓ GPS, DVD
  - ✓ Internet
- Wireless applications
  - ✓ Tyre pressure control
  - ✓ Access control

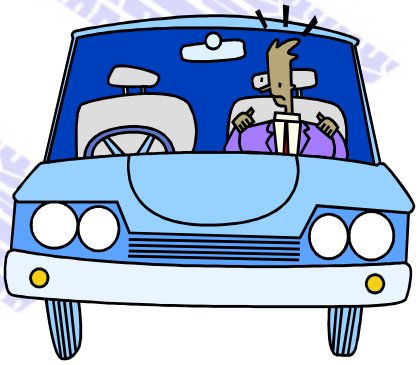
Vehicle electronic networks
(Body electronics)



Source: IEEE Computer

Vehicle electronic networks
Emerging solutions

- More specialized and reliable control networks for X-by-Wire systems
  - ✓ Brake by wire
  - ✓ Steer by wire



Project Participants

Fiat Auto

P&PE - ACEE - E&SI
Software Methodologies

Center for Software Certification
Consiglio Nazionale delle Ricerche

Center for Software Certification

- **The Center for Software Certification (CCS) is an organism of the Italian National Research Council that performs <u>independent</u> certification activity in the area of Information Technology**
- **CCS is part of ISTI, an Institute of the Italian National Research Council that performs research in Computer Science, Information Technology and related application areas, mostly within the framework of national and international research projects**

Center for Software Certification

- Certification models and methods are founded on the long acquired competencies of the ISTI and on public domain standards as well
- CCS staff members participate in national and international Working Groups for ISO standard definition in the field of Software Engineering

Center for Software Certification

The Center has been active since 1984, providing services to national and international customers, in the following fields:

- Software Product Evaluation [ISO/IEC 9126 & ISO/IEC 14598]
- Software Product Evaluation and Certification according to defined requirements and standards
- Software Process Assessment (*process improvement*, *capability determination*) [ISO/IEC 15504, SPICE]

Fiat Auto

Product & Process Engineering
Electronic & System Engineering

Tasks

- To define requirements (HW, SW, communication, reliability, etc.) for the electronic systems and components used in all Fiat, Lancia and Alfa Romeo vehicles.
- To define requirements for the vehicle "body" electronic systems
  - *Lock-unlock doors, passive entry system, anti-thief system, seat movement, windows, mirrors, etc.*

Fiat Auto

Product & Process Engineering
Electronic & System Engineering

Objectives

- To monitor the Software technologies used in the embedded electronic systems
- To use standard methodologies in order to evaluate the "process capability" of the suppliers
- To control the software life-cycle of the embedded systems
- To define and apply methodologies to validate functional requirements of the embedded systems.
- To define the SW architecture used in embedded systems

Issues in automotive

- ***The number of software-based components in automotive systems is increasing*:**
  - ➔**Need to control the development cost of software-based component.**
  - ➔**Need to manage the development of the software-based component, to increase the quality of the final products.**
  - ➔**Need to define new relationships with the suppliers, to better manage sw-based component.**

The ESCAPE Project Goals

- **To set up a methodology supporting the management of software projects and suppliers**
- **To improve FIAT process to select suppliers**
- **To improve the software development process of suppliers**
- **To provide FIAT with methods to determine the risks associated to software suppliers**
- **To give FIAT a better control on the software development project and on the quality of the resulting product**

The ESCAPE Project Action Plan

1. Selecting a methodology
2. Selecting suppliers
3. Assessing the software process
4. Defining improvement plans
5. Conducting "maintenance" assessments

Reasons for SPA

- Software Process Assessment is a way to better understand (and manage) the software process development of a supplier. The assessment is done to:
  - ✓ define a "capability" level and a "risk" level for each supplier
  - ✓ have a criterium to choose suppliers based on their "capability"
  - ✓ understand weak and strong areas of the development process
  - ✓ define better functional requirements
  - ✓ define better system verification and validation procedures

Reasons for SPICE

- Definite software-oriented approach
- Applicability over a wide range of application domains, businesses and sizes of organizations
- Output as process profiles at different levels of detail
- Comparability, reliability and consistency of results
- Independence of organizational structures, life cycle models, technologies and development models
- Adaptability of the assessment scope to cover specific processes of interest
- Re-usability of assessment results, both for process improvement and capability determination

➜ SIG on Automotive initiative

Assessment Preparation

- Planning the Assessment
  - ✓ On-site visit
  - ✓ Time/Cost constraints
  - ✓ Technical constraints
  - ✓ Assessment risk identification
- Defining the Assessment Purpose
  - ✓ Capability Determination
  - ✓ [Process Improvement]
- Defining the Assessment Scope
  - ✓ Process Selection

Assessment Preparation Assessment Scope

1. ***Requirements elicitation process*** **(CUS.3)**
2. ***System requirements analysis and design process*** **(ENG.1.1)**
3. ***Software design process*** **(ENG.1.3)**
4. ***System integration and testing process*** **(ENG.1.7)**
5. ***Project management process*** **(MAN.2)**

CUS.3 Requirements elicitation process

- ***Process purpose*****: to gather, process and track evolving customer needs and requirements throughout the life of the software product/service so as to establish a requirement baseline that serves as the basis for defining the needed software work products.**
- ***Process outcomes***:
  - continuing communication with the customer will be established
  - agreed customer requirements
  - a mechanism will be established for continuous monitoring of customer needs
  - a mechanism will be established for ensuring that customers can easily determine the status and disposition of their requests
  - enhancements arising from changing technology and customer needs will be identified and their impact managed

ENG.1.1 System requirements analysis and design process

- ***Process purpose*: to establish the system requirements (functional and non-functional) and architecture, identifying which system requirements should be allocated to which elements of the system and to which releases.**
- ***Process outcomes***:
  - requirements of the system will be developed that match the customer's stated needs
  - a solution will be proposed that identifies the main elements of the system
  - the requirements will be allocated to each element of the system
  - a release strategy will be developed that defines the priority for implementing system requirements
  - the system requirements will be approved and updated as needed
  - the requirements, proposed solution, and their relationship will be communicated to all affected parties

ENG.1.3 Software design process

- ***Process purpose*: to define a design for the software that implements the requirements and can be tested against them.**
- ***Process outcomes***:
  - an architectural design will be developed that describes the major software components that will implement the software requirements
  - internal and external interfaces of each software component will be defined
  - a detailed design will be developed that describes software units that can be built and tested
  - consistency will be established between software requirements and software design

ENG.1.7 System integration and testing process

- ***Process purpose*: to integrate the software component with other components, producing a complete system that will satisfy the customer's expectations expressed in the system requirements.**
- ***Process outcomes***:
  - an integration strategy will be developed to build system unit aggregates according to the release strategy
  - acceptance criteria for each aggregate will be developed to verify compliance with the system requirements allocated to the units
  - system aggregates will be verified using the defined acceptance criteria
  - an integrated system demonstrating compliance with the system requirements will be constructed
  - test results will be recorded
  - a regression strategy will be developed for retesting aggregates or the integrated system, should a change be made
  - regression testing will be carried out as necessary

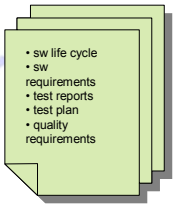
MAN.2 Project management process

- ***Process purpose*: to identify, establish, coordinate and monitor activities, tasks and resources necessary for a project to produce a product and/or service meeting the requirements.**
- ***Process outcomes***:
  - Definition of the scope of the work for the project
  - Feasibility evaluation of achieving the goals of the project with available resources and constraints
  - Estimation of the tasks and resources necessary to complete the work
  - Identification of the interfaces between elements in the project, and with other projects and organizational units
  - Development and implementation of the plans for the project execution
  - Actions to correct deviations from the plan and to prevent recurrence of problems identified in the project will be taken when project targets are not achieved.

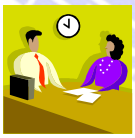
Project implementation
pre-assessment activities

- **Introductory meeting**
  - ✓ To introduce the SPICE (ISO15504) approach
  - ✓ To review the assessment purpose, scope and constraints
  - ✓ To introduce the assessment activities and the provisional assessment plan

- **Pre-assessment questionnaire**
  - ✓ To gather preliminary information on the projects to be used as process instances



Project implementation
on-site activities



- **Briefing**
  - ✓ *Assessment purpose, scope, constraints and model*
  - ✓ *Confidentiality policy*
  - ✓ *Assessment schedule*
- **Data Acquisition & Validation**
  - ✓ *Presentations*
  - ✓ *Document analysis*
  - ✓ *Interviews*

} Checklist-based

- **Process rating (provisional)**
- **Debriefing**

The Rating Dilemma

- Different *rating methods* can be applied
- ranging from the mere processing of measured indicators up to the unaided assessor's judgement
- Need to establish the *requirements* to be satisfied for a rating method to be valid
- Trade-off: assessor's judgement driven by checklists

Confidentiality Policy

- Care has been taken on convincing the supplier that process assessment does not disclose sensitive information about particular techniques used in software development nor details on proprietary software or algorithms
  - process assessment methods do not need investigation on technical aspects: they only investigate on knowledge, experience, skill, confidence, benefits, resource allocation and management of such aspects.
  - any unwanted leakage of information is covered by the security policy and security process of the assessing team.

Project implementation
post-assessment activities

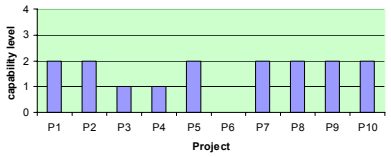
- **Process rating (final)**
  - ✓ ***For each process assessed, assign a rating to each process attribute***
  - ✓ ***Record the set of process attribute ratings as the process profile and calculate the capability level rating***
- **Reporting the results**
  - ✓ ***Prepare the assessment report***
  - ✓ ***Present the assessment results***
  - ✓ ***Finalize and distribute the assessment report***

Project status

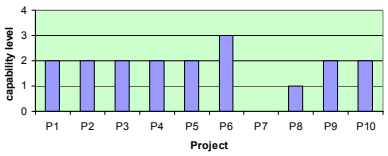
- **Seven assessments performed (on 10 projects) so far**
- **Four more assessments scheduled this year**
- **Further assessments planned (next year), including re-assessments for improvement verification**

Project results

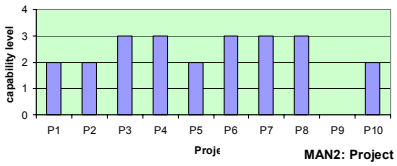
CUS3: Requirements Elicitation Process



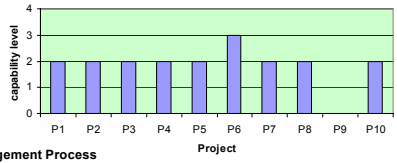
ENG1.1: System Requirement Analysis and Design Process



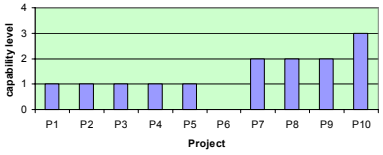
ENG1.3: Software Design Process



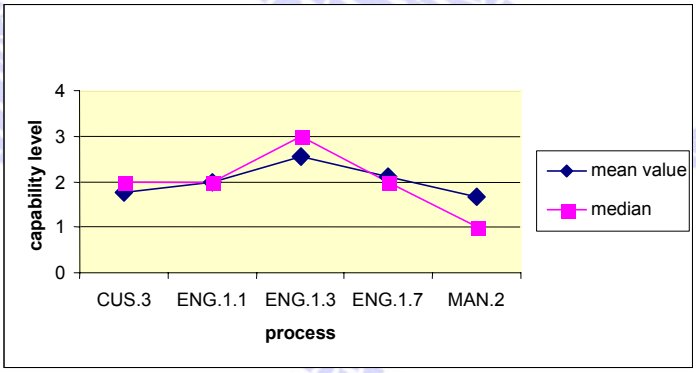
ENG1.7: System Integration and Testing Process



MAN2: Project Management Process



Project results
(Synthesis)



Resulting considerations

- The demand for the use of electronics and software solutions has been dramatically growing in the last years and will be even more in the next future.
- The answer is not completely adequate, as it comes from an environment (both customers and suppliers) that is not enough prepared to the transition - for historical, cultural and technical reasons: innovation is often apparent only, actually based on giving old concepts new names.
- From the case study, some trade-offs seem to be crucial to the automotive community and their investigation can provide research topics to help automotive organizations respond to the challenges presented by today's global competitive environment.

Concluding issues

- Trends
  - Requirement analysis as a key issue
  - Awareness of the customer role in the acquisition process
  - Need for new SW development models
- Trade-offs
  - Platform-oriented vs customer-oriented
  - Resource (memory size, processor performance, design complexity) saving vs maintainability and reliability
- Open issues
  - Interoperability at subsystem level (ECU)
  - Safety and security implications

QW2002 Paper 8A2

Mr. Robert Sabourin
(AmiBug.Com, Inc.)

Training Testing Professionals: Making the transition to Web Based Application Testing

Key Points

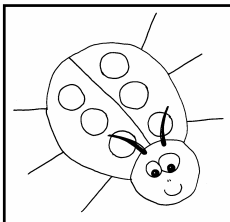
- Training can be established to transition testing team from mainframe to web applications
- Many practical lessons learned and problems to avoid are presented
- Testing professionals should be taught how their current knowledge can be applied

Presentation Abstract

The Public Works department of the Government of Canada is currently participating in several major infrastructure software development projects which will enable existing applications and facilities to be accessible as multi-tiered web based applications. (some internal and some publicly available over the Internet). This paper describes a simple and effective training program developed to enable the existing, established, team of software testing professionals, to effectively test the new Web based and Government On- Line versions of these applications.

About the Author

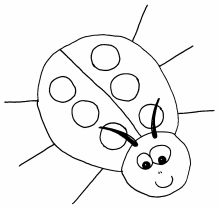
Robert Sabourin, P. Eng. has 20+ years management experience leading teams of software development professionals to consistently deliver projects on-time, on-quality and on-budget. He is a well-respected member of the software engineering community who has managed, trained, mentored and coached hundreds of top professionals in the field. He frequently writes and speaks to conferences around the world on software engineering, SQA, testing, management and internationalization.



Training Testing Professionals: Making the transition to Web Based Application Testing

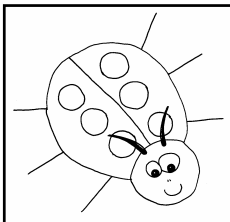
Robert Sabourin
President
AmiBug.Com, Inc.
Montreal, Canada
rsabourin@amibug.com
www.amibug.com

 © Robert Sabourin, 2002



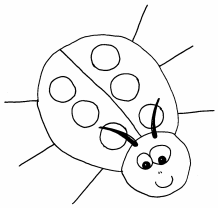
Training Testing Professionals: Making the transition to Web Based Application Testing

 © Robert Sabourin, 2002



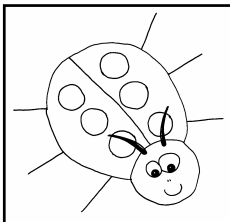
Training Testing Professionals: Making the transition to Web Based Application Testing

- Robert Sabourin , *Software Evangelist*
- President
- AmiBug.Com Inc.
- Montreal, Quebec, Canada
- rsabourin@amibug.com



AmiBug.Com, Inc.

- Software Development & SQA Consulting
- Services
  - Training, Coaching and Professional Development
  - Light Effective Process
  - Team Building and Organization
  - We help people to get things done!



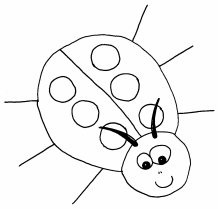
I am a Bug

Robert & Catherine Sabourin

ISBN: 0-9685774-0-7

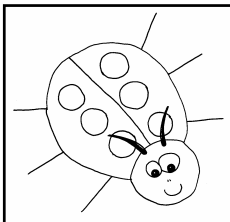
www.amazon.com

In the style of a children's book. Explains elements of software development process in a fun easy to read format.



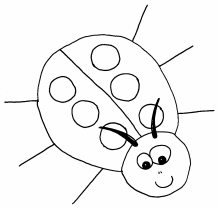
Public Works and Government Services Canada

- Government of Canada
- Infrastructure, procurements
- Centralized services



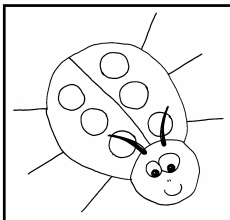
Public Works and Government Services Canada

- Original mainframe based client server solution uses combination of intelligent PC terminals and business logic implemented on mainframe systems with large DB2 backend database



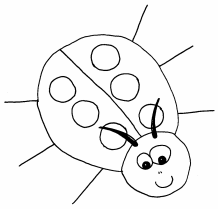
Public Works Canada

- Current Major Development Projects
  - Common Department Financial System (CDFS)
    - System Management
    - Access and Reporting
    - Expenditure Management
    - Revenue Management
    - General Accounting
    - Budget and Forecasting



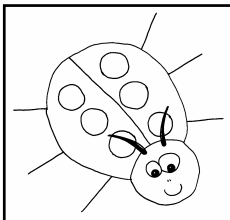
Public Works Canada

- Current Major Development Projects
  - Common Department Financial System (CDFS)
    - Develop web technologies based from end which can be accessed from desktops of internal users at government offices via Internet browsers
    - Use middleware developed using an IBM Websphere architecture
    - Share DB2 backend with co-existing “traditional” mainframe solution



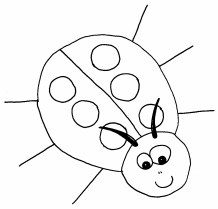
Public Works Canada

- Current Major Development Projects
  - Common Department Financial System (CDFS)
    - Current system has a defect tracking system with over 1400 users across Canada
    - Defect tracking tied in with internal customer support and maintenance services



Public Works Canada

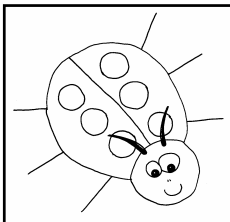
- Government On-Line
  - All Government services to be available on-line on the Internet
    - Complete new web based front end
    - Public information about services
    - Procurement support including supplier registration and tracking
    - Point of contact
    - Point of service



Government On-Line

Canada

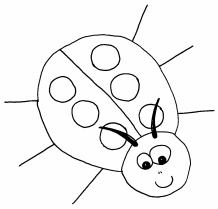
- *"By 2004, Canadians will be able to request and receive all key federal services through secure, interactive and timely on-line transactions."*
- `www.gol-ged.gc.cap`



Government On-Line

Training

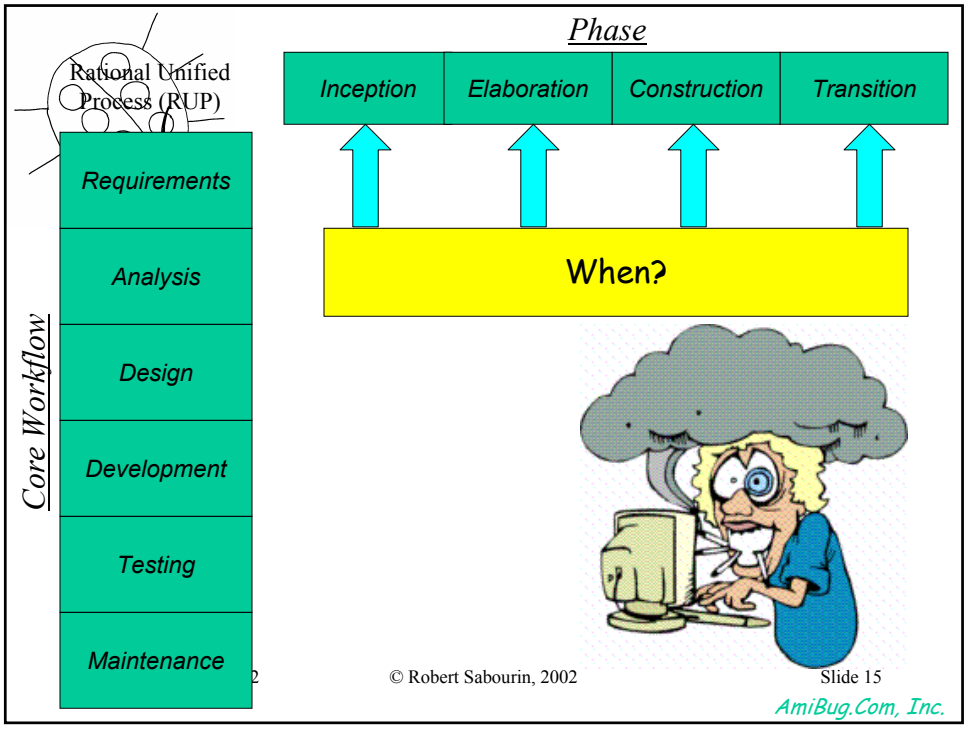
- Observed Q3+Q4-2001 and Q1-2002
  - High percentage of students are experienced testers moving to GOL applications from Client Server or Mainframe applications
  - High percentage have never used basic testing techniques such as:
    - Equivalence Partitioning
    - Boundary Analysis
    - State Modeling
    - Analytic Methods
    - Any Code Coverage Techniques
  - High percentage have never read more than one book about testing
  - High percentage have never attended a course or conference about SQA or Testing or Software Engineering
  - *… but they all are enthusiastic and want to learn!*

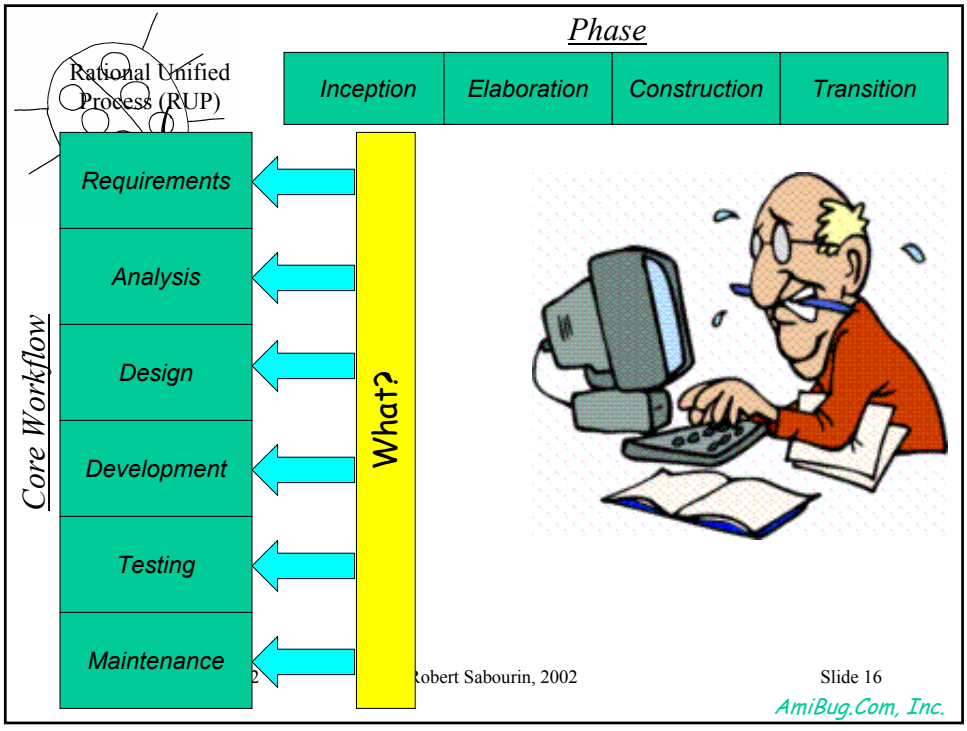


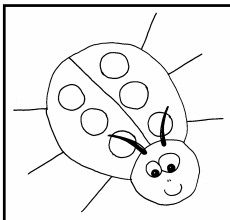
What is needed!

Training

- Get back to basics!
  - Testing 101 courses and books all over the place
  - Practical books
  - Awareness of the many wonderful methods and techniques available in an objective and open minded approach
  - Encourage a foundation in fundamentals before overly skill based approaches

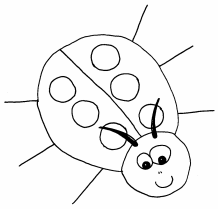

Rational Unified Process (RUP)
Phase
Inception
Elaboration
Construction
Transition
Requirements
Analysis
Design
Development
Testing
Maintenance
Core Workflow
When?
© Robert Sabourin, 2002
Slide 15
AmiBug.Com, Inc.


Rational Unified Process (RUP)
Phase
Inception
Elaboration
Construction
Transition
Requirements
Analysis
Design
Development
Testing
Maintenance
Core Workflow
What?
Robert Sabourin, 2002
Slide 16
AmiBug.Com, Inc.



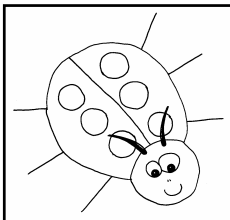
Starting Point

- Testing teams
  - Large team
    - Over 100 testing professionals
    - Senior in testing experience
    - Mainframe focused
    - Very limited experience testing Web systems
    - Good understand of how to find problems with financial systems
    - Good understand of bug reporting and triage



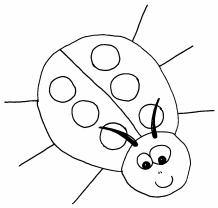
Starting Point

- Established Testing Methods
  - Testing with detailed, complex, procedures which includes a mix of instructions, outlining step by step exactly what the tester should do and what the expected results should be
  - Black box oriented functional testing
  - Some concepts of scenarios
  - Testing designed based on solid analysis of requirements and systems expected usage



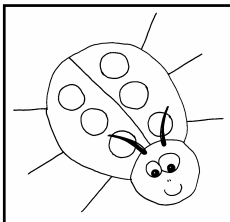
Starting Point

- Basic Web Terminology
  - All testers offered hands-on training in the basics of HTML and all relevant Web Technologies used in the CDFS and GOL projects
  - Three tiered web development architecture being used
    - Which function is implemented on which tier
    - Associated failure modes!



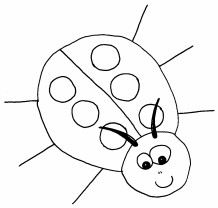
Training Approach

- Back to basics
  - Start with existing testing process in place
  - Teach how test cases are derived from basic concepts such as
    - State modeling
    - Equivalence partitioning
    - Boundary analysis
    - Positive and negative test design
  - Use relevant familiar examples



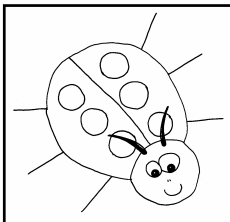
Training Approach

- Decouple test procedure from test data
  - Data independent
- Decouple user input from test instructions
  - GUI independent
- Use familiar examples, show how the procedures can be written in a technology independent manner
- Benefit of reuse and shared testing analysis between two systems (old and new)



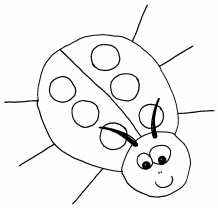
Training Approach

- Testing Lab
  - Teach by example how platform testing can be achieved using a testing lab with Operating System and Browser swapping
  - *A lab was set up and used as an example in training courses*



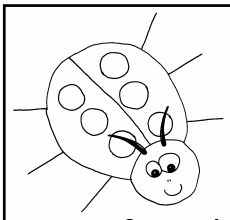
Training Approach

- Course structure
  - Three of ½ day workshops
    - All terminology carefully reviewed with team to ensure consistent
    - Dry run and complete content review
    - ½ day design can be done on site with minimal disruption of projects.



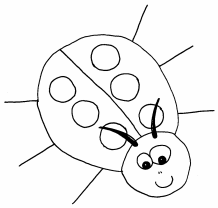
Course Outline

- Philosophy, terminology (purpose, risk)
- Types of testing and the development process (workflow, prioritization)
- Quality Factors and what can break (failure modes)
- Testing Web/User Interface Tier
  - Lab Set Up / Platforms
  - Browser compatibility testing
  - Check list approaches
  - Tools and techniques
- Testing Application Tier
  - Analytic techniques
  - Exploratory techniques
  - Scenario Based Data Driven approaches
- Testing Data Tier
  - Create, Update, Delete
  - Search Sort
  - Shared Database issues
  - SQL – static



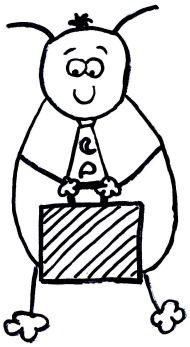
Results

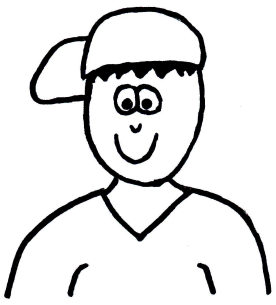
- 6 sessions to date (June 2002), 90 students
- Group has set up completely functional test lab
- Set up new bug prioritization workflow
- Finding – and getting fixed - large number of important bugs well in advance of deployment
- 2 Follow up sessions planned for September
- Frequent follow-up meetings, and correspondence, to help resolve conflicts and confusion as problems are encountered
- New examples and updated content as feedback is collected
- New tests being developed will be easier to adapt to future changes to GUI and any technology shifts



Thank You

- Questions?





QW2002 Paper 9A1

Ed Lycklama
(Sitraka)

Performance and Scalability Bottlenecks in J2EE Applications

Key Points

- Common causes of performance problems in J2EE applications
- The differences between performance and scalability and how they can be detected
- Real world scenarios

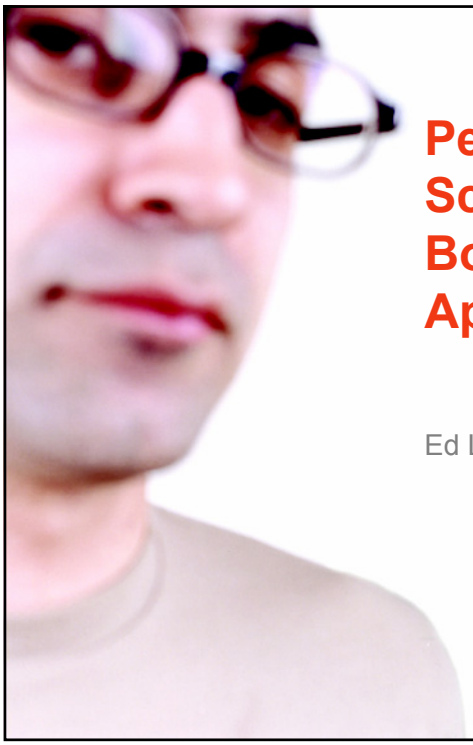
Presentation Abstract

The J2EE platform is quickly emerging as a standard for building enterprise applications. Although the platform has been designed to support high-performance applications, many organizations are discovering that their applications are failing to meet agreed upon performance criteria.

This talk addresses the common causes of performance problems in J2EE applications. Beginning with a discussion of the differences between performance and scalability, this presentation separately discusses both types of problems in J2EE applications, illustrated with some common real-world scenarios. We will also show how these types of problems can be detected and diagnosed in both a load-testing and production environment.

About the Author

Ed Lycklama M. Sc., CTO & Co-Founder As chief technology officer and co-founder of Sitraka, Mr. Lycklama holds primary responsibility for directing the company's technology strategy and managing Sitraka's intellectual capital. He has also led the development of a number of Sitraka's core products.
Mr. Lycklama served as chief architect in the design of Sitraka's award-winning JProbe suite of Java performance and analysis products and XRT suite of Motif widgets. He also serves on Sitraka's board of directors and has been a driving force behind the company's exceptional growth. As an expert in Java application performance issues, Mr Lycklama is a wellrespected and frequent speaker at major industry events.
Mr. Lycklama completed a Master's degree in Computer Science, with a focus on algorithms for distributed systems, at the University of Toronto (1987). He also holds a B.Math from the University of Waterloo (1984).



Performance and Scalability Bottlenecks in J2EE Applications

Ed Lycklama - CTO, co-founder

sitraka

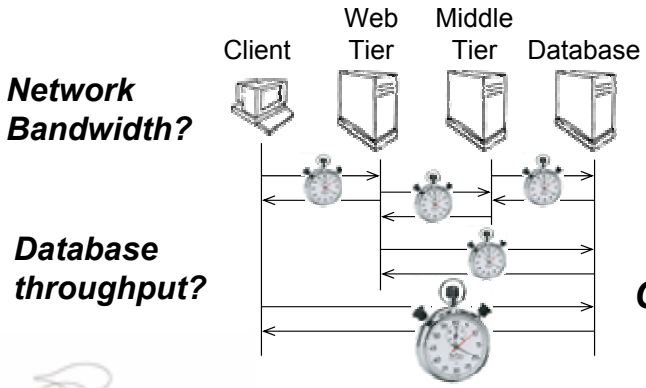
J2EE Reality

- J2EE application performance a serious problem despite:
  - Adequate hardware capability
  - Adequate network capability
  - Modern JVMs
- Many problems not discovered until:
  - Components are assembled
  - Application is distributed across machines
  - Realistic data and load simulations are used

sitraka
the Java™ advantage

The Technical Challenge

Complex N-tier Clustered Architecture



Network Bandwidth?

Database throughput?

Memory usage?

CPU usage?

Component efficiency?

sitraka
the Java™ advantage

Characterizing Performance Bottlenecks

- By functional tier (servlets, EJB, DB, etc)
  - reality: lots of do's and don'ts
- By J2EE role (developer, deployer, administrator)
  - roles aren't always clear-cut
  - too much interdependency
- By underlying principle
  - best way to learn "performance patterns"
  - categorize new problems

sitraka
the Java™ advantage

Understanding J2EE Performance Obstacles

1. Failure to understand system performance
2. Ignoring the benefits of caching
3. Inter-component (and inter-system) interactions
4. Single-threaded thinking
5. Sharing resources

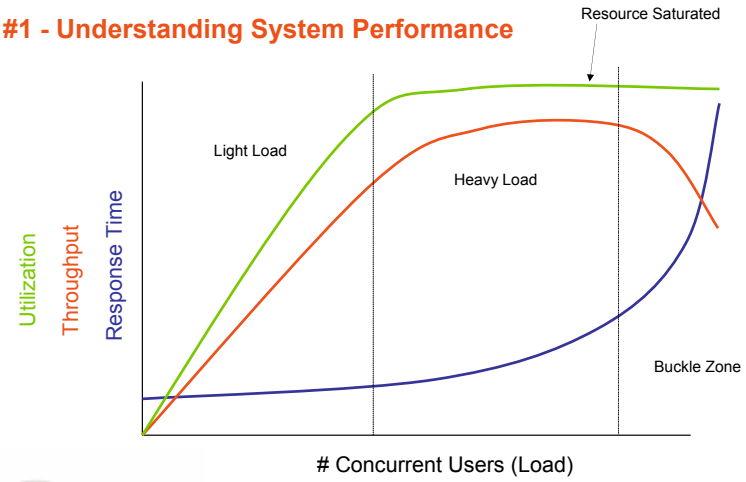
sitraka
the Java™ advantage

#1 - Understanding System Performance

- Performance vs. Scalability
- Important Concepts:
  - response time
  - throughput
  - # concurrent users

sitraka
the Java™ advantage

#1 - Understanding System Performance



sitraka
the Java™ advantage

#1 - Understanding Performance

Little's Law

N = #users in the system

X = average throughput

R = average response time

Then N=RX when observed over a "long enough" period

Little's Law applies to any device or system, regardless of service methodology

sitraka
the Java™ advantage

#1 - Understanding Performance

Little's Law Applied

Useful in back-of-the-envelope calculations, or simple capacity planning

Example: Porting a legacy system to J2EE

- services 100 request/sec at avg. response of 2 sec
- then N = RX = 100*2 = 200 concurrent users
- each concurrent user occupies an execute thread
- 20 threads per server ==> min. 10 app servers

sitraka
the Java™ advantage

#2 - Ignoring Caching

- Retrieving data repeatedly that never (or even rarely) changes is expensive
- Each tier has opportunities for caching results
- The further upstream you cache, the better your performance results will be

sitraka
the Java™ advantage

#2 - Ignoring Caching

Presentation Tier

- Browser can cache pages based on information in the HTML headers (age, expires, etc.)
  - implement getLastModified() from HTTPServlet
- cache static data in servlet's init() method
  - only called once in servlet's lifetime
- JSP caching
  - vendor-specific way of caching output
  - refer to vendor's documentation

sitraka
the Java™ advantage

#2 - Ignoring Caching

Application Tier

- JNDI lookup is very expensive
  - cache EJBHome, JMSConnectionFactory, etc.
  - Use ServiceLocator pattern to cache these expensive lookups
  - result: only perform one expensive JNDI lookup per service, instead of once per service use

sitraka
the Java™ advantage

#2 - Ignoring Caching

Data Access Layer

- Identify read-only data from DB and cache
  - avoid redundant trips to database
- EJB's call ejbLoad/ejbStore around transactional boundaries unless given hints
  - read-only EJB's - never call ejbStore()
  - BMP: keep track of whether its been modified
  - single app-server (no clusters): set db_is_shared to false, then app-server will cache results

sitraka
the Java™ advantage

#3 - Inter-Component Communication

- J2EE is a distributed, multi-tier architecture with many interconnecting components
  - presentation tier -> biz logic -> persistent store
  - Java code running on app-server, which runs on JVM, which runs on OS -- each has its own performance and tuning quirks
- complex set of interactions, must think about performance holistically

sitraka
the Java™ advantage

#3 - Inter-Component Communication

Remote Method Invocation (RMI)

- RMI is the default way in which all remote J2EE communication is done
- when truly remote, network communications overhead is significant
- serialization of objects is expensive
  - entire object graph is persisted and sent to receiving object
- needs to be considered in any system

sitraka
the Java™ advantage

#3 - Inter-Component Communication

Remote Method Invocation (RMI)

- Pass-by-value vs. pass-by-reference
  - local optimization when EJB's are co-located
  - must explicitly turn on (app-server specific)
- Local EJB's in newer app-servers
  - when you know EJB will be in same JVM
  - use EJBLocalObject interface in EJB 2.0-based application servers

sitraka
the Java™ advantage

#3 - Inter-Component Communication

SessionFacade

- All EJB interactions should go through SessionBean
  - never let client objects (e.g. servlets) invoke entity beans directly
  - preserve transactional and security behavior
  - avoid excessive fine-grained network traffic

sitraka
the Java™ advantage

#3 - Inter-Component Communication

Communication with the Database

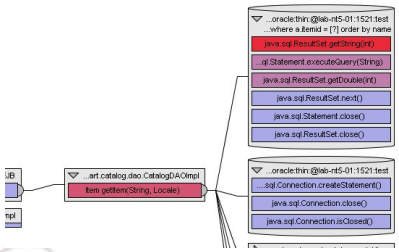
- Common performance bottleneck
- Typical problems:
  - Inefficient queries - sending SQL data that asks the database to do more work than necessary.
  - Excessive querying - efficient queries called too frequently.
  - Large Data Sets - processing large sets of data in ResultSets

sitraka
the Java™ advantage

#3 - Inter-Component Communication

Diagnosing Database problems

- Standard DB tools can examine performance of individual queries on the DB itself
- Need to determine performance impact on individual requests



sitraka
the Java™ advantage

#4 - Single-threaded Thinking

- J2EE application servers run multiple requests concurrently on separate threads
- EJB's are guaranteed to only run on a single thread at a time
- Tempting to ignore concurrency issues
- But, ...

sitraka
the Java™ advantage

#4 - Single-threaded Thinking

- Any shared state or data must be protected
- servlet's can be used in multiple threads
- database use requires concurrency protection to provide data consistency and isolation

- Result: you can't ignore it

sitraka
the Java™ advantage

#4 - Single-threaded Thinking
Servlet SingleThreadModel

- Any servlet that implements SingleThreadModel will only use one-thread per-instance concurrently
  - generally app-servers create a small pool
- Problem: severe throughput limitation
- Problem: any shared data must still be protected

- Solution: protect any common data with `synchronized` keyword

sitraka
the Java™ advantage

#4 - Single-threaded Thinking

Synchronized Methods

- Java: synchronized implements a mutex, only one entrant allowed
- If the size of this “critical section” is large, can be severe throughput limiter
- Strive to keep synchronized sections as short as possible

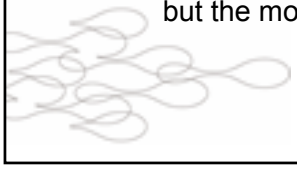
sitraka

the Java™ advantage

#4 - Single-threaded Thinking

Transaction Isolation Levels

- There are five levels, in increasing strictness:
  - _NONE
  - _READ_UNCOMMITTED
  - _READ_COMMITTED
  - _REPEATABLE_READ
  - _SERIALIZABLE
- NOTE: not all DB’s support all five
- The stricter the isolation level, the less concurrency, but the more “correct”



sitraka

the Java™ advantage

#4 - Single-threaded Thinking

Transaction Isolation Levels

- Although SERIALIZABLE is the most “correct”, its also the most throughput restrictive
- consider READ_COMMITTED with optimistic locking if there is a low likelihood of two transactions updating the same database rows
    - updates only fail if there has been a collision
- WebLogic 7 has built-in support
    - you can do-it-yourself in other versions

sitraka
the Java™ advantage

#5 - Resource Sharing

Background

- Standard application resources:
    - CPU
    - Memory
    - I/O
    - Network
    - Database
- In J2EE, there are additional resources

sitraka
the Java™ advantage

#5 - Resource Sharing

J2EE Application Servers

- Application Servers control resources:
  - Execute threads that request run on
  - JDBC connection pools
  - stateful session bean cache
  - stateless session and entity-bean pools
  - JDBC prepared statement cache size
- Correct settings depend on application
- Complex to correlate direct effect of multiple configuration changes

sitraka
the Java™ advantage

#5 - Sharing Resources

Diagnosis

- Application knowledge
- Test applications under load
- System-wide performance assurance tools

sitraka
the Java™ advantage

#5 - Sharing Resources

Memory

- The garbage collector is not a panacea!
- Loitering objects can remain with the JVM heap beyond successive garbage collections.
- Developers must manage object references to remove loitering objects.

sitraka
the Java™ advantage

#5 - Sharing Resources

Memory

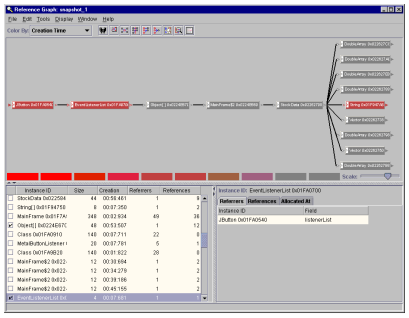
- Erosion of performance over time and an increasing process size
- OutOfMemoryError thrown by the JVM as it abnormally terminates
- Use a J2EE performance assurance tool to watch heap size.

sitraka
the Java™ advantage

#5 - Sharing Resources

Memory

- Undertake a heap analysis using memory debugger to identify the unnecessary references that cause loitering objects.



sitraka
the Java™ advantage

#5 - Sharing Resources

CPU

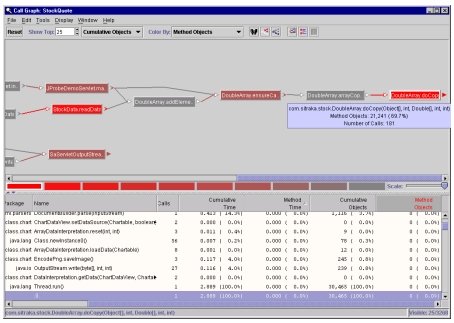
- J2EE application responsiveness often depends upon the efficiency of the software you write.
- Expensive sorting algorithms, slow XML parsing, etc. all have their toll

sitraka
the Java™ advantage

#5 - Sharing Resources

CPU

- Use a profiler to identify where the problem lies.



sitraka
the Java™ advantage

Understanding J2EE Performance Obstacles

1. Failure to understand system performance
2. Ignoring the benefits of caching
3. Inter-component (and inter-system) interactions
4. Single-threaded thinking
5. Sharing resources

sitraka
the Java™ advantage

Performance and Scalability Bottlenecks in J2EE Applications
Ed Lycklama - CTO, co-founder
sitraka

QW2002 Paper 9A2

Joshua Kitchen
(IBeta)

Legacy Data Conversion: Making Coffee

Key Points

- Likely Difficulties/Risks:
  a. Managerial
    i. Lack of visibility at the project management/project sponsor level
    ii. Scarcity of testing resources
    iii. Lack of environments/Poor environmental performance
    iv. Lack of business analyst/subject matter expert involvement
    v. Turf issues
  b. Technical
    i. High number of defects
- Do's and Don'ts of Data Conversion Testing (rules of thumb)
- Key Indicators (How do I know I'm doing it right/wrong)
  a. Application schedule
  b. Data model slip
  c. Performance tuning schedule
  d. Resource rhythm
  e. Defect counts, including sub-counts of defect classification

Presentation Abstract

Converting data from a legacy application is a lot like making coffee.

The existing data (grounds/beans) work well enough in their current state (if you like pulling beans out of the bag or eating spoonfuls of grounds, but then we already know we want a new app, don't we?). To work properly, however, in our new application the data must undergo change. So first the grinding, either from beans or from a courser grade of grounds (cleansing garbage data from the system prior to trying to converting it), then the pouring into the filter and running hot water through it (conversion to new schema, applying different integrity rules, interpolating new fields from existing ones, etc.) until the finished coffee finally resides in the pot (the new application database). In theory, if all went well, what's in the pot is what was desired; the user can drink it and be well satisfied with the temperature, the flavor, the aroma, the viscosity…but theory and fact often part company. So who do we want to drink the first cup, our user group/customer or our testing group? That was my answer, too.

About the Author

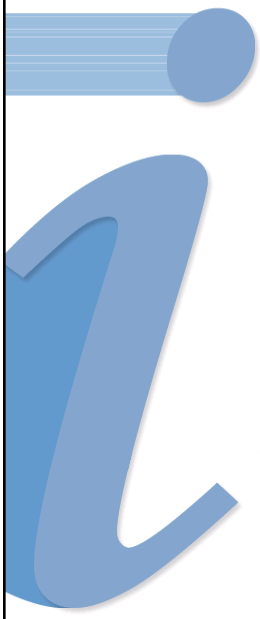
Joshua Kitchen hails from 'lots of different places'; he was commissioned from Cincinnati, Ohio, but after graduating the Naval Academy in 1990, he spent the next decade making his home in a variety of different locales, including Florida, Idaho, Hawaii, and California. A mathematics and system background was augmented by exhaustive training in engineering courtesy of the Navy's nuclear power program, and proven at sea aboard the USS Indianapolis, on which he received his final commission of lieutenant and his qualification as ship's engineer. Post-Navy, he moved on to work with expert systems for Pilkington in their quality assurance department, and in the process earned his Black Belt in Six-Sigma techniques. From Pilkington, he moved on to iBeta, a software quality assurance firm in Denver, Colorado, where he finished his ASQ certification in software quality assurance engineering and proceeded to implement his training and experience from a variety of sources as one of the companies directors.

iBeta
Software Quality Assurance

Legacy Data Conversion: Making Coffee

Converting data from a legacy application is a lot like making coffee.

The **existing data** (grounds/beans) work well enough in their current state (if you like pulling beans out of the bag or eating spoonfuls of grounds, but then we already know we want a new app, don't we?). To work properly, however, in our new application the **data must undergo change yet remain essentially the same**. So first the grinding, either from beans or from a courser grade of grounds (cleansing garbage data from the system prior to trying to converting it), then the pouring into the filter and running hot water through it (conversion to new schema, applying different integrity rules, interpolating new fields from existing ones, etc.) until the finished coffee finally resides in the pot (the new application database). In theory, if all went well, what's in the pot is what was desired; the user can drink it and be well satisfied with the temperature, the flavor, the aroma, the viscosity…but theory and fact often part company. So who do we want to drink the first cup, our user group/customer or our testing group? That was my answer, too.



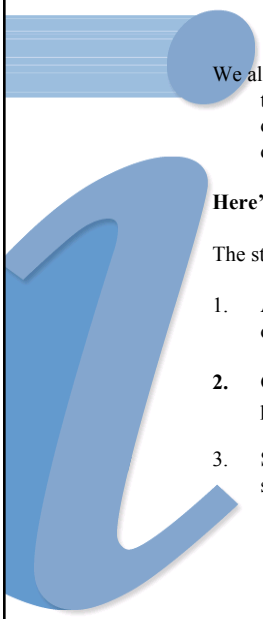
Tackling the testing problem can follow the coffee analogy also; look at the different test techniques that are typically applied during data conversion testing:

Requirements testing (the beans were supposed to be ground to some fineness; did the coffee turn out that fine, or is it overly course?)

Functional testing (no matter what happens, someone's still got to taste the coffee)

Domain testing (pick out beans that best represent the bag)

Performance testing (don't just drop a tea spoon in and point at colored water; if conversion has to occur over a two-day weekend, ***that's*** the product we most want to test on, not something that's been simmering on the pot for a couple of weeks, converted one slow piece at a time until it's well blended)



We all have our own ideas on making coffee, our own preferences; likely nothing in this white paper will be new. As in making coffee, however, rarely do we sit down and go over the risks and rules of thumb to be used in effectively testing data conversion.

Here's a recipe I used.

The structure of this presentation is simple:

1. A look at the **difficulties/risks** encountered, with some perspective on overcoming or mitigating them.

2. **Collected of rules of thumb** that worked well (or not so well) during test planning and execution.

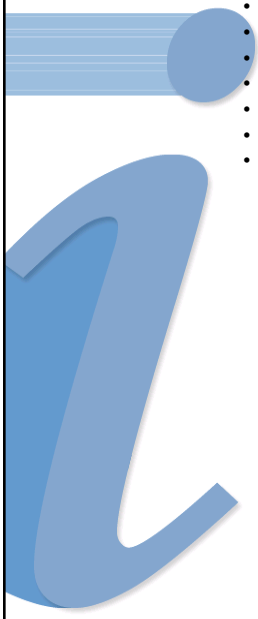
3. Summary of **items to watch** for before, during, and after. Indicators help only so far as we pay attention to them.

Likely Difficulties and Risks

- Lowest priority, at least at the onset
- Technical subject matter
- Low to no budget
- Business involvement
- Development involvement
- High, hard to fix and isolate, defect rates
- Whose bug is it, anyway?

Important Indicators

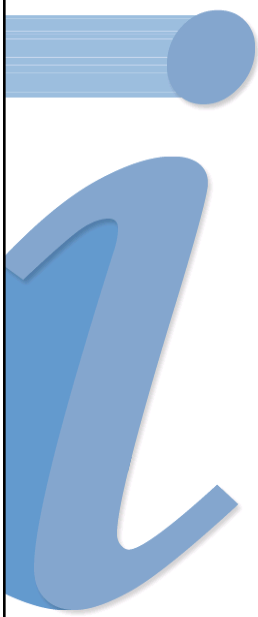
- Overall project schedule slip
- Data model slip
- Performance tuning schedule
- Defect counts by testing category



Do's and Don'ts

- Understand the development process in use
- Have a defined test approach
- Establish close relationships with key people
- Train your team
- Have a communications plan
- Revisit requirements often
- Understand the other group's plans
- Negotiate your testing scope throughout the process
- Always be prepared for major environmental issues
- Use your tools
- Avoid becoming the sponge for everyone else's project
- Avoid distractions
- Never, ever negotiate away your time
- Never become a one-person show

Likely Difficulties and Risks



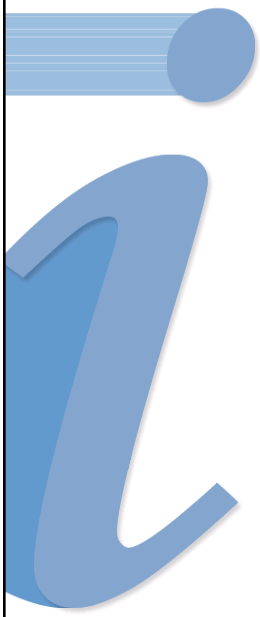
Data conversion tends to be **something of an afterthought** in most projects (now, where did I put the coffee? Must be up here somewhere…), except those that launch specifically with this goal in mind.

So many other things **seem to be of higher priority**: requirements for the upcoming application, initial coding and testing of that application, pushing the new application to the required levels of performance, the various 'must have' change control requests, high priority defects requiring major re-engineering of the application, usability issues…the list goes on and on.

And, too, **data conversion is a more technical**, under the covers event; most of the details are extremely esoteric, the results are far from exciting, and reporting on progress tends to be rather dry. 'I can't send email from the app!' gets far more attention than 'Integrity enforcement at the app level is complicating conversion of people records'.

Thus, data **conversion tends to be thought of last**, usually about the same time that planning begins for product rollout, an unfortunate timing as it usually means this effort is last in line for resources, on the critical path, and with little time to plan let alone code and test.

Likely Difficulties and Risks

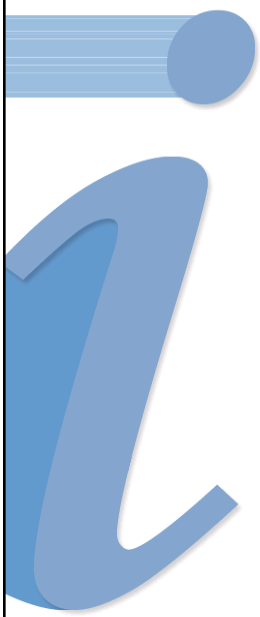


With data conversion behind and ill understood from the onset, testing of data conversion is likewise behind the curve. Gradually, however, the ramifications of insufficiently testing the conversion process and final converted data sink in at various levels and the testing effort receives more and more (often becoming unwelcome; everyone wants to either help make a pot or get a taste) attention.

A key factor in the success of the data conversion testing effort becomes getting **enough visibility early on**; this visibility is critical to getting the resources and time assigned before they become critical, enabling the data conversion testing effort and, by proxy, the data conversion effort, to succeed.

Waiting too long makes data conversion testing a bitter drink no one wants: under resourced, out of time, and with huge expectations on the part of every stakeholder.

Likely Difficulties and Risks

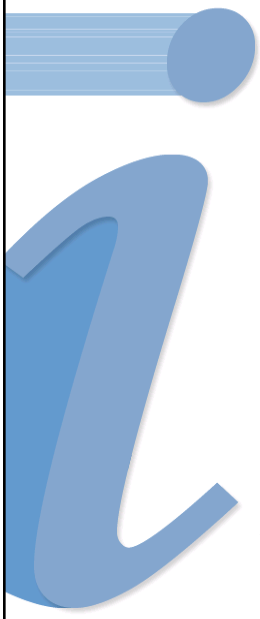


The key steps to getting the right amount of (early) attention to data conversion testing are familiar to any software project manager: take charge, draw up a rough plan or approach, present to stakeholders (**and present to stakeholders, and present to stakeholders, and present…).**

Taking charge of a sub-project that needs doing, yet isn't funded or recognized, is a great deal like petting a shark: an interesting experience, something to tell one's grand-children about, but one earning the nickname 'Lefty' if one is not careful. However, there's really no other way to go about it; if it needs to be done, and wasn't in the project plan from inception, someone's going to have to be a hero and stand up for it, or it will likely fail.

This paper focuses on the testing piece of data conversion, so if you're deciding on standing up for testing, likely you've a counterpart who's decided to stand up for the development. **Make friends; you'll be in this together, and for the duration, else you'll be failing separately.**

Likely Difficulties and Risks

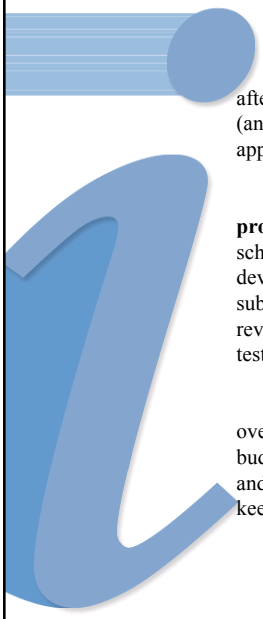


Once you've made friends with your development counterpart, drawing up a rough plan should be fairly easy; we'll cover additional planning considerations elsewhere in this white paper. It will take you a bit of time to draw up, as most good plans that account for specific resources needed do, but the majority of the issues you fight your way through will be organizational rather than technical, so there should be few surprises (knock on wood).

The presentation bit will be harder, as **you'll have to impress a non-technical group with the technical ramifications of failure**. It may seem, at this point, that you are antagonistic to your partner on the development side; after all, you're the one telling stakeholders that your buddy's project may fail. But by making clear the potential issues associated with data conversion (i.e. That the entire project could fail if this piece isn't taken care of), you're trying to not only get resources of your own, but also to increase the resources development has available.

Make no mistake, data conversion testing is quality control, finding errors generated because of short timelines, inadequate or missing requirements, etc; anything you can do to make development's life easier, any resource you can free up to assist their end of the project, anything you can do to prevent them from having to rush (without impacting the timeline you're setting up for yourself), or take shortcuts, will make the project's life (and yours) easier.

Likely Difficulties and Risks

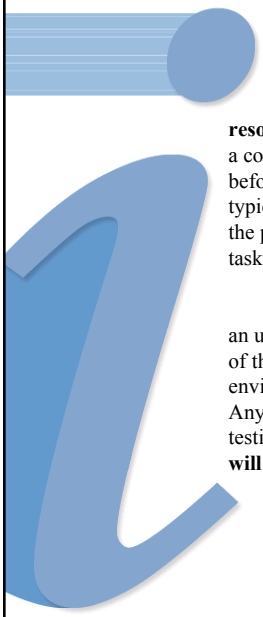


As mentioned earlier, data conversion testing tends to be an afterthought; by the time you come onto the scene, budgets have been made (and hopefully approved) for development, rollout, and testing of the application.

The **polite battle for resources will continue throughout the process**; accept this. The applications development group will be behind schedule, and wish to appropriate your testing environment for additional development; the roll-out group will wish to appropriate your analysts and subject matter experts to assist in planning, training, or additional requirements review; the other testing groups likewise want your environment, equipment, testing staff, or to pass some of their testing scope off to your group.

To say not to give in to these demands/requests for the sake of the overall project would be naïve; the end goal is still to come out on time, on budget, with the feature set and reliability targeted. Caution is useful, however, and a well-managed scope and risk management document is a good tool to keep your team out trouble.

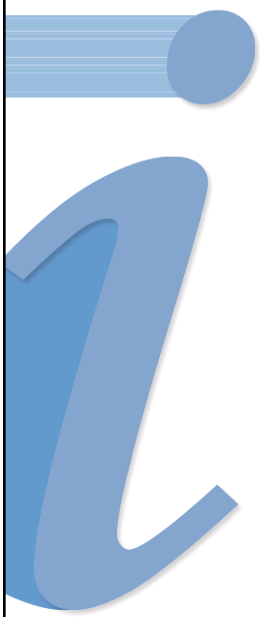
Likely Difficulties and Risks



Your testing environment becomes your most critical resource, because you need a minimum of two (both a controlled legacy and a controlled developed application for comparing data and functionality before and after data transformation), and because your environment is typically larger than most others, at least until final integration and rollout of the product: you need legacy data, and lots of it, if you're to complete your task.

As an additional hardware/software cost to the project, you'll have an uphill battle to get set up; likely you will have to work for at least a portion of the time in and around development's environments, the other testing environments, or perhaps on older equipment handed down from production. Any of these events will slow your testing when compared to the rest of the testing group…and **any time a milestone in another group is missed, you will likely lose your testing environment in favor of that group**.

Likely Difficulties and Risks

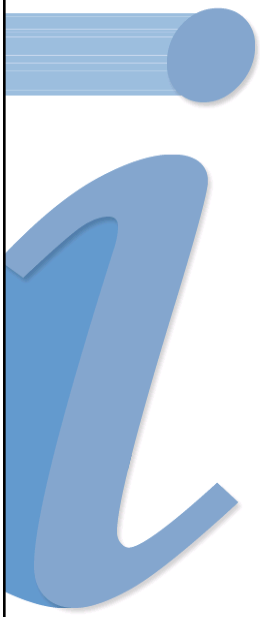


Good business analysts (BA's/SME's) are worth their weight in gold; fortunately or unfortunately, the rest of the development and test groups are aware of this as well. Data conversion is particularly vulnerable to poor requirements gathering issues, and the analyst is the person most likely to minimize this threat. The primary pull on your BA's time will be development; after all, if development doesn't know what to build, they can't build it, and if they can't build it, you've nothing to test.

Participating in development's requirements gathering process (if you can) will bring big dividends to your testing group down the road: you establish good relationships with BA's, get a jump on test planning, and grow an understanding of the application all in one step. It's a very worthwhile step; take it.

A **negative**, at least from your perspective, of BA involvement may occur when they are **over involved**; they may push test priorities (Test this first! And this! And this!), over-prioritizing defects (No, the system MUST not do this! What? The system doesn't do THAT? Unacceptable.), and generally attempting to do your job (part-time; most BA's, by definition, could contribute significantly if managing a testing effort. However, they have other jobs and can hurt more than help if their role in your project is not managed.)

Likely Difficulties and Risks



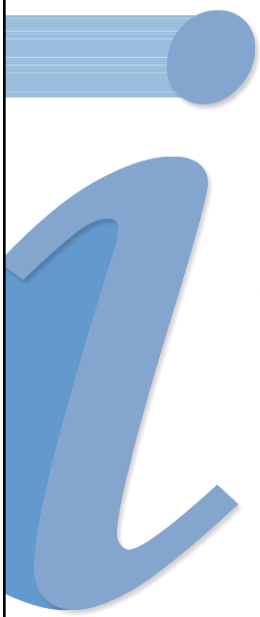
BA's tend to have the ears of the primary stakeholders; if their concerns, both voiced and unvoiced, are not answered they often go to these stakeholders with their concerns. This leads to a variety of unfortunate issues, as you're well aware/have experienced. Listen to the BA's.

Don't necessarily treat them with kid gloves when they offer a strong opinion, but don't ignore them either. Regular communication via email, or better, regular meetings in which they can make their concerns known and have them answered go a long way to making them happy with the testing group and productive members of same.

Everyone may want to drive. Everyone has priorities; these priorities won't match each other's or yours in many respects. Senior people (Dev, BA's, VP's) are necessary to projects to succeed and, by definition, these are people used to driving.

If they are not hearing what they need to hear, or seeing what they need to see (i.e. progress, milestones being met, etc.) they have a tendency to insist on more and more reports, and on direct involvement in your testing process. This distracts them from their tasks, which are essential for your success and are definitely necessary for the success of the project as a whole.

Likely Difficulties and Risks

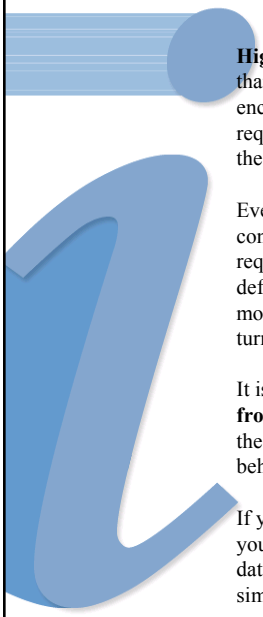


Give them what they need: **regular, organized reporting**. Let them smell the coffee brewing. If they understand what you're are doing, and where you are in your project path, they can rest (perhaps not easily) assured that the right things are being done at the right time, and they will stay out of your way.

Development is a slightly different animal. When development wants to drive the testing process, generally two things can happen: the testing staff is pulled off testing to assist in development (either in reality, in which case they are punching code for a development manager, or figuratively, in which case they are diving into testing detail more along the lines of debugging rather than testing, an inefficient use of their time) or testing is ineffective (defects are downgraded in priority, repeatedly deferred to other code iterations, or simply dismissed as unimportant).

Development must not drive your testing. There's no easy way to say this, but they must not, or your position will swiftly become irrelevant, redundant, and a waste of project resources.

Likely Difficulties and Risks



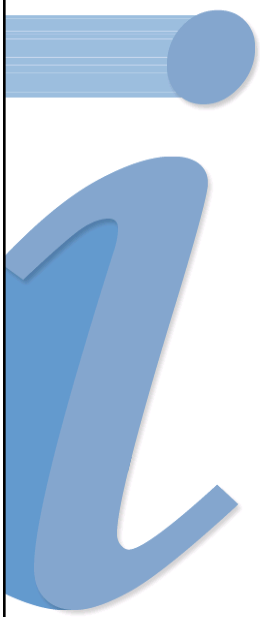
High defect rates. Data conversion may encounter a higher number of defects than other groups; while not inevitable, it is a distinct possibility you will encounter more defects due to three data conversion specific issues: accelerated requirements creep, coordination between different groups of developers, and the impact of database changes.

Every project has requirement creep; its a fact of software development. Data conversion in particular will suffer from this creep, not only from its own requirements, but from the requirements of the application. A defect/requirement added to the application team will push a change to the data model which will in turn push a change to data conversion code which will in turn push a change in the testing…this is inevitable.

It is absolutely essential to **understand when the data model is due to be frozen**; without a grasp on this date, and the understanding of its importance in the other development/testing groups, your team will churn farther and farther behind the rest of the project.

If your development group is different from the application development group, you will have to deal with another layer or two of communication between the database team and the application team. Poor communication often leads to simple, but telling errors in converting the data.

Likely Difficulties and Risks

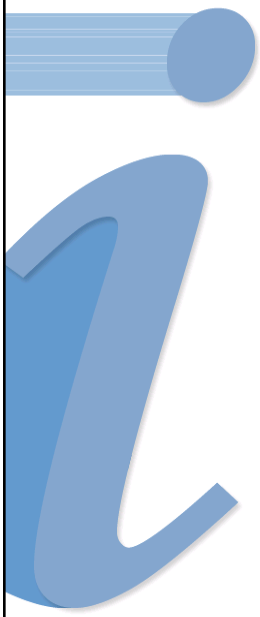


Due to the typically late start and technical issues surrounding requirements gathering for data conversion, **defects often arise from misunderstood or missed requirements**. Data conversion is typically very technical, more concerned with field lengths and data types than with application level processing or interface display; the requirements often differ enormously from the requirements of the application team, but require the input from people (the BAs) who may not have a firm grasp of the technical ramifications involved. This combined with the late start often results in good code that nevertheless results in an unsuitable conversion of legacy data (i.e. defects).

Whose bug is it, anyway? There will be recurring debate between the application and data conversion development groups as to the cause of a given defect, with either the application's processes or the data conversion process being the proximate cause.

Essentially, there are two applications in use: the primary application under development and the secondary application needed to transform the data; in most cases, a defect found by your testing group could be found in either one. Heading off impasses in this area is critical to successfully resolving defects and speeding the testing process to completion. If you can't sort which caused the silt in the bottom of the pot, the grounds or the filter, you'll never make a good cup except by accident.

Do's and Don'ts

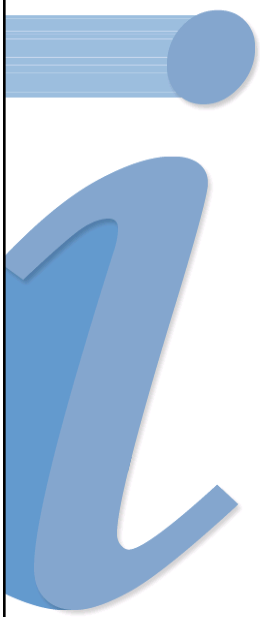


Understand the development process in use. Understanding the process development is using will help the data conversion testing process immensely (especially if you grow to understand they don't have a process!).

This applies not just to the data conversion group; to get defects fixed in a timely fashion, knowing who does what (Mr. Smith the defect coordinator assigns defects to the engineers on staff), and how they do it (a fix has to have a code sample to go with it, so no defect can be marked fixed in the database without that code sample) will significantly smooth the process.

Where possible, **borrow from their processes**; things move faster when both testing and development are speaking the same language.

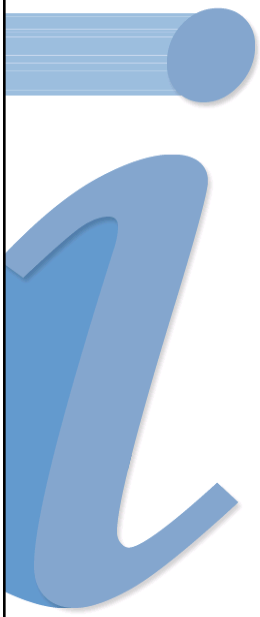
Do's and Don'ts



Have a defined test approach. Conversion testing will be the last car in a very long train of development and testing; as we've already discussed, this testing is likely an afterthought. Having a defined, robust, and well-understood test approach is useful both in communicating the scope and intent of testing and speeding that testing along successfully.

For data conversion testing, I recommend a **four-part approach** as a beginning followed by focus and refinement as testing continues. In order of priority and general utility, a good organization scheme is what we referred to as screen to screen (S2S), functionality, database partitioning, and defect regression.

Do's and Don'ts



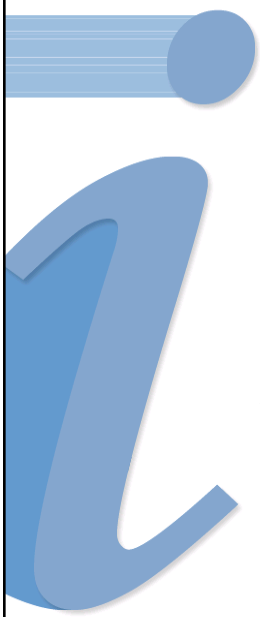
The **biggest bang for the buck** is in the screen to screen comparisons: an instance of the old system is placed online and a snapshot of the same database is used on an instance of the new application.

Testing then becomes a matter of **'camera one, camera two'** comparison. Organization further helps this effort; a simple Excel spreadsheet, for example, with a list of the fields and their locations in the new app, along with the equivalent fields and locations from the old app, is extremely handy. Addition of the transformation logic to the document further enhances its utility.

After the S2S mapping, **running a subset of the functionality testing** on the converted data is essential to bug hunting. Despite all the requirements gathering, development, and testing to date, until the system is testing on the live data no real 'feel' for product progress exists.

Particularly when different teams are doing the conversion and application development, **errors creep into the boundaries between the software and its data**; test data, no matter how well crafted, never truly compares to the real thing.

Do's and Don'ts



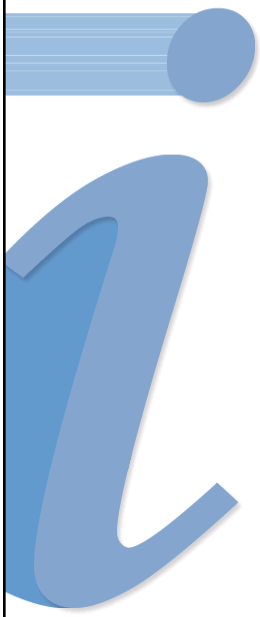
Database partitioning is a form of equivalence class partitioning in which the database is sub-divided into categories of data, then again, and again, until the types of records noted are a representative sample of the legacy system. This testing gives you a large handful of records that can then be used to validate the entire converted system.

Be prepared to run a defect regression set, at least initially, to validate the defect repair failure rate. If that rate is high, resign yourself to re-testing a significant portion of your previously reported defects with each major change of code.

As you test, **review** which of your approaches net **the most relevant defects**, and which aren't netting enough to be worth your time; prioritize the former and reduce the time spent on the later.

Different development staff and strategies lend themselves to different types of errors; each of the four approaches above lends themselves to finding particular types of errors.

Do's and Don'ts

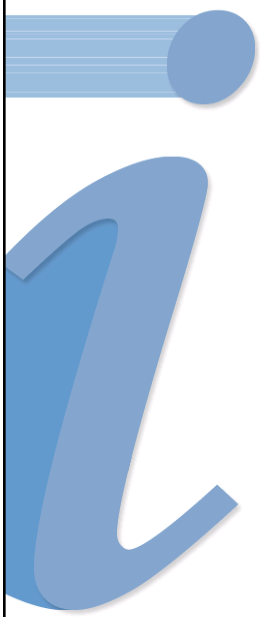


Establish close relationships with key people. As alluded to earlier, the people you will need to make your testing successful are already very tasked; at best, you are likely to have only partial resources for some key activities. BAs provide you with information you need; project managers can keep you aware of key timelines or political issues; friendly developers can help you understand the underlying mechanics of the new application, or data conversion code, and help your defect resolution along; other testing groups on the same application can give you a heads up on new functionality or recurring issues with the application; Systems folks can get your environment up in record time; DbAs can get that database refresh done for you quicker and cleaner than you expected.

Or not.

Disregarding the political aspects of having/making friends in key places, knowing these people and how they work (what information and time they need to perform for you, what other priorities they have on their plate, etc.) will make your testing cycle (and preparations for it) **much, much easier**.

Do's and Don'ts

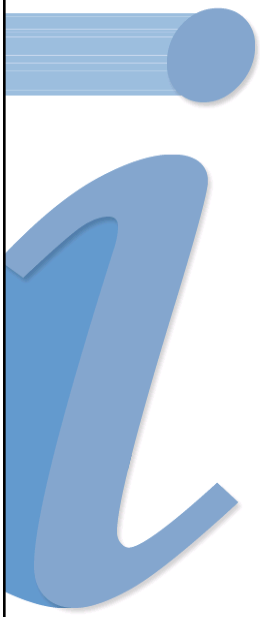


Train your team. Largely due to the 'loaner' nature of your resources, it is strongly recommended that you train on your testing process again and again, especially if there are gaps between iterations. The people assigned to you will be working their other tasks in, around, and between your testing cycles and will need refreshing on your process and instruction on any changes you may (you will) make to said process.

Without a good (not perfect, but good) understanding of the new application's functionality, testing how that functionality performs (or rather, fails to perform) when subjected to real data is difficult. A good approach is to either mix in some testers from the application group (if they can be spared), or put some of your test group into the system test group for a space. If the application group is writing detailed testing scripts, borrowing these scripts will significantly aid in data conversion testing.

The databases must have been reviewed for both the conversion team (old database, plus transformation mapping to the new database) and the application team (new database); understanding the database schema is essential to timely completion of testing, as queries for test data must be run against the old database to find the records of interest in the new database (reviewed through the application).

Do's and Don'ts

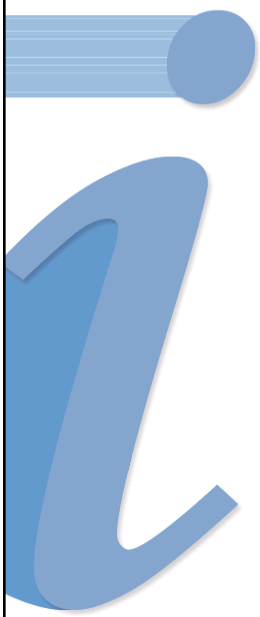


Have a communications plan. Testers need to report to you: defects, testing completed, significant issues in the testing environments, etc. You will need to report up as well: project gantts, test status and completion, opinions of the overall quality of the converted data. You will also have to track defects to completion, answer development's questions on those defects, and digest their answers.

Make a list (check it twice, three times, four…. Know what the deliverable is, who is suppose to produce it, when it is to be produced, and, last but not least, make sure you know what its for! Anything used for communication but without a defined purpose is likely a waste of your time, and likely your testers'/associated developers'/project manager's time as well.

A good laundry list for a communications plan runs along the following lines: weekly tester meeting, weekly test lead (cross-group) meeting, monthly project meeting, daily defect review meeting, pre-cycle meeting (with development, testing, and support present), project gantt, testing assignments, testing results, defect reports, pre-testing check off, development's project gantt (both development groups), etc.

Do's and Don'ts

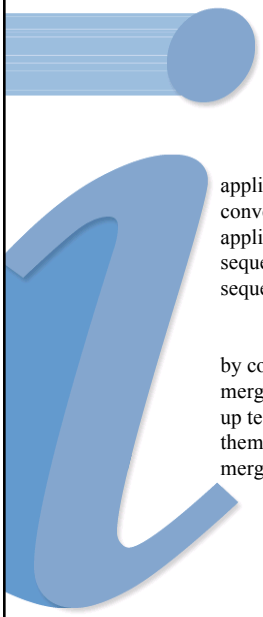


Revisit requirements often. Requirements will often change dramatically on you in data conversion; what was first a hard and fast will-not-budge business requirement may turn a hundred and eighty degrees around once that requirement is put into practice and business can see the results. By revisiting them with business and development, you ensure the three groups are still on the same page, and save your testing staff a tremendous amount of time in wasted testing against out of date requirements.

This 'revisiting' can occur a variety of formats; it is useful to set aside a portion of the defect review meetings to discuss these kinds of issues (as they will happen anyway during testing, setting the time aside and designating it as such will both speed the defect review and accomplish the task: development, testing, and business in the same room discussing the same requirements and how best to proceed.

Just make sure decisions get documented.

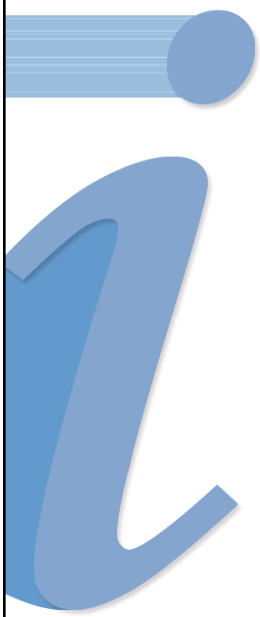
Do's and Don'ts



Understand the other group's plans, in particular the rollout and application testing plans. Both of these plans may (should) require your converted data as a key milestone; they may have particular needs. In a legacy application with multiple databases, these groups may already have a required sequence in which those databases will be integrated into the final product. This sequence will drive your decision(s) of what to test and when.

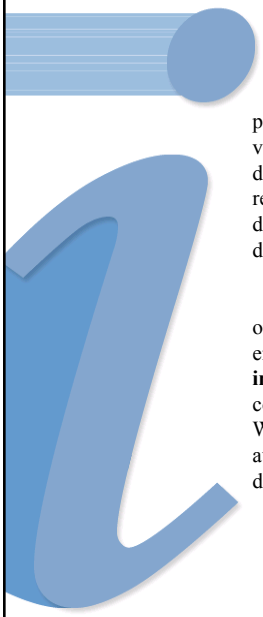
Prioritize the data you intend to test by these other schedules, and by complexity; if possible (big time saver) build larger and larger sets of data by merging these databases in the order they in which they will rollout. You'll end up testing larger and larger collections of data, spreading your test cases across them; you'll be testing wider rather than deeper, looking for holes in the merging process.

Do's and Don'ts



Negotiate your testing scope throughout the process. As noted above, testing scope creep is something that will occur, for various reasons; working with this creep rather than fighting it will move you further ahead. Use scope creep as an opportunity to negotiate for additional resources and time rather than attempting to bar it altogether; should the increase in scope be removed for some reason (and it may, as the requirements associated with data conversion in particular are fluid), you'll lose the added work, but in many cases be able to hold onto the resources.

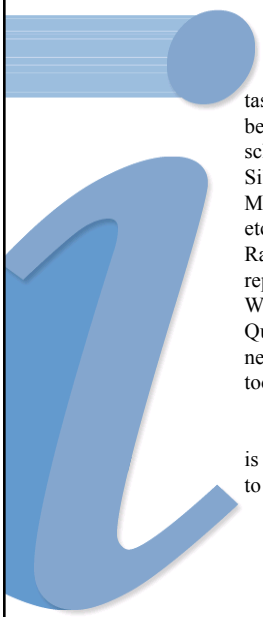
Do's and Don'ts



Always be **prepared for major environmental issues**, particularly as the version of your application approaches the version under test by the system test group; the code will be more developed at that point (read: complicated due to numerous defect repairs, change requests, etc.) and the likelihood that the converted data will create 'significant environmental issues' when first deployed rises markedly.

If the standard smoke testing by systems test is an hour or two, count on a full day or more of smoke test/break environment cycles. In your smoke tests, **focus on database intensive events**: build new records, using high levels of detail; conduct extensive database searches against large records; etc. When the environment goes down, be sure to have resources available to restart or re-deploy the software; having a database developer and DbA on call as well can be very helpful.

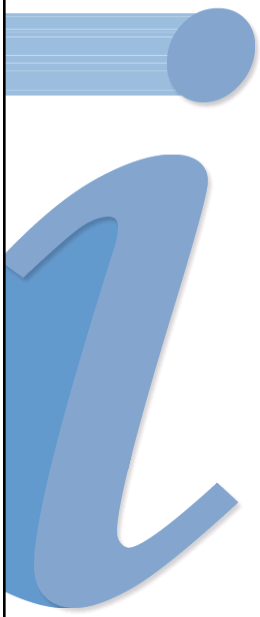
Do's and Don'ts



Use your tools. Spreadsheets can be used for any management task; coffee can be made with a simple pot and hot water, but a much better way of brewing is to use professional tools. Manage your test schedule and control your tests with Mercury's Test Director, Segue's SilkPlan Pro, Rational's TestStudio, etc. Manage your defects with Merant's PVCS Tracker, Seapine's Test Track Pro, Segue's SilkRadar, etc. Control your code and files with Merant's PVCS Version Manager, Rational's ClearCase, McCabe's TRUEChange, etc. Automate the repetitive portion of your testing with Mercury's QuickTest or WinRunner, Parasoft's JTest or WebKing, McCabe's McCabe Test, etc. Query the old database for specific records you'll need to look for in the new database; push those queries into some of the available regression tools, where possible, and let the machine do the work.

More so than any other testing group, data conversion testing is repetitive; the more you automate, the easier your overall task is going to be.

Key Indicators

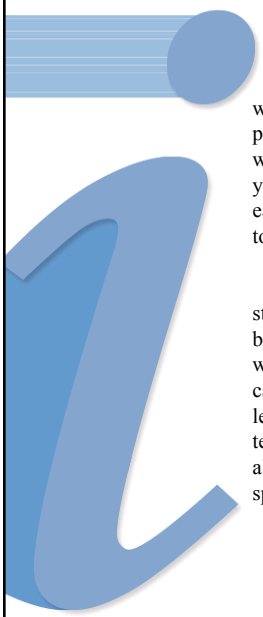


The application schedule will drive yours; anticipate slips of your own based on the tempo of the application's schedule slips. For instance, if the data conversion development staff typically lags application's iteration schedule by a month, and they slip two weeks, you can expect an additional week to two weeks of slip in about a month unless the data conversion development group expends energy to head it off. This indicator keys off application complexity issues, not other slip drivers; a slip of schedule due to unavailability of testing personnel in the system test group will represent a bonus to your timeline rather than an opportunity for a slip down the road.

The data model is important enough to you that any slip it its versioning release or final freeze will directly impact your timeline; keep a close eye on this portion of the overall timeline, and if you have resources that can assist in the event of a slip, make them available to help. Knowledge of the data model is key in any event, and if you can mitigate a slip here, you'll be mitigating a slip to your testing schedule.

The performance tuning schedule, and its impact on the data model, is another useful indicator. If they're behind, they'll likely extend the time before the final data model freeze with performance enhancements, resulting in additional development by the data conversion group, and requiring both additional testing and a later start for your group than would be otherwise.

Key Indicators



There will be a rhythm to your resources, as they're often borrowed; when your BAs or testing staff are busy working on another portion of the project, you'll get only a few hours a day from them, if that. Attempt to work within this rhythm, anticipate it in your project schedule, and push your schedule out as far as feasible. Communicate this schedule and get early commitments from your resources, so when you've got data available to be tested you'll also have the resources on hand to test it.

Defect counts are important for any project. When the defect rates start to taper off, when the fixed rate begins to climb, when the delta between these two measures begins to shrink, things are obviously going well. However, you also must pay attention to your test approach: which category of testing is netting you the most bugs for time spent? Which the least? When should one category get the lion's share of your valuable testing time, and when should one be cut out from your testing cycle altogether? Watching your defects, where they come from, and the time spent to find them will give you the answer.

Likely Difficulties and Risks

- Lowest priority, at least at the onset
- Technical subject matter
- Low to no budget
- Business involvement
- Development involvement
- High, hard to fix and isolate, defect rates
- Whose bug is it, anyway?

Important Indicators

- Overall project schedule slip
- Data model slip
- Performance tuning schedule
- Defect counts by testing category

Do's and Don'ts

- Understand the development process in use
- Have a defined test approach
- Establish close relationships with key people
- Train your team
- Have a communications plan
- Revisit requirements often
- Understand the other group's plans
- Negotiate your testing scope throughout the process
- Always be prepared for major environmental issues
- Use your tools
- Avoid becoming the sponge for everyone else's project
- Avoid distractions
- Never, ever negotiate away your time
- Never become a one-person show

iBeta
Software Quality Assurance

Address:
iBeta Software Quality Assurance
2101 South Blackhawk Street
Suite 100
Aurora, CO 80014

Phone: (303) 627-1110
Fax: (303) 627-1233

Web Address: http://www.iBeta.com

General inquiries: info@ibeta.com
Sales inquiries: sales@ibeta.com

Legacy Data Conversion: Making Coffee

Joshua Kitchen

Converting data from a legacy application is a lot like making coffee.

The existing data (grounds/beans) work well enough in their current state (if you like pulling beans out of the bag or eating spoonfuls of grounds, but then we already know we want a new app, don't we?). To work properly, however, in our new application the data must undergo change yet remain essentially the same. So first the grinding, either from beans or from a courser grade of grounds (cleansing garbage data from the system prior to trying to converting it), then the pouring into the filter and running hot water through it (conversion to new schema, applying different integrity rules, interpolating new fields from existing ones, etc.) until the finished coffee finally resides in the pot (the new application database). In theory, if all went well, what's in the pot is what was desired; the user can drink it and be well satisfied with the temperature, the flavor, the aroma, the viscosity…but theory and fact often part company. So who do we want to drink the first cup, our user group/customer or our testing group? That was my answer, too.

Tackling the testing problem can follow the coffee analogy also; look at the different test techniques that are typically applied during data conversion testing:

Requirements testing (the beans were supposed to be ground to some fineness; did the coffee turn out that fine, or is it overly course?)

Functional testing (no matter what happens, someone's still got to taste the coffee)

Domain testing (pick out beans that best represent the bag)

Performance testing (don't just drop a tea spoon in and point at colored water; if conversion has to occur over a two-day weekend, ***that's*** the product we most want to test on, not something that's been simmering on the pot for a couple of weeks, converted one slow piece at a time until it's well blended)

We all have our own ideas on making coffee, our own preferences; likely nothing in this white paper will be new. As in making coffee, however, rarely do we sit down and go over the risks and rules of thumb to be used in effectively testing data conversion.

Here's a recipe I used.

The structure of this presentation is simple:

1. A look at the difficulties/risks encountered, with some perspective on overcoming or mitigating them.
2. Collected of rules of thumb that worked well (or not so well) during test planning and execution.
3. Summary of items to watch for before, during, and after. Indicators help only so far as we pay attention to them.

Likely Difficulties and Risks

Lowest priority, at least at the onset
Technical subject matter
Low to no budget
Business involvement
Development involvement
High, hard to fix and isolate, defect rates
Whose bug is it, anyway?

Do's and Don'ts

Understand the development process in use
Have a defined test approach
Establish close relationships with key people
Train your team
Have a communications plan
Revisit requirements often
Understand the other group's plans
Negotiate your testing scope throughout the process
Always be prepared for major environmental issues
Use your tools
Avoid becoming the sponge for everyone else's project
Avoid distractions
Never, ever negotiate away your time
Never become a one-person show

Important Indicators

Overall project schedule slip
Data model slip
Performance tuning schedule
Defect counts by testing category

Likely Difficulties/Risks

Lowest priority, at least at the onset
Technical subject matter
Low to no budget
Business involvement
Development involvement
High, hard to fix and isolate, defect rates
Whose bug is it, anyway?

Data conversion tends to be something of an afterthought in most projects (now, where did I put the coffee? Must be up here somewhere…), except those that launch specifically with this goal in mind.

So many other things seem to be of higher priority: requirements for the upcoming application, initial coding and testing of that application, pushing the new application to the required levels of performance, the various 'must have' change control requests, high priority defects requiring major re-engineering of the application, usability issues…the list goes on and on.

And, too, data conversion is a more technical, under the covers event; most of the details are extremely esoteric, the results are far from exciting, and reporting on progress tends to be rather dry. 'I can't send email from the app!' gets far more attention than 'Integrity enforcement at the app level is complicating conversion of people records'.

Thus, data conversion tends to be thought of last, usually about the same time that planning begins for product rollout, an unfortunate timing as it usually means this effort is last in line for resources, on the critical path, and with little time to plan let alone code and test.

With data conversion behind and ill understood from the onset, testing of data conversion is likewise behind the curve. Gradually, however, the ramifications of insufficiently testing the conversion process and final converted data sink in at various levels and the testing effort receives more and more (often becoming unwelcome; everyone wants to either help make a pot or get a taste) attention.

A key factor in the success of the data conversion testing effort becomes getting enough visibility early on; this visibility is critical to getting the resources and time assigned before they become critical, enabling the data conversion testing effort and, by proxy, the data conversion effort, to succeed.

Waiting too long makes data conversion testing a bitter drink no one wants: under resourced, out of time, and with huge expectations on the part of every stakeholder.

The key steps to getting the right amount of (early) attention to data conversion testing are familiar to any software project manager: take charge, draw up a rough plan or approach, present to stakeholders (and present to stakeholders, and present to stakeholders, and present…).

Taking charge of a sub-project that needs doing, yet isn't funded or recognized, is a great deal like petting a shark: an interesting experience, something to tell one's grand-children about, but one earning the nickname 'Lefty' if one is not careful. However, there's really no other way to go about it; if it needs to be done, and wasn't in the project plan from inception, someone's going to have to be a hero and stand up for it, or it will likely fail.

This paper focuses on the testing piece of data conversion, so if you're deciding on standing up for testing, likely you've a counterpart who's decided to stand up for the development. Make friends; you'll be in this together, and for the duration, else you'll be failing separately.

Once you've made friends with your development counterpart, drawing up a rough plan should be fairly easy; we'll cover additional planning considerations elsewhere in this white paper. It will take you a bit of time to draw up, as most good plans that account for specific resources needed do, but the majority of the issues you fight your way through will be organizational rather than technical, so there should be few surprises (knock on wood).

The presentation bit will be harder, as you'll have to impress a non-technical group with the technical ramifications of failure. It may seem, at this point, that you are antagonistic to your partner on the development side; after all, you're the one telling stakeholders that your buddy's project may fail. But by making clear the potential issues associated with data conversion (i.e. That the entire project could fail if this piece isn't taken care of), you're trying to not only get resources of your own, but also to increase the resources development has available.

Make no mistake, data conversion testing is quality control, finding errors generated because of short timelines, inadequate or missing requirements, etc; anything you can do to make development's life easier, any resource you can free up to assist their end of the project, anything you can do to prevent them from

having to rush (without impacting the timeline you're setting up for yourself), or take shortcuts, will make the project's life (and yours) easier.

As mentioned earlier, data conversion testing tends to be an afterthought; by the time you come onto the scene, budgets have been made (and hopefully approved) for development, rollout, and testing of the application.

The polite battle for resources will continue throughout the process; accept this. The applications development group will be behind schedule, and wish to appropriate your testing environment for additional development; the roll-out group will wish to appropriate your analysts and subject matter experts to assist in planning, training, or additional requirements review; the other testing groups likewise want your environment, equipment, testing staff, or to pass some of their testing scope off to your group.

To say not to give in to these demands/requests for the sake of the overall project would be naïve; the end goal is still to come out on time, on budget, with the feature set and reliability targeted. Caution is useful, however, and a well-managed scope and risk management document is a good tool to keep your team out trouble.

Your testing environment becomes your most critical resource, because you need a minimum of two (both a controlled legacy and a controlled developed application for comparing data and functionality before and after data transformation), and because your environment is typically larger than most others, at least until final integration and rollout of the product: you need legacy data, and lots of it, if you're to complete your task. As an additional hardware/software cost to the project, you'll have an uphill battle to get set up; likely you will have to work for at least a portion of the time in and around development's environments, the other testing environments, or perhaps on older equipment handed down from production. Any of these events will slow your testing when compared to the rest of the testing group…and any time a milestone in another group is missed, you will likely lose your testing environment in favor of that group.

Business involvement. Good BA's/SME's are worth their weight in gold; fortunately or unfortunately, the rest of the development and test groups are aware of this as well. Data conversion is particularly vulnerable to poor requirements gathering issues, and the analyst is the person most likely to minimize this threat. The primary pull on your BA's time will be development; after all, if development doesn't know what to build, they can't build it, and if they can't build it, you've nothing to test.

Participating in development's requirements gathering process (if you can) will bring big dividends to your testing group down the road: you establish good relationships with BA's, get a jump on test planning, and grow an understanding of the application all in one step. It's a very worthwhile step; take it.

A negative, at least from your perspective, of BA involvement may occur when they are over involved; they may push test priorities (Test this first! And this! And this!), over-prioritizing defects (No, the system MUST not do this! What? The system doesn't do THAT? Unacceptable.), and generally attempting to do your job (part-time; most BA's, by definition, could contribute significantly if managing a testing effort. However, they have other jobs and can hurt more than help if their role in your project is not managed.)

BA's tend to have the ears of the primary stakeholders; if their concerns, both voiced and unvoiced, are not answered they often go to these stakeholders with their concerns. This leads to a variety of unfortunate issues, as you're well aware/have experienced. Listen to the BA's.

Don't necessarily treat them with kid gloves when they offer a strong opinion, but don't ignore them either. Regular communication via email, or better, regular meetings in which they can make their concerns known and have them answered go a long way to making them happy with the testing group and productive members of same.

Everyone may want to drive. Everyone has priorities; these priorities won't match each other's or yours in many respects. Senior people (Dev, BA's, VP's) are necessary to projects to succeed and, by definition, these are people used to driving. If they are not hearing what they need to hear, or seeing what they need to see (i.e. progress, milestones being met, etc.) they have a tendency to insist on more and more reports, and on direct involvement in your testing process. This distracts them from their tasks, which are essential for your success and are definitely necessary for the success of the project as a whole.

Give them what they need: regular, organized reporting. Let them smell the coffee brewing. If they understand what you're are doing, and where you are in your project path, they can rest (perhaps not easily) assured that the right things are being done at the right time, and they will stay out of your way.

Development is a slightly different animal. When development wants to drive the testing process, generally two things can happen: the testing staff is pulled off testing to assist in development (either in reality, in which case they are punching code for a development manager, or figuratively, in which case they are diving into testing detail more along the lines of debugging rather than testing, an inefficient use of their time) or testing is ineffective (defects are downgraded in priority, repeatedly deferred to other code iterations, or simply dismissed as unimportant).

Development must not drive your testing. There's no easy way to say this, but they must not, or your position will swiftly become irrelevant, redundant, and a waste of project resources.

High defect rates. Data conversion may encounter a higher number of defects than other groups; while not inevitable, it is a distinct possibility you will encounter more defects due to three data conversion specific issues: accelerated requirements creep, coordination between different groups of developers, and the impact of database changes.

Every project has requirement creep; its a fact of software development. Data conversion in particular will suffer from this creep, not only from its own requirements, but from the requirements of the application. A defect/requirement added to the application team will push a change to the data model which will in turn push a change to data conversion code which will in turn push a change in the testing…this is inevitable.

It is absolutely essential to understand when the data model is due to be frozen; without a grasp on this date, and the understanding of its importance in the other development/testing groups, your team will churn farther and farther behind the rest of the project.

If your development group is different from the application development group, you will have to deal with another layer or two of communication between the database team and the application team. Poor communication often leads to simple, but telling errors in converting the data.

The new application's data model will keep changing due to the efforts of the application development group and from other testing groups, particularly the performance group. Performance tuning may significantly change the model in a stroke (e.g. de-normalizing the data model, or portions of the data model); an additional feature or defect repair may make a minor change, but require a major data conversion code revision.

Due to the typically late start and technical issues surrounding requirements gathering for data conversion, defects often arise from misunderstood or missed requirements. Data conversion is typically very technical, more concerned with field lengths and data types than with application level processing or interface display; the requirements often differ enormously from the requirements of the application team, but require the input from people (the BAs) who may not have a firm grasp of the technical ramifications involved.

This combined with the late start often results in good code that nevertheless results in an unsuitable conversion of legacy data (i.e. defects).

Whose bug is it, anyway? There will be recurring debate between the application and data conversion development groups as to the cause of a given defect, with either the application's processes or the data conversion process being the proximate cause.

Essentially, there are two applications in use: the primary application under development and the secondary application needed to transform the data; in most cases, a defect found by your testing group could be found in either one. Heading off impasses in this area is critical to successfully resolving defects and speeding the testing process to completion. If you can't sort which caused the silt in the bottom of the pot, the grounds or the filter, you'll never make a good cup except by accident.

Any integrity enforcement at the application level, if not carefully understood, can create havoc in the application when it is run on converted data. Records can be completely un-viewable, processes can crash, the database can error…a host of things can go wrong. Understanding what integrity is enforced at the application level and what is enforced at the database level is vital for successful testing of converted data.

Calculations or inferences drawn by the application from the data is likewise an area of understanding critical to successful testing; if an item is wrong when viewed at the interface a significant amount of time can be wasted trying to understand that the application is altering the data before it is being viewed.

Do's and Don'ts of Data Conversion Testing (rules of thumb)

Understand the development process in use
Have a defined test approach
Establish close relationships with key people
Train your team
Have a communications plan
Revisit requirements often
Understand the other group's plans
Negotiate your testing scope throughout the process
Always be prepared for major environmental issues
Use your tools
Avoid becoming the sponge for everyone else's project
Avoid distractions
Never, ever negotiate away your time
Never become a one-person show

Understand the development process in use. Understanding the process development is using will help the data conversion testing process immensely (especially if you grow to understand they don't have a process!). This applies not just to the data conversion group; to get defects fixed in a timely fashion, knowing who does what (Mr. Smith the defect coordinator assigns defects to the engineers on staff), and how they do it (a fix has to have a code sample to go with it, so no defect can be marked fixed in the database without that code sample) will significantly smooth the process. Where possible, borrow from their processes; things move faster when both testing and development are speaking the same language.

If the process is written, good; if it is being followed, even better. Look for the holes, places defects can be lost and not repaired for significant periods of time. Ask about escalation; how long can a defect remain in the system at a given priority before it moves up in the queue? What is an acceptable fix failure rate? 20%? 10%? 50%? What does the development staff consider a reasonable number? Then take that number back to the other testing groups and project managers and make them aware of it; a high expected failure rate is an indication you'll need additional time testing, as you can expect a defect regression test set to grow significantly during your testing cycle.

What deliverables that development produces can help your testing? The data model, for one; a summary of application enforce integrities is another; a list of records that will be deleted from the database is a third. These items can answer key questions and save time that would otherwise be spent in multiple meetings or brainstorming sessions.

Have a defined test approach. Conversion testing will be the last car in a very long train of development and testing; as we've already discussed, this testing is likely an afterthought. Having a defined, robust, and well-understood test approach is useful both in communicating the scope and intent of testing and speeding that testing along successfully.

For data conversion testing, I recommend a four-part approach as a beginning followed by focus and refinement as testing continues. In order of priority and general utility, a good organization scheme is what we referred to as screen to screen (S2S), functionality, database partitioning, and defect regression.

The biggest bang for the buck is in the screen to screen comparisons: an instance of the old system is placed online and a snapshot of the same database is used on an instance of the new application. Testing then becomes a matter of 'camera one, camera two' comparison. Organization further helps this effort; a simple Excel spreadsheet, for example, with a list of the fields and their locations in the new app, along with the equivalent fields and locations from the old app, is extremely handy. Addition of the transformation logic to the document further enhances its utility.

After the S2S mapping, running a subset of the functionality testing on the converted data is essential to bug hunting. Despite all the requirements gathering, development, and testing to date, until the system is testing on the live data no real 'feel' for product progress exists. Particularly when different teams are doing the conversion and application development, errors creep into the boundaries between the software and its data; test data, no matter how well crafted, never truly compares to the real thing.

Database partitioning is a form of equivalence class partitioning in which the database is sub-divided into categories of data, then again, and again, until the types of records noted are a representative sample of the legacy system. This testing gives you a large handful of records that can then be used to validate the entire converted system.

Be prepared to run a defect regression set, at least initially, to validate the defect repair failure rate. If that rate is high, resign yourself to re-testing a significant portion of your previously reported defects with each major change of code.

As you test, review which of your approaches net the most relevant defects, and which aren't netting enough to be worth your time; prioritize the former and reduce the time spent on the later. Different development staff and strategies lend themselves to different types of errors; each of the four approaches above lends themselves to finding particular types of errors.

Establish close relationships with key people. As alluded to earlier, the people you will need to make your testing successful are already very tasked; at best, you are likely to have only partial resources for some key activities. BAs provide you with information you need; project managers can keep you aware of key timelines or political issues; friendly developers can help you understand the underlying mechanics of the new application, or data conversion code, and help your defect resolution along; other testing groups on the same application can give you a heads up on new functionality or recurring issues with the application; Systems folks can get your environment up in record time; DbAs can get that database refresh done for you quicker and cleaner than you expected.

Or not.

Disregarding the political aspects of having/making friends in key places, knowing these people and how they work (what information and time they need to perform for you, what other priorities they have on their plate, etc.) will make your testing cycle (and preparations for it) much, much easier.

Train your team. Largely due to the 'loaner' nature of your resources, it is strongly recommended that you train on your testing process again and again, especially if there are gaps between iterations. The people assigned to you will be working their other tasks in, around, and between your testing cycles and will need refreshing on your process and instruction on any changes you may (you will) make to said process.

Without a good (not perfect, but good) understanding of the new application's functionality, testing how that functionality performs (or rather, fails to perform) when subjected to real data is difficult. A good approach is to either mix in some testers from the application group (if they can be spared), or put some of your test group into the system test group for a space. If the application group is writing detailed testing scripts, borrowing these scripts will significantly aid in data conversion testing.

The databases must have been reviewed for both the conversion team (old database, plus transformation mapping to the new database) and the application team (new database); understanding the database schema is essential to timely completion of testing, as queries for test data must be run against the old database to find the records of interest in the new database (reviewed through the application).

Have a communications plan. Testers need to report to you: defects, testing completed, significant issues in the testing environments, etc. You will need to report up as well: project gantts, test status and completion, opinions of the overall quality of the converted data. You will also have to track defects to completion, answer development's questions on those defects, and digest their answers.

Make a list (And check it twice, three times, four... Know what the deliverable is, who is suppose to produce it, when it is to be produced, and, last but not least, make sure you know what its for! Anything used for communication but without a defined purpose is likely a waste of your time, and likely your testers'/associated developers'/project manager's time as well.

A good laundry list for a communications plan runs along the following lines: weekly tester meeting, weekly test lead (cross-group) meeting, monthly project meeting, daily defect review meeting, pre-cycle meeting (with development, testing, and support present), project gantt, testing assignments, testing results, defect reports, pre-testing check off, development's project gantt (both development groups), etc.

Revisit requirements often. Requirements will often change dramatically on you in data conversion; what was first a hard and fast will-not-budge business requirement may turn a hundred and eighty degrees around once that requirement is put into practice and business can see the results. By revisiting them with business and development, you ensure the three groups are still on the same page, and save your testing staff a tremendous amount of time in wasted testing against out of date requirements.

This 'revisiting' can occur a variety of formats; it is useful to set aside a portion of the defect review meetings to discuss these kinds of issues (as they will happen anyway during testing, setting the time aside and designating it as such will both speed the defect review and accomplish the task: development, testing, and business in the same room discussing the same requirements and how best to proceed.

Just make sure decisions get documented.

Separate requirements meetings are also useful even during the testing cycle if it becomes apparent that significant changes need to be made to the application and/or data conversion code. Testing's presence in these meetings are essential as they may change the scope of testing, invalidate large sections testing previously scheduled, and generally discuss the overall fate of the application.

Understand the other group's plans, in particular the rollout and application testing plans. Both of these plans may (should) require your converted data as a key milestone; they may have particular needs. In a legacy application with multiple databases, these groups may already have a required sequence in which those databases will be integrated into the final product. This sequence will drive your decision(s) of what to test and when.

Prioritize the data you intend to test by these other schedules, and by complexity; if possible (big time saver) build larger and larger sets of data by merging these databases in the order they in which they will rollout. You'll end up testing larger and larger collections of data, spreading your test cases across them; you'll be testing wider rather than deeper, looking for holes in the merging process.

Negotiate your testing scope throughout the process. As noted above, testing scope creep is something that will occur, for various reasons; working with this creep rather than fighting it will move you further ahead. Use scope creep as an opportunity to negotiate for additional resources and time rather than attempting to bar it altogether; should the increase in scope be removed for some reason (and it may, as the requirements associated with data conversion in particular are fluid), you'll lose the added work, but in many cases be able to hold onto the resources.

Always be prepared for major environmental issues, particularly as the version of your application approaches the version under test by the system test group; the code will be more developed at that point (read: complicated due to numerous defect repairs, change requests, etc.) and the likelihood that the converted data will create 'significant environmental issues' when first deployed rises markedly.

If the standard smoke testing by systems test is an hour or two, count on a full day or more of smoke test/break environment cycles. In your smoke tests, focus on database intensive events: build new records, using high levels of detail; conduct extensive database searches against large records; etc. When the environment goes down, be sure to have resources available to restart or re-deploy the software; having a database developer and DbA on call as well can be very helpful.

Use your tools. Spreadsheets can be used for any management task; coffee can be made with a simple pot and hot water, but a much better way of brewing is to use professional tools. Manage your test schedule and control your tests with Mercury's Test Director, Segue's SilkPlan Pro, Rational's TestStudio, etc. Manage your defects with Merant's PVCS Tracker, Seapine's Test Track Pro, Segue's SilkRadar, etc. Control your code and files with Merant's PVCS Version Manager, Rational's ClearCase, McCabe's TRUEChange, etc. Automate the repetitive portion of your testing with Mercury's QuickTest or WinRunner,

Parasoft's JTest or WebKing, McCabe's McCabe Test, etc. Query the old database for specific records you'll need to look for in the new database; push those queries into some of the available regression tools, where possible, and let the machine do the work.

More so than any other testing group, data conversion testing is repetitive; the more you automate, the easier your overall task is going to be.

Avoid becoming the sponge for everyone else's project needs; if someone needs an environment you own, and it would honestly further the project, by all means help. Be prepared to deal, however, and look clearly at your timeline and resources; play it straight with them, and see if there's something they can commit to that will help you (and again, the project as a whole) down the line. Get these commitments identified and agreed to early, and make them visible at the project management level to lock them in.

Avoid distractions. Simply because someone says a particular part of the sky is falling, even if they have significant presence on the project, don't drop what you're doing to comply. Meet them halfway; talk through the issues, preferably face to face. If need be, calm them down and discuss how testing for this particular threat may take resources away from other key areas. Make them a friend to aid you in getting additional resources and time. And keep in mind that their point may be very valid, but insulate your testing staff from this point using the above-recommended methods until you've got a chance to validate the concern (or debunk it) and have a plan to deal with it.

Never, ever negotiate away your time, even to meet critical project milestones. As the last caboose on the train, you'll be lucky to come in anywhere close to the original project deadlines; the rest of the cars will be in the terminal and waiting impatiently for you to arrive. That said, there is one (1) instance in which time (calendar) should be negotiated away; when it results in tangible (i.e. existing and set up) environments/testing staff who's presence on your portion of the project would reduced your timeline well within the time you're negotiating away.

Never give up.

Never become a one-person show. If you've got a staff of one, and that one is you, you'll be in trouble no matter what your testing skills.

Despite the need to guard your own time and resources, keep an eye to the project overall; if your group is doing well, but there are others not doing so well, be ready to lend a hand. In the end, you only succeed if everyone succeeds.

Key Indicators

Overall project schedule slip
Data model slip
Performance tuning schedule
Defect counts by testing category

The application schedule will drive yours; anticipate slips of your own based on the tempo of the application's schedule slips. For instance, if the data conversion development staff typically lags application's iteration schedule by a month, and they slip two weeks, you can expect an additional week to two weeks of slip in about a month unless the data conversion development group expends energy to head it off. This indicator keys off application complexity issues, not other slip drivers; a slip of schedule due to unavailability of testing personnel in the system test group will represent a bonus to your timeline rather than an opportunity for a slip down the road.

The data model is important enough to you that any slip it its versioning release or final freeze will directly impact your timeline; keep a close eye on this portion of the overall timeline, and if you have resources that can assist in the event of a slip, make them available to help. Knowledge of the data model is key in any event, and if you can mitigate a slip here, you'll be mitigating a slip to your testing schedule.

The performance tuning schedule, and its impact on the data model, is another useful indicator. If they're behind, they'll likely extend the time before the final data model freeze with performance enhancements, resulting in additional development by the data conversion group, and requiring both additional testing and a later start for your group than would be otherwise.

There will be a rhythm to your resources, as they're often borrowed; when your BAs or testing staff are busy working on another portion of the project, you'll get only a few hours a day from them, if that. Attempt to work within this rhythm, anticipate it in your project schedule, and push your schedule out as far as feasible. Communicate this schedule and get early commitments from your resources, so when you've got data available to be tested you'll also have the resources on hand to test it.

Defect counts are important for any project. When the defect rates start to taper off, when the fixed rate begins to climb, when the delta between these two measures begins to shrink, things are obviously going well. However, you also must pay attention to your test approach: which category of testing is netting you the most bugs for time spent? Which the least? When should one category get the lion's share of your valuable testing time, and when should one be cut out from your testing cycle altogether? Watching your defects, where they come from, and the time spent to find them will give you the answer.

QW2002 Paper 2I1

Dr. Ludmila Cherkasova, Dr. Yun Fu, Dr. Wenting Tang & Dr. Amin Vahdat (Hewlett Packard Labs)

Web Site Performance and Qos Monitoring

Key Points

- A novel technique and tool for measuring the web site's end-to-end response time
- A set of new metrics to assess a web site performance and efficiency
- A practical performance case study using three commercial web sites

Presentation Abstract

Understanding and measuring end-to-end service performance perceived by the clients is a challenging task. In this work, we describe a novel technique for measuring the web site's end-to-end response time which based on reconstruction of web page accesses from passively captured network packets at a web server side. This technique can effectively determine the set of objects composing web page without parsing or interpreting the HTML syntax. Exploiting this technique, we built a tool, called EtE monitor. EtE monitor does not require any changes or modifications to site content or server side infrastructure, or client browsers, and can be used for sites with static or dynamically generated content.

Relative to existing approaches, EtE monitor offers a set of new benefits. Timestamps extracted from network packet level provide invaluable information about connection setup time, server side processing time, and network related transfer time, which can not be directly obtained from other sources. Additionally, this technique allows us to analyze many other useful and practical metrics, such as number of aborted page accesses, number of objects retrieved directly from the server versus from network and client browser caches, number of client connections used to retrieve the web page, amount of packets resent in the response to reflect network congestion on the path to a client, etc.

About the Author

Ludmila Cherkasova joined Hewlett-Packard Laboratories, Palo-Alto, California, in 1991, where she currently is a senior scientist in the Internet and Computing Platforms Research Center. Among her latest works are WebMetrix (a tool for web service providers), MediaMetrics (a tool for media service providers), content-aware load balancing strategies for web server clusters, a scaleable architecture for shared web hosting service, the admission control and scheduling algorithms that improve performance of overloaded web servers. Her research interests include the performance of Internet-based services, computer systems, network protocols, and characterization of next generation system workloads. She received a PhD in Computer Science from Novosibirsk Institute of Computing Systems, Russia, in 1984. She is a member of ACM and USENIX.

Yun Fu is a Ph.D candidate at Computer Science Department of Duke University. He received his B.S. in computer science and Master degree in computer engineering from Nankai Univeristy in 1995 and 1998 respectively, and his MS in computer science from Duke University in 2001. His research interests include distributed systems, networking and operating systems.

Wenting Tang is a member of technical staff in Internet System and Storage Laboratory, Hewlett Packard, Palo Alto, California. His research interests include Internet service performance, content delivery networks, streaming media and large-scale distributed systems. He received his PhD in computer science from Michigan State University. He is a member of the IEEE.

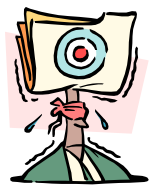
Amin Vahdat joined the faculty of the the Computer Science department at Duke University after receiving a PhD from the University of California, Berkeley in 1998. His research focuses on system support for highly

Web Site Performance and QoS Monitoring: *Measuring Client Response Time and More …* at the Web Server

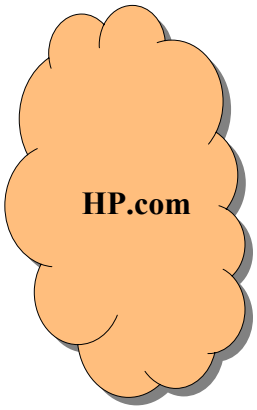
Lucy Cherkasova, Yun Fu, Wenting Tang, and Amin Vahdat

HPLabs and Duke University

Service provider problems...



???



- A lot of research is done to optimize web server performance in order to improve client experience

BUT

- Do we know *what is the client experience?*
- *What are the critical latency components* in the end-to-end response time?
- Do we know whether the improvements on the web server side *indeed improve end-user experience*?
- Do we know *who the clients* are and *where they are located* on the Internet?

End-to-End Web Service Measurement: Why Is It Important?

- Two *main factors* impact the response time perceived by the clients: *network latency* and *server side processing time*
- Many web sites use *complex multi-tiered architecture*
- A set of *new technologies*, such as servlets and Javaserver pages, extend the web servers to generate information-rich *dynamic web pages* and to leverage existing business systems
- Combination of these technologies could lead to increased server-side processing time especially in *distributed environment*
- *New ad-hoc business metric:* web service is considered to be "unavailable" if its response time exceeds *6 seconds*
- The service providers need a quantitative analysis of the major latency components contributing to the response time to achieve given business and QoS objectives:
  - Invest in more powerful site infrastructure or
  - Choose a CDN service?

Why Is It Difficult?

- Web pages are *complex* objects with *multiple embedded images*
- HTTP protocol is *stateless*: different images are requested by client browser independently:
  - Some of them are issued concurrently
  - Some of them use persistent connections
  - Some of them are obtained from proxies
  - Some of them are obtained from user browser caches
- The response time of a web page observed by the client is *the result of download of all page related images*

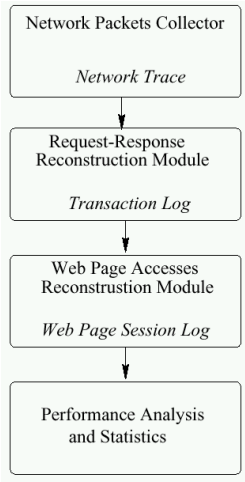
What Are Currently Available Solutions?

- *Active periodic probing* of a particular web page from a fixed number of clients across the Internet
  - Keynote service
    - Keynote "clients" are not the real web site clients
    - Allows monitoring of a particular web page
    - Always pulls the entire page (with all embedded images) from the server
- *Page instrumentation technique* based on downloadable JavaScript or Java Applet to a client web browser
  - HP Open View "Web Transaction Observer"
    - The measurement starts after download of the main html page (significant portion of the response time is missing)
    - Does not provide latency breakdown unless the web server is also instrumented
  - eBusiness Assurance (eBA, from Candle Corp)
  - Quality of Service (QoS) Monitor (IBM, Tivoli)
  - Research paper by Rajamony and Elnozahy from IBM (Austin) uses JavaScript to instrument the links to particular pages. Somewhat more limited: cannot measure directly accessed pages, e.g "*index.html*"…

What Do We Propose?

- **EtE monitor**
  - Passive monitoring tool for end-to-end response time measurement
  - Non-intrusive, does not require any changes or modifications to a site content, or server side infrastructure, or client browsers
  - Can be used for sites with static or dynamically generated content
- What does it provide?
  - End-to-end response measurement for ***all the pages*** and ***all the clients*** accessing the site
  - Analysis of response components:
    - ***Server processing*** time portion
    - ***Network transfer*** time portion
  - Reports the % of data delivered from the server vs the % of data cached on the client side
  - Reports the % of ***aborted page accesses*** and the related performance reasons
  - Analysis of the ***most frequently accessed documents*** and their response time
  - Client clustering by ASes (Autonomous Systems)
    - Requests (bytes) clustering by ASes and the corresponding response time
  - ***And more …..***

EtE Monitor Architecture

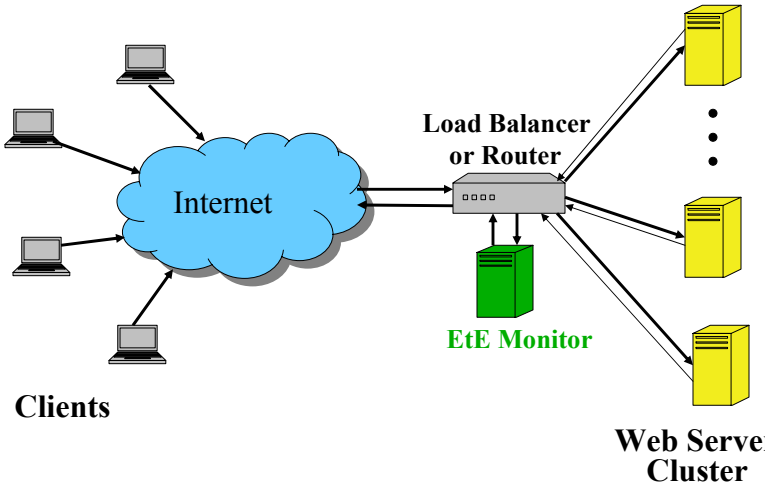


1. The ***Network Packet Collector*** module: collects network packets using *tcpdump* and records them in *Network Trace* enabling offline analysis.
2. In the ***Request-Response Reconstruction*** module, EtE monitor reconstructs all TCP connections from the *Network Trace* and extracts HTTP transactions (a request with corresponding response) from the payload. EtE monitor stores the HTTP header lines and other related information in the *Transaction Log*
3. The ***Web Page Reconstruction*** module is responsible for grouping the request-response pairs into logical web page accesses and stores them in the *Web Page Session Log*
4. The ***Performance Analysis and Statistics*** module summarizes a variety of performance characteristics integrated across all client accesses

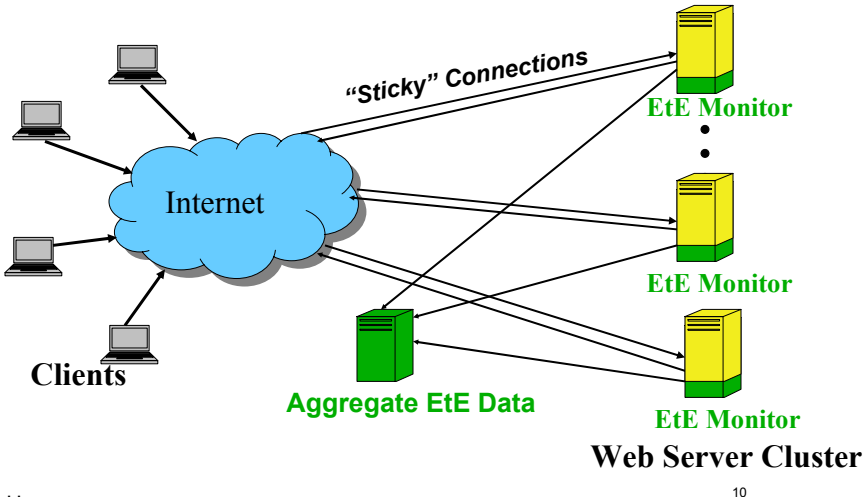
EtE Monitor Deployment Configurations

- EtE monitor can be deployed in several different ways:
  - An independent ***network appliance*** at a point on the network where it can capture all HTTP transactions for a web server or web server cluster
  - A ***software solution*** installed on a web server to monitor web transactions on a particular web server
  - A ***mixed software solution*** with some modules (*Network Packet Collector* module and *Request-Response Reconstruction* module) installed on a web server to collect network packet and process them into *Transaction Log*, and the other two modules residing on some other ISP server to minimize the EtE monitor performance overhead at a given web server

Deployment Configuration: EtE monitor as a Network Appliance



Deployment Configuration: EtE monitor as a Software Solution



Request-Response Reconstruction Module

- The TCP connections are rebuilt from *Network Trace* using:
  - The client IP address
  - The client port number
  - The request (response) TCP sequence number
- Within the payload of the rebuilt TCP connections, HTTP transactions are delimited as defined by HTTP protocol
- After reconstructing the HTTP transactions, the monitor records the HTTP header lines and other information of interest in the *Transaction Log* and discards the transaction body

Request-Response Reconstruction Module (continuation)

- Each *entry* in the *Transaction Log* includes:
  - The client IP address
  - A unique flow ID for TCP connection
  - The requested *URL*
  - The content type
  - The payload size
  - The *referer* field
  - The *via* field
  - Whether the request was aborted
  - The number of packets resent in the response
  - The corresponding timestamps

Page Reconstruction Module

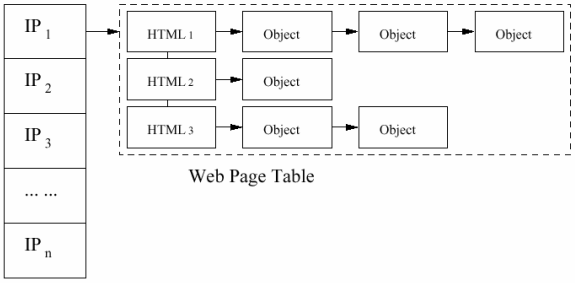
- To measure the client perceived end-to-end response time for retrieving a web page, we ***need to group the objects in a web page access***
- We use *two-pass heuristic method* and *statistical filtering mechanism* to reconstruct different client page access
  - *First pass:* EtE monitor uses the HTTP requests ***with referer*** field to build a ***Knowledge Base*** of web pages and their embedded objects
  - *Second pass*:
    - EtE monitor reconstructs the page accesses ***without referer*** field using the *Knowledge Base* of web pages and some additional heuristics
    - EtE monitor uses *statistical analysis to identify valid access patterns* and filter the accesses grouped incorrectly

Example

Example of initial *html.file* request and the following embedded object request with corresponding *referer* field:

```
request:
  Get /index.html HTTP/1.0
  Host: www.hpl.hp.com

response:
  HTTP/1.0 200 OK
  Content-Type: text/html

request:
  Get /img1.jpg HTTP/1.0
  Host: www.hpl.hp.com
  Referer: http://www.hpl.hp.com/index.html

response:
  HTTP/1.0 200 OK
  Content-Type: image/jpeg
```

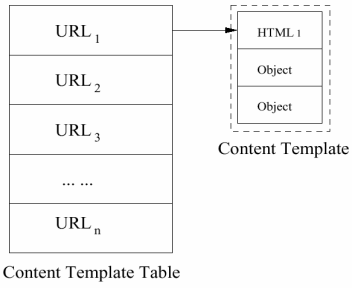
First Pass: Client Access Table



EtE monitor stores web page access information into a hash table **using client IP addresses**:

- If the content type is *text/html*, a new web page entry is created in the *Web Page Table*
- For other types, the request URL is inserted according to its *referer* field

Building a Knowledge Base of Web Pages



- From the *Client Access Table*, EtE monitor determines the ***content template*** of any given web page as a combined set of all objects that appear in all access patterns for this page

Second Pass: Reconstruction of Web Page Accesses

- With the help of *Knowledge Base*, EtE monitor processes the entire *Transaction Log* again, and creates a new *Client Access Table*
- This time it processes the objects without *referer* field:
  - EtE monitor consults the *Knowledge Base* while checking all the page entries in the *Web Page Table* to find the page an object might be embedded in, and appends it at the end of that page
  - If none of the web page entries in the *Web Page Table* contains the object based on the *Knowledge Base* then
    - EtE monitor searches for the page accessed with the same flow ID
    - Otherwise it appends the object to the latest accessed page (additionally it uses *configurable think time threshold* to delimit web pages)
    - If the *think time threshold* is exceeded, the object is dropped

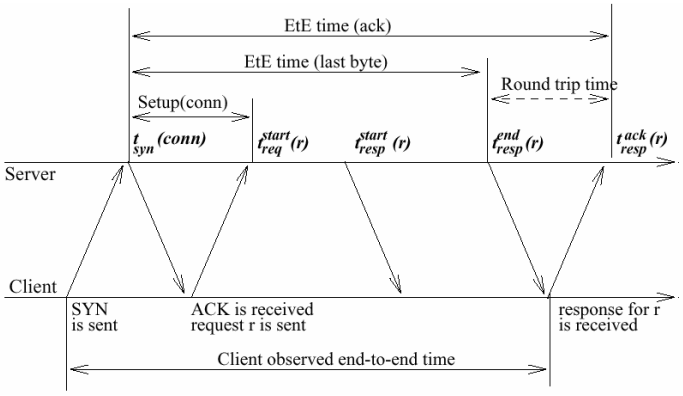
Identifying Valid Accesses Using Statistical Analysis of Access Patterns

- Although the above two-pass process is very efficient, there could still be some accesses grouped incorrectly
- We use a statistical analysis to better approximate the actual content of web pages and filter out the incorrectly constructed accesses

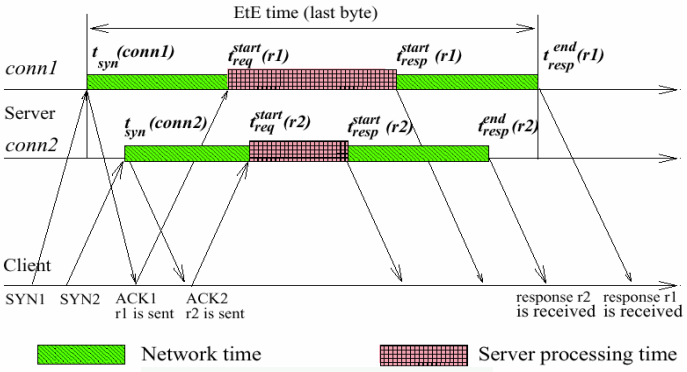
Metrics to Measure Web Service Performance

- Response time metrics
  - End-to-end response time observed by the client for a web page download
  - Latency breakdown: server related and network related portions
  - Connection set-up time
- Metrics evaluating web service caching efficiency
  - Server file hit ratio
  - Server byte hit ratio
- Aborted pages and QoS
  - Why the accesses are aborted:
    - Bad performance?
    - Client browsing patterns?

Example: 1-object page retrieval (basic timestamps)



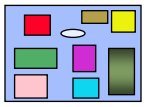
Latency Breakdown for Multiple Concurrent Connections: Server Processing vs Network



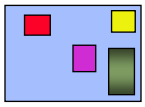
$$CumNetwork(P) = \sum_{j=1}^{N} Network(conn_j),$$

$$CumServer(P) = \sum_{j=1}^{N} Server(conn_j).$$

Metrics Evaluating Web Service Caching Efficiency

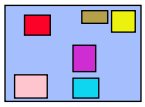


Original web page *url1* (page template):
- 10 objects,
- 100 Kbytes.



Access to *url1: Acc1*
- 5 objects,
- 70 Kbytes.

FileHitRatio(Acc1) = 5/10, 50%
ByteHitRatio(Acc1)=70/100, 70%



Access to *url1: Acc2*
- 7 objects,
- 80 Kbytes.

FileHitRatio(Acc1) = 7/10, 70%
ByteHitRatio(Acc1)=80/100, 80%

ServerFileHitRatio(url1) = *(5/10 + 7/10) / 2*, 60%
ServerByteHitRatio(url1) = *(70/100 + 80/100) / 2*, 75%

The smaller *is the better!*

Aborted Pages and QoS

- *User perceived QoS* is another important metric: aborted connections might be indicative of the poor site performance
- Need to consider only the subset of aborted page accesses with high end-to-end response time and what is the cause of performance problems? Are they *network* or *server* related?
- This will filter out aborted page accesses due to client browsing habits

Case Studies

- **HPL external site (HPL)**
  - From July12, 2001 to August 11, 2001
  - The site has mostly static content
- **Open View Support site (OV-Support)**
  - From October 11, 2001 to October 25, 2001
  - The site uses JavaServer Pages technology for dynamic page generation
- **HP External Support Site (IT-Support)**
  - From March 25, 2002 to April 8, 2002
  - The web pages are both dynamic and customized

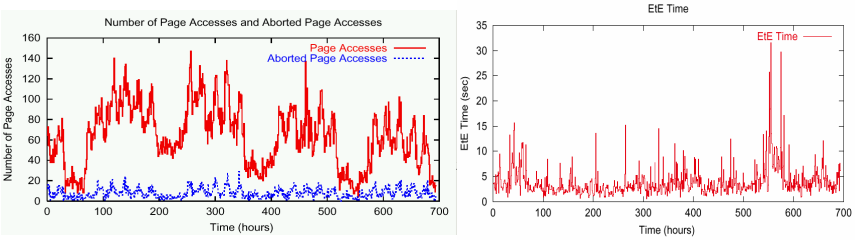
Sites Statistics At-A-Glance (HPL and OV-Support)

Metrics	HPL *url1*	HPL *url2*	Support *url1*	Support *url2*
EtE time	3.5 sec	3.9 sec	2.6 sec	3.3 sec
% of accesses above 6 sec	8.2%	8.3%	1.8%	2.2%
% of aborted accesses	12.3%	26%	4.8%	8.5%
% of aborted accesses above 6 sec	1.3%	2.8%	0.1%	0.2%
% of accesses from clients-proxies	16.8%	19.8%	11.2%	11.7%
EtE time from clients-proxies	4.2 sec	3 sec	4.5 sec	3 sec
Network-vs-Server ratio in EtE time	99.6%	99.7%	96.3%	93.5%
Page size	99 KB	60.9 KB	127 KB	100 KB
Server file hit ratio	38.5%	58%	22.9%	28.6%
Server byte hit ratio	44.5%	63.2%	52.8%	44.6%
Number of objects	4	2	32	32
Number of connections	1.6	1	6.5	9.1

Sites Statistics At-A-Glance (US-Support)

Metrics	*url1*	*url2*	*url3*	*url4*
EtE time	3.2 sec	3.5 sec	4 sec	2.6 sec
% of accesses above 6 sec	7%	10.7%	14.3%	3.9%
% of aborted accesses above 6 sec	0.8%	1.9%	1.5%	1.1%
% of accesses from clients-proxies	5.3%	21.9%	6.8%	4.1%
EtE time from clients-proxies	6.4 sec	5.6 sec	7 sec	6.1 sec
Network-vs-Server ratio in EtE time	86.1%	69.8%	47.6%	46.4%
Page access size	85.5 KB	62 KB	48.5 KB	58.3 KB
Number of requests	10.3	2	3.5	1.8
Number of connections	8.4	1.7	3.4	1.3

HPLabs Site Case Study

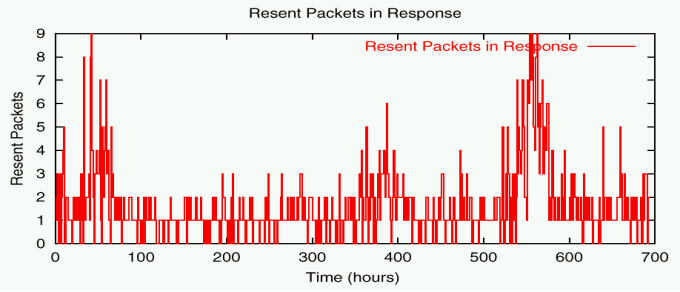


HPL site during a month (accesses to *index.html page*)

- Figure shows the EtE time to *index.html* on hourly scale during a month
- In spite of overall good performance, hourly averages reflect *significant variation* in response time observed by the clients

- Periods of increased latency correspond to weekends!
  What is the problem?

Understanding the Client Population

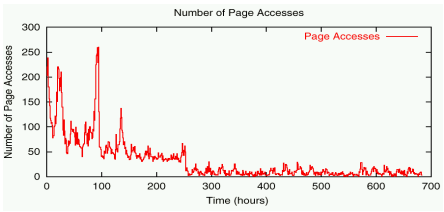


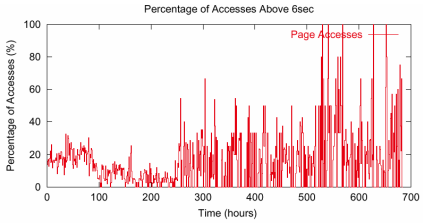
- *Resent packets* typically reflect network congestion or network–related bottlenecks
- Periods of increased resent packets correspond to weekends

- The explanation: *the client population significantly "changes" during weekends*
- Most of the clients access the web site from home via low-bandwidth connections

It is extremely important to understand the client population!
Active probing approach using artificial clients (with typically "good" connection to the Internet) lacks this information

Performance Analysis of Accesses to *itanium.html*





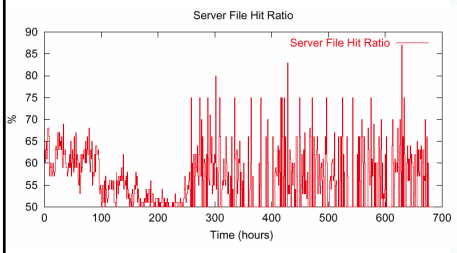
First Figure:

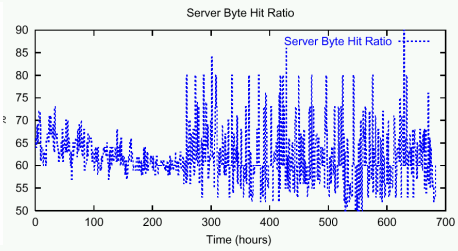
- Number of accesses to *itanium.html* page
- From being the most popular page in the beginning of the study, it gets to the 7th place after 10 days

Second Figure

- Percentage of accesses above 6 sec to *itanium.html* page
- *Question:* why is the latency observed by the clients getting higher?

Caching Efficiency of the Page



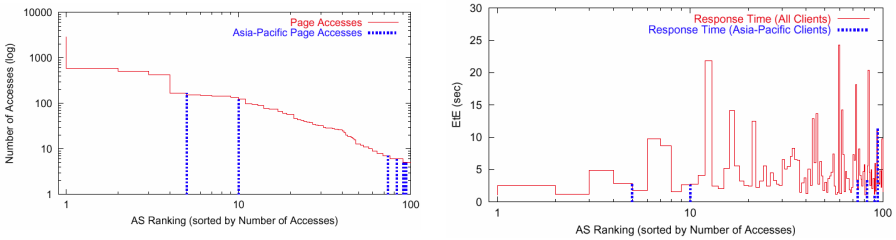


When the page is getting less popular, "colder", the number of objects and bytes retrieved from the original server increases significantly: i.e. *fewer network caches store the page related objects*

It translates into increased response time observed by the client

Active probing technique *cannot* reflect the caching efficiency of the site
The tools based on instrumentation technique *cannot* provide insight into this problem either

Clients Clustering by ASes (IT-Support Site)



- Clients grouped by ASes show a heavy tail distribution
- These figures allow us to see large client clusters and their corresponding end-to-end response time
- The ability of EtE monitor to measure performance metrics for *a certain group of clients* is particularly attractive for Service Providers *to validate required SLAs*

Validation Experiments

- We performed two groups of experiments
  - To validate the accuracy of EtE measurements
  - To evaluate the page access reconstruction power of EtE
    - How dependent are the reconstruction results on the existence of *referer* field information?
- The results are encouraging:
  - EtE provides a very close approximation of the response time
  - EtE monitor does a good job of page reconstruction even when the requests do not have any *referer* field!
  - However, *two-pass heuristic method* and *statistical filtering mechanism* we use to reconstruct page accesses increase the number of reconstructed pages by about 20-30%

Limitations

- EtE monitor is not appropriate for sites that *encrypt* much of their data (e.g., via SSL)
- EtE monitor is not appropriate for sites that *"outsource" most of their content* to CDNs
- Similar limitation applies to pages with *"mixed" content*: if a portion of the page is served from some other remote sites. In this case, EtE will measure only response time for local site content
- For *clients coming behind the proxy*, EtE monitor measures the response time as observed from the proxy
- Since the tool is based on heuristics and statistics to reconstruct the page content, the best results are obtained when *the sample size is large enough*
- *Dynamically generated content* creates additional challenges for EtE monitor (typical for other analysis tools too): a configuration file provided by a site administrator is needed

Conclusion and Future Work

- *Understanding performance characteristics* of Internet services is *critical* to evolving and engineering the web services to match:
  - Changing demand levels
  - Client populations
  - Global network characteristics
- EtE monitor, based on a novel technique, offers a number of benefits *unavailable from other tools and by other means.*
- EtE monitor can be extended to work in *"almost real-time"* to provide timely information about web services and their performance.
- Extended analysis on client clustering will provide an opportunity to use the information from EtE monitor for ***intelligent decision making*** on service placement and service optimization

Acknowledgements

- The tool and the study would not be possible without a generous help of our HP colleagues:
  - HPLabs team:
    - Mike Rodriquez, Annabelle Eseo, and Peter Haddad
  - HPO, Managed Web Services:
    - Guy Mathews
  - OpenView team:
    - Steve Yonkaitis, Bob Husted, Norm Follett, and Don Reab
  - US support team
    - Claude Villermain, Vincent Rabiller, Pierre-Emmanuel Delforge

Their help is highly appreciated !

Web Site Performance and QoS Monitoring: Measuring Client Response Time and More... at the Web Server

Ludmila Cherkasova[†], Yun Fu[†‡], Wenting Tang[†], and Amin Vahdat[‡]

[†]Hewlett-Packard Laboratories
1501 Page Mill Road, Palo Alto, CA 94303, USA
email: {lucy_cherkasova, yun_fu, wenting_tang}@hp.com

[‡]Dept of Computer Science, Duke University
Durham, NC 27708, USA
email: {fu,vahdat}@cs.duke.edu

Abstract. *In this work, we describe a novel technique for measuring a web site's end-to-end response time based on reconstruction of web page accesses from passively captured network packets at the web server side. This technique can effectively determine the set of objects composing a web page without parsing or interpreting the HTML syntax. Exploiting this technique, we built a tool, called EtE monitor. EtE monitor does not require any changes or modifications to site content or server side infrastructure, or client browsers, and can be used for sites with static or dynamically generated content.*

Relative to existing approaches, EtE monitor offers a set of new benefits. Timestamps extracted from network packet level provide invaluable information about connection setup time, server side processing time, and network related transfer time, which cannot be directly obtained from other sources. Additionally, this technique allows us to analyze many other useful and practical metrics, such as number of aborted page accesses, number of objects retrieved directly from the server versus from network and client browser caches, number of client connections used to retrieve the web page, etc. Our initial implementation and performance analysis across three different commercial web sites confirm the utility of our approach.

1 Introduction

The rapid growth and business-critical use of the internet have made performance measurement an essential service for web sites. Understanding and measuring end-to-end service performance is a challenging task. In general, a web page is composed of an HTML file and several embedded objects such as images. A browser retrieves a web page by issuing a series of HTTP requests for all objects. The requests can be sent through one persistent TCP connection or multiple concurrent connections. However, HTTP does not provide any means to delimit the beginning or the end of a web page in order to effectively measure the overall response time for corresponding web page retrieval.

Two main factors impact the response time perceived by the clients: the network latency and the server side processing time.

Many web sites are using complex multi-tiered architecture where the user requests are received by a front-tier web server playing a role of a web interface. The web server in turn accesses an application server that itself may issue the requests to a back-end database using middleware technologies such as

CORBA, RMI, etc. A set of new technologies, such as servlets [10] and Javaserver Pages [11], are the popular choice for extending and enhancing web servers. Servlets and Javaserver Pages technology allow web developers and designers to rapidly develop and easily maintain, information-rich, dynamic web pages that leverage existing business systems.

However, these new technologies for web page generation and more complex web site architecture could lead to increased server-side processing time and, as a result, require more careful performance assessment of overall site design and understanding its performance implication to the end-user observed latency.

The user satisfaction with web site response quality influences how long the user stays at the web site, and determines user's future visits to the site. Web site response time observed by the actual users becomes a critical metric to measure and improve. The importance of end-to-end response time for web page download led to a new performance metric introduced by businesses for measuring a web service efficiency. A web site or web service is considered to be "unavailable" if its response time exceeds 6 sec [12].

Currently, there are two most popular techniques used to assess the web site's end-to-end response time: 1) active probing technique [13, 17, 23, 19] based on periodic polling of web services using a geographically distributed set of artificial clients, and 2) a special web page instrumentation with an additional code (typically written in JavaScripts or using Java applets) which gets downloaded in a client browser when a web page is accessed [7, 9, 2, 20]. In Section 2, we discuss the merits and drawbacks of these techniques in more detail.

In this work, we outline a novel technique [6] to measure web site's end-to-end response time based on reconstruction of web page accesses from passively captured network packets on web server side. This technique can effectively determine the set of objects composing web page without parsing or interpreting the HTML syntax. Then by using this information and a few other heuristics, we reconstruct web page accesses. Exploiting this technique, we built a tool, called **EtE monitor**. We tested this tool to assess performance of three different web sites at Hewlett-Packard Corporation.

Relative to existing techniques, EtE monitor offers a number of benefits:

- Our system can determine the breakdown between the server and network overhead associated with retrieving a web page. This information is necessary to understand where performance optimizations should be directed, for instance to improve server-side performance or to leverage existing content distribution networks (CDNs) to improve network locality.

- EtE monitor tracks all accesses to web pages for a given service. Many existing techniques are typically restricted to a few probes per hour to URLs that are pre-determined to be popular. Our approach is much more agile to changing client access patterns. What real clients are accessing determines the performance that EtE monitor evaluates. Finally, given the Zipf popularity of service web pages [1], our approach is able to track the characteristics of the heavy tail that often makes up a large overall portion of web site accesses.

- Given information on all client accesses, clustering techniques [15] can be utilized to determine network performance characteristics by network region or autonomous system. System administrators can use this information to determine which content distribution networks to partner with (depending on their points of presence) or to determine multi-homing strategies with particular ISPs.

- EtE monitor captures information on page requests that are manually aborted by the client, either because of unsatisfactory web site performance or specific client browsing patterns (e.g., clicking on a link before a page has completed the download process). Existing techniques

cannot model user interactions in the case of active probing or miss important aspects of web site performance such as TCP connection establishment in the case of web page instrumentation.

- Finally, EtE monitor is able to determine the actual benefits of both browser and network caches. By learning the likely composition of individual web pages, our system can determine when certain embedded objects of a web page are not requested and conclude that those objects were retrieved from some cache in the network.

The rest of this paper is organized as follows. In the next section, we survey existing techniques and products and discuss their merits and drawbacks. Section 3 outlines the EtE monitor architecture, with additional details in Sections 4-6. In Section 7, we present the results of three performance studies, which have been performed to test and validate EtE monitor and its approach. The studied web sites include static web pages, dynamic web pages and customized web pages. We discuss the limitations of the proposed technique in Section 8 and present our conclusions and future work in Section 9.

2 Related Work

A number of companies use active probing techniques to offer measurement and testing services today, including Keynote [13], NetMechanic [17], Software Research [23], and Porivo Technologies [19]. Their solutions are based on periodic polling of web services using a set of geographically distributed, synthetic clients. In general, only a few pages or operations can typically be tested, potentially reflecting only a fraction of all user's experience. Further, active probing techniques cannot typically capture the potential benefits of browser and network caches, in some sense reflecting "worst case" performance. From another perspective, active probes come from a different set of machines than those that actually access the service. Thus, there may not always be correlation in the performance/reliability reported by the service and that experienced by end users. Finally, it is more difficult to determine the breakdown between network and server-side performance using active probing, making it more difficult for customers to determine where best to place their optimization efforts.

Another popular approach is to embed instrumentation code with web pages to record access times and report statistics back to the server. For instance, WTO (Web Transaction Observer) from HP OpenView suite [7] uses JavaScript to implement this functionality. With additional web server instrumentation and cookie techniques, this product can record the server processing time for a request, enabling a breakdown between server and network processing time. However in general, single web pages with non-HTML Content-Type fields, such as *application/postscript*, *application/x-tar*, *application/pdf*, or *application/zip*, can not be instrumented. Further, this approach requires additional server-side instrumentation and dedicated resources to actively collect performance reports from clients. A number of other products and proposals [9, 2, 20] employ similar techniques.

Similar to our approach, web page instrumentation can also capture end-to-end performance information from real clients. But since the JavaScript code is downloaded to a client web browser with the instrumented HTML file, and is executed after the page is downloaded, typically only the response time for retrieving the subsequent embedded images can be measured: it does not capture the connection establishment time and the main HTML file download time (which can be a significant portion of overall response time).

To avoid the above drawbacks, some recent work [20] proposes to instrument the hyperlinks for measuring the response times of the web pages that the links point to. This technique exploits similar ideas of downloading a small amount of code written in JavaScript to a client browser when

a web page is accessed via a hyperlink. However, under this approach, the response times for pages like *index.html* (i.e. the web pages which are accessed directly, not via links to them) cannot be measured.

There have been some earlier attempts to passively estimate the response time observed by clients from network level information. SPAND [21, 22] determines network characteristics by making shared, passive measurements from a collection of hosts and uses this information for server selection, i.e. for routing client requests to the server with the best observed response time in a geographically distributed web server cluster.

The NetQos, Inc. [18] provides a tool for application performance monitoring, which exploits similar ideas proposed in this paper: it collects the network packet traces from server sites and reconstructs the request-response pairs (the client requests and the corresponding server responses) and estimates the response time for those pairs.

However, the client-perceived web server responses are the retrievals of web pages (a web page is composed of an HTML file and several embedded objects such as images, and not just a single request-response pair). Thus, there is an orthogonal problem of grouping individual request-response pairs into the corresponding web page accesses. EtE monitor provides this additional step of client page access reconstruction to assess the true end-to-end time observed by the client when downloading a web page.

3 EtE Monitor Design

EtE monitor consists of four program modules shown in Figure 1:

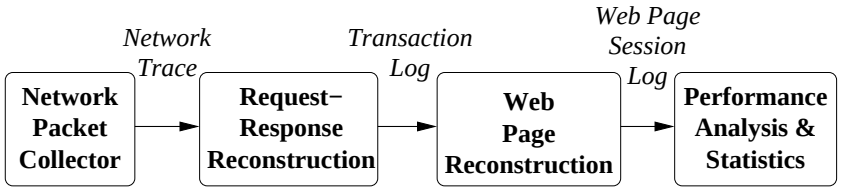


Figure 1: EtE Monitor Architecture.

1. The *Network Packet Collector* module.

   It collects network packets using *tcpdump*[24] and records them to a *Network Trace*, enabling offline analysis.

2. The *Request-Response Reconstruction* module.

   In the *Request-Response Reconstruction* module, EtE monitor reconstructs all TCP connections from the *Network Trace* and extracts HTTP transactions (a request with the corresponding response) from the payload. EtE monitor does not consider encrypted connections whose content cannot be analyzed. After obtaining the HTTP transactions, the monitor stores some HTTP header lines and other related information in the *Transaction log* for future processing (excluding the HTTP payload). To rebuild HTTP transactions from TCP-level traces, we use a methodology proposed by Feldmann [5] and described in more detail and extended to work with persistent HTTP connections by Krishnamurthy and Rexford [14].

   For efficiency, this module is written in C.

3. The *Web Page Reconstruction* module.

   A web page is generally composed of one HTML file and some embedded objects such as images or JavaScripts. When a client requests a particular web page, the client's browser should retrieve all the page embedded images from a web server to display the requested page. The client browser retrieves these embedded images separately. Each object is retrieved by an individual HTTP request. Entries of the *Transaction Log* contain the information about these individual HTTP requests. Our next step is to relate different individual HTTP requests in the web sessions corresponding to a particular web page accesses. As the outcome of this step, we build *Web Page Session Log*.

   For flexibility, this module is written in Perl.

4. *Performance Analysis and Statistics Presentation* module.

   After different request-response pairs are grouped into web page retrieval sessions, we can measure the client perceived end-to-end response time for a web page download, and many other useful metrics to reflect the service efficiency.

EtE monitor can be deployed in several different ways:

1. EtE monitor configured as an independent *network appliance*.

   EtE monitor should be placed at a point in the network where it can capture all HTTP transactions for a web server, e.g., the same subnet of the web server. It should be a point where a web server traffic in both directions can be captured: the request traffic to web server and the response traffic from the server.

   If a web site consists of multiple web servers, EtE monitor should be placed at the common entrance and exit of all the web servers as shown in Figure 2.

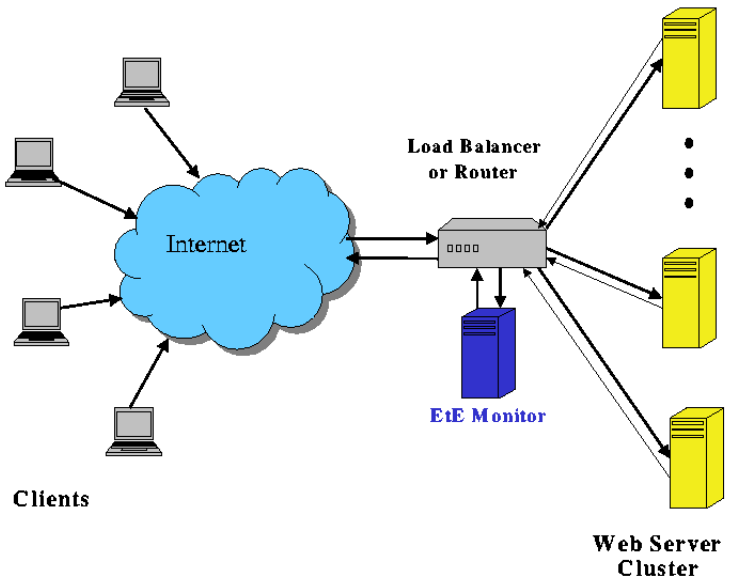


Figure 2: EtE monitor deployed as an independent *network appliance*.

If a web site is supported by geographically distributed web servers, there could be no such a common point. However, most typically, web servers in a web server farm (or cluster) are using "sticky connections", i.e., once the client has established a TCP connection with a particular web server, the consequent client's requests are sent to the same server. In this case, EtE monitor configured as a network appliance can still be used to capture a flow of transactions (to and from) of a particular web server, representing a part of all web transactions for the web site, and the measured data can be considered as sampling.

2. EtE monitor configured as a *software solution* deployed on a web server.

   EtE monitor can be installed as a software solution at a web server, and used for web transactions monitoring at this particular server.

   If a web site consists of multiple web servers, then as in the previous case, the EtE monitor does work when each web server is using "sticky connections" as shown in Figure 3. In this case, the EtE monitor can be installed as a software solution at all the servers or at a randomly selected web server in the site configuration, and the measured data should be aggregated to present overall traffic or can be considered as sampling.

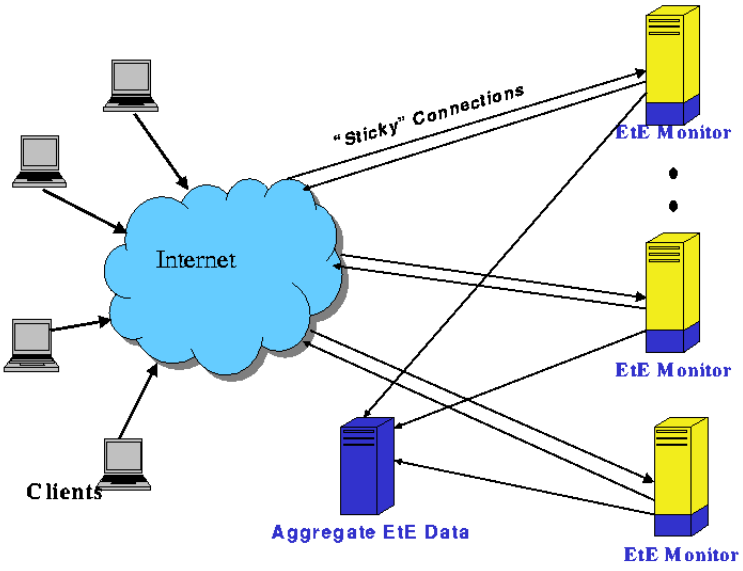


Figure 3: EtE monitor deployed as mixed or sw solution.

3. EtE monitor configured as *mixed software solution* with some modules deployed on a web server and some modules deployed on independent nodes outside of the web server. The architecture is similar to the one shown in Figure 3.

   To minimize the performance impact of additional computations on a web server, only two modules of EtE monitor are deployed at a web server (or web servers): the Network Packets Collector module and the Request-Response Reconstruction module. The resulting *Transaction Log* is two to three orders of magnitude smaller than the original *Network Trace*. It is transferred to a different, independent node with other two modules installed: Web Page Reconstruction and Performance Analysis and Statistics Presentation modules. These modules process the *Transaction Logs* received from web servers and produce the performance results.

4 TCP Connection and HTTP Request Reconstruction

As described above, the Request-Response Reconstruction module reconstructs all observed TCP connections. The TCP connections are rebuilt from the *Network Trace* using client IP addresses, client port numbers, and request (response) TCP sequence numbers. Within the payload of the rebuilt TCP connections, HTTP transactions can be delimited as defined by the HTTP protocol. Meanwhile, the timestamps, sequence numbers and acknowledged sequence numbers for HTTP requests can be recorded for later matching with the corresponding HTTP responses.

When a client clicks a hypertext link to retrieve a particular web page, the browser first establishes a TCP connection with the web server by sending a SYN packet. If the server is ready to process the request, it accepts the connection by sending back a second SYN packet acknowledging the client's SYN [1]. At this point, the client is ready to send HTTP requests to retrieve the HTML file and all embedded objects. For each request, we are concerned with the timestamps for the first byte and the last byte of the request since they delimit the request transfer time and the beginning of server processing. We are similarly concerned with the timestamps of the beginning and the end of the corresponding HTTP response. Besides, the timestamp of the acknowledgment packet for the last byte of the response explicitly indicates that the browser has received the entire response.

EtE monitor detects aborted connections by observing either

- a RST packet sent by an HTTP client to explicitly indicate an aborted connection or
- a FIN/ACK packet sent by the client where the acknowledged sequence number is less than the observed maximum sequence number sent from the server.

After reconstructing the HTTP transactions (a request and the corresponding response), the monitor records the HTTP header lines of each request in the *Transaction Log* and discards the body of the corresponding response.

Each entry in the log includes a number of fields: (1) a unique flow ID for the TCP connection, (2) the client's IP address, (3) the requested URL, (4) the content type, (5) the *referer* field, (6) the *via* field, (7) whether the request was aborted, (8) the number of packets resent during the connection (potentially an indication of the presence of network congestion), (9) the size and timestamps of the request and response. Some fields in the entry are used to rebuild web pages, while other fields can be used to measure end-to-end performance.

5 Grouping HTTP Requests into Web Pages

To measure the client perceived end-to-end response time for retrieving a web page, one needs to identify the objects that are embedded in a particular web page and to measure the response time for the client requests retrieving these embedded objects from the web server. In other words, to measure the client perceived end-to-end response time, we must group the object requests into web page accesses. Although we can determine some embedded objects of a web page by parsing the HTML for the "container object", some embedded objects cannot be easily discovered through static parsing. For example, JavaScript is used in web pages to retrieve additional objects. Without executing the JavaScript, it may be difficult to discover the identity of such objects.

[1] Whenever EtE monitor detects a SYN packet, it considers the packet as a new connection unless there is an earlier SYN packet with the same source port number from the same IP address. A retransmitted SYN packet is not considered as a newly established connection. However, if a SYN packet is dropped, e.g. by intermediate routers, there is no way to detect the dropped SYN packet on the server side.

Automatically, determining the content of a page requires a technique to delimit individual page accesses. One recent study [4] uses an estimate of client think time as the delimiter between two pages. While this method is simple and useful, it may be inaccurate in some important cases. For example, consider the case where a client opens two web pages from one server at the same time. Here, the requests for the two different web pages interleave each other without any think time between them. Another case is when the interval between the requests for objects within one page may be too long to be distinguishable from think time (perhaps because of the network conditions).

Different from previous work, our methodology uses heuristics to determine the objects composing a web page, i.e. the *content* of the web page, and applies statistics to adjust the results. EtE uses the HTTP *referer* field as a major "clue" to group objects into a web page. The *referer* field specifies the URL from which the requested URL was obtained. Thus, all requests for the embedded objects in an HTML file are recommended to set the *referer* fields to the URL of the HTML file. However, since the *referer* fields are set by client browsers, not all browsers set the fields. To solve this, EtE monitor first builds a *Knowledge Base* from those requests with *referer* fields, and uses more aggressive heuristics to group the requests without *referer* fields based on the *Knowledge Base* information.

Subsection 5.1 outlines *Knowledge Base* construction of web page objects. Subsection 5.2 presents the algorithm and technique to group the requests in web page accesses using *Knowledge Base* information and a set of additional heuristics.

5.1 Building a Knowledge Base of Web Page Objects

The goal of this step is to reconstruct a special subset of web page accesses, which we use to build a *Knowledge Base* about web pages and the objects composing them. Before grouping HTTP transactions into web pages, EtE monitor first sorts all transactions from the *Transaction Log* using the timestamps for the beginning of the requests in increasing time order. Thus, the requests for the embedded objects of a web page must follow the request for the corresponding HTML file of the page. When grouping objects into web pages (here and in the next subsection), we consider only transactions with *successful* responses, i.e. with status code 200 in the responses.

The next step is to scan the sorted transaction log and group objects into web page accesses. Not all the transactions are useful for the *Knowledge Base* construction process. During this step, some of the *Transaction Log* entries are excluded from our current consideration:

- Content types that are known not to contain embedded objects are excluded from the knowledge base, e.g., *application/postscript*, *application/x-tar*, *application/pdf*, *application/zip* and *text/plain*. For the rest of the paper, we call them *independent, single page* objects.
- If the *referer* field of a transaction is not set and its content type is not *text/html*, EtE monitor excludes it from further consideration.

To group the rest of the transactions into web page accesses, we use the following fields from the entries in the *Transaction Log*: the request URL, the request *referer* field, the response *content type*, and the client IP address. EtE monitor stores the web page access information into a hash table, the *Client Access Table* depicted in Figure 4, which maps a client's IP address to a *Web Page Table* containing the web pages accessed by the client. Each entry in the *Web Page Table* is a web page access, and composed of the URLs of HTML files and the embedded objects. Notice that EtE monitor makes no distinction between statically and dynamically generated HTML files. We consider embedded HTML pages, e.g. framed web pages, as separate web pages.

When processing an entry of the *Transaction Log*, EtE monitor first locates the *Web Page Table* for

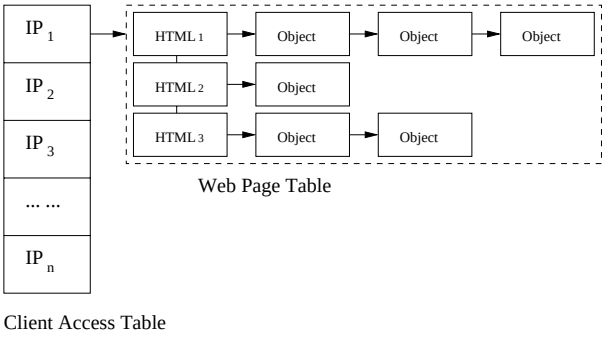


Figure 4: *Client Access Table.*

the client's IP in the *Client Access Table*. Then, EtE monitor handles the transaction according to its content type:

1. If the content type is *text/html*, EtE monitor treats it as the beginning of a web page and creates a new web page entry in the *Web Page Table*.

2. For other content types, EtE monitor attempts to insert the URL of the requested object into the web page that contains it according to its *referer* field. If the referred HTML file is already present in the *Web Page Table*, EtE monitor appends this object at the end of the entry. If the referred HTML file does not exist in the client's *Web Page Table*, it means that the client may have retrieved a cached copy of the object from somewhere else between the client and the web server. In this case, EtE monitor first creates a new web page entry in the *Web Page Table* for the referred HTML file. Then it appends the considered object to this page.

From the *Client Access Table*, EtE monitor determines the *content template* of any given web page as a combined set of all the objects that appear in all the access patterns for this web page. Thus, EtE monitor scans the *Client Access Table* and creates a new hash table, as shown in Figure 5, which is used as a *Knowledge Base* to group the accesses for the same web pages from other client's browsers that do not set the *referer* fields.

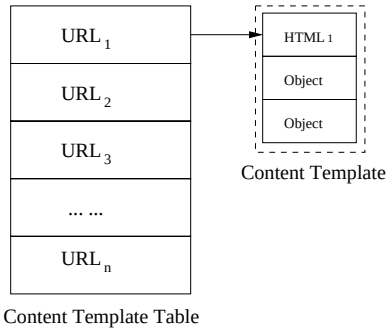


Figure 5: *Knowledge Base* of web pages: maps URLs to the corresponding accessed Content Templates.

Since in this pass, the *Client Access Table* is based on explicit reference relationship, the *Content Template Table* constructed from it is relatively trustable and can be used as a *Knowledge Base* to group the accesses for the same web pages from other client's browsers that do not set the *referer* fields.

5.2 Reconstruction of Web Page Accesses

With the help of the *Knowledge Base*, EtE monitor processes the entire *Transaction Log* again. This time, EtE monitor does not exclude the entries without *referer* fields. It significantly extends the number of correctly processed web page accesses. Using data structures similar to those introduced in Section 5.1, EtE monitor scans the sorted *Transaction Log* and creates a new *Client Access Table* to store all accesses as depicted in Figure 4. For each transaction, EtE monitor locates the *Web Page Table* for the client's IP in the *Client Access Table*. Then, EtE monitor handles the transaction depending on the content type:

1. If the content type is *text/html*, EtE monitor creates a new web page entry in the *Web Page Table*.
2. If a transaction is an independent, single page object, EtE monitor marks it as individual page without any embedded objects and allocates a new web page entry in the *Web Page Table*.
3. For other content types that can be embedded in a web page, EtE monitor attempts to insert it into the web page that contains it.

   - If the *referer* field is set for this transaction, EtE monitor attempts to locate the referred page in the following way. If the referred HTML file is in an existing page entry in the *Web Page Table*, EtE monitor appends the object at the end of the entry. If the referred HTML file does not exist in the client's *Web Page Table*, EtE monitor first creates a new web page entry in the table for the referred page and marks it as *nonexistent*. Then it appends the object to this page. If the *referer* field is not set for this transaction, EtE monitor uses the following policies. With the help of the *Knowledge Base*, EtE monitor checks each page entry in the *Web Page Table* from the latest to earliest. If the *Knowledge Base* contains the *content template* for the checked page and the considered object does not belong to it, EtE monitor skips the entry and checks the next one until a page containing the object is found. If such an entry is found, EtE monitor appends the object to the end of the web page.
   - If none of the web page entries in the *Web Page Table* contains the object based on the *Knowledge Base*, EtE monitor searches in the client's *Web Page Table* for a web page accessed via the same flow ID as this object. If there is such a web page, EtE monitor appends the object to the page.
   - Otherwise, if there are any accessed web pages in the table, EtE monitor appends the object to the latest accessed one.

   If none of the above policies can be applied, EtE monitor drops the request.

   Obviously, the above heuristics may introduce some mistakes. Thus, EtE monitor also adopts a *configurable think time threshold* to delimit web pages. If the time gap between the object and the tail of the web page that it tries to append to is larger than the threshold, EtE monitor skips the considered object. In this paper, we adopt a configurable think time threshold of 4 sec.

Although the above two-pass process can effectively provide accurate web page access reconstruction in most cases, there could still be some accesses grouped incorrectly. To filter out such accesses, we must better approximate the actual content of a web page.

All the accesses to a web page usually exhibit a set of different access patterns. For example, an access pattern can contain all the objects of a web page, while other patterns may contain a subset of

them (e.g., because some objects were retrieved from a browser or network caches). We assume the same access patterns of those incorrectly grouped accesses should rarely appear repeatedly. Thus, we use the statistical analysis on access patterns to determine the actual content of web pages and exclude the incorrectly grouped accesses. This technique is described in more detail in our earlier paper [6].

6 Web Service Performance Metrics

In this section, we introduce a set of metrics and the ways to compute them in order to measure a web service efficiency. These metrics can be categorized as:

- metrics approximating the end-to-end response time observed by the client for a web page download. Additionally, we provide a means to calculate the breakdown between server processing and networking portions of overall response time.
- metrics evaluating the caching efficiency for a given web page by computing the server file hit ratio and server byte hit ratio for the web page.
- metrics relating the end-to-end performance of aborted web pages to the QoS.

6.1 End-to-End Response Time Metrics

We use the following functions to denote the critical timestamps for connection *conn* and request r:

- $t_{syn}(conn)$: time when the first SYN packet from the client is received for establishing the connection *conn*;
- $t_{req}^{start}(r)$: time when the first byte of the request r is received ;
- $t_{req}^{end}(r)$: time when the last byte of the request r is received;
- $t_{resp}^{start}(r)$: time when the first byte of the response for r is sent;
- $t_{resp}^{end}(r)$: time when the last byte of the response for r is sent;
- $t_{resp}^{ack}(r)$: time when the ACK for the last byte of the response for r is received.

Additionally, for a web page P, we have the following variables:

- N - the number of distinct connections used to retrieve the objects in the web page P;
- $r_1^k, ...r_{n_k}^k$ - the requests for the objects retrieved through the connection $conn_k$ $(k = 1, ..., N)$, and ordered accordingly to the time when these requests were received, i.e.,

$$t_{req}^{end}(r_1^k) \leq t_{req}^{end}(r_2^k) \leq ... \leq t_{req}^{end}(r_{n_k}^k).$$

The extended version of HTTP 1.0 and later version HTTP 1.1 [8] introduce the concepts of *persistent connections* and *pipelining*. Persistent connections enable reuse of a single TCP connection for multiple object retrievals from the same IP address. Pipelining allows a client to make a series of requests on a persistent connection without waiting for the previous response to complete (the server must, however, return the responses in the same order as the requests are sent).

We consider the requests $r_i^k, ..., r_n^k$ to belong to the same *pipelining group* (denoted as $PipeGr = \{r_i^k, ..., r_n^k\}$) if for any j such that $i \leq j - 1 < j \leq n$, $t_{req}^{start}(r_j^k) \leq t_{resp}^{end}(r_{j-1}^k)$.

Thus for all the requests on the same connection $conn_k$: $r_1^k, ..., r_{n_k}^k$, we define the maximum pipelining groups in such a way that they do not intersect, e.g.,

$$\underbrace{r_1^k, ..., r_i^k}_{PipeGr_1}, \underbrace{r_{i+1}^k}_{PipeGr_2}, ..., \underbrace{r_{n_k}^k}_{PipeGr_l}.$$

For each of the pipelining groups, we define three portions of response time: total response time ($Total$), network-related portion ($Network$), and lower-bound estimate of the server processing time ($Server$).

Let us consider the following example. For convenience, let us denote $PipeGr_1 = \{r_1^k, ..., r_i^k\}$.

Then

$$Total(PipeGr_1) = t_{resp}^{end}(r_i^k) - t_{req}^{start}(r_1^k),$$

$$Network(PipeGr_1) = \sum_{j=1}^{i} (t_{resp}^{end}(r_j^k) - t_{resp}^{start}(r_j^k)),$$

$$Server(PipeGr_1) = Total(PipeGr_1) - Network(PipeGr_1).$$

If no pipelining exists, a pipelining group only consists of one request. In this case, the computed server time represents precisely the server processing time for a given request-response pair. If a connection adopts pipelining, the "real" server processing time might be larger than the computed server time because it can partially overlap the network transfer time, and it is difficult to estimate the exact server processing time from the packet-level information. However, we are still interested to estimate the "non-overlapping" server processing time as this is the portion of the server time on a critical path of overall end-to-end response time. Thus, we use as an estimate the lower-bound server processing time, which is explicitly exposed in the overall end-to-end response.

If connection $conn_k$ is a newly established connection to retrieve a web page, we observe additional connection setup time:

$$Setup(conn_k) = t_{req}^{start}(r_1^k) - t_{syn}(conn_k)\ ^2,$$

otherwise the setup time is 0. Additionally, we define $t^{start}(conn_k) = t_{syn}(conn_k)$ for a newly established connection, otherwise, $t^{start}(conn_k) = t_{req}^{start}(r_1^k)$.

Similarly, we define the breakdown for a given connection $conn_k$:

$$Total(conn_k) = Setup(conn_k) + t_{resp}^{end}(r_{n_k}^k) - t_{req}^{start}(r_1^k),$$

$$Network(conn_k) = Setup(conn_k) + \sum_{j=1}^{l} Network(PipeGr_j),$$

$$Server(conn_k) = \sum_{j=1}^{l} Server(PipeGr_j).$$

Now, we define similar latencies for a given page P:

$$Total(P) = \max_{j \le N} t_{resp}^{end}(r_{n_j}^j) - \min_{j \le N} t^{start}(conn_j),$$

For the rest of the paper, we will use the term *EtE time* interchangeably with $Total(P)$ time.

$$CumNetwork(P) = \sum_{j=1}^{N} Network(conn_j),$$

[2]The connection setup time as measured by EtE monitor does not include dropped SYNs, as discussed earlier in Section 4.

$$CumServer(P) = \sum_{j=1}^{N} Server(conn_j).$$

All the above formulae use $t_{resp}^{end}(r)$ to calculate response time. An alternative way is to use as the end of a transaction the time $t_{resp}^{ack}(r)$ when the ACK for the last byte of the response is received by a server. Figure 6 shows an example of a simplified scenario where a 1-object page is downloaded by the client: it shows the communication protocol for connection setup between the client and the server as well as the set of major timestamps collected by the EtE monitor on the server side. The connection setup time measured on the server side is the time between the client SYN packet and the first byte of the client request. This represents a close approximation for the original client setup time. If the ACK for the last byte of the client response is not delayed or lost, $t_{resp}^{ack}(r)$ is a more

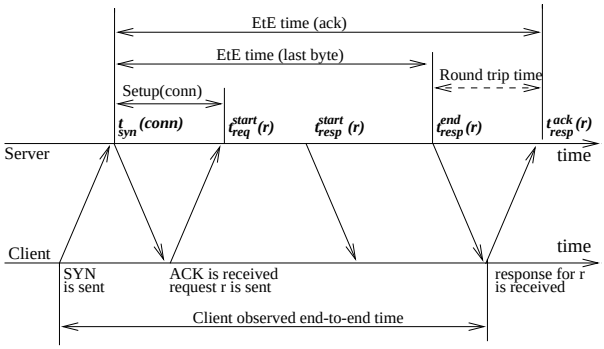


Figure 6: An example of a 1-object page download by the client: major timestamps collected by the EtE monitor on the server side.

accurate approximation of the end-to-end response time observed by the client: it "compensates" for the latency of the first client SYN packet that is not measured on the server side. The difference between the two methods, i.e. *EtE time (last byte)* and *EtE time (ack)*, is only a round trip time, which is on the scale of milliseconds. Since the overall response time is on the scale of seconds, we consider this deviation an acceptably close approximation. However, to avoid the problems with delayed or lost ACKs, EtE monitor determines the end of a transaction as the time when the last byte of a response is sent by a server.

The functions $CumNetwork(P)$ and $CumServer(P)$ give the sum of all the network-related and server processing portions of the response time over all connections used to retrieve the web page. However, the connections can be opened concurrently by the browser. To evaluate the concurrency impact, we introduce the page concurrency coefficient *ConcurrencyCoef(P)*:

$$ConcurrencyCoef(P) = \frac{\sum_{j=1}^{N} Total(conn_j)}{Total(P)}.$$

Using page concurrency coefficient, we finally compute the network-related and the service-related portions of response time for a particular page P:

$$Network(P) = CumNetwork(P)/ConcurrencyCoef(P),$$
$$Server(P) = CumServer(P)/ConcurrencyCoef(P).$$

EtE monitor can distinguish the requests sent to a web server from clients behind proxies by checking the HTTP *via* fields. If a client page access is handled via the same proxy (which is typically the case, especially when persistent connections are used), EtE monitor provides correct measurements for end-to-end response time and other metrics, as well as provides interesting statistics on the percentage of client requests coming from proxies. Clearly, this percentage is an approximation, since not all the proxies set the *via* fields in their requests. Finally, EtE monitor can only measure the response time to a proxy instead of the actual client behind it.

6.2 Measuring Web Service Caching Efficiency

Real clients of a web service may benefit from the presence of network and browser caches, which can significantly reduce their perceived response time. However, none of the existing performance measurement techniques provide any information on the impact of caches on web services: what percentage of the files and bytes are delivered from the server comparing with the total files and bytes required for delivering the web service. This impact can only be partially evaluated from web server logs by checking response status code 304, whose corresponding requests are sent by the network caches to validate whether the cached object has been modified. If the status code 304 is set, the cached object is not expired and need not be retrieved again.

To evaluate the caching efficiency of a web service, we introduce two metrics: *server file hit ratio* and *server byte hit ratio* for each web page.

For a web page P, assume the objects composing the page are $O_1, ..., O_n$. Let $Size(O_i)$ denote the size of object O_i in bytes. Then we define $NumFiles(P) = n$ and $Size(P) = \sum_{j=1}^{n} Size(O_j)$.

Additionally, for each access P^i_{access} of the page P, assume the objects retrieved in the access are $O^i_1, ..., O^i_{k_i}$, we define $NumFiles(P^i_{access}) = k_i$ and $Size(P^i_{access}) = \sum_{j=1}^{k_i} Size(O^i_j)$. First, we define *file hit ratio* and *byte hit ratio* for each page access in the following way:

$$FileHitRatio(P^i_{access}) = NumFiles(P^i_{access})/NumFiles(P),$$

$$ByteHitRatio(P^i_{access}) = Size(P^i_{access})/Size(P).$$

Let $P^1_{access}, ..., P^N_{access}$ be all the accesses to the page P during the observed time interval. Then

$$ServerFileHitRatio(P) = \frac{1}{N} \sum_{k \leq N} FileHitRatio(P^k_{access}),$$

$$ServerByteHitRatio(P) = \frac{1}{N} \sum_{k \leq N} ByteHitRatio(P^k_{access}).$$

The lower numbers for *server file hit ratio* and *server byte hit ratio* indicate the higher caching efficiency for the web service, i.e., more files and bytes are served from network and client browser caches.

Often, a corporate web site has a set of templates, buttons, logos, and shared images, which are actively reused among a set of different pages. A user, browsing through such a site, can clearly benefit from the browser cache. The proposed caching metrics are useful for evaluating the efficiency of caching and comparing different site designs.

6.3 Aborted Pages, QoS and Client Browsing Behavior

User-perceived QoS is another important metric to consider in EtE monitor. One way to measure the QoS of a web service is to measure the frequency of aborted connections. The logic behind this is that if a web site is not fast enough a user will get impatient and hit the stop button, thus aborting the connection. However, such simplistic interpretation of aborted connections and web server QoS has several drawbacks. First, a client can interrupt HTTP transactions by clicking the browser's "stop" or "reload" button while a web page is downloading, or clicking a displayed link before the page is completely downloaded. Thus, only a subset of aborted connections are relevant to poor web site QoS or poor networking conditions, while other aborted connections are caused by client-specific browsing patterns. On the other hand, a web page can be retrieved through multiple connections. A client's browser-level interruption can cause all the currently open connections to be aborted. Thus,

the number of aborted page accesses more accurately reflects client satisfaction than the number of aborted connections.

For aborted pages, we distinguish the subset of pages Π_{bad} with the response time higher than the given threshold X_{EtE} (in our case, $X_{EtE} = 6\ sec$). Only these pages might be reflective of the bad quality downloads. While a simple deterministic cut off point cannot truly capture a particular client's expectation for site performance, the current industrial *ad hoc* quality goal is to deliver pages within 6 sec [12]. We thus attribute aborted pages that have not crossed the 6 sec threshold to individual client browsing patterns. The next step is to distinguish the reasons leading to poor response time: whether it is due to network or server-related performance problems, or both.

7 Practical Case Studies

In this section, we present three case studies to illustrate the benefits of EtE monitor in assessing web site performance.

- The first site is the HP Labs external site (*HPL Site*), *http://www.hpl.hp.com*. Static web pages comprise most of this site's content. We deployed EtE monitor as an independent network appliance and measured performance of this site for a month, from July 12, 2001 to August 11, 2001.
- The second site is a support site for a popular HP product family, which we call *OV-Support Site*. It uses JavaServer Pages [11] technology for dynamic page generation. The architecture of this site is based on a geographically distributed web server cluster with Cisco Distributed Director [3] for load balancing, using "sticky connections" or "sticky sessions", i.e. once a client has established a TCP connection with a particular web server, the subsequent client's requests are sent to the same server. We used a mixed solution deployed on one of the servers in the cluster to measure the site performance for 2 weeks, from October 11, 2001 to October 25, 2001.
- The third site under study is a support site, which provides a variety of technical information and tools on software, hardware, and the network to help customers manage their multivendor computing environment. We call it *IT-Support* site. The architecture of this site is based on a web server cluster with Cisco Distributed Director [3] for load balancing, using "sticky connections". The pages returned to the clients are both dynamic and customized. We deployed EtE monitor as an independent network appliance and measured the site performance at one of the site web servers for 2 weeks, from March 25, 2002 to April 8, 2002.

Table 1 summarizes the three site's performance *at-a-glance* during the measured period using the two most frequently accessed pages at each site.
The average end-to-end response time of client accesses to these pages reflects good overall performance. However in the case of HPL and IT-Support sites, a sizeable percentage of accesses take more than 6 sec to complete (7%-10.7%), with a portion leading to aborted accesses (0.8%-2.8%). The OV-Support site had better overall response time with a much smaller percentage of accesses above 6 sec (1.8%-2.2%), and a correspondingly smaller percentage of accesses aborted due to high response time (0.1%-0.2%).

As discussed earlier, the HPL content is static, the OV-Support site content is generated using dynamic pages, while IT-Support content is both dynamic and customized. This increased complexity in generation of the corresponding pages is reflected through the increased fraction of the server processing time in the overall response time for the corresponding three sites. But in general, the

Metrics	HPL *url1*	HPL *url2*	OV-Support *url1*	OV-Support *url2*	IT-Support *url1*	IT-Support *url2*
EtE time	3.5 sec	3.9 sec	2.6 sec	3.3 sec	3.2 sec	3.5 sec
% of accesses above 6 sec	8.2%	8.3%	1.8%	2.2%	7%	10.7%
% of aborted accesses above 6 sec	1.3%	2.8%	0.1%	0.2%	0.8%	1.9%
% of accesses from clients-proxies	16.8%	19.8%	11.2%	11.7%	5.3%	21.9%
EtE time from clients-proxies	4.2 sec	3 sec	4.5 sec	3 sec	6.4 sec	5.6 sec
Network-vs-Server ratio in EtE time	99.6%	99.7%	96.3%	93.5%	86.1%	69.8%
Page size	99 KB	60.9 KB	127 KB	100 KB	NA	NA
Server file hit ratio	38.5%	58%	22.9%	28.6%	NA	NA
Server byte hit ratio	44.5%	63.2%	52.8%	44.6%	NA	NA
Number of objects	4	2	32	32	NA	NA
Number of connections	1.6	1	6.5	9.1	8.4	1.7

Table 1: *At-a-Glance* statistics for *www.hpl.hp.com* and *support* site during the measured period.

network transfer time dominates the performance for all considered URLs in the table, ranging from 69.8% for the IT-Support site to 99.7% for the HPL site.

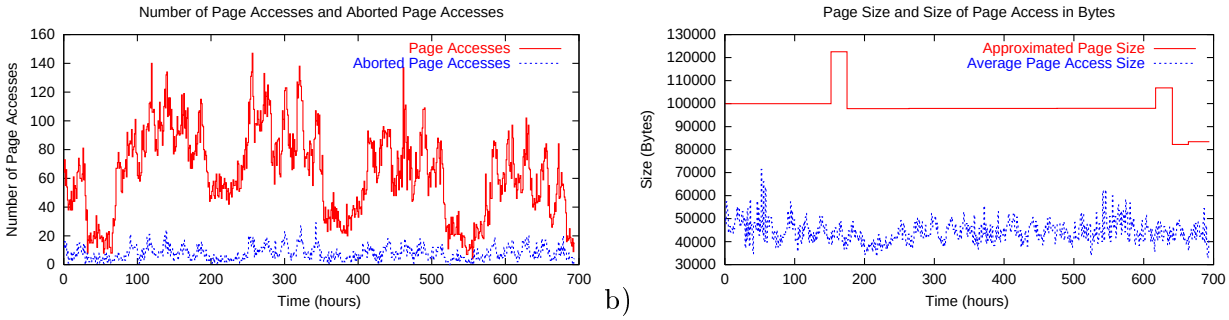
The pages from the HPL and OV-Support sites are comparable in size. However, the two pages from the HPL site have a small number of objects per page (4 and 2 correspondingly), while the OV-Support site pages are composed of 32 different objects. Page composition influences the number of client connections required to retrieve the page content. Additionally, statistics show that network and browser caches help to deliver a significant amount of page objects: in the case of the OV-Support site, only 22.9%-28.6% of the 32 objects are retrieved from the server, accounting for 44.6%-52.8% of the bytes in the requested pages.

The web pages published on the IT-Support site are both dynamic and customized. The pages returned to the clients are dynamically generated based on a set of preferences and parameters customized to the end clients. For example, a client may select among 10 different language options for the site content. The page accessed via the same URL but with different language options might have a set of differently specialized, embedded objects and images. So, each access to a logically identical URL is defined by a different URL expression. The service providers of this site provided us with a set of policies (regular expressions) on how to generate customized URLs, which are used to aggregate client's accesses to these URLs and measure the performance. In this case, the "size" of a web page and the "corresponding set of embedded objects" are not uniquely identified. EtE monitor identifies a **combined set** of embedded objects during the construction of the knowledge base of web pages, and uses this information to correctly reconstruct page accesses. As a result, some metrics measured by EtE monitor become meaningless such as: the average page size, the number of embedded objects, the file and byte hit ratios, etc. However, service providers can use their knowledge about specific web pages of interest together with EtE monitor reported information on the average size of client page accesses and the average number of requests for objects in a page in order to approximate the caching efficientcy of the site.

Given the above summary, we now present more detailed information from our site measurements. For the HPL site, the two most popular pages during the observed period were *index.html* and a page in the news section describing the Itanium chip (we call it *itanium.html*).

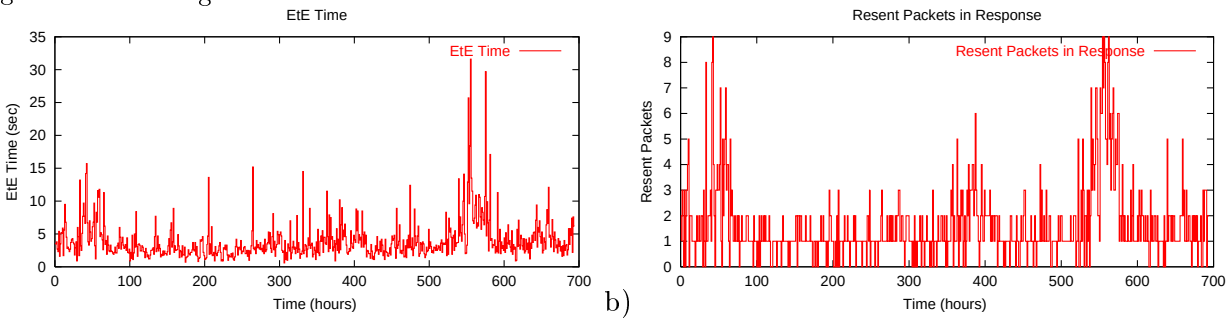
Figure 7 a) shows the number of page accesses to *index.html*, as well as the number of aborted page accesses during the measured period. The graph clearly reflects weekly access patterns to the site.

Figure 7 b) reflects the *approximate page size*, as reconstructed by EtE monitor. We use this data to additionally validate the page reconstruction process. While debugging the tool, we manually compare the content of the 20 most frequently accessed pages reconstructed by EtE monitor against



a) b)

Figure 7: HPL site during a month: a) Number of all and aborted accesses to *index.html* ; b) Approximated page size and average access size to *index.html*.



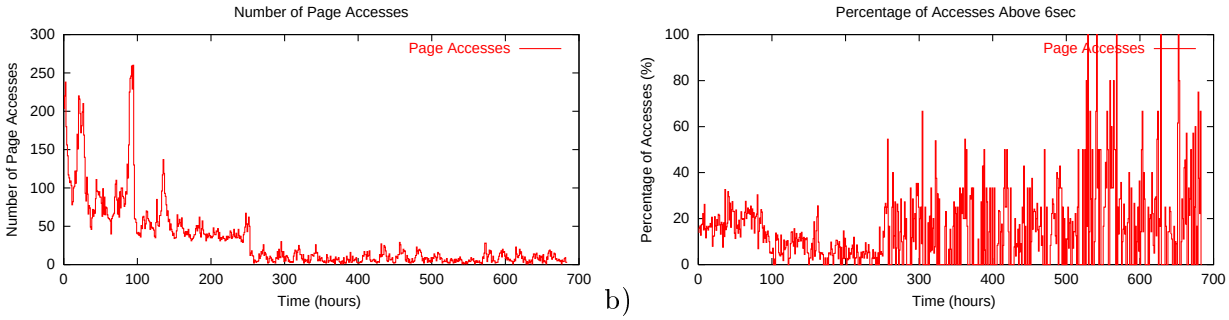
a) b)

Figure 8: HPL site during a month: a) end-to-end response times for accesses to *index.html*; b) number of resent packets in response.

the actual web pages: the EtE monitor page reconstruction accuracy for popular pages is very high, practically 100%. Figure 7 b) allows us to "see" the results of this reconstruction process over the period of the study. In the beginning, it is a straight line exactly coinciding with the actual page size. At hour mark 153, it jumps and returns to a next straight line interval at the 175 hour mark. As we verified, the page has been partially modified during this time interval. The EtE monitor "picked" both the old and the modified page images, since they both occurred during the same day interval and represented a significant fraction of accesses. However, the next day, the *Knowledge Base* was "renewed" and had only the modified page information. The second "jump" of this line corresponds to the next modification of the page. The gap can be tightened, depending on the time interval EtE monitor is set to process. The other line in Figure 7 b) shows the average page access size, reflecting the server byte hit ratio of approximately 44%.

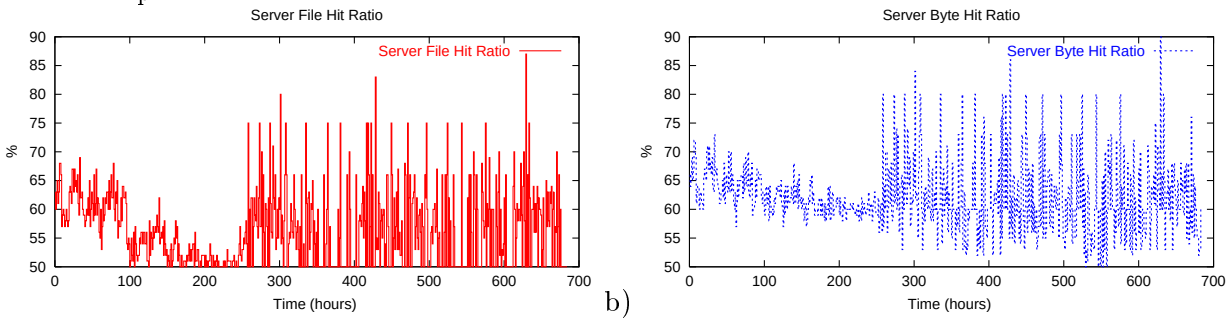
Figure 8 a) shows the end-to-end response time for accesses to *index.html* on an hourly scale during a month. In spite of good average response time reported in at-a-glance table, hourly averages reflect significant variation in response times. This graph helps to stress the advantages of EtE monitor and reflects the shortcomings of active probing techniques that measure page performance only a few times per hour: the collected test numbers could vary significantly from a site's instantaneous performance characteristics.

Figure 8 b) shows the number of resent packets in the response stream to clients. There are three pronounced "humps" with an increased number of resent packets. Typically, resent packets reflect network congestion or the existence of some network-related bottlenecks. Interestingly enough, such periods correspond to weekends when the overall traffic is one order of magnitude lower than weekdays (as reflected in Figure 7 a)). The explanation for this phenomenon is that during weekends the client population of the site "changes" significantly: most of the clients access the site from home using modems or other low-bandwidth connections. This leads to a higher observed end-to-end response time and an increase in the number of resent packets (i.e., TCP is likely to cause drops more often



a) b)

Figure 9: HPL site during a month: a) number of all accesses to *itanium.html*; b) percentage of accesses with end-to-end response time above 6 sec.



a) b)

Figure 10: HPL site: a) server file hit ratio for *itanium.html*; b) server byte hit ratio for *itanium.html*.

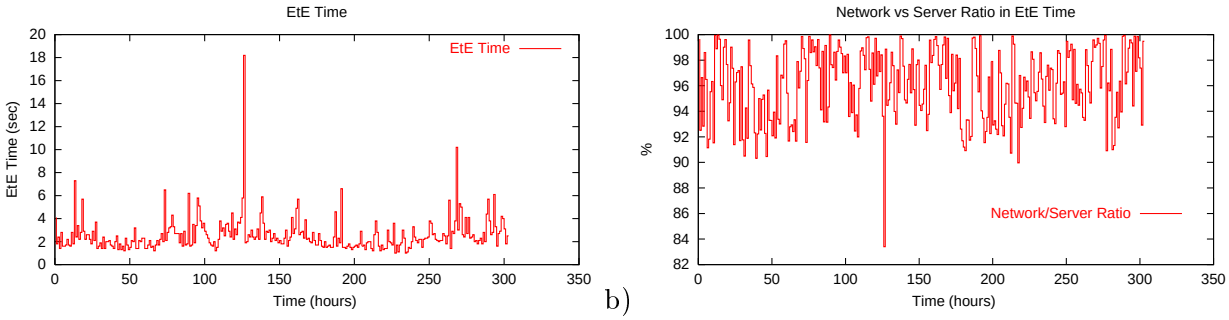
when probing for the appropriate congestion window over a low-bandwidth link). These results again stress the unique capabilities of EtE monitor to extract appropriate information from network packets, and reflect another shortcoming of active probing techniques that use a fixed number of artificial clients with rather good network connections to the Internet. For site designers, it is important to understand the actual client population and their end-to-end response time and the "quality" of the response. For instance, when large population of clients have limited bandwidth parameters, the site designers should consider making the pages and their objects "lighter weight".

Figure 9 a) shows the number of page accesses to *itanium.html*. When we started our measurement of the HPL site, the *itanium.html* page was the most popular page, "beating" the popularity of the main *index.html* page. However, ten days later, this news article started to get "colder", and the page got to the seventh place by popularity.

Figure 9 b) shows the percentage of accesses with end-to-end response time above 6 sec. The percentage of high response time jumps significantly when the page becomes "colder". The reason behind this phenomenon is shown in Figure 10, which plots the server file hit and byte hit ratio. When the page became less popular, the number of objects and the corresponding bytes retrieved from the server increased significantly. This reflects that fewer network caches store the objects as the page becomes less popular, forcing clients to retrieve them from the origin server.

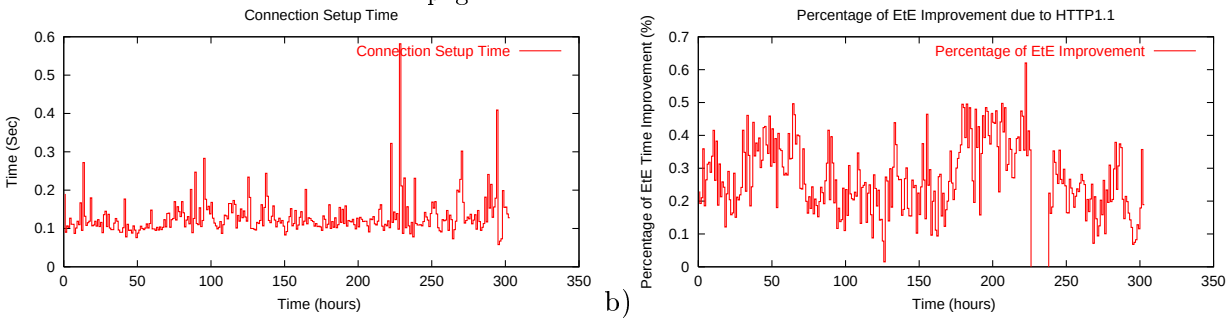
Figure 9 b) and Figure 10 explicitly demonstrate the network caching impact on end-to-end response time. When the caching efficiency of a page is higher (i.e., more page objects are cached by network and browser caches), the response time measured by EtE monitor is lower. Again, active probing techniques cannot measure (or account for) the page caching efficiency to reflect the "true" end-to-end response time observed by the actual clients.

We now switch to the analysis of the OV-Support site. We will only highlight some new observations specific to this site. Figure 11 a) shows the average end-to-end response time as measured by EtE monitor when downloading the site main page. This site uses JavaServer Pages technology for



a) b)

Figure 11: OV-Support site during 2 weeks: a) end-to-end response time for accesses to a main page; b) network-server time ratio for the main page.



a) b)

Figure 12: OV-Support site during 2 weeks: a) connection setup time for the main page; b) an estimated percentage of end-to-end response time improvement if the server runs HTTP1.1.

dynamic generation of the content. Since dynamic pages are typically more "compute intensive," it has a corresponding reflection in higher server-side processing fraction in overall response time. Figure 11 b) shows the network-server time ratio in the overall response time. It is higher compared to the network-server ratio for static pages from the HPL site. One interesting detail is that the response time spike around the 127 hour mark has a corresponding spike in increased server processing time, indicating some server-side problems at this point. The combination of data provided by EtE monitor can help service providers to better understand site-related performance problems.

The OV-Support site pages are composed of a large number of embedded images. Two most popular site pages, which account for almost 50% of all the page accesses, consist of 32 objects. The caching efficiency for the site is very high: only 8-9 objects are typically retrieved from the server, while the other objects are served from network and browser caches. The site server is running HTTP 1.0 server. Thus typical clients used 7-9 connections to retrieve 8-9 objects. The *ConcurrencyCoef* (see Section 6), which reflects the overlap portion of the latency between different connections for this page, was very low, around 1.038 (in fact, this is true for the site pages in general). This indicates that the efficiency of most of these connections is almost equal to sequential retrievals through a single persistent connection.

Figure 12 a) shows the connection setup time measured by EtE monitor. We perform a simple computation: how much of the end-to-end response time observed by current clients can be improved if the site server would run an HTTP 1.1 server, allowing clients to use just two persistent connections to retrieve the corresponding objects from the site? In other words, how much of the response time can be improved by eliminating unnecessary connection setup time? Figure 12 b) shows the estimated percentage of end-to-end response time improvement available from running an HTTP 1.1 server. On average, during the observed interval, the response time improvement for *url1* is around 20% (2.6 sec is decreased to 2.1 sec), and for *url2* is around 32% (3.3 sec is decreased to 2.2 sec).

Figure 12 b) reveals an unexpected "gap" between 230-240 hour marks, when there was "no improvement" due to HTTP 1.1. More careful analysis shows that during this period, all the accesses retrieved only a basic HTML page using 1 connection, without consequent image retrievals. The other pages during the same interval have a similar pattern. It looks like the image directory was not accessible on the server. Thus, EtE monitor, by exposing the abnormal access patterns, can help service providers get additional insight into service related problems.

Client population analysis is another area attracting service provider's speical interest. Knowledge about the largest client clusters and their response times is extremely useful for service providers to make wise decisions on additional server and cache placement [15].

EtE monitor can provide information about client clustering by associating them with corresponding ASes (Autonomous Systems). The service providers of the *IT-Support* site have a special concern about their clients from the Asia-Pacific region (AS numbers between 9216-10239 represent Asia-Pacific domains). Table 2 shows the average response times for Asia-Pacific clients (AP clients) and percentage of their accesses to the four most popular *URL*s under study.

Metrics	*url1*	*url2*
EtE time (All Clients)	3.2 sec	3.5 sec
EtE Time (Asia-Pacific Clients)	3.9 sec	4.7 sec
% of Asia-Pacific Clients Accesses	4.8%	6.1%

Table 2: Percentage of the client accesses from the Asia-Pacific regionto the IT-Support site and end-to-end response times for these accesses.

As Table 2 shows, the AP client accesses constitute from 4.8% to 6.1% of all clients accesses for two most popular *URL*s. The end-to-end response times observed by Asia-Pacific clients for *url1* and *url2* are only slightly higher than the corresponding average response times observed by all the clients, which was counter-intuitive.

EtE monitor provides a daily analysis of client population accessing a set of URLs of interest. This information provides a useful quantitative view on response times to the major client clusters. It can be used for site efficiency design to determine if a geographically distributed web cluster is needed to improve site performance. Such information can also be used for content delivery networks to make appropriate decisions on data placement for a given client population.

The ability of EtE monitor to reflect a site performance for different ASes (and groups of IP addresses) happens to be a very attractive feature for service providers. When service providers have special SLA-contracts with certain groups of customers, EtE monitor provides a unique ability to measure the response time observed by those clients and to validate QoS targets for those contracts. This area is a promising direction for our future work on EtE monitor.

8 Limitations

There are a number of limitations to our EtE monitor architecture. Since EtE monitor extracts HTTP transactions by reconstructing TCP connections from captured network packets, it is unable to obtain HTTP information from encrypted connections. Thus, EtE monitor is not appropriate for sites that encrypt much of their data (e.g., via SSL).

In principle, EtE monitor must capture all traffic entering and exiting a particular site. Thus, our software must typically run on a single web server or a web server cluster with a single entry/exit point where EtE monitor can capture all traffic for this site. If the site "outsources" most of its popular content to CDN-based solutions then EtE monitor can only provide the measurement information

about the "rest" of the content, which is delivered from the original site. For sites using CDN-based solutions, the active probing or page instrumentation techniques are more appropriate solutions to measure the site performance. A similar limitation applies to pages with "mixed" content: if a portion of a page (e.g., an embedded image) is served from a remote site, then EtE monitor cannot identify this portion of the page and cannot provide corresponding measurements. In this case, EtE monitor consistently identifies the portion of the page that is stored at the local site, and provides the corresponding measurements and statistics. In many cases, such information is still useful for understanding the performance characteristics of the local site.

The EtE monitor does not capture DNS lookup times. Only active probing techniques are capable of measuring this portion of the response times. Further, for clients behind proxies, EtE monitor can only measure the response times to the proxies instead of to the actual clients.

As discussed in Section 3, the heuristic we use to reconstruct page content may determine incorrect page composition. Although the statistics of access patterns can filter invalid accesses, it works best when the sample size is large enough.

Dynamically generated web pages introduce another issue with our statistical methods. In some cases, there is no consistent *content template* for a dynamic web page if each access consists of different embedded objects (for example, some pages use a rotated set of images or are personalized for client profiles). In this case, there is a danger that metrics such as the *server file hit ratio* and the *server byte hit ratio* introduced in Section 6 may be inaccurate. However, the end-to-end time will be computed correctly for such accesses.

There is an additional problem (typical for server access log analysis of e-commerce sites) about how to aggregate and report the measurement results for dynamic sites where most page accesses are determined by *URL*s with client customized parameters. For example, an e-commerce site could add some client specific parameters to the end of a common URL path. Thus, each access to this logically same URL has a different URL expression. However, service providers may be able to provide the policy to generate these URLs. With the help of the policy description, EtE monitor is still able to aggregate these URLs and measure server performance.

9 Conclusion and Future Work

Today, understanding the performance characteristics of Internet services is critical to evolving and engineering Internet services to match changing demand levels, client populations, and global network characteristics. Existing tools for evaluating web service performance typically rely on active probing to a fixed set of URLs or on web page instrumentation that monitors download performance to a client and transmits a summary back to a server. This paper presents, EtE monitor, a novel approach to measuring web site performance. Our system passively collects packet traces from the server site to determine service performance characteristics. We introduce a two-pass heuristic method and a statistical filtering mechanism to accurately reconstruct composition of individual page and performance characteristics integrated across all client accesses.

Relative to existing approaches, EtE monitor offers the following benefits: i) a breakdown between the network and server overhead of retrieving a web page, ii) longitudinal information for all client accesses, not just the subset probed by a third party, iii) characteristics of accesses that are aborted by clients, and iv) quantification of the benefits of network and browser caches on server performance. Our initial implementation and performance analysis across two sample sites confirm the utility of our approach. We are currently investigating the use of our tool to understand the client performance on a per-network region. This analysis can aid in the placement of wide-area replicas or in the choice

of an appropriate content distribution network. Finally, our architecture is general to analyzing the performance of multi-tiered web services. For example, application-specific log processing can be used to reconstruct the breakdown of latency across tiers for communication between a load balancing switch and a front end web server, or communication between a web server and the storage tier/database system.

Acknowledgments: Both the tool and the study would not have been possible without generous help of our HP colleagues: Mike Rodriquez, Steve Yonkaitis, Guy Mathews, Annabelle Eseo, Peter Haddad, Bob Husted, Norm Follett, Don Reab, and Vincent Rabiller. Their help is highly appreciated. Our special thanks to Claude Villermain who helped to identify and to correct a subtle bug for dynamic page reconstruction.

References

[1] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching, and Zipf-like Distributions: Evidence, and Implications, In Proceedings of IEEE INFOCOM, March, 1999.

[2] Candle Corporation: eBusiness Assurance. `http://www.candle.com/`.

[3] Cisco DistributedDirector., `http://www.cisco.com/`.

[4] F.D. Smith, F.H. Campos, K. Jeffay, and D. Ott. What TCP/IP Protocol Headers Can Tell Us About the Web. In Proceedings of ACM SIGMETRICS, Cambridge, May, 2001.

[5] A. Feldmann. BLT: Bi-Layer Tracing of HTTP and TCP/IP. Proceedings of WWW-9, May 2000.

[6] Y. Fu, L. Cherkasova, W. Tang, A. Vahdat. EtE: Passive End-to-End Internet Service Performance Monitoring. In Proceedings of USENIX Annual Technical Conference, June 13-15, Monterey, CA, 2002.

[7] HP Corporation. OpenView Products: Web Transaction Observer. `http://www.openview.hp.com/`.

[8] R. Fielding, J. Gettys, J. Mogul, H. Nielsen, and T. Berners-Lee. Hypertext Transfer Protocol - HTTP/1.1. RFC 2616 , IETF, June 2001. http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[9] IBM Co.Tivoli Web Management Solutions, `http://www.tivoli.com/products/demos/twsm.html`.

[10] JavaServlet Technology. `http://java.sun.com/products/servlet/`.

[11] JavaServer Pages. http://java.sun.com/products/jsp/technical.html.

[12] T. Keeley. Thin, High Performance Computing over the Internet. Invited talk at Eighth Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'2000).

[13] Keynote Systems, Inc. `http://www.keynote.com`.

[14] B. Krishnamurthy and J. Rexford. Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement, pp.511-522, Addison Wesley, 2001.

[15] B. Krishnamurthy and J.Wang, On Network-Aware Clustering of Web Clients. Proceedings of ACM SIGCOMM 2000, August 2000.

[16] D. Mosberger and T. Jin. Httperf—A Tool for Measuring Web Server Performance. J. of Performance Evaluation Review, Volume 26, Number 3, December 1998.

[17] NetMechanic, Inc. `http://www.netmechanics.com`.

[18] NetQoS, Inc. `http://www.netqos.com`.

[19] Porivo Technologies, Inc. `http://www.porivo.com`.

[20] R. Rajamony, M. Elnozahy. Measuring Client-Perceived Response Times on the WWW. USENIX Symposium on Internet Technologies and Systems (USITS), March 2001, San Francisco.

[21] S. Seshan, M. Stemm and R. Katz. SPAND: Shared Passive Network Performance Discovery USENIX Symposium on Internet Technologies and Systems, 1997.

[22] Mark Stemm, Randy Katz, Srinivasan Seshan. A Network Measurement Architecture for Adaptive Applications. Proc. of IEEE INFOCOM, 2000.

[23] Software Research Inc. `http://www.soft.com`.

[24] `http://www.tcpdump.org`.

available and high performance network services. As a graduate student he participated in the design and implementation of the UC Berkeley Network of Workstations project. Some of this work was later commercialized as part of the Inktomi search engine. At Duke University, A. Vahdat leads the Internet Systems Software Group (ISSG). A. Vahdat is a winner of the prestigious National Science Foundation Faculty Development CAREER award and a member of ACM, IEEE, and USENIX.

QW2002 Paper 2I2

Dr. Selim Aissi
(Intel)

Web Services QoS Indicators

Presentation Abstract

Web Services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. QoS of a Web Service is its ability to have some level of assurance that its service and traffic requirements are satisfied in a timely fashion. That means that all the components of a Web Service need to be considered when evaluating its QoS: backbone, core, access, and applications.

So, what are the criteria (Quality Indicators) of a usable and reliable Web Service? When dealing with QoS, it is critical to have an end-to-end view of Web Services because they represent a paradigm shift in the way applications are developed. Modules are no longer modules by one party. Some of the functionality a user will get by calling Web services. But then, that user will need to be very careful which services are useful and which are not. Some sites are doing a great service by providing acceptance tests and expected results, but they still provide a granular view of QoS since they only validate only one component. This presentation addresses the end-to-end challenges of Web Services QoS.

About the Author

Dr. Aissi has been involved in the development of Safety-Critical Embedded Systems in the R&D, military, automotive, and wireless appliances for over twelve years. He worked at the University of Michigan, General Dynamics (M1A2 Abrams Battlefield Tank Division), General Motors (Embedded Controller Excellence Center), Applied Dynamics International, and Intel Corporation. Dr. Aissi has played several management and architecture roles at Intel, and he is currently a Sr. Architect at Intel's Corporate Technology Group in Hillsboro, Oregon, USA and serves as the Intel representative to several Web Services Standards Working Groups, including the OASIS CPPA and the UDDI Technical Committees. He holds a Ph.D. in Aerospace Engineering from the University of Michigan.

Web Services QoS Indicators

Selim Aissi, Ph.D.

Senior Security Architect
Corporate Technology Group
Intel Corporation

September 4, 2002

intel.

Intel Research & Development

Agenda

- What are Web Services?
- So, what's new?
- The 7 Web Services Quality Indicators
  - Accessibility
  - Availability
  - Reliability
  - Interoperability
  - Integrity
  - Performance
  - Security
  - Regulatory
- Concluding Remarks

intel.

Intel Research & Development

What are Web Services?

- Self-describing, self-contained, modular applications
- Platform and implementation neutral
- Based on open standards for description, discovery, and invocation
- Programmatically connect business process together
- Typically transactional, requiring integration with existing systems

Applications that interact with each other using Web standards

intel.

Intel Research & Development

What are Web Services?

- Basic Definitions of Core Layers-

- Common Internet Transport Protocols.
  - Although not specifically tied to any transport protocol, Web Services build on ubiquitous Internet connectivity and infrastructure to ensure nearly universal reach and support.
  - In particular, Web Services take advantage of HTTP, the same connection protocol used by web servers and browsers.
- Extensible Markup Language (XML).
  - XML is a widely accepted format for exchanging data and its corresponding semantics.
  - It is a fundamental building block for nearly every other layer in the Web Services stack.
- Simple Object Access Protocol (SOAP).
  - SOAP is a protocol for messaging and RPC-style communication between applications.
  - Based on XML and uses common Internet transport protocols like HTTP to carry its data.

intel.

Intel Research & Development

What are Web Services?

- Basic Definitions of Core Layers-

- Web Services Description Language (WSDL).
  - WSDL is an XML-based description of how to connect to a particular web service.
  - A WSDL description abstracts a particular service's various connection and messaging protocols into a high-level bundle and forms a key element of the UDDI directory's "green pages." IBM recently submitted WSDL to the W3C, and it will likely be adopted in some form.
- Universal Description, Discovery, and Integration (UDDI).
  - UDDI represents a set of protocols and a public directory for the registration and real-time lookup of web services and other business processes.
  - Version 3 of the specification is available

intel.

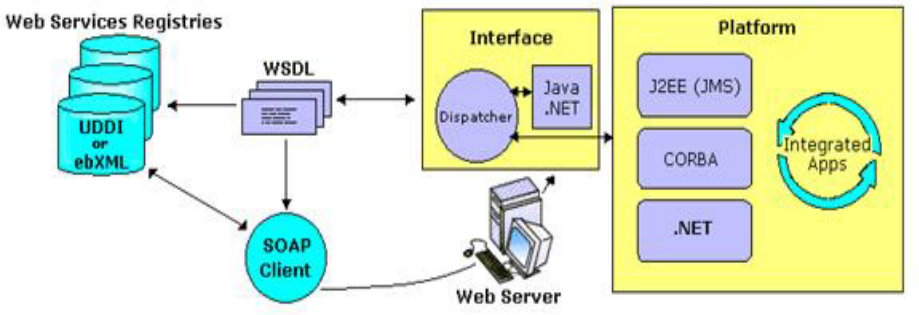
5

Intel Research & Development

What are Web Services?

- Example-

- Example Web Services Architecture



intel.

6

Intel Research & Development

Agenda

- What are Web Services?
- So, what's new?
- The 7 Web Services Quality Indicators
  - Accessibility
  - Availability
  - Reliability
  - Interoperability
  - Integrity
  - Performance
  - Security
  - Regulatory
- Concluding Remarks

So, what's new?

- continued-

- Web Services Standardizes
  - Data definition, exchange, transformation (XML)
  - Remote procedure call protocol (SOAP)
  - Description of Web Services (WSDL)
  - Methods of publishing Web Services (UDDI)
- Based on realistic assumptions of how IT operates
  - Heterogeneous - different languages, data structures, operating systems and hardware
  - Loosely coupled - asynchronous, message-based integration between applications

Agenda

- What are Web Services?
- So, what's new?
- The 8 Web Services Quality Indicators
  - Accessibility
  - Availability
  - Reliability
  - Interoperability
  - Integrity
  - Performance
  - Security
  - Regulatory
- Concluding Remarks

intel

Intel Research & Development

Accessibility

- Accessibility is the QoS Indicator of a service that represents the *degree it is capable of serving a Web service request*.
- It may be expressed as a probability measure denoting the success rate or chance of a successful service instantiation at a point in time.
- There could be situations when a Web Service is available but not accessible.
- High accessibility of a Web Service can be achieved by building highly scalable systems.
- *Scalability* refers to the ability to consistently serve the requests despite variations in the volume of requests.

intel

Intel Research & Development

Availability

- Availability is the QoS Indicator of whether the Web Service is present or ready for immediate use.
- Availability represents the probability that a service is available.
- Larger values represent that the service is always ready to use while smaller values indicate unpredictability of whether the service will be available at a particular time.
- Also associated with availability is Time-To-Repair (TTR). *TTR* represents the time it takes to repair a service that has failed.
- Ideally smaller values of TTR are desirable.

intel.

11

Intel Research & Development

Reliability

- Reliability is the QoS Indicator of a Web Service that represents *the degree of being capable of maintaining the service and service quality*.
- The number of failures per month or year represents a measure of reliability of a Web Service.
- In other words, *reliability* refers to the *assured and ordered delivery* for messages being sent and received by service requestors and service providers.

intel.

12

Intel Research & Development

Interoperability

- Interoperability is the QoS Indicator of whether a Web Service can consume services derived from non-homogeneous applications.
- Current implementation of standards varies enough to make heterogeneous solutions difficult.
- WS-I (Web Services Interoperability) Organization is about to publish various Interoperability Profiles: Basic, Security, etc.
- Issues to address in design and verification/validation:
  - What does “soap:actor” mean on SOAP headers?
  - “SOAPAction” HTTP header
  - Fault-propagation (semantics, “soap fault with HTTP status 200-500”, etc.)
  - Custom fault codes (remember: SOAP allows for the definition of custom SOAP fault codes)
  - Intermediaries

intel.

13

Intel Research & Development

Integrity

- Integrity is the QoS Indicator of how a Web Service maintains the correctness of the interaction in respect to the source.
- Proper execution of Web Service transactions will provide the correctness of interaction.
- A *transaction* refers to a sequence of activities to be treated as a single unit of work.
- All the activities have to be completed to make the transaction successful.
- When a transaction does not complete, all the changes made are rolled back.

intel.

14

Intel Research & Development

Performance

- Performance is the QoS Indicator of Web Service, which is measured in terms of throughput and latency.
- Higher throughput and lower latency values represent good performance of a Web Service.
- *Throughput* represents the number of Web Service requests served at a given time period.
- *Latency* is the round-trip time between sending a request and receiving the response.

intel.

15

Intel Research & Development

Security

- Security is the QoS Indicator of the Web Service of providing confidentiality and non-repudiation by authenticating the parties involved, encrypting messages, and providing access control.
- Security has added importance because Web Service invocation occurs over the public Internet.
- The service provider can have different approaches and levels of providing security depending on the service requestor.

intel.

16

Intel Research & Development

Regulatory

- Regulatory is the QoS Indicator of the Web Service in conformance with the rules, the law, compliance with standards, and the established service level agreement.
- Web Services use many standards such as SOAP, UDDI, and WSDL.
- Strict adherence to correct versions of standards (e.g., SOAP V1.2) by service providers is necessary for proper invocation of Web Services by service requestors.

Agenda

- What are Web Services?
- So, what's new?
- The 7 Web Services Quality Indicators
  - Accessibility
  - Availability
  - Reliability
  - Interoperability
  - Integrity
  - Performance
  - Security
  - Regulatory
- Concluding Remarks

Concluding Remarks

- 8 main Web Services QoS indicators are described
- Accessibility, availability, reliability, interoperability, integrity, performance, security, and regulatory.
- Indicators need to be addressed in the implementation and verification/validation of Web Service applications.

References:

www.webservices.org
www-106.ibm.com/developerworks/
msdn.microsoft.com/library/

Thank you!

Selim Aissi, Ph.D.

selim.aissi@intel.com

QW2002 Paper 3I1

Dr. John M. Ryan
(Safe Shopping Network)

Measuring and Improving Web Site Quality: A Consumer Focused System

Key Points

- Web site functions represent a series of process steps
- These process steps may be defined, tested and scored
- Scores may be used to improve web site performance

Presentation Abstract

The safeshoppingnetwork.com establishes consumer based standards for web site functionality (security, privacy, safety, usability, performance and availability). These standards are then used to develop and adopt test software capable of delivering scores to data bases. Web sites are then ranked based on the published scores thus providing the consumer with guidance in each tested area. For the merchant, the scores along with published web-wide statistics (over 1000 cases) provide guidance for improvement.

This paper explores the standards, test strategies and software, discusses resulting statistical distributions and provides guidance for web site quality improvement.

The need to explore configuration control as a key strategy to reducing variation is also discussed.

About the Author

Dr. John M. Ryan one of the company's co-founders and serves as the COO and Senior Vice President of Test Technology. Dr. Ryan's extensive background in statistics, network controlled quality and highly efficient flow systems led to his development of the company's Internet based across-the-web test technologies. This is Dr. Ryan's second Internet start-up having served as president of the Marine Network.

Dr. John Ryan has previously held a number of international management positions with Intel, Seagate Technology and Read-Rite Corporation and as a private consultant where he has developed and implemented world-class manufacturing and quality systems throughout Asia. Dr. Ryan is a frequently published author in several quality, factory of the future and software journals.

QW2002 Paper 3I1

Dr. John M. Ryan
(Safe Shopping Network Test Development Center)

Measuring and Improving Web Site Quality: A Consumer Focused System

Security

Usability

Safety

Safe Shopping Network
Test Technology Center

Performance

Privacy

Availability

www.SafeShoppingNetwork.com
Know Before You Go!

Scoring Standards (100 Points Each)

Safety: Cart functionality, product delivery, security

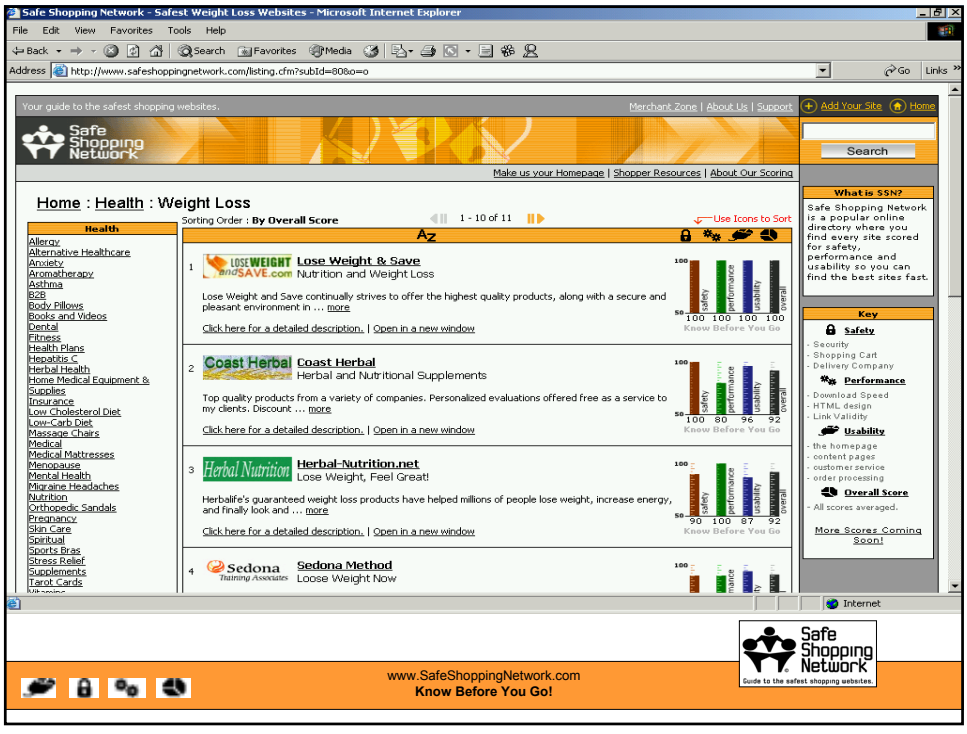
Performance: Download time, link validity, HTML validity

Usability: Homepage understandability/eye appeal/balance, content/product page layouts, customer service, order processing.

Privacy Policy: Exists, not rent/sell/give individual info, data collected, opt-out, child protection, etc.

Security: Vulnerability Scanning (1700 items by QualysGuard)

Availability: Page downloads/response time 30 day monitor/15 minute intervals

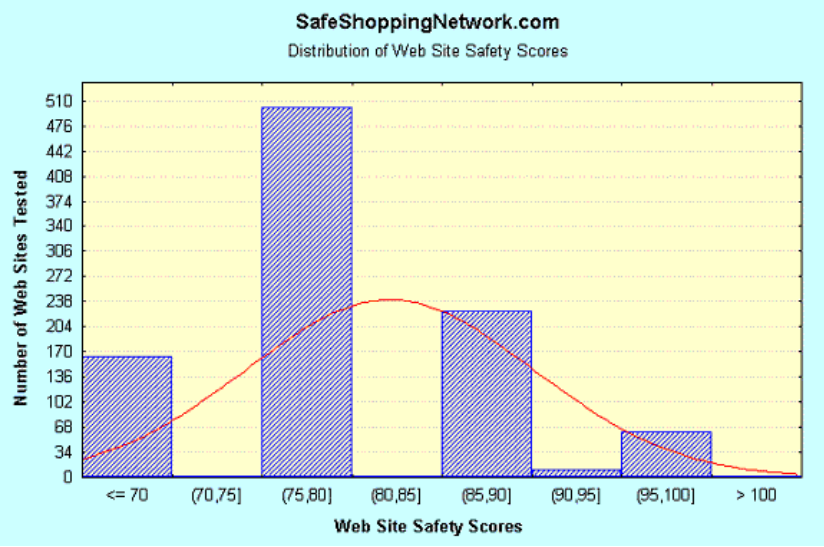


The Importance of Web Site Quality Used as a Ranking Mechanism

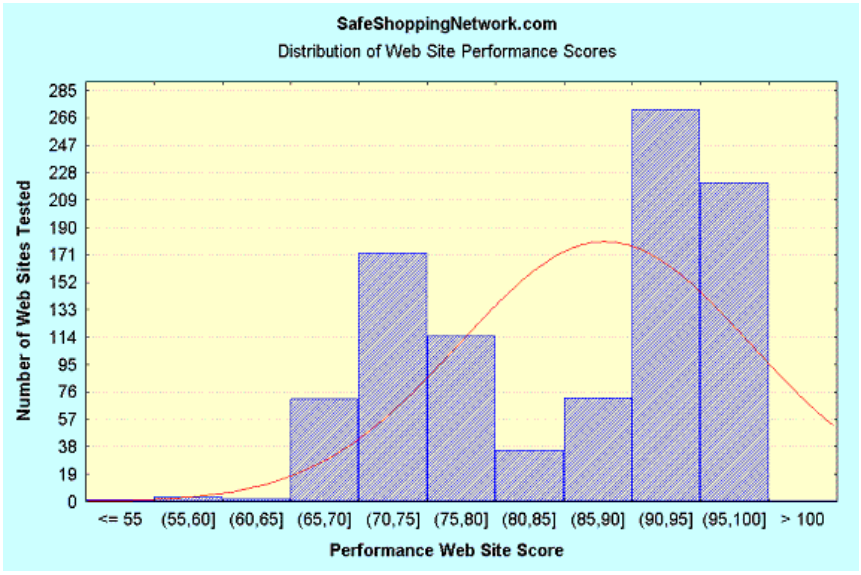
For the Weight Loss category (selected from the "Health" list, the **sites are ranked by the overall scores** each web site has achieved. In this case, Lose Weight & Save has topped the list.

Higher rankings on the list generally means that this **web site will receive more hits, more traffic and higher sales – based on the quality of the web site** (not CPM/CPC or the amount of money spent on securing other search engine positioning).

Quality = Sales = Profit

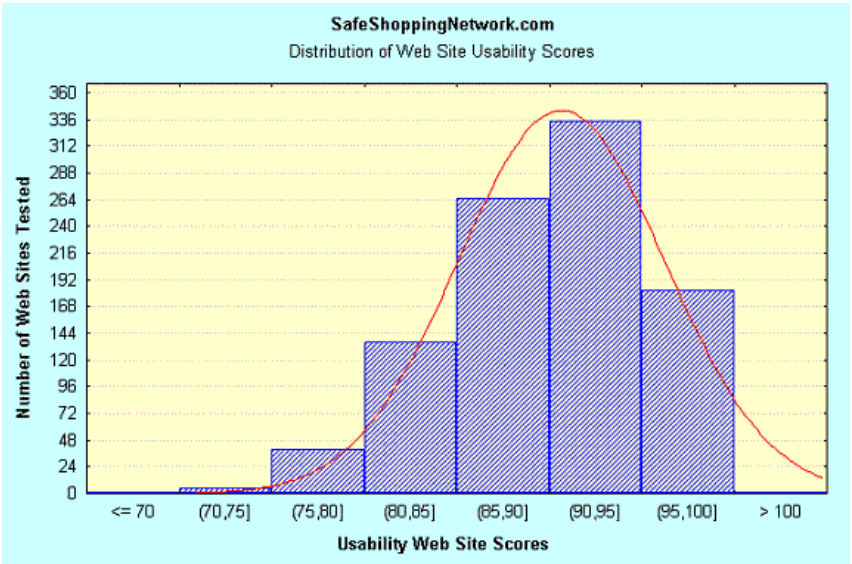


www.SafeShoppingNetwork.com
Know Before You Go!



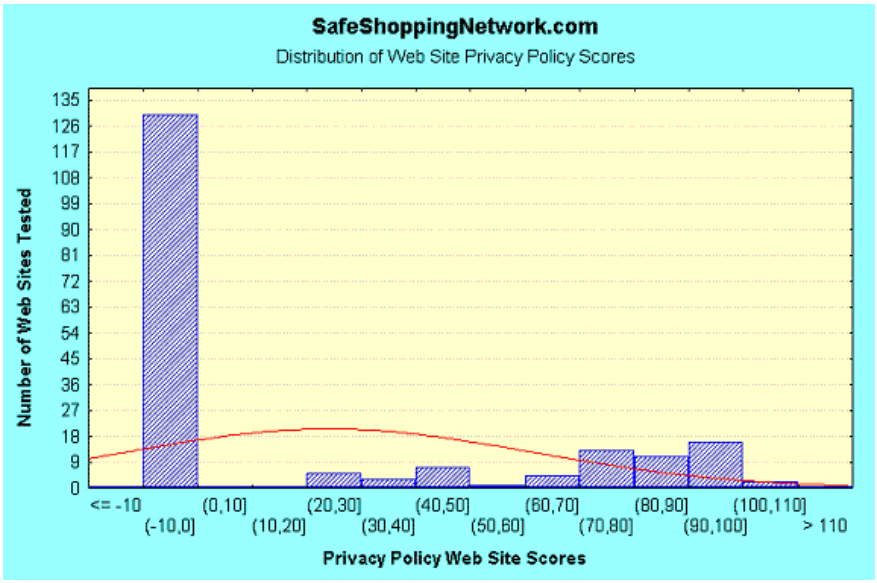
Safe Shopping Network
Guide to the safest shopping websites.

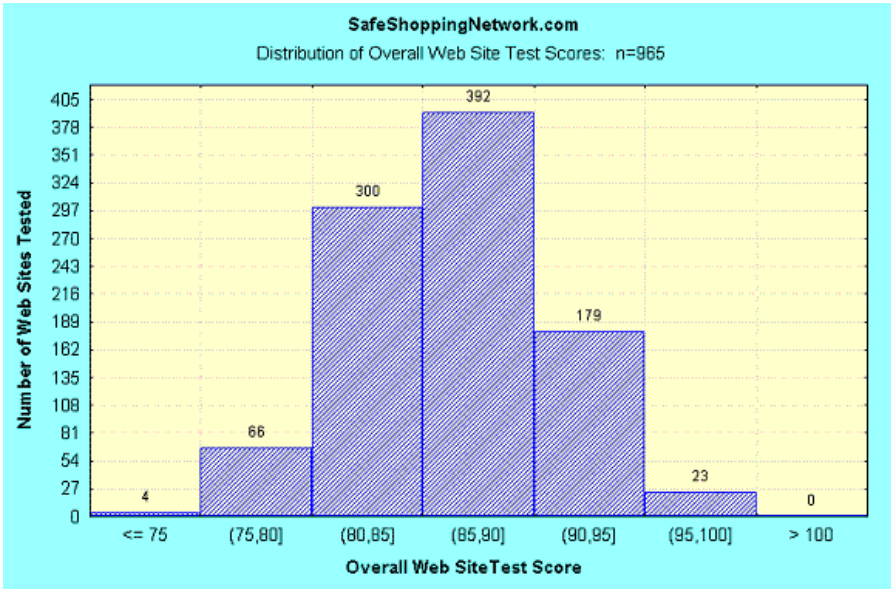
www.SafeShoppingNetwork.com
Know Before You Go!



Safe Shopping Network
Guide to the safest shopping websites.

www.SafeShoppingNetwork.com
Know Before You Go!


SafeShoppingNetwork.com
Distribution of Web Site Privacy Policy Scores
Number of Web Sites Tested
135
126
117
108
99
90
81
72
63
54
45
36
27
18
9
0
<= -10
(-10,0]
(0,10]
(10,20]
(20,30]
(30,40]
(40,50]
(50,60]
(60,70]
(70,80]
(80,90]
(90,100]
(100,110]
> 110
Privacy Policy Web Site Scores


SafeShoppingNetwork.com
Distribution of Overall Web Site Test Scores: n=965
Number of Web Sites Tested
405
378
351
324
297
270
243
216
189
162
135
108
81
54
27
0
4
66
300
392
179
23
0
<= 75
(75,80]
(80,85]
(85,90]
(90,95]
(95,100]
> 100
Overall Web SiteTest Score

Correlations Among Major Test Factors

	USABILITY	PERFORMANCE	SAFETY	PRIVACY
USABILITY	1.00	.09	.41	.25
PERFORMANCE		1.00	.15	-.04
SAFETY			1.00	.21
PRIVACY				1.00

Performance Scores Correlation Analysis

"Performance" is tested across-the-web by software. Three items are explored and scored by the software: HTML, Download time and broken (or unbroken) links.

As expected all three of these items correlated significantly with each other and correlated very positively with the overall Web Site Performance Score.

All three scores correlate significantly with the Overall Web Site Score (> . 41).

Usability Correlation Analysis

•First Impression, Balance and Eye Appeal form a significant "Visual" cluster. This can be explained in that all three items are related to how clean and marketable a web site looks when visited.

• Customer Contact items form a cluster separate from other Usability scores. Web sites that do not provide clear and multiple ways for customers to contact the e-commerce business management, get low scores.

• If the pages are set up in a clear manner, they tend to be easy to navigate.

• Web sites that provide a diverse selection of product types and strong visual appeal tend to get high scores on Usability. Apparently, having a good site designer pays off.

Safety Score Correlation Analysis

"Safety" is tested across-the-web by our patented software. Three items are explored and scored by the Safety Test: Availability of E-commerce shippers available, SSL enabled vs un-enabled, and the addition of shopping cart costs.

As expected all three of these items correlated significantly with each other (> .73) and correlated very positively (> .71)with the overall Web Site Safety Score.

All three scores also correlate highly (> .62) with the Overall Web Site Score.

Privacy Policy Correlation Analysis

(Continuously improving the standards, tests and configuration requirements)

The Privacy Policy scores were analyzed using pair-wise deletion due to the large number of sites that had no privacy policy published. Excluding the standard which awards points for a statement regarding child protection, all inter-correlation coefficients exceeded + . 36 with most well above the + .49 level.

We would expect the inclusion of a Privacy Policy to form a strong correlation cluster and with the exception of the standard which awards points for the inclusion of a child protection statement, the group did form a cluster.

Removing the requirement for a child protection statement will be studied further by the Safe Shopping Network Test Technology Center.

Early Conclusions From the Correlation Study

There are a number of predicted score cluster components operating within the overall scoring scheme employed by Safe Shopping Network test strategies. These clusters will be further studied to search for improved test strengthening schemes.

It appears that the child protection standard included as part of the Privacy Policy scoring should be studied further for possible elimination.

Improving Web Site Quality

Test scores in hand, the web site manager is armed with enough information to set priorities and begin improving the site.

A comparison of web site scores to established data.

Category	Item Tested	Score Low	Score Avg	Score High	Site Scores	Lower Quartile	Median	Upper Quartile
Performance	Web load	0	38.38	50	40	30	32.5	50
	Web HTML	0	22.18	25	25	20	25	25
	Total Performance	0	82.89	100	85	75	80	95
Safety	Cart Delivery	0	15.16	20	20	0	20	20
	Cart Function	0	19.16	45	20	12.5	15	35
	Cart Security	0	26.69	35	35	25	35	35
	Total Safety	0	61.01	100	75	45	70	90
	UPS Average	0	75.6	100	79.7	67.5	80	86.67

www.SafeShoppingNetwork.com
Know Before You Go!

Shopping Network
Guide to the safest shopping websites.

Summary

A. Set the standards
B. Test the sites
C. Improve the standards and test technology
D. Provide web sites with the tools to improve quality
E. Develop new low cost test suite software
F. Allow few configuration alternatives which violate quality standards.

The Safe Shopping Network is currently establishing partnerships with:

A. Appropriate test software and hardware companies.

B. Interested in partnerships with merchant portal companies in a revenue sharing agreement.

Please leave your card if your company is in a position to discuss a partnership.

QWE2002 Paper 3I2

Dr. Edward Miller
(eValid, Inc.)

InBrowser WebSite Testing: The Client-Side Approach

Key Points

- Modern websites are very complex, with many new kinds of objects.
- InBrowser testing -- done in the browser context -- is very simple.
- Although certain limits exist, InBrowser testing has many compelling features.

Presentation Abstract

Web sites are becoming increasingly more complex due to:
1.The inclusion of (e.g.) Flash objects, Java Applets, XML, JavaScript.
2. The increased use of Multiple Windows, Secure Log-Ins, Message Pop-ups and Web-launched applications. Testing these sites requires tools that can intuitively and accurately adapt to such complexity.

The first part of this talk will address many of the difficulties in testing a modern web site from the perspective of a professional web site quality tester. The second part will discuss the 'Browser-centric' test tool as a solution to some of these difficulties. The main focus will be on the qualities that a test tool needs to meet the myriad of requirements that the web site tester is faced with.

About the Author

Dr. Edward Miller is Chairman and President of Software Research, Inc., San Francisco, California, and Chief Technical Architect for software test tools development and software engineering quality questions. Dr. Miller has worked in the software quality management field for 25 years in a variety of capacities, and has been involved in the development of families of automated software, analysis and Web quality tools. He was chairman of the 1985 1st International Conference on Computer Workstations, and has participated in IEEE conference organizing activities for many years. He is the Chairman of the Quality Week Conferences since 1988. He is the author of Software Testing and Validation Techniques, an IEEE Computer Society Press tutorial text. Dr. Miller received his Ph.D. (Electrical Engineering) degree from the University of Maryland, an M.S. (Applied Mathematics) degree from the University of Colorado, and a BSEE from Iowa State University.

InBrowser WebSite Testing: The Client-Side Approach

Dr. Edward Miller

eValid, Inc.
901 Minnesota Street
San Francisco, CA 94107 USA
Email: miller@soft.com

Presentation Outline

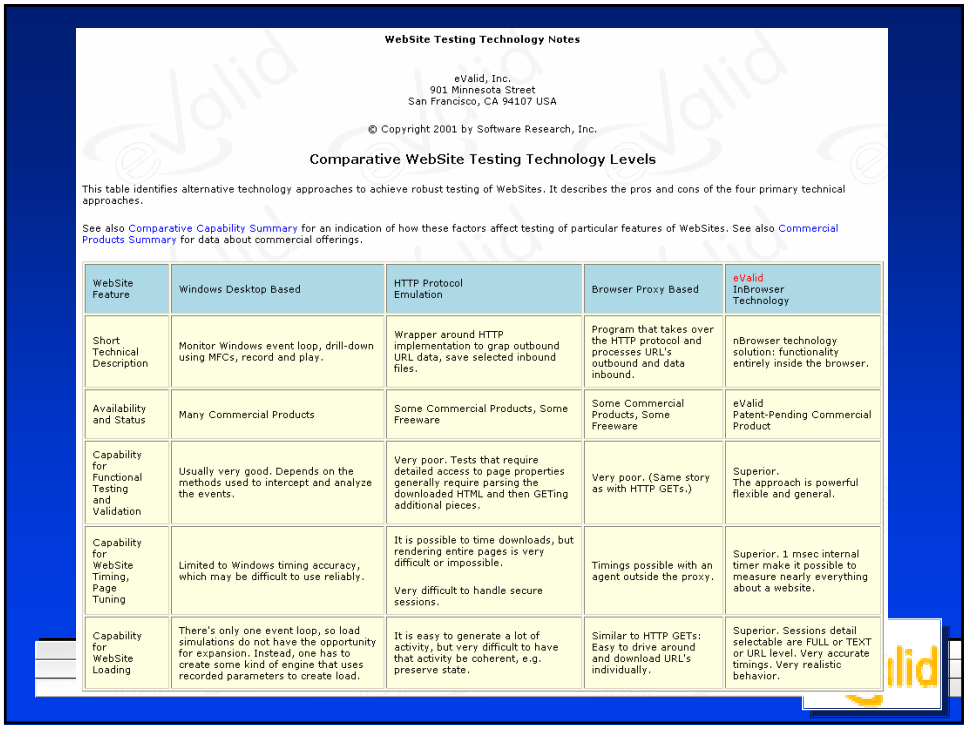
- Overview of Technology
- General Operation Description
- Validation & Verification Modes
- Timing and Tuning Process
- Load Imposition Mode
- SiteMap Mode
- Performance Considerations

Most Common Problems in WebSites

- Quality/Content
  - Broken Links
  - Missing/Broken Components
- Performance
  - Too-Slow Download
  - Incorrect Download
- Interaction
  - Failed 1st Layer Transactions
    - Login/Authentication
    - Specialized Controls
  - Delayed nth Tier Transactions

Alternative Technologies

- Windows Desktop
  - Client/Server Testing
  - Windows Events
  - Browser is "opaque"
- Unix Xwindows
  - Client/Server Testing
  - X-Display Events
  - Browser is "opaque"
- Browser Internal
  - Everything is open



WebSite Testing Technology Notes

eValid, Inc.
901 Minnesota Street
San Francisco, CA 94107 USA

© Copyright 2001 by Software Research, Inc.

Comparative WebSite Testing Technology Levels

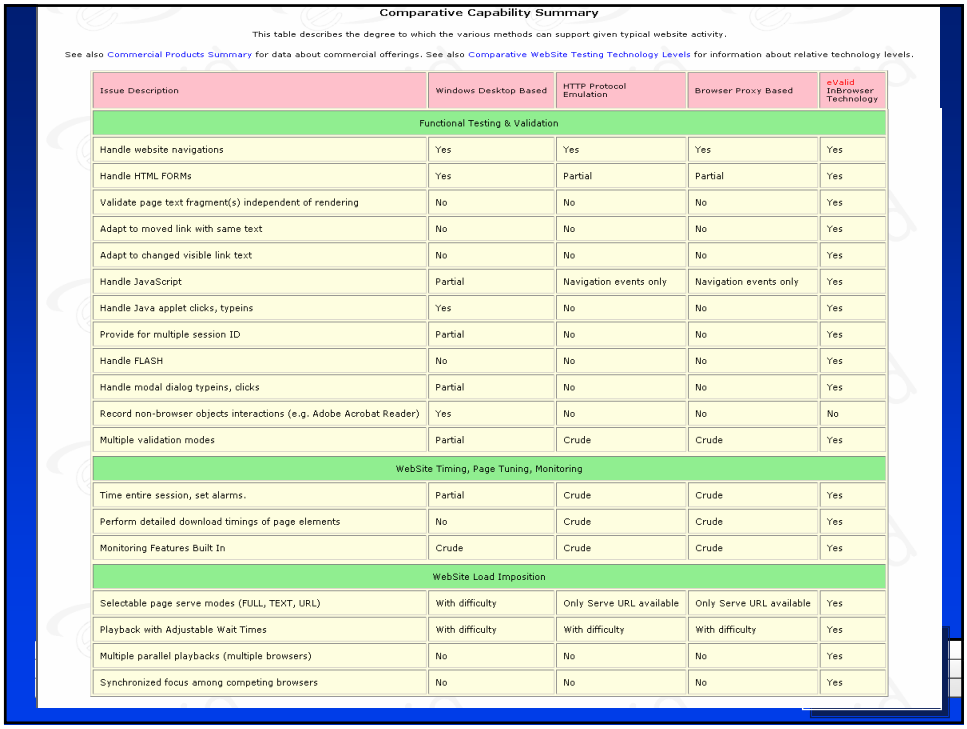
This table identifies alternative technology approaches to achieve robust testing of WebSites. It describes the pros and cons of the four primary technical approaches.

See also Comparative Capability Summary for an indication of how these factors affect testing of particular features of WebSites. See also Commercial Products Summary for data about commercial offerings.

WebSite Feature	Windows Desktop Based	HTTP Protocol Emulation	Browser Proxy Based	eValid InBrowser Technology
Short Technical Description	Monitor Windows event loop, drill-down using MFCs, record and play.	Wrapper around HTTP implementation to grap outbound URL data, save selected inbound files.	Program that takes over the HTTP protocol and processes URL's outbound and data inbound.	nBrowser technology solution: functionality entirely inside the browser.
Availability and Status	Many Commercial Products	Some Commercial Products, Some Freeware	Some Commercial Products, Some Freeware	eValid Patent-Pending Commercial Product
Capability for Functional Testing and Validation	Usually very good. Depends on the methods used to intercept and analyze the events.	Very poor. Tests that require detailed access to page properties generally require parsing the downloaded HTML and then GETing additional pieces.	Very poor. (Same story as with HTTP GETs.)	Superior. The approach is powerful flexible and general.
Capability for WebSite Timing, Page Tuning	Limited to Windows timing accuracy, which may be difficult to use reliably.	It is possible to time downloads, but rendering entire pages is very difficult or impossible. Very difficult to handle secure sessions.	Timings possible with an agent outside the proxy.	Superior. 1 msec internal timer make it possible to measure nearly everything about a website.
Capability for WebSite Loading	There's only one event loop, so load simulations do not have the opportunity for expansion. Instead, one has to create some kind of engine that uses recorded parameters to create load.	It is easy to generate a lot of activity, but very difficult to have that activity be coherent, e.g. preserve state.	Similar to HTTP GETs: Easy to drive around and download URL's individually.	Superior. Sessions detail selectable are FULL or TEXT or URL level. Very accurate timings. Very realistic behavior.

Windows Technology Options

- Windows Desktop
  - Operates from Windows Desktop Event Loop
  - GUI objects partially opaque
- HTTP Protocol
  - Records outbound URLs and response pages
- Browser Proxy
  - Records activity in/out from browser on HTML
- InBrowser Technology
  - Runs inside IE-compatible browser
  - Full context
  - Realistic timings



Comparative Capability Summary

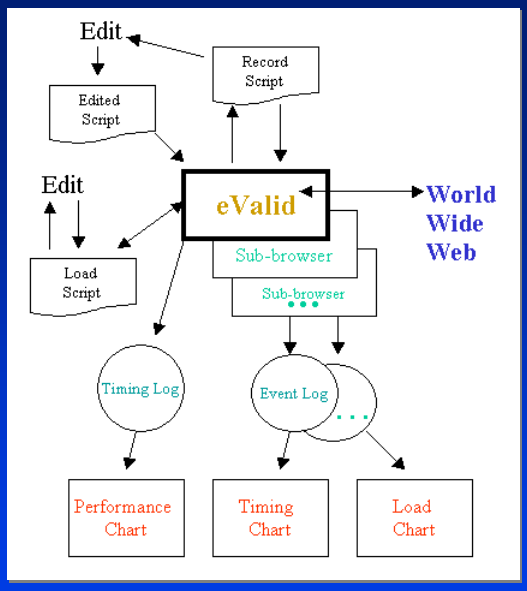
This table describes the degree to which the various methods can support given typical website activity.

See also Commercial Products Summary for data about commercial offerings. See also Comparative WebSite Testing Technology Levels for information about relative technology levels.

Issue Description	Windows Desktop Based	HTTP Protocol Emulation	Browser Proxy Based	eValid InBrowser Technology
Functional Testing & Validation				
Handle website navigations	Yes	Yes	Yes	Yes
Handle HTML FORMs	Yes	Partial	Partial	Yes
Validate page text fragment(s) independent of rendering	No	No	No	Yes
Adapt to moved link with same text	No	No	No	Yes
Adapt to changed visible link text	No	No	No	Yes
Handle JavaScript	Partial	Navigation events only	Navigation events only	Yes
Handle Java applet clicks, typeins	Yes	No	No	Yes
Provide for multiple session ID	Partial	No	No	Yes
Handle FLASH	No	No	No	Yes
Handle modal dialog typeins, clicks	Partial	No	No	Yes
Record non-browser objects interactions (e.g. Adobe Acrobat Reader)	Yes	No	No	No
Multiple validation modes	Partial	Crude	Crude	Yes
WebSite Timing, Page Tuning, Monitoring				
Time entire session, set alarms.	Partial	Crude	Crude	Yes
Perform detailed download timings of page elements	No	Crude	Crude	Yes
Monitoring Features Built In	Crude	Crude	Crude	Yes
WebSite Load Imposition				
Selectable page serve modes (FULL, TEXT, URL)	With difficulty	Only Serve URL available	Only Serve URL available	Yes
Playback with Adjustable Wait Times	With difficulty	With difficulty	With difficulty	Yes
Multiple parallel playbacks (multiple browsers)	No	No	No	Yes
Synchronized focus among competing browsers	No	No	No	Yes

eValid Technology Testing Pros/Cons

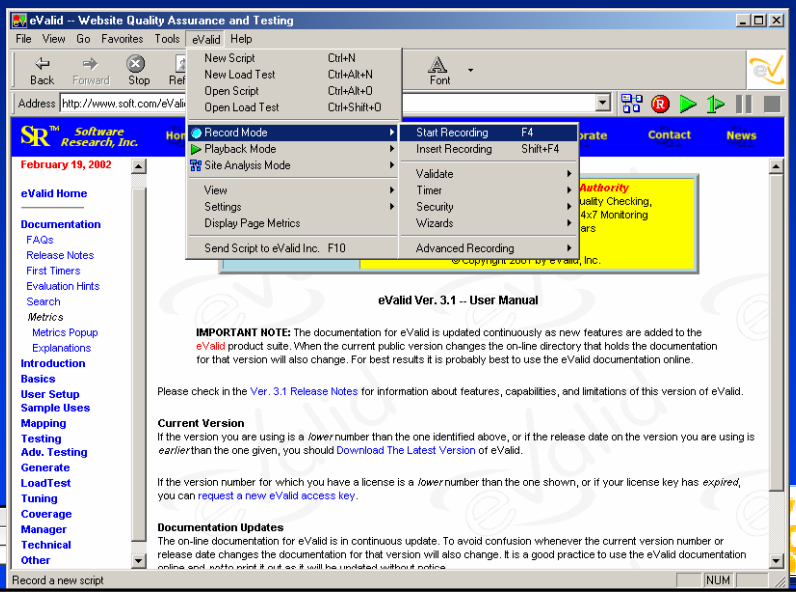
- Advantages
  - User View
  - Realistic
  - Natural operation
  - Accurate timings
- Disadvantages
  - Browser Variations
  - UNIX platform support



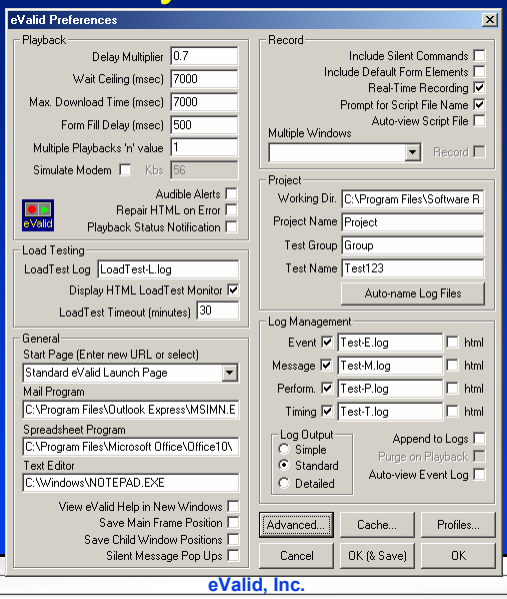
eValid Characteristics

- IE Base (IE 5.5+)
- NT/2000/XP
- Simple Script Language
- Point and Click Interface
- Online Documentation
- Multiple Copy Playback
- Built in Spider

eValid Opening Menu & Pulldown

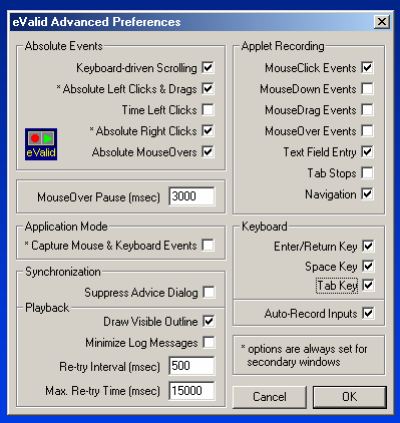


Record/Play Preferences Menu

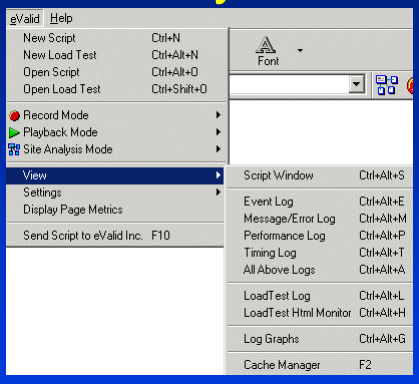


eValid, Inc.

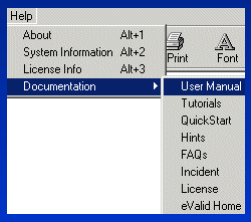
Advanced Preferences Meun



eValid Functional Testing Record/Play Pulldown



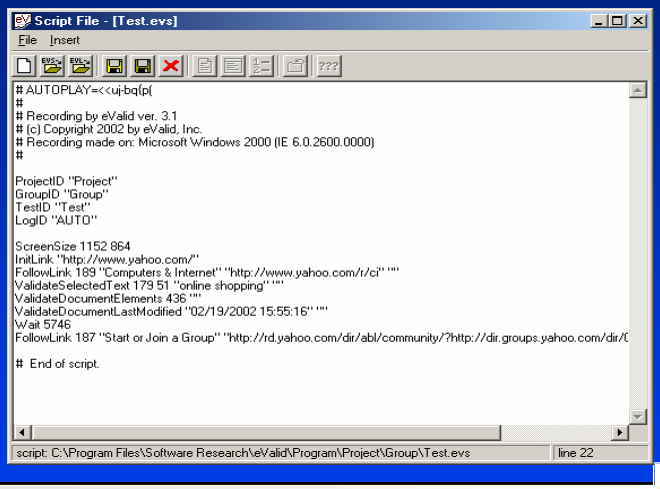
eValid Documentation/Support Pulldown



Recording Modes

- Navigation (including Frames)
- Modal Dialog
- Applet
- Application Mode
- Dependent Sub-Window
- Independent Sub-Window

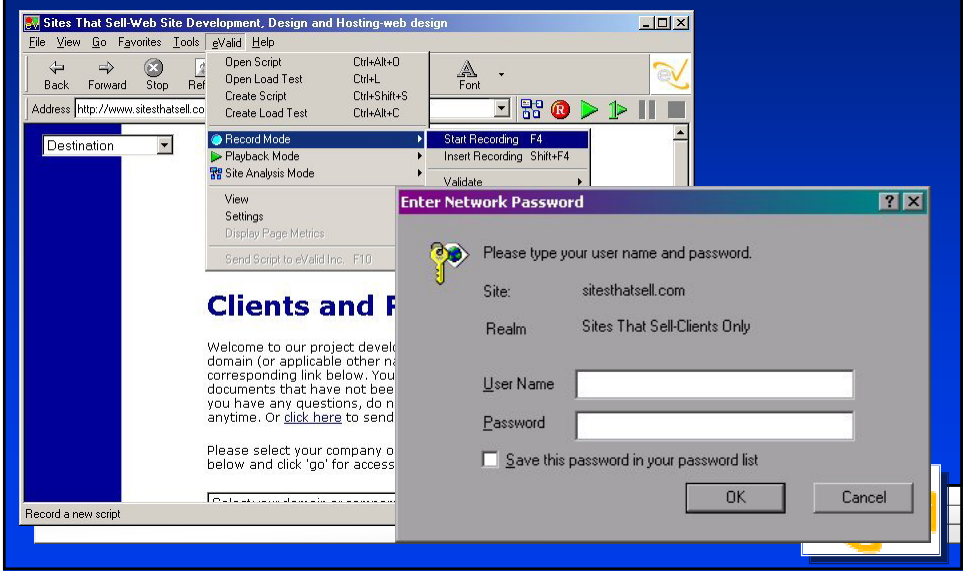
Example Script Window



Modal Dialog Recording

- Modal Dialog Has Screen Focus
- Access Recording via Script Window
- Pulldown for Allowable Input Sequences
- Extrinsic Commands on Same Window

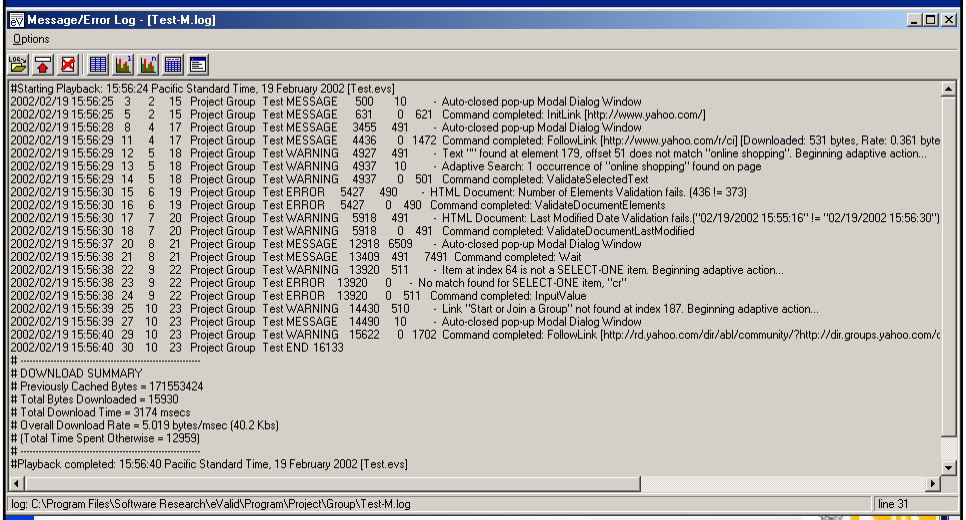
Sample of Modal Dialog Recording



Validation Modes

- Text Fragments
- Images, Links, Objects
- Image Part (with Synchronization)
- Document Properties
- Table Cells
- Element Ids (ID Tags)
- Applets (Size, Identification)

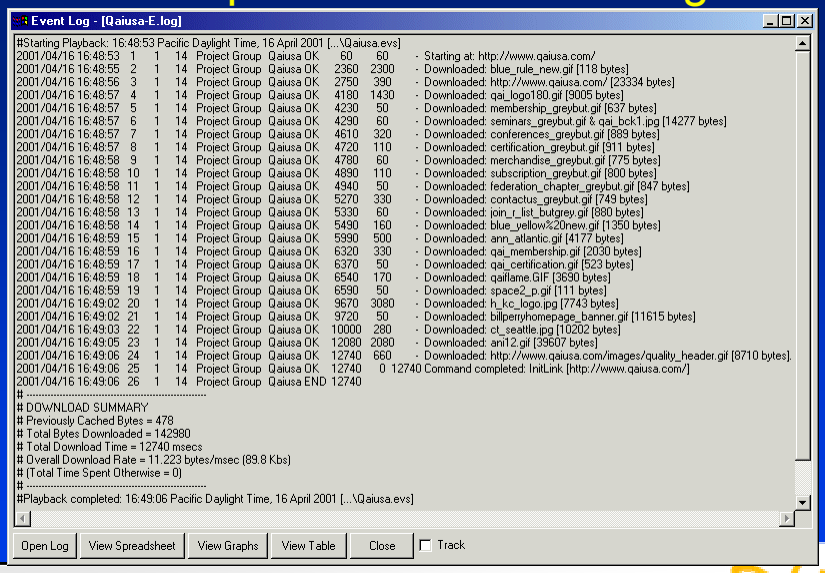
Example Message/Error Log



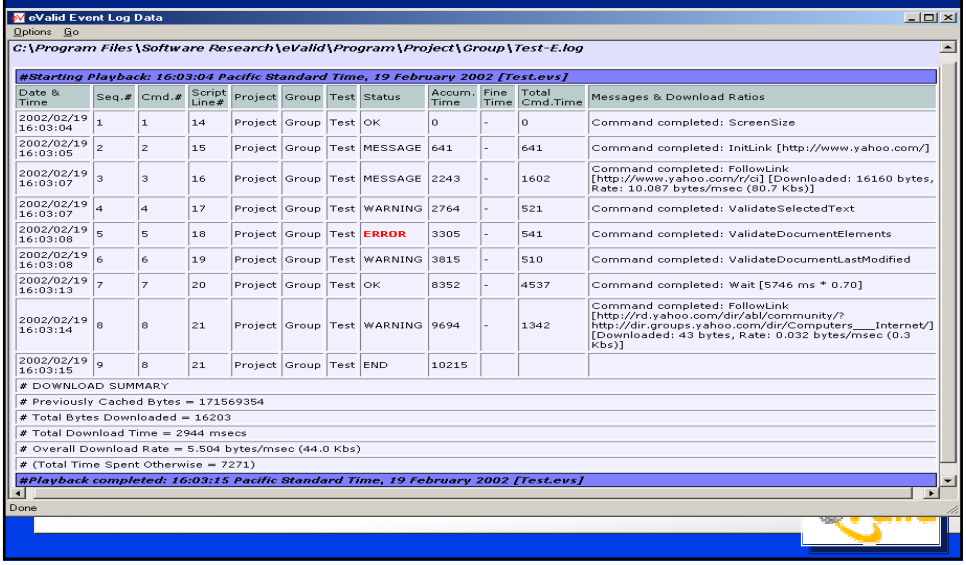
Performance Timing/Tuning

- Single and Multiple Download Timings
- Overall User-level Response Times
- Perceived User-level Response Times, Thresholds
- Web Effects
- Page Element Timings (Detailed Timing)

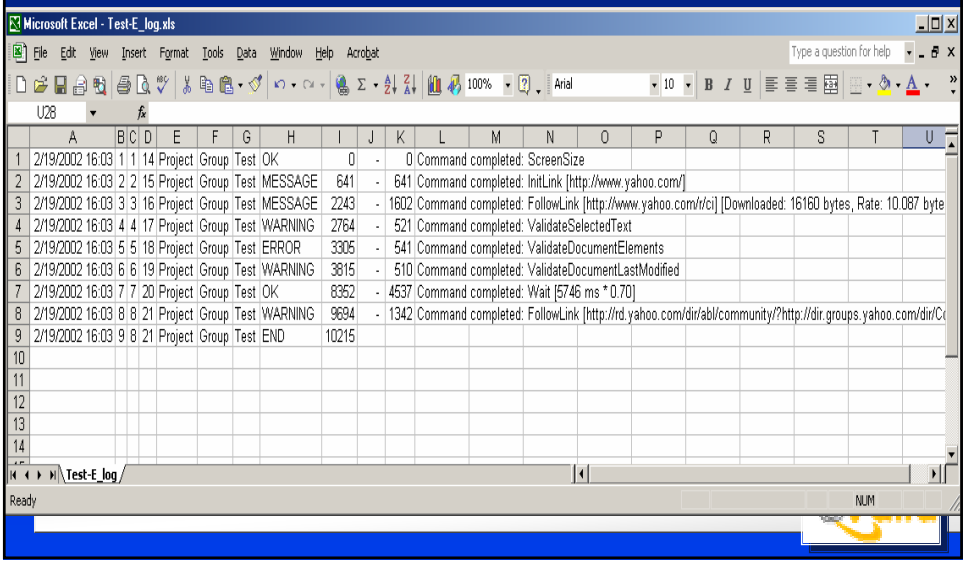
Example Detailed Event Log



Example Event Log in HTML

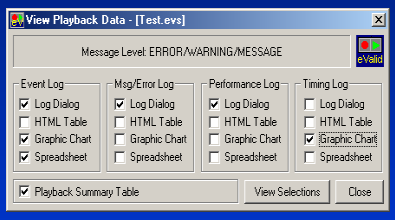


Example Event Log as Spread Sheet

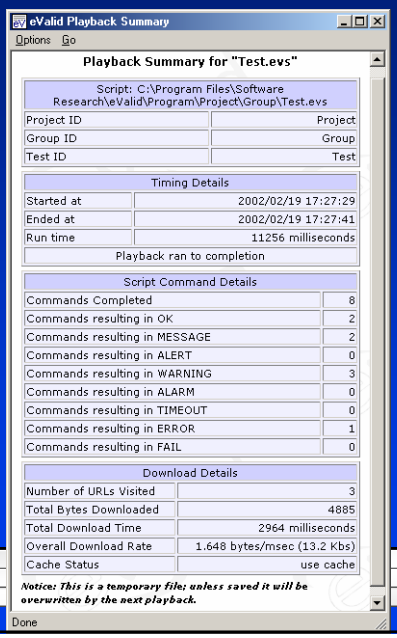


	A	B	C	D	E	F	G	H	I	J	K	L
1	2/19/2002 16:03	1	1	14	Project	Group	Test	OK	0	-	0	Command completed: ScreenSize
2	2/19/2002 16:03	2	2	15	Project	Group	Test	MESSAGE	641	-	641	Command completed: InitLink [http://www.yahoo.com/]
3	2/19/2002 16:03	3	3	16	Project	Group	Test	MESSAGE	2243	-	1602	Command completed: FollowLink [http://www.yahoo.com/r/ci] [Downloaded: 16160 bytes, Rate: 10.087 byte
4	2/19/2002 16:03	4	4	17	Project	Group	Test	WARNING	2764	-	521	Command completed: ValidateSelectedText
5	2/19/2002 16:03	5	5	18	Project	Group	Test	ERROR	3305	-	541	Command completed: ValidateDocumentElements
6	2/19/2002 16:03	6	6	19	Project	Group	Test	WARNING	3815	-	510	Command completed: ValidateDocumentLastModified
7	2/19/2002 16:03	7	7	20	Project	Group	Test	OK	8352	-	4537	Command completed: Wait [5746 ms * 0.70]
8	2/19/2002 16:03	8	8	21	Project	Group	Test	WARNING	9694	-	1342	Command completed: FollowLink [http://rd.yahoo.com/dir/abl/community/?http://dir.groups.yahoo.com/dir/C
9	2/19/2002 16:03	9	8	21	Project	Group	Test	END	10215			
10												
11												
12												
13												
14												

Playback Report Menu



Playback Summary



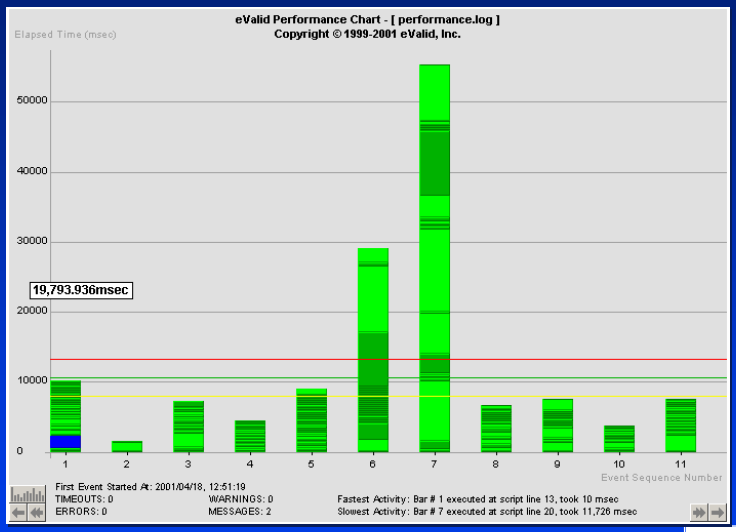
Alernative Playback Modes

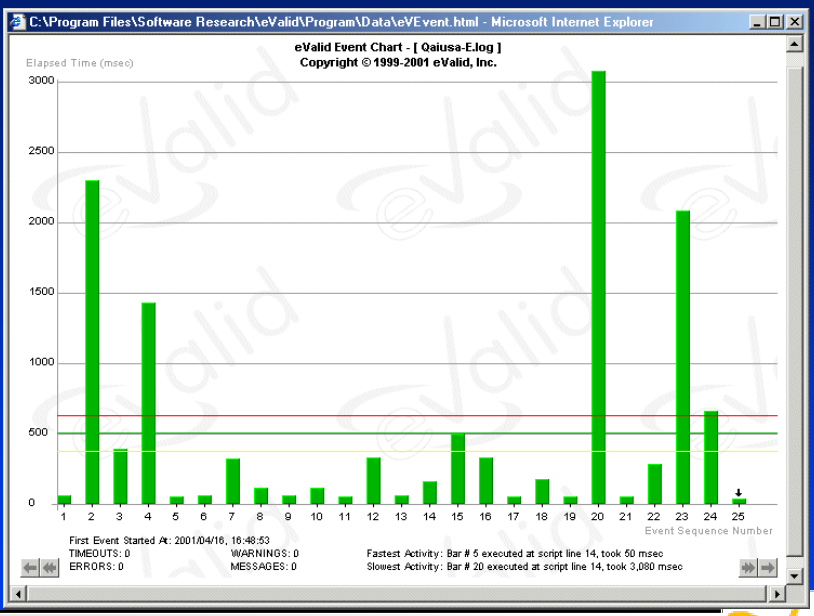
- Play Multiple
- Play Forever
- CallScript/GoScript
- OnErrorCallScript/OnErrorGoScript
- Parameter Passing
- Limitations
- Batch Mode Interface

Detailed Timing Data Collection

- Cache Disabled
- Detailed Timing Enabled
- Resolution 1.0 msec.
- Download Time and Size From Internals
- Base Page
- CSS's
- IMG's
- Final Page Rendering

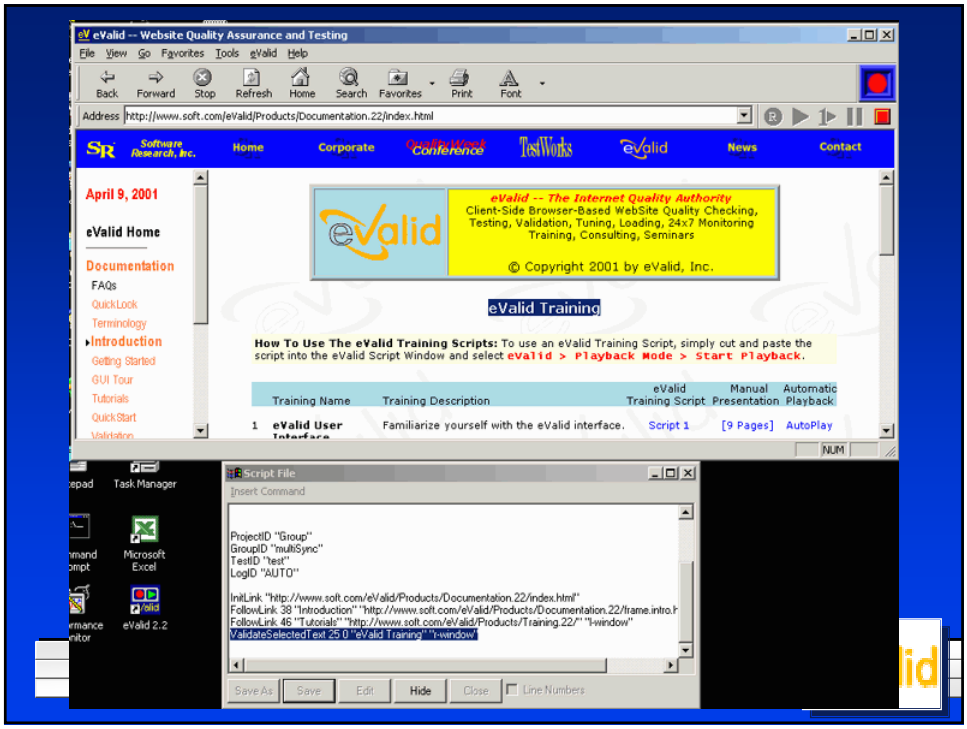
Sample "Stack Chart" for Timing/Tuning





Dynamic Testing: Unsolved Problems

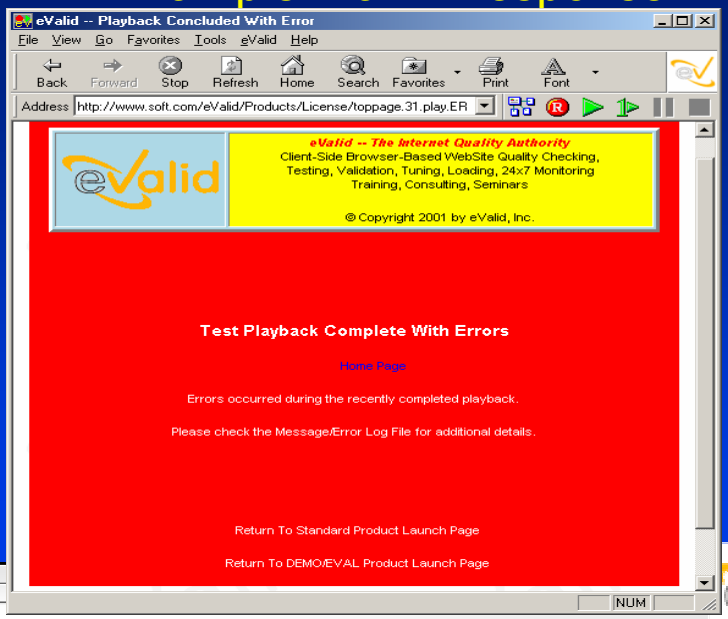
- Repeatability With Drags
- Databases that “Remember” (Initialization)
- Transient Page Effects
- Multi-Media Displays (Except Last Page)
- “Flash” Media Presentations
- Asynchronous Interactions



Alarm/Event Timing

- Overall Timer
  - Total Time
  - Total Byte Counts
- Page Timing
  - Base Page
  - LINKed Files
    - JavaScript (*js)
    - Cascading Style Sheets (*css)
  - Images

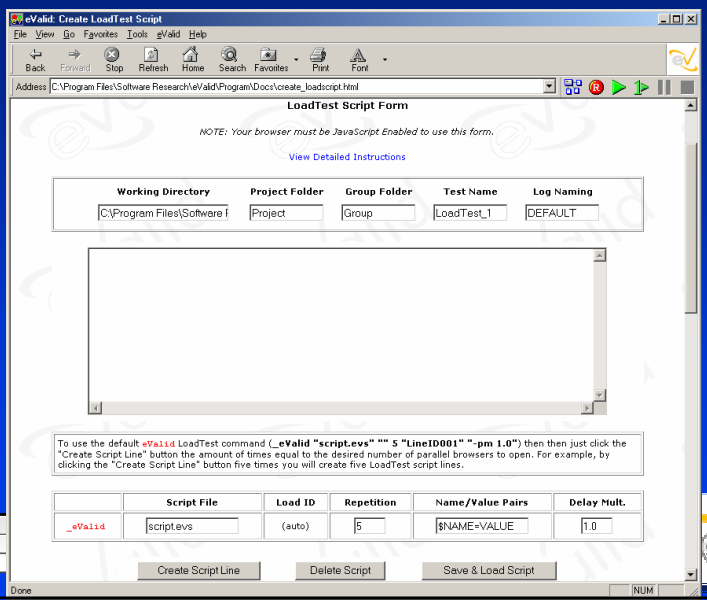
Example Alarm Response



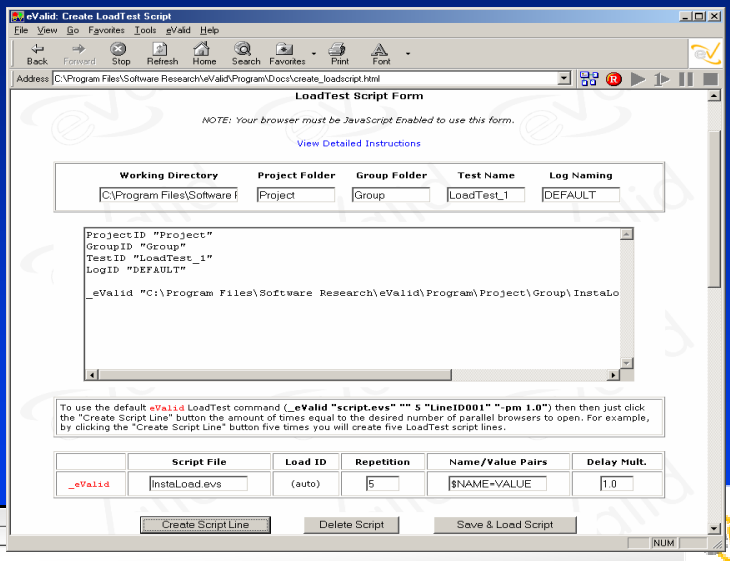
Load and Capacity Checking/Testing

- Load Imposition (Client Side)
- Load Measurement (Server Side)
- User Scenarios (Abstract User)
- Realism
- Bandwidth Issues
- Client Side Machine Capacity

Sample Create LoadTest Setup



Completed LoadTest Page



User Class Definition/Use

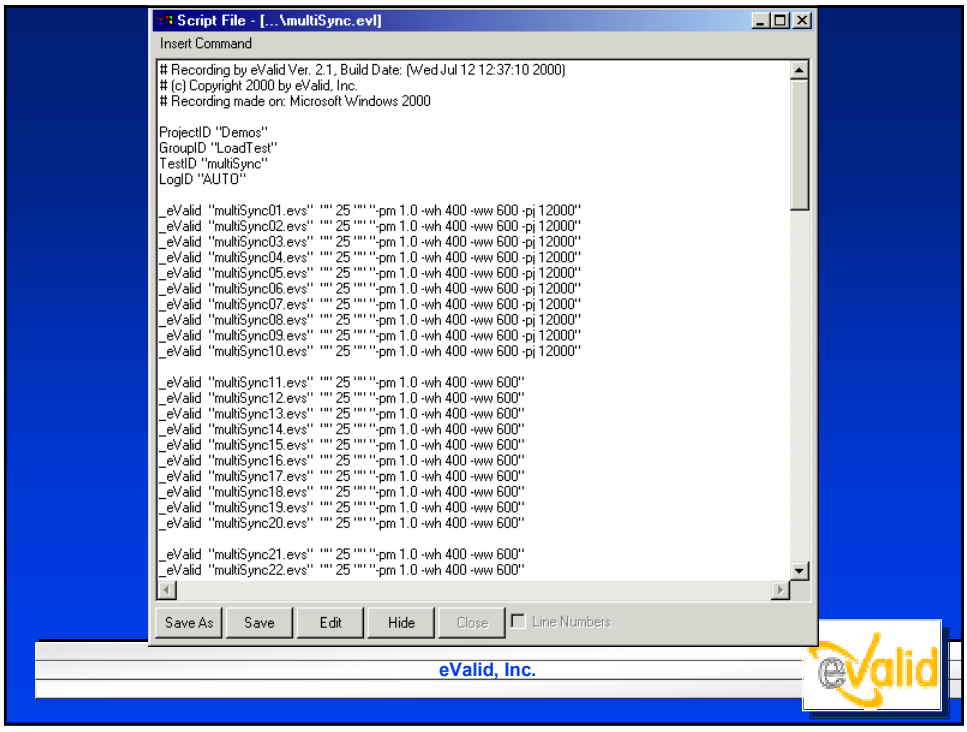
- Artificial “User Class”
- Single *evs file
- Variable Delay Multiplier
- Overall Test Result Timing Aggretation
- No Restrictions on Number of Classes
- No Restrictions on Composition
- Infinite Capability to Emulate Real Users

Machine Capacity Issues

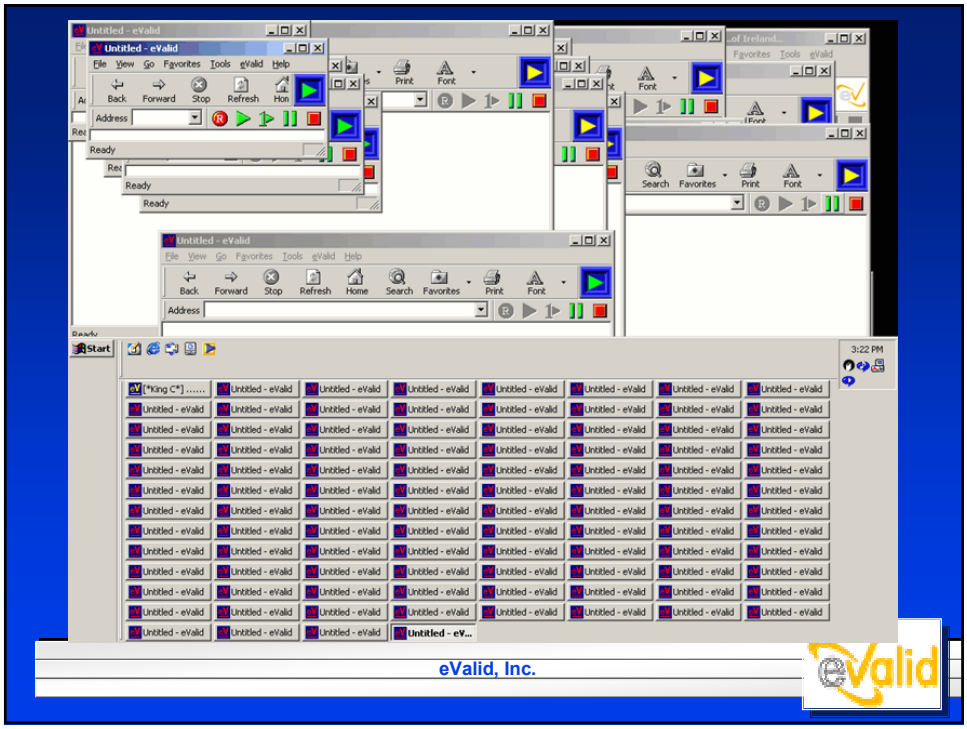
- RAM
- Virtual Memory
- Heap Space
- Machine Bus Speed
- LAN Speed
- Display Driver Memory
- Other

Typical Capacity

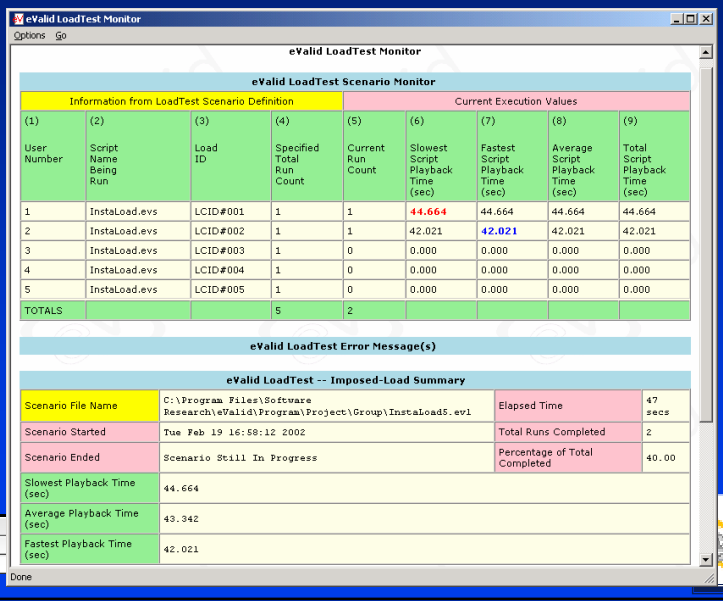
- 500+ MHz P-III
- 512 MB RAM
- 64 MB RAM in Video Card
- Heap Space Adjusted
- Virtual Memory Adjusted
- Latest OS Patches
- Result: 75-125 eValid Instances per Machine
- Multiple-Machine Playback Option



eValid, Inc.



Example LoadTest Interim Report



eValid LoadTest Monitor

Options Go

eValid LoadTest Monitor

eValid LoadTest Scenario Monitor

Information from LoadTest Scenario Definition				Current Execution Values				
(1) User Number	(2) Script Name Being Run	(3) Load ID	(4) Specified Total Run Count	(5) Current Run Count	(6) Slowest Script Playback Time (sec)	(7) Fastest Script Playback Time (sec)	(8) Average Script Playback Time (sec)	(9) Total Script Playback Time (sec)
1	InstaLoad.evs	LCID#001	1	1	44.664	44.664	44.664	44.664
2	InstaLoad.evs	LCID#002	1	1	42.021	42.021	42.021	42.021
3	InstaLoad.evs	LCID#003	1	0	0.000	0.000	0.000	0.000
4	InstaLoad.evs	LCID#004	1	0	0.000	0.000	0.000	0.000
5	InstaLoad.evs	LCID#005	1	0	0.000	0.000	0.000	0.000
TOTALS			5	2				

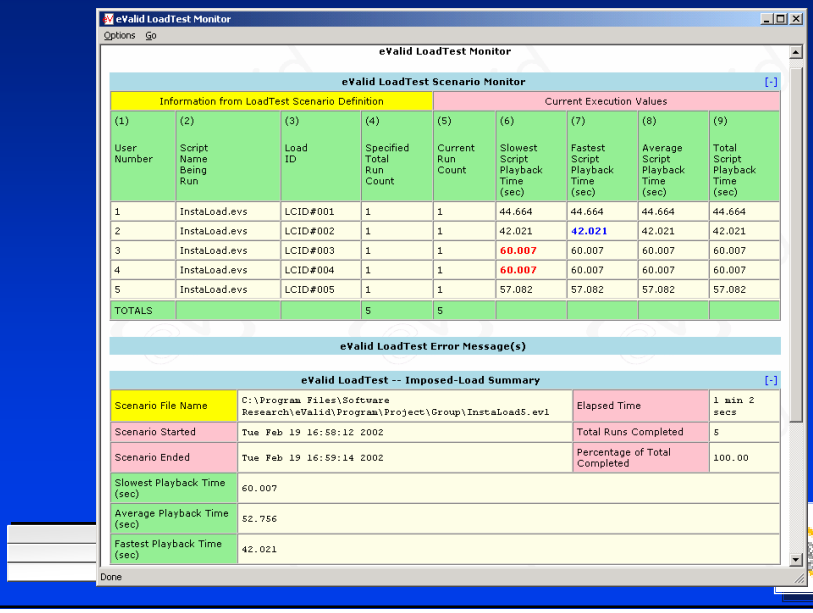
eValid LoadTest Error Message(s)

eValid LoadTest -- Imposed-Load Summary

Scenario File Name	C:\Program Files\Software Research\eValid\Program\Project\Group\InstaLoad5.evl	Elapsed Time	47 secs
Scenario Started	Tue Feb 19 16:58:12 2002	Total Runs Completed	2
Scenario Ended	Scenario Still In Progress	Percentage of Total Completed	40.00
Slowest Playback Time (sec)	44.664		
Average Playback Time (sec)	43.342		
Fastest Playback Time (sec)	42.021		

Done

Example LoadTest Final Report



eValid LoadTest Monitor

Options Go

eValid LoadTest Monitor

eValid LoadTest Scenario Monitor [-]

Information from LoadTest Scenario Definition				Current Execution Values				
(1) User Number	(2) Script Name Being Run	(3) Load ID	(4) Specified Total Run Count	(5) Current Run Count	(6) Slowest Script Playback Time (sec)	(7) Fastest Script Playback Time (sec)	(8) Average Script Playback Time (sec)	(9) Total Script Playback Time (sec)
1	InstaLoad.evs	LCID#001	1	1	44.664	44.664	44.664	44.664
2	InstaLoad.evs	LCID#002	1	1	42.021	42.021	42.021	42.021
3	InstaLoad.evs	LCID#003	1	1	60.007	60.007	60.007	60.007
4	InstaLoad.evs	LCID#004	1	1	60.007	60.007	60.007	60.007
5	InstaLoad.evs	LCID#005	1	1	57.082	57.082	57.082	57.082
TOTALS			5	5				

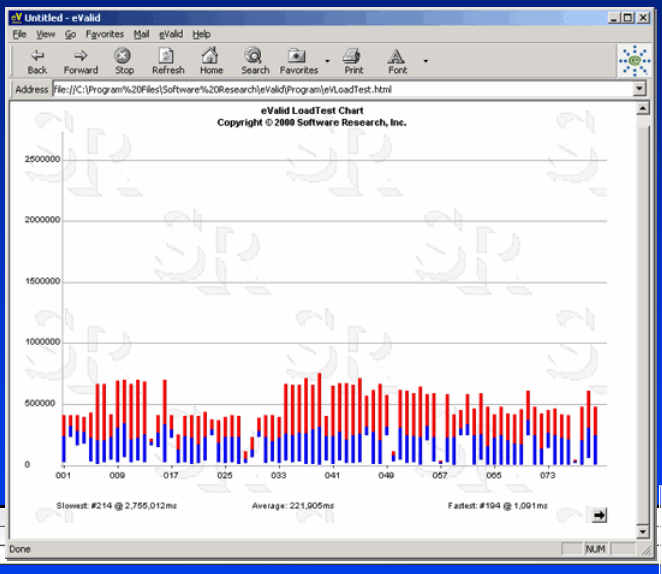
eValid LoadTest Error Message(s)

eValid LoadTest -- Imposed-Load Summary [-]

Scenario File Name	C:\Program Files\Software Research\eValid\Program\Project\Group\InstaLoad5.evl	Elapsed Time	1 min 2 secs
Scenario Started	Tue Feb 19 16:58:12 2002	Total Runs Completed	5
Scenario Ended	Tue Feb 19 16:59:14 2002	Percentage of Total Completed	100.00
Slowest Playback Time (sec)	60.007		
Average Playback Time (sec)	52.756		
Fastest Playback Time (sec)	42.021		

Done

Sample Multi-User Load Scenario Chart



Conclusions & Recommendations

- Nature of WebSite Testing
- Complexity Required
- Realism Requirement

QW2002 Paper 4I1

Giri Vijayaraghavan and Cem Kaner
(Florida Tech Dept of Computer Sciences)

Bugs in Your Shopping Cart: A Taxonomy

Key Points

- Presents an outline of potential problems testers can find in ecommerce sites
- Applies the outline to shopping cart functionality--identifies 300+ specific risks
- Illustrates one common approach to risk-based testing

Presentation Abstract

Imagine being asked to test a website's shopping cart. If you hadn't tested one before, where would you start? What experience would you draw on? Where would you look for more information? Even very experienced testers have blind spots when they try to generate test ideas for an application that they have not tested. This session presents a simple outline that will help you generate test ideas and limit your blind spots. The outline is the result of a year's research on classifying e-commerce related failures and risks. The result has 60 top-level categories and examples of errors (potential issues to test for) under most categories. In many cases, we also link to examples of e-commerce defects that have been publicized in the press.

Using the list, you could pick a category of interest (such as accessibility or software upgrade), read descriptions of several types of problems that fit within that category, and so identify a few issues that would be appropriate to test for in your application. Based on feedback to the authors of Testing Computer Software, we believe that many testers will be able to use this list to identify potential problems that they would otherwise have missed.

We intend the outline to serve similar functions to Kaner / Falk / Nguyen's bug appendix in Testing Computer Software (TCS): help testers generate ideas; help test plan inspectors check a large set of tests for thoroughness and coverage; help testers and other stakeholders identify risks during discussions of prioritizing the testing effort. Kaner expects this to become the successor to the TCS list, and we have structured the outline (e.g. more top-level categories and fewer levels) based on feedback from many TCS readers.

About the Author

Giri Vijayaraghavan, giirii@hotmail.com, www.girivijay.com, is a Master's student in Computer Science at Florida Institute of Technology. This paper summarizes his thesis research on "E-commerce risks and failures.". Giri holds a Bachelors degree in Computer Science and Engineering and has worked as Research Assistant in various funded projects of IBM and Texas Instruments and as an intern at Fidelity Investments-eBusiness.

Cem Kaner, J.D., Ph.D., kaner@kaner.com, www.kaner.com, is Professor of Computer Sciences at Florida Institute of Technology. He is senior author of Testing Computer Software, of Lessons Learned in Software Testing, and of Bad Software: What To Do When Software Fails.

Bugs in Your Shopping Cart: A Taxonomy

Giri Vijayaraghavan
and Cem Kaner
International Software Quality Week
September, 2002

Challenges of E-Commerce Testing

- ***Global market:*** Many languages, currencies and cultural norms, global shipping, different legal systems (conflicting advertising / sales laws), etc.
- ***Multiple platforms:*** End-to-end, a single transaction might involve several systems on several continents, running on different processor and O/S families.
- ***Multiple clients:*** Client operating systems, browsers, peripherals, and ever-so-many different utilities.
- ***Multiple customer profiles:*** The user profile varies greatly by gender, age, language, wealth, etc.
- ***Multiple partner apps.*** The application under test relies on services provided by third parties, which are totally outside the application developer's control. For example, most of the credit card processing steps are done by the credit card company's system.

Copyright © Cem Kaner, 2000-2002.

Challenges of E-Commerce Testing

- ***Components from strangers. T***hird party components allow programmers to snap together large programs (or make big changes) quickly. Just because the components were written by someone else doesn't mean they're reliable.
- ***Multiple clocks, no standard time.*** There is no shared reference clock and so events can occur, or appear to occur, out of sequence.
- ***Random delays complicate timing.*** Different services will be provided at different speeds (and end user client(s) will receive and respond at varying speeds). Race conditions and other time-related problems more likely.
- ***Rapid change.*** Rapid, iterative development. Constantly changing UI.

Challenges of E-Commerce Testing

Most people coming to e-commerce testing will face a steep, multi-dimensional learning curve.

The Bug Taxonomy

- An outline that categorizes and lists a large number of potential bugs.
- The tester who uses the taxonomy can sample from the list, selecting a potential problem for analysis.
  - The tester's question is whether the software under test could have a bug analogous to the one from the list.
  - If so, the next question is what type of test would expose this type of bug.
- A good taxonomy
  - Has enough detail for a motivated, intelligent newcomer to the area to be able to understand it.
  - Is broad enough to raise at least a few issues new to someone with moderate experience in the area.
- A good taxonomy is a useful tool for informing a tester who is new to the area about the types of problems to be tested for.

Copyright © Cem Kaner, 2000-2002.

Using A Bug Taxonomy

- Kaner published an extensive taxonomy (over 400 bugs) in 1988 (updated by Kaner, Falk & Nguyen, 1993) and was frequently contacted by readers who used the list. Based on that feedback, we expect the following uses of the Shopping Cart list:
  - Tester unfamiliar with an aspect of the program looks for potential failure modes in the risk list, then explores the program looking for those types of failures.
  - Tester who has run out of good test ideas looks for plausible failure modes in the risk list, then creates tests looking for those types of failures.

Copyright © Cem Kaner, 2000-2002.

Using A Bug Taxonomy

- Test manager, training new testers, walks with the group through selected examples from the risk list in order to convey to the trainees the breadth of their work.
- Tester, auditing a test plan, samples from the taxonomy, selects plausible failures, then checks the test plan to determine what tests (if any) could have detected the failure. If none, the test plan has a hole.

Copyright © Cem Kaner, 2000-2002.

Relation to FMEA

- Failure Mode and Effects Analysis involves a multi-disciplinary evaluation of a product.
- The analysts identify the functions of the product, the ways they could fail (the failure modes), the potential impacts and severity of those failures, likelihood of the failures and potential causes of them.
- http://www.fmeca.com/ffmethod/fmeaproc.htm
- Given this information, the analysts (or management) prioritize their research.
- This is strongly analogous to *risk-based testing*, in which tests are designed to check whether certain potential bugs are actually in the product, and are then prioritized if there isn't enough time to run them all.

Copyright © Cem Kaner, 2000-2002.

Categorizing the Risks

We put together a list of about 60 categories of potential bugs, such as performance, understandability, accessability. Within a given category, we list

- Potential failures;
- Information about causes of the failures, if we have that information;
- References; and
- Examples of actual failures within the category's theme, that were reported in the trade press.

Categorizing the Risks--Example

Cache Server failure

Cache servers are used as intermediaries for web requests and retains previously requested copies of resources. The use of a cache server is to handle common requests locally and improve site performance by better speed up and reduced overhead on the web servers.

The common issues discussed here are can be found in more detail in "Known HTTP Proxy/Caching Problems [1]"

- Cache may return an outdated shopping cart document if the header is misrepresented or last modified date is omitted
- If shopping cart content is dynamic in nature, then cache server will not be able to serve new content
- If caching proxy server fails during shopping session, sometimes the browser fails to bypass server and may need to be reconfigured and shopping cart state may be lost

Categorizing the Risks--Example

Cache Server failure (continued)

- If shopping cart uses any form of encoded response, the proxy might cache it and send it to a non-encoding capable client
- Sensitive shopping cart content may get cached by Interception proxies that break client cache directives like "No cache" or "Must revalidate"
- The cache server may end up blocking some methods used by the shopping cart software, because the method contained in the request is unknown to the proxy so instead it generates the default HTTP 501 Error as a response.
- Shopping carts that use IP address to track state of the cart, may fail because Interception proxies at ISP level may alter client's IP to that of the proxy itself
- A caching proxy mesh might break HTTP content serialization resulting in the user getting older content when the shopping cart page loads

Categorizing the Risks--Example

Cache Server failure (continued)

Examples of related bugs and other known issues

- Is Web caching bad for the Internet?
  http://www.cnn.com/2000/TECH/computing/04/18/web.cache.idg/
- Known HTTP Proxy/Caching Problems
  http://www.wrec.org/Drafts/draft-ietf-wrec-known-prob-03.txt

How We Developed the List

- Brainstormed a first draft top-level list (we thank James Bach for his assistance)
- Searched electronic bug databases (such as bugnet.com and cnet.com) for examples
- Searched open source software for bug databases for specific products. These gave us examples and indications of the types of bugs possible
- Brainstormed additional types of problems
- Circulated the list and the outline for peer review

What's Coming Soon

- Current taxonomy is a subset of the full set for Shopping Cart
  - We have rougher but extensive draft material for about 20 more categories. This is available on request to people who have a legitimate, *immediate* need.
  - A more polished version of these, and a few others, will be in Giri Vijayaraghavan's M.Sc. thesis, which will probably be completed in December 2002. Check for it at www.testingeducation.org in January 2003.

References

- S. Amland, "Risk-Based Testing and Metrics," *EuroSTAR 99, Barcelona, Spain*, 1999.
- C. Dilley, "Known HTTP Proxy/Caching Problems," May 14, 2001
- A. Biraj Rath, NIIC (USA), Inc., P. M. Raj Nath, Microsoft Corporation, P. M. Mukesh Agarwal, Microsoft Corporation, P. M. Jas Lamba, NIIC (USA), Inc., R. George Gianopoulos, Microsoft Corporation, and C. Laura Hargrave, Microsoft Corporation, "Testing Process," July 2000
- Kaner, Falk & Nguyen, 1993 “Testing Computer Software” 2nd Edition
- FMECA, "FMEA Methodology.” http://www.fmeca.com/ffmethod/definiti.htm

Copyright © Cem Kaner, 2000-2002.

Bugs in your shopping cart: A Taxonomy

Giri Vijayaraghavan
Cem Kaner
Florida Institute of Technology

Presented at the International Software Quality Week, San Francisco, CA, September 2002.

Introduction

Imagine being asked to test a web site's shopping cart. If you hadn't tested one before, where would you start? What experience would you draw on? Where would you look for more information? Even very experienced testers have blind spots when they try to generate test ideas for an application that they have not tested. This paper presents a simple outline that will help you generate test ideas and limit your blind spots. The outline is the result of a year's research on classifying e-commerce-related failures and risks. The result has 60 top-level categories and examples of errors (potential issues to test for) under most categories. In many cases, we also link to examples of e-commerce defects that have been publicized in the press.

Goals of this outline

Using the list, you could pick a category of interest (such as accessibility or software upgrade), read descriptions of several types of problems that fit within that category, and so identify a few issues that would be appropriate to test for in your application. Based on feedback to the authors of *Testing Computer Software*, we believe that many testers will be able to use this list to identify potential problems that they would otherwise have missed.

We intend the outline to serve similar functions to Kaner / Falk / Nguyen's bug appendix in *Testing Computer Software* (TCS):

- Help testers generate ideas;
- Help test plan inspectors check a set of tests for thoroughness and coverage;
- Help testers and other stakeholders identify risks during discussions of prioritizing the testing effort.

Kaner expects this to become the successor to the TCS list, and we have structured the outline (e.g. more top-level categories and fewer levels) based on feedback from many TCS readers

The nature of e-commerce testing

“E-commerce testing involves testing high value, high risk, and high performance business critical systems”[1]. The cost of a failure in an e-commerce system is hence formidable and generally the testing process involves considerable effort in designing and integrating an effective risk management process into the testing to avoid or minimize the cost of failures.

The distributed nature of the underlying e-commerce applications also adds to the complexity in developing a comprehensive risk based testing approach because it makes it difficult to exactly pinpoint where the risk of failure exists. Which application failed? Where did it fail? And when did it fail? E-commerce testing only involves testing the individual underlying components and web applications for potential risks and failures, along with the e-commerce "web site" as a whole.

A bug in an underlying vendor specific component such as a Web server, transaction server or database system may be missed or ignored because the focus of testing might be on the site as a whole or on the code specifically written for the web application. Unfortunately, an inconspicuous bug in a third-party component might cause a serious failure when the system is in production, leading to bad publicity, and lost reputation, customer confidence, and revenue.

The problem of transitioning to e-commerce testing from other platforms

E-commerce testing involves numerous challenges, such as new or unfamiliar technologies, or technology-driven business processes or logic that the tester doesn't understand how to test. Learning the necessary information or skills may be complex but can be simplified with strategies for quickly generating test ideas or test plans that address potential risks. We believe that an outline of well-researched potential failures can help flatten the steep learning curve involved with e-commerce testing.

It may be worthwhile for any tester who is moving into the e-commerce-testing arena to remember the following issues:

- ***"Test Global and Test Distributed":*** E-commerce systems are truly global in spirit and structure. The different underlying systems may be on different continents, but they appear to integrate seamlessly over large, distributed and non-homogenous business networks and other communication channels. Risk analysis and test planning should allow for potential problems caused, for example, by a side effect of a software upgrade on a transaction server physically located in Holland on a transaction taking place between a customer in Korea and a retailer in the US.

- *3 Ms where 1st M:* ***Multiple Platform*** *2nd M:* ***Multiple Clients (Browsers)*** *3rd M:* ***Multiple customer profiles****:* This will be a considerable change for a tester who comes to e-commerce testing from other traditional testing fields. In the stand-alone or older client servers, the user's platform, the client type and the nature of the user might be well known to the developers and testers. E-commerce systems involve more uncharted territory, with greater diversity of operating systems, browsers and other system software and hardware.
  The user profile varies greatly in terms of age, gender, taste and usage-testing business software has become as configurationally complex as testing

consumer software, but with more serious consequences in the event of failure.

- ***Learning to count in "web years"****:* Changes and updates are inherent in E-commerce sites. Content and target platforms change quickly, without much time for planning and regression testing of each change. This can be challenging for testers coming from traditional business applications. The e-commerce tester must learn to generate effective sets of test ideas rapidly.

- ***The risk of testing in a "not-so-representative test environment"***
  In an e-commerce world, creating a completely representative test environment is often impossible. With a limit on how much can be actually simulated in a test lab, there is real risk of not knowing how the application will behave in some environments in the field.

How to use this Outline?

- ***Generate new test ideas:***
  The outline provides about 60 top-level categories with examples of errors under each one of them. The categorized outline inspires test idea generation if the tester considers the function-under-test, and then considers how the function would fail with respect to one of the categories.
- ***Use the test ideas here, for applicable projects***
  This paper is a subset of a broader project that will be published in Giri Vijayaraghavan's Master's thesis. You will probably be able to find that thesis posted on the Net at www.testingeducation.org and www.girivijay.com in January or February 2003. The thesis fills in details for more categories of shopping cart problems and less extensively considers failure modes for some other functions. We use the shopping cart as the example function because it has rich functionality (and thus many different failure modes). We use an example function to make the application of the categories more concrete. The same types of problems will show up in many other types of e-commerce application functions. Vijayaraghavan's taxonomy is not exhaustive. We think the 60 top-level categories are sufficient, but below that level, you'll be able to add plenty of your own examples if you use the outline. We recommend that you download Vijayaraghavan's thesis and customize it as you use it on different projects. The more tailored it is to your company's applications, the more long-term value it will have for you. If you do this, we ask that you send us any non-confidential additions that you've made, so that we can add them to a master list.
- ***Audit test plans by inspecting for tests for potential errors***
  It's difficult to find the blind spots in a long test plan. There is so much detail available that it's hard to see what's missing. A list of potential failure modes provides you with and independent cross-check. To audit

the plan, consider each category in turn. Ask yourself whether failures within that category are *possible* in the software under test, and if so, what they might look like. Pick two or three possible failures and then check the test plan. Does it have tests that would catch these problems? If not, you've found a hole in the plan.

- ***As teaching material to assist in the training of testers who are new to e-commerce testing.***

  The clear structure of the outline, the detailed list of possible risks and failures, and the concrete examples can help new testers broaden their understanding of the range of problems open to their discovery. You can also use them as anchoring points for discussions. For example, some test groups set up a weekly lunch meeting to improve their knowledge or skills. One of these lunches, you might talk about how to recognize accessibility errors. Another day, you might talk about how to recognize update-installation side effects. This outline can help you focus those discussions.

- ***As a presentation tool for explaining to managers the different types of failures that can occur in an e-commerce site.***

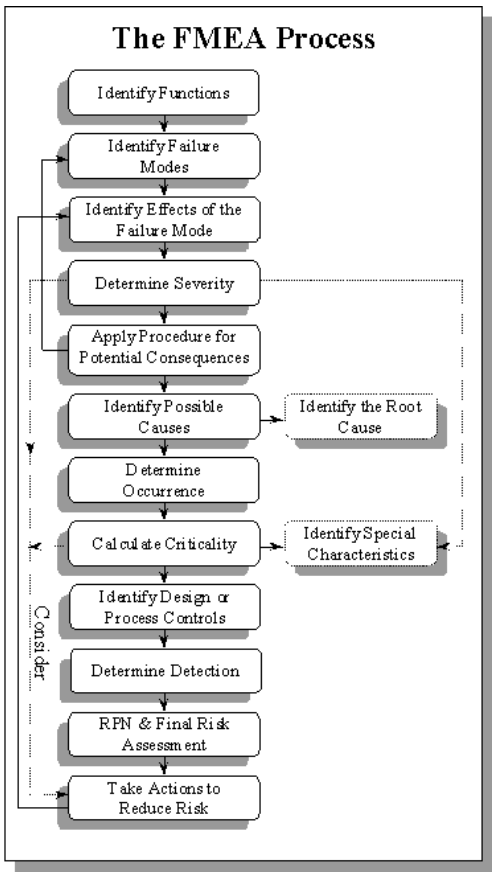
  Managers don't necessarily understand the breadth of the scope of risks involved in e-commerce projects. Use this outline to develop a list of the types of issues that can be tested for. Cross-reference to published examples of errors that have shown up in the trade press (use ours or find your own). A well-organized presentation based on demonstrable risks can go a long way toward getting you adequate funding.

Failure mode and effects analysis (FMEA)

This outline fits within the tradition of failure mode and effects analysis (FMEA). "FMEA has been widely used in the automobile, medical equipment, and aviation industries to focus development teams on potential failures and help them decide which risk are the most important to mitigate and what actions might be taken to mitigate them"[2].

Typically a FMEA process starts with identifying functions and identifying failure modes associated with the functions.

A typical FMEA process looks like fig 1.0:



Source: http://www.fmeca.com/ffmethod/fmeaproc.htm

We follow a similar approach in software: risk-based software test management involves identifying core system functions, identifying the different risks or failure modes associated with the functions, determining their severity, sometimes calculating their criticality and then prioritizing the risks to be addressed in the testing approach (Stale Amland, Risk-Based Testing and Metrics, EuroSTAR 99, Barcelona, Spain, 1999 [3]) FMEA differs from the risk-based test management approach in an important way--the FMEA approach often involves more careful attention to the fine details of the potential failure. In this respect, FMEA is more like the risk-based test design tradition, in which we work out the design details of tests based on the failure we are trying to detect (Cem Kaner, Black Box Software Testing course notes, available soon at www.testingeducation.org.)

If FMEA were applied to an e-commerce site, it would involve:

Identifying the functions

Identifying functions and then generating the different function modes and failure modes is a tedious and time-consuming process, especially when we try to analyze a complex system such as a shopping cart driven e-commerce site. Decomposing an e-commerce site into constituent functions is further complicated by inter-dependencies between components and the overlapping architectures of e-commerce sites. My Master's thesis (Vijayaraghavan, in preparation) analyzes *User-Authentication (Login/Logout)*, *Search*, and *Shopping*

Cart as a few of the functions in a shopping-cart-based, business-to-consumer e-commerce site.

Identifying failure modes

One of our main goals for the outline (and the thesis) is for it to serve as a brainstorming tool to help you generate the different failure modes associated with functions under test.

It's difficult to generate a comprehensive list of failure modes for a function for a variety of reasons:

- Lack of knowledge about all related and existing software bugs that have caused failures in the past or possess the potential to cause a failure in the future.
- Lack of knowledge about historical data on past failures and the causes behind them.
- Lack of an organized '"Bug Taxonomy" that can remind the tester of patterns of potential failure/risk.

The sample list of "shopping cart" failure modes, which was generated using the outline provides references to known software bugs in e-commerce systems, provides historical data of past failures by citing issues publicized in the press.

The sample 'function': An E-commerce Shopping cart

An Introduction

The familiar metaphor of a shopping cart that is present in many e-commerce sites has an important function of keeping track of the user's state while he/she is "shopping". A simple cart may just maintain a list of items that the user places in it and maintains state until the user finishes shopping and exits the system (by closing the browser). Complex and advanced carts have more sophisticated functionality such as real-time credit card processing and real-time order tracking.

As e-commerce sites grow in size and popularity, they tend to add more and more features to their shopping carts. Shopping carts have grown from simple state tracking functions to a highly sophisticated and creative piece of software offering a flexible range of user options.

The diversity, the creative imagination and technological innovation that have gone into the design of these next-generation-shopping carts, make them fascinating and challenging to test. Some issues that you may want to consider before testing a shopping cart are:

- The same issue of testing under "web years" that we discussed about general e-commerce testing applies here. Hence testers will have to reckon with testing under reduced time and sudden spikes in workload.
- A shopping cart may not scale. It may work well for 100 users but not for 1000. Testers should estimate performance standards early in testing.
- They are prolific and vary greatly in terms of design, size, complexity, and underlying technology. Hence no standard best practices exist that can

provide a single-point reference on how to efficiently and comprehensively test a shopping cart.

- Because of the rapid evolution and change in their design and features, it is common for testers to encounter legacy-shopping carts built with outdated technology.
- In contrast, the rapid changes in design and functionality might be a challenge to the less tech-savvy tester, as they add more learning pressure on the tester.

We thank Ms. Karen Johnson for sharing her experiences on testing a shopping cart and for her inputs on some of the issues/bugs mentioned in this paper.

Different types of shopping carts

Testing each of the above shopping carts can be very different because of the difference in the way they have been built and hosted. Except for some **in-house shopping carts**, shopping carts tend to have a large number of 3rd party components, which are sometimes beyond the scope of the testing group.

3rd party built and hosted shopping carts are generally a cheaper alternative. They are employed by small-scale e-commerce sites with fewer staff to design and maintain the system. Sometimes the 3rd party host may be the same as your site-hosting provider. But 3rd party hosting wrenches much of the control away from the tester, since the bulk of the components are not in the tester's domain. But the positive side of this type of shopping cart is that the cart is smaller, less complex and generally easier to test.

Out-of-the-box shopping **carts** are customizable, pre-fabricated, and ready for deployment. Installation, customization, and configuration bugs are some of the common types of issues that testers will encounter when testing shopping carts of this genre. Though many serious security holes have been caught due to bad settings and bad configurations in out-of-the-box carts, much of the deeper level of testing becomes the responsibility of the vendor who originally developed the cart software.

And finally there are many ***free shopping cart scripts*** available for download in CGI or ASP script sites. Very simple and small-scale e-commerce sites tend to use these carts. From the tester's perspective, the risk is in the script code. While some of these scripts are well done, others are hastily developed by amateurs.

The Outline

The Structure of the outline:

This introductory material explains the structure of the outline, which is the core of this paper. The outline presented in this paper is a top-level list of about 60 risk categories and the *list of shopping cart failures* is a long and detailed list of different failures that can occur in a shopping cart under each category of the outline. The shopping cart failure list is an example of how the outline can be used to generate test ideas.

Both the Outline and the list of shopping cart failures are work in progress. A more complete list will be available in the Master's thesis and later publications.

The list of shopping cart failures has been categorized under the 60+ categories shown by the outline. Some categories have relevant sub-categories for ease of understanding and structure. We provide definitions and cite relevant literature for each category. The categories and sub-categories hold detailed lists of risks, errors, bugs, and failures that a tester might find when testing a shopping cart. Most of the categories also have examples of published bugs to illustrate the risks. Some categories also have thoughts on methods to test for this risk. More details on the qualitative categories can found in ISO 9126 [4]

Thoughts on how to use this list to generate test ideas for non-shopping cart applications

We developed the list of shopping cart failures to study the use of the outline as a test idea generator. We think the list is a sufficiently broad and well-researched collection that it can be used as a starting point for testing other applications.
For example, the list contains a generalized collection of risks due to database failures (Database media failure, database statement failure, database instance failure and database-user process failure). This collection of database failures is relevant to all projects that use databases. Failures in web server, database server or cache server are not specific to just to a shopping cart driven e-commerce sites but to all web sites that use them. A tester testing a travel/airline ticket web site or an auction web site may find the list as relevant to his testing as a tester working on a shopping cart web site.

Outline: Table Format

Performance
Reliability
Software Upgrade
UI Risks/Errors/Usability
Maintainability
Conformance
Adaptability
Stability
Analyzability
Operability
Understandability
Fault tolerance
Inter-operability
Accuracy
Internationalizability
Visibility
Accessibility
Localizability
Suitability
Compliance
Maturity

Recoverability
Learnability
Efficiency
Changeability
Testability
Installability
Replaceability
Insufficient Capacity Planning
Backend Connectivity failure
Bandwidth Sufficiency risks (During Peak)
Human Error
Calculation/Computation
Navigation Flow
Process Flow
Data flow
Transaction flow
User-System Interaction
Third-party Software failure
Database-Statement Failure
Database-Instance Failure
Database-User-Process Failure
Database-Media Failure
Memory failures/Memory Leaks
I/O Data Type Conflicts/Data Mismatch
Network risks
Functionality
Error Message /Exception Handling
Server-Side-Hardware Failure
Client-side-hardware failure
Third-party-hardware failure
Browser problems
Document Confidentiality
System Security
Client Privacy
Database Server failure
Cache Server Failure
Web-server failure
Transaction Server failure
ISP problems

List of shopping cart failures

Database Server failure

A database server is software that manages data in a database. It updates, deletes, adds changes, and protects data [Network Magazine]. Database servers provide both the access

control and concurrency control. So while testing a shopping cart, if you find empty catalogs, unpopulated data fields and authentication problems, then you should check the database server. Some of the issues discussed here are based upon the discussion in the paper “Managing Database Server Performance within an Electronic Commerce Framework” [5] Here are different ways a shopping cart can fail, when the database server goes wrong:

- Inability to load or populate data in the product catalog.
- Inability to load or populate order data in the shopping cart.
- Inability to load or populate customer profiles.
- DB server failure may lead to a complete failure of data retrieval in the system since DB server manages/serves the data in the system.
- Increase in response time during "browse" transaction. Browse transaction generates high frequency, random, sequence of queries on the database server.
- The "shopping cart" transaction fails to update/load the billing details/price in the basket. Shopping cart transaction places medium weight, high frequency read/write operation.
- Increase in response time to load/update billing details, price lists and total in the basket.
- Failure or delay to commit the customer order to the database in the "Buy transaction".
- User-registration failure, unable to execute read-write process during user registration.
- Search process fails to execute since DB server failure may cause failure of read-only search process to fail.
- Increase in "search" time may indicate performance problems in Database server.

Cache Server failure

Cache servers are used as intermediaries for web requests and retains previously requested copies of resources. The use of a cache server is to handle common requests locally and improve site performance by better speed up and reduced overhead on the web servers.

The common issues discussed here are can be found in more detail in “Known HTTP Proxy/Caching Problems [7]”

- Cache may return an outdated shopping cart document if the header is misrepresented or last modified date is omitted
- If shopping cart content is dynamic in nature, then cache server will not be able to serve new content
- If caching proxy server fails during shopping session, sometimes the browser fails to bypass server and may need to be reconfigured and shopping cart state may be lost
- If shopping cart uses any form of encoded response, the proxy might cache it and send it to a non-encoding capable client
- Sensitive shopping cart content may get cached by Interception proxies that break client cache directives like "No cache" or "Must revalidate"
- The cache server may end up blocking some methods used by the shopping cart software, because the method contained in the request is unknown to the proxy so instead it generates the default HTTP 501 Error as a response.
- Shopping carts that use IP address to track state of the cart, may fail because Interception proxies at ISP level may alter client's IP to that of the proxy itself
- A caching proxy mesh might break HTTP content serialization resulting in the user getting older content when the shopping cart page loads

Examples of related bugs and other known issues
Is Web caching bad for the Internet?
http://www.cnn.com/2000/TECH/computing/04/18/web.cache.idg/
Known HTTP Proxy/Caching Problems
http://www.wrec.org/Drafts/draft-ietf-wrec-known-prob-03.txt

DATABASES

The approach used here to classify the different databases can be found in more detail in "Oracle9i Database Administration: Recover Databases [6]", According to the definitions provided in it,

Database statement-failure: "Statement failure occurs when there is a logical failure in the handling of a statement".

Database-Instance Failure:

"Instance failure occurs when a problem prevents a database instance from continuing to run. An instance failure can result from a hardware problem, such as a power outage, or a software problem, such as an operating system crash. Instance failure also results when you issue a SHUTDOWN ABORT or STARTUP FORCE statement"

Database-User-Process Failure:

"A process failure is a failure in a user, server, or background process of a database instance such as an abnormal disconnect or process termination"

Database-Media Failure:

"An error can occur when trying to write or read an file on disk that is required to operate a database. This occurrence is called media failure because there is a physical problem reading or writing to files on the storage medium".

Database statement-failure

- User may be attempting to issue a statement referencing a table in the shopping cart that does not exist
- A user may be attempting to issue a statement referencing a table in the product catalog, user database for which they have do not have permission to access
- Flawed statement or Flawed query used by the web developer may make shopping cart data inaccessible to the user.
- Inability of a user to submit information, which is to be stored into a database because of inadequate table space allocation for the user/operation
- Flawed statement/query may lead to in-correct addition/deletion of items in the basket
- Incorrect access of tables may lead to incorrect computations/calculations of shipping/taxes
- Failure to clearly specify required fields, optional fields and edit permissions may lead to problems when data is being written back into the tables.
- Inefficient queries on the shopping cart tables

Database-Instance Failure

- The number of simultaneous connections allowed is less that the maximum number required by the system for shopping cart transactions
- Power outage when shopping cart database is being accessed and no recovery routines exist.
- Check for issues where using a product database and multiple item forms together would cause an error

Examples of related bugs and other known issues

Database Glitches at Walmart.com
http://www.internetnews.com/ec-news/article.php/4_739221

Database-User-Process Failure

- Risk of user being unable to return to shopping cart after navigating away from the page since contents of cart not been saved
- User unable to add/delete/modify contents of the basket
- Client PC hangs during shopping cart transaction and user state not saved/retrievable
- Failure of the shopping cart database to rollback process on detection of user process failure

Database-Media Failure

- Not enough memory on the system on which the shopping cart database resides
- Disk failures/Hard drive crashes, and other irreversible media corruption of the shopping cart database may cause complete loss of data
- Corruption of shopping cart Database backup

Error Messages/ Exception Handling

Provided below is a detailed list of errors that you might encounter in an e-commerce site with a shopping cart and it might be useful to test for appropriate error messages. Testers should find this list useful to test a shopping cart site for error handling and check if the error handler handles these common errors. It has also been sub-categorized for ease of use on the basis of the kind of errors the system has been designed to handle.

- **Quantity error-handling**
  - Ability to erratically checkout an empty shopping cart and check if error message is displayed.
  - Ability to add negative numbers to the quantity field. Check for appropriate error handling
  - Accepts decimal entries for quantity but ignores the decimal point and either accepts the first or last digit alone, so 7.0 may be interpreted as 7 or 0 and no error handling exists to prompt or correct the error.
  - Accepts decimal entries for quantity but again ignores decimal point and accepts the quantity comprising of both the digits, so 7.0 may be interpreted as 70! And no error message to prompt or correct the error.
  - Quantity field not size-constrained and no error message to prompt user of acceptable values or data range.
  - An over-sensitive error handler may not let a user increase/decrease/edit the quantity field at an editable stage and may risk rendering the data entry final!
- **Information error-handling**
  - Forms requiring registration information, shipping address information, billing address information employ script based entry validation to validate entries but sometimes the scope of the script exceeds its limit and pops an error message for entry fields outside the limit of the script or optional fields.
  - Some address fields contain two parts, address 1 and address 2 in order to accommodate lengthy addresses. But some error handlers count both the fields as compulsory and pop error messages to users who leave address 2 empty (because their address is short and fits right into the first one!)
  - Long addresses may get clipped and no error message or routine exists to warn the user about the size constraint.
  - Lack of error routine to check for valid US zip code in the address section.
  - Check for trigger-happy error messages that sometimes pop up to a non-US shopper's dismay, to validate an empty US zip code.
  - Error message pops up informing the user of incomplete information entry but does not highlight the field where error exists.
- **Interaction and Transaction error-handling**
  - "*An Internal Server error*" may be displayed without any fix to the user, sometimes this error, which may be due to a missing term in the URL, can be fixed by appending a term, like say &reference to the address.
  - **"***Inventory module error message***"** may be displayed with no explanation to the user; sometimes this error occurs when two users access the last

item and the inventory control tries to update the order so that only one user gets access to the item.
    - If you encounter an "*ODBC error* message" when you click Checkout, you may be missing your "session ID", error handler should be enabled to handle this common error or should provide help to customers with simple fix to this errors.
    - *"Timeout Error messages"* If any routine exists to check the time of inactivity and auto times out any shopping cart, such existence of timeout routines should be communicated to the user beforehand.
- **Payment/ Credit-Card error-handling**
    - An incorrect expiration date (be sure to use a two-digit year, such as "02") and supporting error message to prompt the user.
    - *"Invalid Card Number error message"*, if the card processing is a real time event in the cart, then user may be prompted to enter the number again or try a different card
    - Inconsistency between the address in the billing section and the address in the card. Check for user-understandable error message.
    - *Browser version too old to support card processing/secure protocols,* user must be pointed to the browser issue and not leave them in a limbo, with a clueless message pointing to card error instead of browser incompatibility issue.
    - *Invalid ABA Number error message*: If shopper is paying by check, he/she must supply valid checking account and "ABA" numbers.
    - *Site does not support the card used by the customer*; provide a message forehand about the type of cards the site supports.
    - *"Temporary Network Error messages"*: A temporary network problem may cause a data transmission error between the credit card processor and your bank.
    - Check if alternative error handling exists, when third party billing agents fail.

- **General error messages**
    - Unable to understand error message; Cryptic & undecipherable error messages especially in secure areas of the shopping cart may make users abandon their cart in panic.
    - A common mistake in a shopping cart error- handling system is displaying machine errors or compilation errors to the user instead of understandable error messages consistent with the language of the site.
    - Persuasive VBScript or Javascript error message boxes that pop up on an erratic entry but don't close on clicking OK!
    - Error-handling routine re-directs you to another page for explaining the error but provides no way to return back to the original state of the shopping cart.
    - Loads a pop-up error message box, but a 404 "page not found" error displayed in the error pop-up!

- *Over enthusiastic exception handling*: pops error message even after the error has been corrected or error message pops up for correct entries too due to failed script based validation routine.
- Error message box or an action to close the error box *causes illegal operation or illegal memory reference* in the browser software and causes the browser to close in the middle of a transaction.
- Error boxes written in scripts not supported or *incompatible with browser type*.
- *Typos, grammatical errors* in error messages that change the meaning of the intended error message
- *Illegible error message*: A combination of the color scheme of the message box and the font size and color may cause the legibility of the error message to degrade.
- *Security problems caused by bad error handlers*: Sometimes error messages pose serious security risks by exposing sensitive data like port numbers, line number of internal code, type of server and internal configuration of systems. Mixing machine communicated errors and error-handling system may simplify the process of writing error messages but the risk of a security lapse runs high when such error messaging systems unintentionally channels out internal and sensitive data.

Examples of related bugs and other known issues

Problem with "Hotwire.com": lack of a useful error message
http://www.phototour.minneapolis.mn.us/essays/hotwire.html

Human Error

Though human judgement and perception is far more superior to any machine, the human tendency to err is always a risk. All shopping cart centric e-commerce systems involve some human action and intervention in the form of data entry, data upgrade, system upgrade, and system design. The chance of human error is equal on both the retailer's side and the user's side. The Common human errors in shopping cart are incorrect price entry and erroneous handling of back end processes. Below are some common risks that exist due to human errors

- **Human error on the retailer side**
  - Risk of price glitches: incorrect price entry, incorrect data feed, incorrect database configuration and all other forms of incorrect human data entry
  - Quantity glitches, incorrect entry of numeric inputs, input in wrong format.
  - System time incorrectly set, all time stamps on order placements out of sync
  - Administrator forgot to restart the web server or shut it down by mistake
  - Back-end human error: wrong item sent, or package inter-changed etc
  - Shopping cart configured incorrectly
  - Administrator erased custom settings by mistake
  - System reset to default by mistake

- Security breaches and system security compromises due to deliberate or non- deliberate human action
- Forgot to backup the files
- Corrupted the configuration file by mistake
- Erased data or deleted files by mistake
- Physical failures induced to the shopping cart system and its underlying hardware, due to bad handling, accidental damage caused by human action
- Human error in entering the correct email address when sending confirmation of order placement (in non-automated systems)
- Typos, grammatical mistakes, and incorrect language structure usage in content pages
- Any large-scale human disaster or man-made disaster that causes physical damage to underlying e-commerce system.
- Transaction aborted due to non intervention of required personnel

- **Human error on the customer side**
  - In-correct selections, in-correct navigation, in-correct understanding of the shopping process could be some top-level errors on the shopper's side.
  - Adding the wrong quantity, filling up information in the wrong fields, filling up incorrect information, specifying wrong shipping address are some of the other errors that cause the e-commerce system from delivering the items purchased through the shopping cart.
  - Entering the wrong data type, entering in the wrong format (ex. Date), selecting the wrong shipping options.
  - Deliberate or non- deliberate abortion of the transaction process.
  - Loss of shopping cart state and subsequent abandonment of shopping cart due to erroneously closing the browser.
  - Entering wrong credit card number or selecting wrong credit card type or entering the expiration date in the wrong format or order.
  - In-correct usage of the shopping cart functionality, like pressing the confirmation button multiple times, clicking on selection buttons multiple times causing errors in the order placement.
  - Trying to access the shopping cart in an incompatible underlying environment or using an older incompatible version of the browser, or having scripts and cookies disabled.
  - Do not have/ has not installed the required plug-ins or media software that is required to view the shopping cart catalog.
  - Wrong shipping methods requested for Alaska, Hawaii, Puerto Rico, and international addresses**.** Only UPS Second Air, FedEx 2Day, and USPS Priority Mail deliver to these addresses.
  - The Billing and Shipping addresses are reversed
  - Wrong e-mail address entered

Examples of related bugs and other known issues

Ashford.com flaw allows "free" purchases

http://news.com.com/2100-1017-233806.html?legacy=cnet

IBM customers buy $1 laptops in site snafu
http://news.com.com/2100-1017-235771.html?legacy=cnet

Pricing mistake prompts Buy.com rush
http://news.com.com/2100-1017-221397.html?legacy=cnet

AOL nightmare: ordered a digital camera from AOL, received McAfee Office 2000 instead!!
http://www.zdnet.com/anchordesk/talkback/talkback_229502.html

United to honor dirt-cheap online ticket fares
http://www.itworld.com/Tech/2409/CWSTO57853/

Amazon.com hit with pricing glitch
http://www.computerworld.com/industrytopics/retail/story/0,10801,47949,00.html

Attache cases go for a penny apiece after pricing glitch at Staples.com
http://www.computerworld.com/industrytopics/retail/story/0,10801,57891,00.html

Customer outrage prompts Amazon to change price-testing policy
http://www.computerworld.com/industrytopics/retail/story/0,10801,50153,00.html

Amazon charging different prices on some DVDs
http://www.computerworld.com/managementtopics/ebusiness/story/0,10801,49569,00.html

The Price Isn't Right: A keying mistake set the price at a mere $26.89 instead of $299.
http://thestandard.net/article/0,1902,24690,00.html

Amazon glitch spurs shopping spree

http://www.usaicorp.com/cc/clips/2000/080200/amazonglitch.htm

Coding glitches main culprit in e-tail fire sales
http://news.com.com/2100-1017-244280.html?legacy=cnet

Price goofs in e-commerce

http://www.augustachronicle.com/stories/053101/fea_124-3979.shtml

Risks due to *Calculation/Computation* errors

A shopping cart has various calculations and computations like discount calculations, billing calculations, shipping and handling calculations and tax calculations. Summarized below are some common risks due to *calculation and computation errors* that cause shopping carts to fail.

- **Discounts/Coupons and special offer calculations**
  - Coupons in the online world are generally a set of numbers that accord a pre-mentioned discount. Common errors are Incorrect sequences of numbers, a mistakenly swapped set of numbers (denoting a different product and different discount)
  - Coupons being accepted by the system, after its expiration date.
  - An infamous bug has been that of, allowing the same customer to use the coupon multiple times until the total price has been completely discounted.
  - Coupon functions OK, but the billing system does not honor the coupon code and continues to charge the full and non-discounted price.
  - Coupons with conditions are also error-prone, because sometimes the conditions that make the coupon valid have errors and make the coupon deemable under all conditions!
  - Some coupon codes that are still under the development stage and not yet been open for public view get "crawled" by search crawlers. And the public gets away with some free shopping!
  - One other bug mentioned in the bug list below highlights how a user could not place a discount over a credit voucher she had, because the system could process either a discount or credit but cannot process both together!
  - Errors in "Quantity available" or "in stock value" displayed in the catalog. This may be due to incorrect computation of inventory stock value. Risk is there will be a delay in shipping the order or the order may never be delivered.
  - Check for all discount options. Discount by total percentage may work but discount by total weight may not work.
  - Check for issues caused by duplicate items with quantity discounts.

- **Pre-checkout/Check-out calculations**
  - Some shopping carts show the total only after checkout, but show incorrect and incomplete calculations when checked in the pre-checkout stage.
  - Some shopping carts display only the price of the item in the pre-checkout stage but omit all other additional costs like shipping, taxation and tend to show a large cumulative price when user is checking out,
  - Hidden costs not shown in the pre-checkout stage
  - Multiplication errors when multiplying prices in real numbers with integer quantities and subsequent error in displaying the total price.
  - Decimalization errors in checkout figures
  - Errors may occur in currency conversion when more than one type of currency is accepted and also subsequently conversion rate tables may be inaccessible or maybe outdated
- **Taxation calculations**
  - Taxes are applied only to the items, but like one of the bug examples quoted in this section, sales tax was applied to shipping costs!

- Even when an order has been placed for multiple copies of the same item, sometimes tax maybe mistakenly applied only to one copy due to the system's internal calculation code.
- When the issue of state taxes is encountered, common problems include wrong application of 'state' factor to the tax calculations, mix-up in estimating taxes for the region of delivery.

- **Shipping calculations**
  - Some sites which interface with other sites such as UPS to calculate 'Shipping costs' skip calculations on shipping when the other site is down and may not provide accurate total costs.
  - Again sites that interface with external sites for accessing their shipping tables, may incorrectly compute shipping costs when large quantities of heavy items are ordered, this error occurs due to the Maximum limit on the weight that is available in the cost calculator tables. For Ex. The UPS site allows calculation of shipping costs for packages with a maximum total weight of 100 LBS. Any order above that weight may be miscalculated since the calculation will continue to be based on 100 LBS
  - International shipping is another error-ridden area, since most international air shipping costs keep changing and also vary with respect to destination, generally the risk remains that the table used for calculation is outdated.
  - Most carts calculate on the basis of price threshold, weight threshold, quantity threshold, line item threshold and sometimes no charge. Risk is high that error may occur due to erroneously using the wrong model for calculation. A single very heavy shipment may then cost very less, if quantity is mistakenly used for the purpose of calculation.

Examples of related bugs and other known issues

Spring forward leaves eBay behind
http://www.computerworld.com/industrytopics/retail/story/0,10801,59222,00.html

Glitches let net shoppers grab free goods
Botched coupon deals let shoppers waltz out with free or nearly free deals
http://news.com.com/2100-1017-242811.html?legacy=cnet

Macys.com says no to unauthorized coupon codes

http://ciscomp.com.com/2100-1017-251548.html?legacy=cnet

Shoppers seize unauthorized discounts at Macys.com

http://ciscomp.com.com/2100-1017-251334.html?legacy=cnet

AltaVista credits players $1 after contest error

http://news.com.com/2100-1017-242970.html?tag=mainstry

Staples.com nailed again by its own Net coupons

http://ciscomp.com.com/2100-1017-244220.html?legacy=cnet

Online shopping, glitches and gotchas
http://www.nwfusion.com/newsletters/techexec/2001/01156806.html

Risks due to *Software Upgrade* errors

Due to the dynamic nature of their content, web stores and shopping carts undergo frequent updates, upgrades and changes. But these frequent changes tend to frequently break things and cause havoc when the site opens up for business after the upgrade. Listed below are some of the risks posed by software upgrade in shopping carts and e-commerce systems

- **Software upgrade on the server side**
  - A common error is the failure to backup the web-store before Upgrade.
  - Accidentally over-writing the product database file during upgrade
  - Non-removal of staging files before upgrade may lead to corruption of the shopping cart
  - Failure to update or reset correct file permissions in the shopping cart after upgrade process, this causes some pages to show “Unauthorized to view” errors when the user clicks on a catalog page
  - Many software upgrade processes look for folders with standard names. For example, CGI based shopping carts look for standard CGI directory path. Any deviations from the standards pose the risk of an incomplete install/upgrade
  - Some upgrades corrupt the shopping cart by changing the default file types to newer file types. And this newer file type may not be compatible with clients that use it.
  - Files upgraded successfully but did not to make changes go ” live" after upgrade!
  - Failure to check the OS compliance of host server before the upgrade
  - Failure to verify the host server's software and hardware requirements before upgrade
  - Insufficient disk space available for the shopping cart upgrade process and the upgrade stalls before completion
  - Failure to update older and outdated content, before an upgrade or site re-design
  - Risk of mistakenly listing outdated and discontinued products by over-writing new files with older ones.
  - **“***We ran two programs at the same time that will not run together*”, Upgrades performed without checking inter-compatibility between existing or newer software processes within the system.
  - Post upgrade “internal glitch” have caused orders from being processed in shopping carts, they generally occur due to new but mismatched data feed installs, convoluted linking due to addition of new links within the shopping cycle, Older links not removed and new links installed without targets.

- Upgrades to some parts of the system, may cause selective failures in dependent or related sections of the system. A common issue has been upgrades to client information databases, causing User authentication failures due to lockouts and denial of access to login processes.
- A fix to one bug causes another! A common problem in conventional software too. A good example of this type of risk is the example of DoubleClick Ad failure mentioned in a bug listed below.
- A "newer look" or "fresh look" after an upgrade may not always mean an error free look for the site, "newer look" changes the GUI and functionality and this leads to newer problems both in terms of functionality, usability and technical glitches leading to blackouts.
- Another important risk is the risk of security problems that are caused by poor installation and in-complete installation that results in some security features being turned off.
- Software upgrades sometimes sets all options to 'default' automatically after the installation is complete and in this in turn may over-write any existing customized options This leads to change in e-commerce system behavior and settings.

- **Client side response to server side software upgrade**
  - Browser incompatible with the new upgraded server side shopping cart

Examples of related bugs and other known issues

Amazon Endures Third Holiday Outage
http://www.ecommercetimes.com/perl/story/5870.html

Webvan stalls on the way to Thanksgiving dinner

http://news.com.com/2100-1017-248798.html?legacy=cnet

E*Trade users locked out of trading

http://news.com.com/2100-1017-221117.html?legacy=cnet

Problems hit E*Trade for third day

http://news.com.com/2100-1017-221192.html?legacy=cnet&tag=rltdnws

Software Glitch Affects DoubleClick's Domestic Clients

http://www.atnewyork.com/news/article.php/8471_441871

Walmart.com runs into glitches

http://news.com.com/2100-1017-249390.html?tag=prntfr

When Buy.com redesigned its Web site on April 26, it mistakenly listed between 4,000 and 7,000 discontinued laser discs for $1.11 apiece
http://news.com.com/2100-1017-225527.html?tag=rn

Dangerous shop service if installed the right way
http://exploiter.virtualave.net/9904-exploits/hhp-WebShop.txt

Yahoo introduces email bug after attack

http://news.com.com/2100-1023-236686.html?legacy=cnet

Document Confidentiality

How secure is the e-commerce site? Is it safe to give my credit card number? Can someone get my order details and my personal information?
Document confidentiality means protecting private information from being leaked to third parties [8]. Compromises on this issue lead to serious security related failures. This category deals with issues like credit card information leaks, order information leaks, account information leaks, etc.
Shopping carts with advanced features provide direct linking with credit card processing agencies using secure protocols such as SSL or SET (Secure Electronic Transaction). But we need to remember that these secure protocols are also prone to failures and attack by malicious elements and can hence cause sensitive data loss.
Cryptography is a key technology that is used for protecting the system against such leaks and testers testing shopping carts that encrypt data in bill payment transactions need to know some simple ways in which cryptography fails?

- The risk of the cryptography algorithm failing because it contains patterns from the plain text and the algorithm can be guessed.
- The risk of the decryption key being guessed and hence may succumb to attacks such as brute-force attack.
- Risk of using lower bit keys to encrypt data. Lower the number of bits the easier it is to crack the key. 128 bits and higher are considered safe.
- Loss or corruption of a private key
- A key is compromised but failure to replace or remove the compromised key.

Cryptography is a well-published topic and lots of information is available on the web on encryption. It may prove futile for testers to test every aspect of cryptography, due to the complexity of the subject. But a basic idea of the risks associated with violating a few basic rules such as safe key selection or failure to replace corrupted keys will help in validating the security in the transaction stage of the shopping cart.

Apart from encrypting the data transfers to protect information, "Document Confidentiality" also involves physically safe guarding files and documents that contain sensitive and confidential information.

Refer to the bug example **"Shopping Carts Expose Order Data",** where a poorly installed shopping cart exposed the order 'log' file with names, addresses, credit card numbers in a world-readable format. People could search for these log files just by entering simple key words from any search engine!

- Risk of exposing directories that hold sensitive files and allowing an external user to access the directory or folders from the web!
- Risk of setting improper read and writes permissions to these files due to which any external user can access and modify these important files.

- Risk of mistakenly configuring the email list server to include sensitive customer information or attach confidential files in public email listings and postings.
- Script errors that let users edit their URL by changing a few visible parameters like order number and grant them access to other user's records!
- Poor configuration of shopping carts may cause an attacker to gain entry to classified information (refer to the examples for more details)
- Risk of unfixed bugs or new bugs in databases and server software may open up serious security holes (Refer to the example bugs). www.bugnet.com hosts a long list of security bugs in this category.
- Check for issues like where the shipping section shows "billing Information" in non-secure customer, emails.
- Check for issues, where instead of just the last four digits, all numbers of the credit card are exposed.

Examples of related bugs and other known issues

E-Commerce Fears? Good Reasons
http://www.wired.com/news/ebiz/0,1272,44690,00.html

Shopping Carts Expose Order Data

http://www.internetnews.com/ec-news/article.php/4_102621

Shopping Carts exposing CC data
http://exploiter.virtualave.net/9904-exploits/cybercash.cc.txt
http://exploiter.virtualave.net/9904-exploits/perlshop.cc.txt
http://exploiter.virtualave.net/9904-exploits/shopping.cart.cc.data.txt

Expert finds hole in shopping carts

http://zdnet.com.com/2100-11-514435.html

HQ for Exposed Credit Numbers

http://www.wired.com/news/ebiz/0,1272,44613,FF.html

Which? under fire over security scare

http://news.bbc.co.uk/1/hi/sci/tech/1402222.stm

Qwest Glitch Exposes Customer Data
http://online.securityfocus.com/news/431

O'Reilly Leaks Geeks' Info
http://online.securityfocus.com/news/408

United Airlines
Frequent fliers who logged onto United Airlines'
Web site got a look at other people's Mileage
Plus account information for more than 12 hours
http://www.siliconvalley.com/docs/news/tech/072275.htm

Hacker Posts Credit Card Info
http://www.wired.com/news/technology/0,1282,33539,00.html

Risks due to Memory Leaks

These are the bane of many a site, and not surprisingly also a major cause of many e-commerce site crashes and also the cause of deteriorating performance of many e-commerce sites. Shopping carts are complex systems that use scripting code, run on a server, need an underlying OS and also need a browser on the client side to function. Hence a memory leak that occurs in any of the above components can cause the shopping cart to fail indirectly due to memory leaks!

Just as this article on Web testing in ExtremeTech puts it, "Memory leaks can range from irritating to debilitating. A merely irritating memory leak might involve a component growing until it crashes and re-spawns. This is still bad for the site, as a crashed component forces everyone to wait until it restarts. The worst-case scenario is a component that uses up more and more system memory (or worse, kernel memory) without exiting, until the entire system finally grinds to a halt."

The common issues discussed here is a subset of a larger set of memory leak issues and bugs that can be found (in more detail) at Labmice.net [9] (http://www.labmice.net/troubleshooting/memoryleaks.htm)

- **Issues due to memory leaks in scripting code**
    - Some scripts have maximum static string size and violation of which causes memory leaks, hence shopping cart pages that use heavy and long scripts to add navigational or functional capabilities may violate these size constraints resulting in browser throwing up "*Out of Memory"* errors when the cart pages are viewed.
    - Bugs in the Script DLLs may cause memory leaks under specific conditions such as when the limit on the number of loops within a script is exceeded.
    - Some inbuilt functions such as 'string format' functions (as in VBscript) in common script languages have known memory leak problems and frequent use of these functions in a high volume web environment such as ASP enabled shopping cart page may lead to serious memory leak problems.
    - Enough cannot be said about the negative effects of sloppy programming practices, which are the root cause to most memory leaks, and badly written scripts that lock up resources can be equally lethal in shopping cart pages as in anywhere else.
    - Since shopping carts are chiefly database driven, resource locking is a big risk where a database record or a file becomes unavailable for prolonged

periods of time because a particular site component has an exclusive hold on it.

- Older version of constituent components in the scripting environment can also be a potential cause to memory leaks, for example using an older version of perl interpreter engine in a perl-based shopping cart.
- Some design level decisions can also save the e-commerce site from potential memory leaks, one of them is to adopt a modular scripting framework where independent classes can be tested for memory leaks.

- **Issues due to memory leaks in browsers**

  - IE 5.0 and 5.01 have had memory leak problems when images were re-sized using DHTML (Dynamic Hyper-Text Markup Language) . Since it is common to use scripts and DHTML to format images in shopping cart catalogs, care should be taken not to trigger memory leaks. This can potentially freeze the cart page, and cause performance problems and loading errors.
  - Some browser methods like "StartDownload" methods consume excessive memory and do not return them back to the system. Hence shopping carts that offer downloadable files, product, software code and also host large sized image files are in danger of potential risks of memory leaks, triggered by calling these risky browser methods.
  - Browsers are known to load and access several libraries. And shopping carts that spawn duplicate product browsers, pop-ups, ad browsers multiply the total RAM consumption. Since these browsers have known memory leakage problems associated with their versions, they may cause the system to come to a cranking stop, freezing any potential transaction midway.
  - Memory Leaks in IE 3.0x: These versions (3.0/3.01a/3.01b) have progressive memory leaks; one of its manifestations slows down the performance and response of the browser slowly over a period of time. And thus media rich or heavy data content shopping cart pages faces serious performance issues due to these types of memory leaks.
  - Another famous memory leak bug in IE 3.x is it caches page information when using the <Form Method=POST> tag and fails to free the memory until the application is shut down, this is a serious problem since many carts use GET and POST methods in their forms.

- **Issues due to memory leaks in underlying operating system**

  - Undeleted threads are a major source of memory leaks in Operating systems, so if the underlying OS of an e-commerce system starts leaking memory due to undeleted threads then the hosted shopping cart begins to fail due to lack of memory available for its functioning
  - Any third-party process that may be running on the operating system may cause unreleased memory, which may indirectly cause the shopping cart to fail due to underlying OS failure because of lack of sufficient memory.
  - Some standard system libraries in older versions of operating systems may leak memory, so risk of not upgrading to newer underlying operating

systems in e-commerce sites transforms itself into a risk leading to memory leak.
  - Also shopping cart sites, which have user written server side plugins created by server programming APIs, the user code may introduce serious memory leaks.

- **Issues due to memory leaks in server**

  - Incorrect use of multithreading in Web server software can be a problem and may end up in a memory leak, since a shopping cart sits on a web server, a web server failure leads to a shopping cart being rendered non-operative.
  - According to the different IIS FAQ lists and MS knowledge base articles, Inetinfo process in Internet Information Server (IIS) may leak memory when using SSL.
  - Some web servers hosting software downloads or shopping carts offering large documents, eBooks, Media files as downloadable products have a high risk of suffering from memory leaks the equal to the size of the file being uploaded/downloaded if transfer is aborted prematurely. The cause may be due to some methods like *Request.BinaryRead* being called by ASP or other similar scripts.
  - When trying to access member accounts in a shopping cart, small memory leaks may occur when a lookup of the current domain name is performed. A pointer to the domain name may be saved in a global location without freeing the previous domain name already stored there.

Examples of related bugs and other known issues

IIS Memory Leaks
http://www.iisfaq.com/MemoryLeaks/

Memory Leaks in OS
http://www.labmice.net/troubleshooting/memoryleaks.htm

More on Resource Leaks
http://www.willows.com/listarchives/dev/twindev-1998-jul/0155.html

Risks due to *Insufficient Capacity Planning*

According to this Microsoft TechNet paper [10] “Capacity planning is the process of measuring a Web site's ability to serve content to its visitors at an acceptable speed. This is done by measuring the number of visitors the site currently receives and by how much demand each user places on the server, and then calculating the computing resources (CPU, RAM, disk space, and network bandwidth) that are necessary to support current and future usage levels”. The idea of categorizing the risks into the following three categories is derived from the same paper.

- **Risks based on the number of users and usage**
  - Shopping cart performance degrades due to increase in site users disproportionate to existing capacity.
  - No Increase in the number of users but increase in the activity of the users, increase in terms of catalog page hits, latency time, increase in usage of search activity, increases in shopping cart update cycles. The increase in such heavy resource consuming activities may upset the capacity planning equation, which maybe based on the number of users and not usage.
  - The most common cause that leads to sudden load and causes deficiency in system capacity are the seasonal increases in customers especially the "Holiday shoppers". Test shopping cart for performance and scalability under realistic loads
  - Increase in the number of transactions involving third party components like Billing cycles, Credit card authorizations and account transfers, where the insufficiency in the capacity of the third party systems will indirectly cause the shopping cart and the e-commerce site to stall.
  - Resource consumption also depends on the stage of the shopping cycle. For Example: The checkout stage uses more pages, more CPU, more DB transaction cycles and more server utilization than the catalog 'browse' stage. One has to always plan for sufficient capacity and availability for all stages of the shopping cycle keeping in mind the changes in the requirements at each stage.
- **Risks based on computing Infrastructure**
  - CPU Insufficiency may be a big risk if there is an excessive demand placed on CPU by the Web Server or the Database Server. Web servers especially tend to consume more CPU cycles than the corresponding Database server in the system.
  - If the shopping cart spawns a new process every time a user invokes it, and no mechanism exists to limit the maximum number of shoppers then very soon the processes will choke the available CPU and cause the entire system to slowly crash.
  - An operation may cost less in terms of resource consumption, but if the frequency of that operation is high, then very soon we will have a capacity insufficiency risk. Generally product pages and search pages are moderate in their cost but search page operation is very frequent due to which very soon, it may become the largest resource consumer despite its cost being less.
  - Also Shopper capacity is determined by the underlying Operating Systems. For example According to this paper [10] Win NT reaches CPU utilization of 96.40% at shopper's load of 1000 while Win 2000 reaches CPU utilization of 72.89% at shopper's load of 1000.
  - If any shopping cart operation like basket load or catalog load is memory intensive, then the underlying web server may run into memory deficit very soon.

- If any shopping cart operation forces the web server's page-able process to page to disk, it is bad news and will affect the performance of the web server badly.

- **Risks based on site content complexity**
  - Network capacity may become a bottleneck if shopping cart uses high static content like large images and static HTML.
  - Poor site design where heavy elements and heavy content pages are called more often than the lighter ones due to which the resource consumption gets unevenly distributed and resource consumption becomes very high. Higher demand should have 'light' content and we can afford to make pages with lesser demand 'heavy'.
  - Advertisements retrieved from ad databases, Customizations to fit shopper's choice, ActiveX control driven Menus and Java based menus are some of the complex site components that potentially affect the capacity of the system and tend to pose a risk to the functioning of the shopping cart.

Examples of related bugs and other known issues

Crashing Success for the Web?
For Online Retailers, a Make-or-Break Year Could Find Sites Overloaded
http://abcnews.go.com/sections/business/TheStreet/onlineretail991202.html

Charles Schwab Web Site Crashes
http://www.binarythoughts.com/article.cfm?StoryID=237

Encyclopaedia Britannica's New Web Site Crashes

http://www.infowar.com/p_and_s/99/p_n_s_102299e_j.shtml

E-tail sites crash over holiday weekend

http://news.com.com/2100-1017-249048.html?legacy=cnet

Webvan running out of Thanksgiving goodies (may also go into process failure)

http://news.com.com/2100-1017-248881.html?legacy=cnet

Customers locked out of Virgin Megastore's online sale

http://news.com.com/2100-1040-230643.html?legacy=cnet

Accessibility Risks in Shopping Carts

According to statistics provided at www.webaim.org [11] "an estimated 20 percent of the population in the United States (40.8 million individuals) have some kind of disability, and 10 percent (27.3 million individuals) have *severe* disability. The 27.3 million individuals with severe disabilities are limited in the way that they can use the Internet". W3C's Web Accessibility Initiative (WAI) has produced the Web Content

Accessibility Guidelines 1.0 [12], which explain in detail how to make a Web site accessible for people with a variety of disabilities.

Detailed lists of risks that affect shopping carts in terms of accessibility failure are provided below. They have been categorized with respect to the different severe disabilities that affect Internet users and more specifically people who use shopping cart enabled e-commerce sites.

❖ *Visual Impairments*

- **Blindness**
  - ***In General***
    - If the cart catalog is categorized under headings and captions to denote product categories, shipping options, billing options, test the content with a screen reader and check if the document's text or other text equivalents make sense? Test and verify that the change in categorization is understandable to a person who cannot see the headings or captions.
    - If you are using style-sheets to render your shopping cart pages, check if it is possible for screen readers and non-CSS supporting browsers to render the cart page correctly.
    - When dynamically updating shopping cart pages, the equivalents to dynamic content may not get updated when the products, rates, or prices are dynamically updated. Thus only non-updated data would be available to impaired or disabled users.
    - If you are using an image-mapped-shopping cart page, check for redundant links to every navigable section.
    - In shopping cart product pages, where size tables, price tables, shipping tables, schedule tables are used; Test if row and column headers are identified. Otherwise the screen reader will output a stream of non-distinguishable data.
    - Also when testing tables that use two or more logical levels, Check if alternative text exists to identify which cell is a header cell and which contains data.
    - When frames or layers are used to create the browse catalog in the cart, check each frame for title. This will help identification and navigation when read by screen readers.
    - Some of the tools that help disabled or impaired users do not support programmable objects such as applets or scripts. Check for alternative functionality in the shopping cart under a situation when the user has these options turned off
    - Use of pop-up shopping carts, pop-up advertisements, pop-up alerts, sales pitches in new windows takes the control off the active window that's being read and confuses the

reader by making the screen reading software alternate between the different windows!

- Navigation across the shopping cart pages should be consistent and straightforward, Look for looping navigation and random return-backs and other similar issues that cause the screen reader navigation problems.

- ***Text Equivalent***

  - Check for equivalent test for every non-text element in the product catalog of the shopping cart
  - Check for issues where the text equivalent describes the graphic but not the content in the graphic image!
  - Look out for open or broken ALT tag, this disables the screen reader from reading the content inside the tag.
  - If tables are used to format the appearance of equivalent text, test using a screen reader whether the content read from the table makes sense. Tables tend to confuse the screen reader and the screen readers tend to read the content in different cells in a haphazard way.
  - Check if by error the alternative text equivalent is provided in a language not same as the language of content. Apply this test to all sites that have multi-lingual international sites.
  - Check for typos, spelling errors and word jumble in the alternative text. Screen readers do not have the ability or the intelligence to notice the errors and it reads out non-tangible words to the user.
  - Check for unknown abbreviations, acronyms, and unfamiliar complex words and jargon describing non-text elements in the cart
  - Check if the items that have been added into the shopping cart have alternate text, which the screen reader can read out so that the user can verify that the item she just purchased, has been added.
  - Check if the images of credit cards that are displayed at checkout have alternative text. Otherwise, the user will not be able to make out what cards are accepted.
  - Test shipping cost calculators, gift-wrapping cost calculators and other such user-aid tools for equivalent text support. Test each button and field, for alternative text describing the functionality.
  - In the billing and shipping section of a shopping cart, test the order of entry and order of tabs with a screen reader. Sometimes screen readers read tabs in the wrong order.
  - Test multimedia presentations and alternative text for correctness of data and also check if they appear in sync.

- ***Color-Blindness***
  - ***Total Color Blindness***
    - Testers should test if an alternative text exists for text that conveys information by means of color. For example, if all items marked for sale are marked in 'red' or new items are marked in 'blue', then equivalent alternatives should exist in regular black text.
    - Testers should test for alternatives when color is used as the primary way to indicate an action, say links turns purple from blue, when it is clicked. Similarly some shopping carts mark 'visited' categories by changing its color when a user clicks on it. This color change is to aid the shopper in keeping track of what they have seen and what they have not seen? But a person with color blindness may not notice the change in color as he has trouble identifying the colors. An alternative may be using an object like asterix or a cursor instead of color to identify change in state.
  - ***Color deficiencies***
    - People with a color deficiency can see some colors but some pairs of colors look the same to them. So, for example, foreground and background colors may appear the same. Look for difficult color combinations in catalog design, link identification, announcements and so on. Reds, greens, oranges and yellows are the most likely to cause problems. . For color combinations, see http://www.webaim.org/intro/intro
  - **Low Vision**
    - People with low vision impairment use screen enlarger software to increase readability of small text but the enlarger limits the visible area of the browser screen. Screen designs that communicate well at normal text size may be confusing and hard to use when viewed in an enlarged mode.
    - If pages rely on scrolling, screen enlargers will yield pages that require more scrolling. If items on the screen are not appropriately grouped, the user will have to remember and correlate too many details that should be displayed together
    - Another risk lies in the use of graphics with embedded text for product catalogs, because due to enlargement of screen space, the images may get highly pixilated and embedded text may become hard to read.
    - People with age-related visual impairments—such as macular degeneration, glaucoma and cataracts prefer

shopping cart pages and functionality that can be enlarged, scrolled and purely textual devoid of much use of graphics.

- ***Hearing Impairments***
  - **Deafness**
    - ***Lack of visible textual support***
      In sites where sounds signifies a buying process, purchase alerts, error message, instant messaging from a live representative, then testers should look for alternative visible textual support.
    - ***Test for the absence of subtitles or other text***
      Sub tittles or alternative synchronized text, should supplement video catalogs, virtual demos of products product information sessions and help videos for shopping.

  - **Deaf-Blindness**
    - ***Test for special cases of double disability***
      Test for all risks that are applicable to Blindness, but since Deaf-Blind use screen readers that convert text into Braille (through a refreshable Braille device attached to the computer), you have to test to confirm that there is text that is readable by the Braille reader. Sound as an alternative to visible text is workable for the blind but will not work for people with double disabilities, such as deafness and blindness.

- ***Mobility impairments***
  - Mobility impairments range from minor to major problems restricting voluntary muscle movements. Because of lack of dexterity, the user may find it difficult to click on small (single letters and alphabets), product links and catalog navigation links. Similarly, look for small functional buttons or other tiny targets that must be hit precisely.
  - Others using devices to access the keyboard, such as a mouth stick or a head wand have lower dexterity than regular users, so test auto refresh, time outs and auto-exits from secure billing areas and verify that the time available is sufficient for them.
  - Users using devices such as head wand have to shake their head 20 times to browse through 20 links! Hence test if the unnecessary links can be minimized, or short cuts to skim past groups of related links can be offered?
  - Keyboard shortcuts and keyboard functionality is vital to users with mobility impairments; risk is high that they will be neglected if too much mouse-centric functionality is added.

- ***Cognitive impairments***
  - Some of the better-known cognitive impairments are the Downs Syndrome, Alzheimer's disease and the lesser-known cognitive impairments include reading and learning disorders. Individuals with cognitive impairments often benefit from graphics or icons that supplement the text, providing a monotonous small sized 'text only' interface devoid of any meaningful graphics and animations may not benefit such users much.

- ***Seizure disorders***
  - Beware of flickering sales tickers, ad banners, notifications, alerts, or interactive messages that are provided in some shopping carts. These seemingly harmless gizmos may be a serious risk to people with seizure disorders if their frequency of flickering is between 2 Hz to 55 Hz.

Related Internet Links

1. Western Australian Electronic Commerce Center, Information on ***Accessibility and Usability*** (http://www.ecommercecentre.online.wa.gov.au/matrix/acc.htm)

2. **Section 508:** The amended Rehabilitation Act that requires Federal agencies to make their electronic and information technology accessible to people with disabilities. (http://www.section508.gov)

Usability Risks in Shopping Carts

As per the definition provided by usability.gov, Usability is the measure of the quality of a user's experience when interacting with a product or system — whether a Web site, a software application, mobile technology, or any user-operated device.
And hence a general agreement is that a "Usable Web interface is one that is

- Easy to learn
- Easy to use,
- Culturally appropriate
- Not discordant with the user's expectations of how the program should operate,
- Responsive (fast enough responses that the user doesn't notice waiting for the program to catch up).

Provided below is a list of issues that may seriously hamper the usability of a shopping cart in an e-commerce site. Real life bugs support some of the risks and issues and some others have Do's and Don'ts on how to avoid the risk.

- ***Pop-Ups***
  - Do not use pop-up window shopping carts because If the user clicks elsewhere in the main window, the pop-up is sent "behind" the main window
  - Test shopping carts with pop-up/Ad eliminating software turned ON. Pop-up shopping carts may not work if the pop-up eliminator is ON.
  - Check if Pop-up shopping carts have sufficient "real-estate" space when the user adds more items.
- ***Functionality***
  - Provide "remove item" or "add item" buttons instead of asking the user to change "item quantity number"- it is easier and more error free.
  - Look for items that have not been linked back to the “item”/catalog page.
  - Check if the “number of items” in the cart is displayed. Users prefer carts that show the current data and state, like how many items are in the cart? What is the total? Etc
  - Check if the "Continue Shopping" and "Proceed to Checkout" buttons are visible.
- ***Navigation***
  - Look for unnecessary steps between item selection and checkout. The more clicks, the more confusion and the greater the probability that the customer will abandon the transaction.
  - Do not link to any external site/page from the shopping cart page- leads to shopper getting confused/uninterested-leading to shopping cart abandonment.
  - Check if the shopper is able to navigate back to shopping process, after "adding" or "removing" items
  - Check if it is possible to add additional items directly from the cart page, instead of going back to browsed pages. This improves functionality and enhances usability.
  - If providing detailed info on products to users, then test if you are able to return back to the shopping cart from the detailed page and also check if the state of the shopping cart is maintained.

- ***Ease Of Use***

- Check if thumbnail photos of the items can be added to the shopping carts in addition to a text description, this re-assures the customer that the right item has been added to the shopping cart
- Try enhancing the usability by providing an auto-update cart facility after user has added/removed item.
- Check for appropriate positioning of buttons. Place "Continue Shopping" on the left and "Checkout" button the right, users perceive it analogous to "back" and "going forward" respectively
- Presence of standard "credit card" images on the UI adds trust psychologically on the site's security. Check the shopping cart for images or text that might cause mistrust in the user.
- Check the if the UI provides functionality for discounts and coupons. Provide separate field in the UI to display discounts due to coupons, it helps user note the discounts better
- Provide separate columns to display "total" bill as the user adds items to the cart
- Too much information to type into the cart - this common problem should be avoided
- Do not limit the features of the shopping cart--keep it flexible
- Cart is too hard to use -reduce functional complexity
- Check if the user is conveyed the information of order placement. Warn the customer when the transaction becomes final; do not surprise them by abruptly billing their contents.
- Check forms against data requirement. Collect only essential information about the user that is absolutely a must for completing the deal, unnecessary questions and making optional questions compulsory makes the user experience bad.
- Check for Hi-Tech whiz creations like flash display of catalog and constantly flashing blue lights in a shopping cart because it may reduce the usability of the cart. A classic example of a site that got booed away due to its technical gimmickry was www.boo.com
- Check for plug-ins or media files that are not common in any general browser software, and recommend not using them. Expecting users to download software to shop at your site is high handedness! And may cost you heavily in terms of loss of customers to other competitors.
- Provide the user with the functionality to choose the mode of shipment, Check for fixed default radio buttons, non-flexible shipping options, erratic placement of multiple selection checkboxes

- Not sticking to known paths in navigability and sequence of shopping decreases the usability of the shopping cart, Check for odd sequencing issues like re-sequencing shipping costs after the user has been billed and charged. This will confuse the customer about whether the purchase was executed or not!
- When new functionality is added to the shopping cart- Check if it is user-understandable otherwise provide help
- Check for odd naming of known metaphors.
- Check if shipping can be calculated before checkout. Shoppers prefer getting an idea of the total cost of the item.

Eric Myers's Case study [13] on shopping cart experience and Andrew Starling's E-commerce and Usability [14] researches some of the above-mentioned risks in a detailed manner.

Examples of related bugs and other known issues

I hate waiting
http://www.zdnet.com/anchordesk/talkback/talkback_229493.html

You forgot 'overcomplicated technology'!

http://www.zdnet.com/anchordesk/talkback/talkback_229472.html

Dell Computer's site has got to be the worst for hiding charges

http://www.zdnet.com/anchordesk/talkback/talkback_229509.html

Last minute surprises make me crazy too

http://www.zdnet.com/anchordesk/talkback/talkback_229513.html

Cookie crumbled

http://www.zdnet.com/anchordesk/talkback/talkback_229516.html

Convert Browsers to Buyers (Article)
http://builder.cnet.com/webbuilding/pages/Graphics/Ecommerce/index.html

E-commerce and Usability
http://wdvl.internet.com/Authoring/Design/Basics/ecom1.html

Two usability bug examples: Usability Testing
http://www.smartisans.com/usability_testing.htm

Appendix: Terminology

- ***Test idea*:**

A test idea is a brief statement that identifies a test that might be useful. A test idea differs from a test case, in that the test idea contains no specification of the test workings, only the essence of the idea behind the test. Test ideas are generators for test cases: potential test cases are derived from a test ideas list. A key question for the tester or test analyst is which ones are the ones worth trying. (We thank Brian Marick for introducing us to this term, and the level of analysis that comes with it.)

- ***Risk category***
  We call the 60 top-level categories that constitute the outline, risk categories. Each of them identifies a class of problems that might arise in the function under test.
- ***FMEA***
  Failure Modes and Effects Analysis is a common method of risk analysis for complex or life critical systems. The process involves identifying the failure characteristics of individual components and determining the effects of those failures as they propagate across the systems in which they are embedded.
- ***Failure mode***
  By definition, "Failure Modes are sometimes described as categories of failure. A potential Failure Mode describes the way in which a product or process could fail to perform its desired function (design intent or performance requirements) as described by the needs, wants, and expectations of the internal and external Customers"[2]
- ***Software bug taxonomy***
  Taxonomy is the theoretical study of classification including its bases, principles, procedures and rules [Grolier Incorporated 1993] and classifications that are created by categorizing the different software bugs are called bug taxonomies.
- ***Risk-based testing***
  Risk-based testing is a strategy for selecting test ideas and designing tests based on the types of problems you imagine might be present in the product. Risk-based test management is a prioritization scheme--the test management team decides which types of tests or potential problems to invest testing time in based on perceived (or estimated) probability of failure and expected damage that will result if the product actually fails in this way.

References:

[1] A. Biraj Rath, NIIC (USA), Inc., P. M. Raj Nath, Microsoft Corporation, P. M. Mukesh Agarwal, Microsoft Corporation, P. M. Jas Lamba, NIIC (USA), Inc., R. George Gianopoulos, Microsoft Corporation, and C. Laura Hargrave, Microsoft Corporation, "Testing Process," July 2000.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnentdevgen/html/testproc.asp

[2] FMECA, "FMEA Methodology." http://www.fmeca.com/ffmethod/definiti.htm

[3] S. Amland, "Risk-Based Testing and Metrics," *EuroSTAR 99, Barcelona, Spain*, 1999. www.amland.no/PowerPoint/EuroSTAR99Presentation.ppt

[4] I. 9126, "Software engineering -- Product quality -- Part 1: Quality model," *International Organization for Standardization*.

[5] P. Martin, "Managing Database Server Performance within an Electronic Commerce Framework." http://www.cs.queensu.ca/home/cords/publications/m99.pdf

[6] Oracle Corporation, "Oracle9i Database Administration: Recover Databases."

[7] C. Dilley, "Known HTTP Proxy/Caching Problems," May 14, 2001. http://www.wrec.org/Drafts/draft-ietf-wrec-known-prob-03.txt

[8] L. D. Stein, *Web Security: A step-by-step reference guide*, May-2000.

[9] Labmice.net, "Memory Leaks." http://www.labmice.net/troubleshooting/memoryleaks.htm

[10] L. Louis de Klerk (Inobits Consulting Pty., Jason Bender (MSNBC), "Microsoft Enterprise Services White Paper
E-Commerce Technical Readiness," *Microsoft TechNet*, April 2000 version 1.0. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/ecommerce/maintain/optimize/capplan.asp

[11] WebAIM, "Introduction to Web Accessibility," 2002. http://www.webaim.org/intro/intro

[12] W3C-WAI, "Web Content Accessibility Guidelines 1.0," 5-May-1999. http://www.w3.org/TR/WCAG10/

[13] E. G. Myers, "Case Study #4: The Shopping Cart Experience," 2001. http://www.egmstrategy.com/carts4.html

[14] A. Starling, "E-commerce and Usability," November 5, 2001. http://wdvl.internet.com/Authoring/Design/Basics/ecom1.html

QW2002 Paper 4I2

Mr. Ed Adams
(VeriTest)

Deployment of Globalised Wireless Internet Applications

Key Points

- Convergence of Wireless Internet Applications with the Desktop and Server
- Interoperability of Applications on Wireless Internet Devices
- Certification of Internationalised Applications for Standards Compliance

Presentation Abstract

This session will cover the test challenges facing wireless application developers as they integrate their solutions with those provided by traditional application service providers. It will include an overview of the changing technologies, and focus on key areas of testing needed for these mobile solutions. A case study will be presented that illustrates the full range of testing needed for a mobile device (with a microbrowser) in Brazil that is accessing on-line banking information in Japan. For this example, we will consider functionality and compatibility issues, as well as the consequences if the application is not internationalised/ localised

About the Author

As VeriTest's Vice President of Marketing, Worlwide Testing Services, Ed is responsible for evangelizing VeriTest's capabilities in outsourced testing, certification, and consulting. He has over 10 years experience in the software industry in Product Management, Testing, Development, and Marketing, and is a frequent speaker on quality and test automation topics. Prior to joining VeriTest, Ed spent 4 years with Rational Software's automated testing team. He is a member of the Order of the Engineer, ASME, and several university advisory boards. He holds a B.S. in Mechanical Engineering, a B.A. in English Literature, and an MBA from The Carroll School of Management at Boston College.

VeriTest.

Deployment of Globalised Wireless Internet Applications

Ed Adams
Vice President, Worldwide Testing Services
Ed_Adams@veritest.com
QW 2002 Paper 4I2

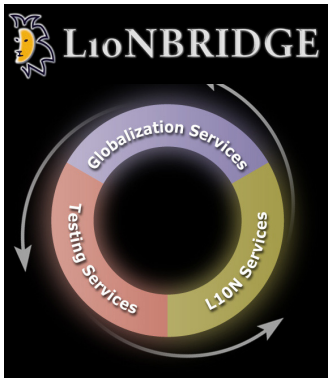
Agenda

- Introduction
- Deployment for Multilingual Environments
- Principles of Good Application Design for Wireless
- Language – Is it relevant ?
- Design Considerations for International
- Testing vs. Certification
- Globalisation Cost/Implications
- Summary

VeriTest.

VeriTest: A division of Lionbridge

- **VeriTest Founded 1987**
  - > Private 12 Years, Public 3 Years (LIOX)
- **Lionbridge Market Position in Globalisation and Testing**
  - > $100M+ Revenues in 2001
  - > 1000+ Employees in 11 Countries
- **Business**
  - > Globalisation Solutions
  - > Localisation Services
  - > Product QA and Certification
- **Industries**
  - > Fortune 1000 in IT, Financial, Life Sciences, Automotive



VeriTest.

3

Certification at VeriTest

Featuring Microsoft Visual Basic for Applications

Ready For Microsoft Windows

Certified For Microsoft Windows
Windows 2000 Datacenter Server
Windows 2000 Advanced Server
Windows 2000 Server
Windows 2000 Professional

Alcatel OmniPCX Office

YES TESTED & APPROVED Novell

TATA Wireless certified solution

Microsoft .net Enterprise Servers

Designed for Microsoft Windows XP

BMC SOFTWARE Certified

palm

A MICROSOFT ConnectED SOLUTION

VERIFIED BY VeriTest. Compatible with Stratus ftServer

TERMINAL SERVER TESTING PROGRAM
Microsoft Windows NT Server 4.0 Terminal Server Edition

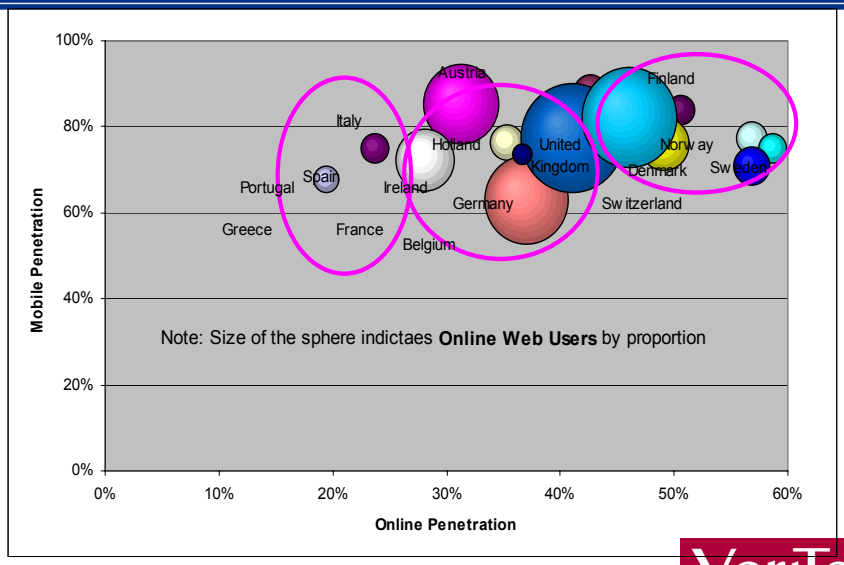
Designed for Microsoft BackOffice

IBM Netfinity Server Proven

VeriTest.

4

Deployment for multilingual audience



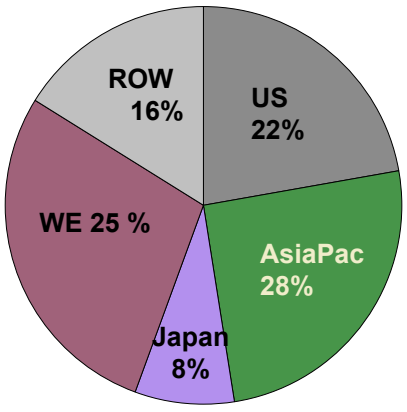
Source: ICMM V8.1, IDC, Feb 2002

5

VeriTest.

Deployment for multilingual audience

% of 2006 Internet Users

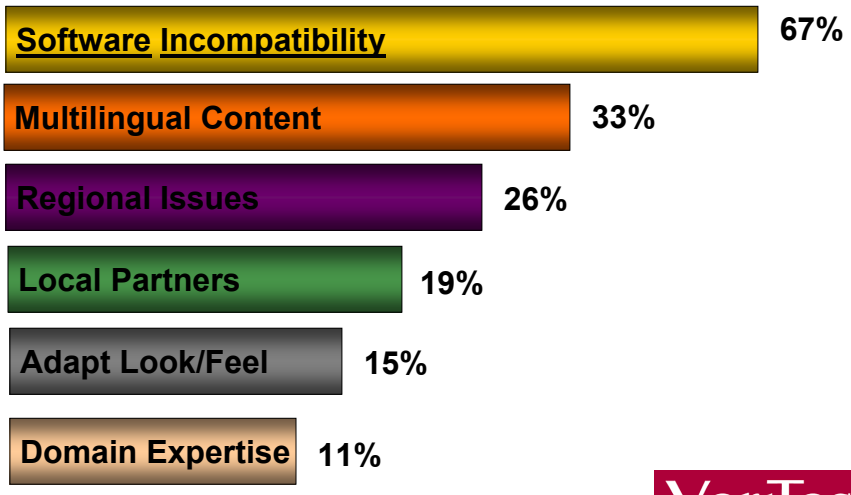


Source: ICMM V8.1, IDC, Feb 2002

6

VeriTest.

Deployment for multilingual – Case Data



7

VeriTest.

Principles of good application design

- Goal:Maximise Customer number & frequency
  - Value of visit drives velocity of return
- Compelling Content
  - Fresh, relevant to the solution – is it interesting ?
  - Clear and concise – it's a small screen!
- Navigation
  - Must be simple, intuitive and consistent
  - Cumbersome access results in
- Extensibility
  - To other markets / demographics

8

VeriTest.

Presentation of Content

- Menu selection screens
  - Limit # of selections (9) and use links to navigate
  - Avoid all wrapping of text
- Card Titles and Headings
  - Openwave / non-openwave browser ?
  - Save valuable screen space
- Registration
  - If possible, eliminate PC based registration !
  - Use cookies and choice menus to facilitate entry of repeated text (.com, .be, .co.uk, etc)
- Splash screens
  - Use 1 graphic, keep it 1 bit, with no timers

9

VeriTest.

Navigation

- Minimise # key strokes to access information
- Minimise requirements to use 'back' key
- Minimise dependency on soft keys
  - Use soft keys for 'OK', 'Skip', 'View', 'Home' etc
  - Most phones support at least one soft key e.g. Nokia use a soft key for the 'back' key function – define in WML
- Login Screens
  - Limit to banking/brokering applications/sites only
- Exit Functionality
  - Highest source of Customer dissatisfaction !!
  - Usability testing should identify exit functionality issues

10

VeriTest.

Language – Is it Relevant ?

- Data Entry / Input Methods….
- Optimising data entry for the application
  - Postal Codes Used ?
  - Date / Time formats used ?
  - Alphabetical sorting order for menu items ?
  - Some European languages have > 26 characters !
  - Limits to address fields
  - Line breaking
- Translation can increase text size by 50%
  - Test for wrapping before releasing

11

Design Considerations – I18N

- I18N = InternationalisatioN
- Test for each level of 'enablement'

1	Level 1 is '8-bit' Clean
2	Level 2 exhibits locale awareness
3	Level 3 implements Resource/Code Separation
4	Level 4 is Multi-Byte Enabled
5	Level 5 is Fully MBCS Enabled supporting input method editors, sorting, writing directionality etc.

VeriTest.

12

Quality Assurance vs. Certification

- QA / Testing
  - Application functions per requirements/design
  - Designed to identify design/usability defects
- Certification
  - Application meets 'public' functionality criteria
    - Stamp of approval for application developer
    - Reliability statement for application consumer
    - Consistency for platform provider

13

VeriTest.

Typical Certification Criteria

- Basic Support
  - Mobile gateways / browsers supported
- UI Support
  - Good 'About' functionality
- Processes
  - Switching between voice and data
- Cards
  - Menu selection guidelines
- Alerts
  - Telco vs. Hardware vs. Software alerts
- Input Methods

14

VeriTest.

Cost/Implications of Failure

- Delayed release of localised applications
- Cost to 'retrofit' applications to ensure global wireless internet compliance
- Opportunity costs => Consumer migration to Cultural / 'Friendly' applications
- Credibility in global market place
  - > Your application will kill you
  - > Not vice versa

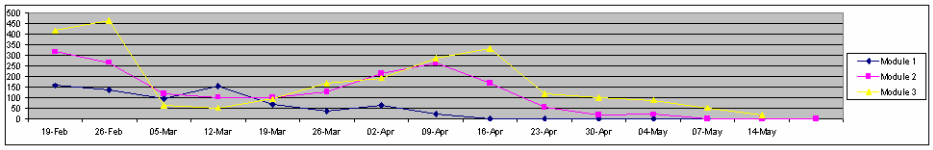
15

VeriTest.

Case Study Data

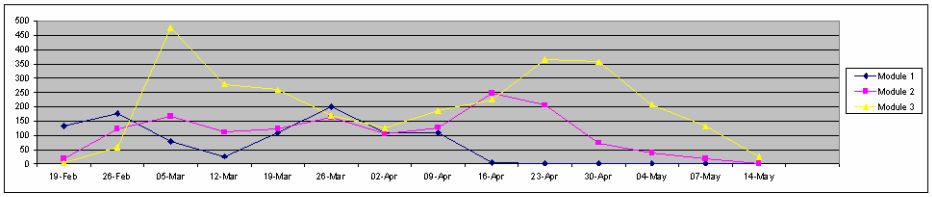
Trend Analysis - Open Bugs, Week Ending

	19-Feb	26-Feb	05-Mar	12-Mar	19-Mar	26-Mar	02-Apr	09-Apr	16-Apr	23-Apr	30-Apr	04-May	07-May	14-May		
Module 1	158	137	97	157	69	36	64	24	0	0	0	0	0	0	0	0
Module 2	317	266	118	103	103	127	216	263	170	56	19	22	1	0	0	0
Module 3	417	465	65	53	95	169	194	288	334	118	102	89	53	18		
Totals	892	868	280	313	267	332	474	575	504	174	121	111	54	18	0	0



Trend Analysis - Corrected and Integrated Bugs, Week Ending

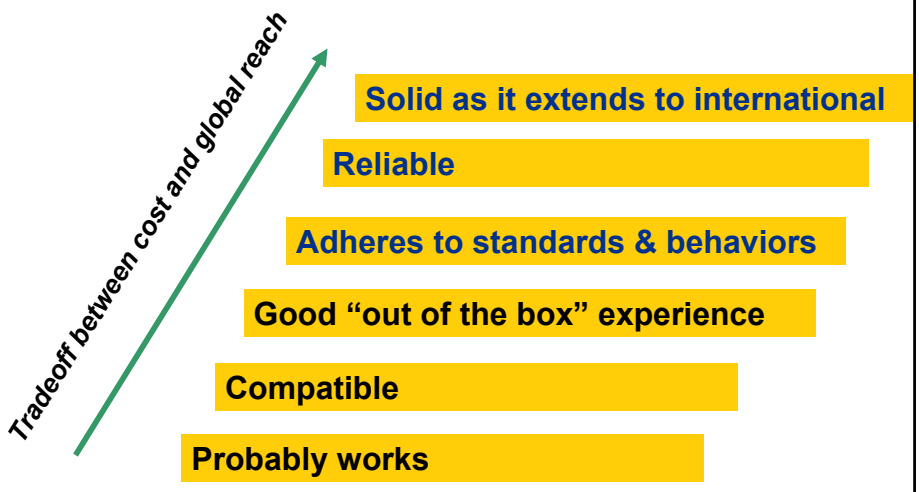
	19-Feb	26-Feb	05-Mar	12-Mar	19-Mar	26-Mar	02-Apr	09-Apr	16-Apr	23-Apr	30-Apr	04-May	07-May	14-May	
Module 1	132	177	79	26	108	199	109	109	4	0	0	0	0	0	
Module 2	18	122	166	111	123	161	106	127	246	205	73	37	18	0	
Module 3	3	59	476	279	259	169	125	187	225	365	357	208	131	25	
Totals	153	358	721	416	490	529	340	423	475	570	430	245	149	25	



16

VeriTest.

Summary approach to release



17

VeriTest.

Summary

- Today, most organisations do not factor 'international' into wireless app design
- QA can play a more significant role in outcome of wireless internet than in '80-'00
- Markets in this decade will be truly global
  - …and so will consumer choice!

18

VeriTest.

Thank You

VeriTest.
Outsourced Testing
Worldwide Labs
www.veritest.com

QW2002 Paper 6I1

Ms. Nancy Landau
(Alltel Technology Services)

Performance Testing Applications In Internet Time

Key Points

- Standardizing test processes.
- Profiling applications and user workflows.
- Working with diverse development teams.

Presentation Abstract

How can a test team simplify the transition from performance testing two-tier client/server applications to testing complex, multi-tier web applications combined with an exponential growth in testing needs? In this presentation, Nancy Landau presents case studies that address changes made in performance testing methods to handle compressed delivery schedules, new architectures and technologies, and changing customer expectations. The experiences focus on performance testing, but the strategies apply to all test efforts.

About the Author

Nancy Landau has 15 years of experience in quality assurance and financial services. She has been involved in design, development, deployment, test, and support of large-scale client/server solutions for the mortgage banking industry. She is the lead client/server performance test analyst for the Residential Lending Division of ALLTEL, a Fortune 500 company.

Performance Testing Applications in Internet Time

Nancy Landau

ALLTEL INFORMATION SERVICES

Objectives

- Review performance testing basics
- Describe fundamentals
- Explain success factors
- Review examples

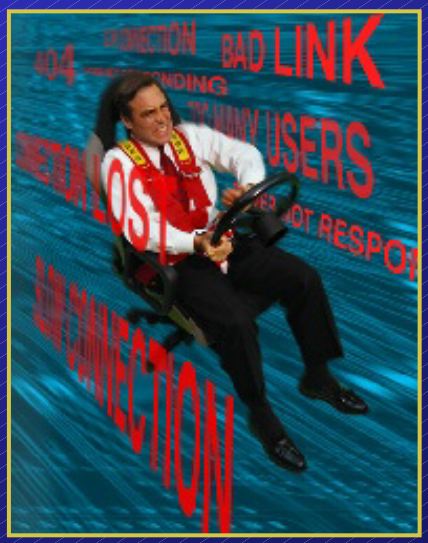
Audience: Web developer, performance engineer, stress test / QA project manager

Terms & Concepts

- Application Under Test (AUT): The software application(s) being tested.
- System Under Test (SUT): The hardware & operating environment(s) being tested.
- Virtual User: Software process that simulates real user interactions with the AUT.
- Process/Workflow: A user function within the AUT.
- Scenario: A set of workflows defined for a set of virtual users to execute.
- Transaction: A subsection of the measured workflow; more granular user events for which response time will be measured.
- Bottleneck: A load point at which the SUT/AUT suffers significant degradation.
- Breakpoint: A load point at which the SUT/AUT suffers degradation to the point of malfunction.
- Scalability: The relative ability or inability of the AUT/SUT to produce consistent measurements regardless of size of workload.

Why Performance Test?

- Internet applications bring performance issues direct to your users
- Slow response times and errors have a direct cost



Test Development

- Automated testing IS software development
- Lifecycle mirrors product development
- Use iterative test development
- Emphasize planning stages
- Plan for reuse

Plan - System Usage

- Get system usage information
  - Identify workflows
  - Define typical user profiles
  - Define transactions and expected results
- Examine typical and peak workloads
- Define access methods - connection speeds, etc.
- Trace use cases to components and hardware

Case Study #1

- Web application for data analysis reports
- Third party development, limited test environment
- Vendor's tests stated 1,000 user capacity
- Internal tests initially failed at 30 users
- Cause 1: Different model workflows
  - Report sizes
  - User actions
  - Cached data

- Cause 2: Capacity extrapolated from 30-user tests
- Lesson: understand usage patterns, avoid "good math gone bad"

Plan - AUT Architecture

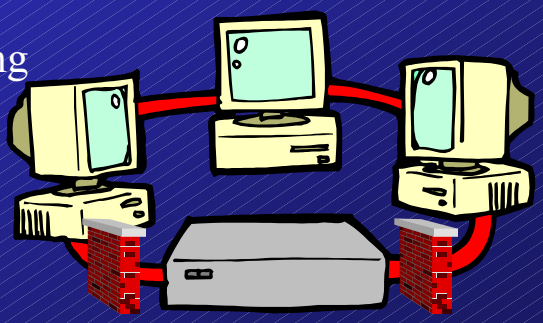
- Perform architectural walkthroughs
- Understand security methods
- Review 3rd party components
- Understand queues
- Identify caching models
- Examine session management
- Verify tool compatibility with AUT

Case Study #2

- Web-based customer service application
- 3rd party component for host connectivity
  - Bottlenecked only on multi-processor servers
  - Vendor provided recompiled component
- Delphi controls, 3270 Active-X
- GUI test tool supported both Delphi and 3270
- BUT not together!
- Lessons: know your components and your tools!

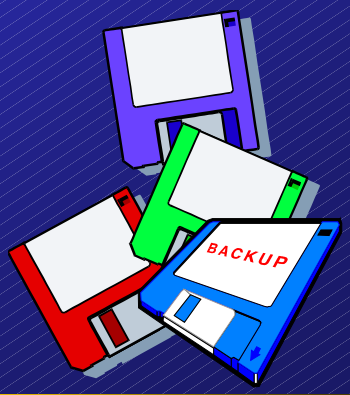
Plan - SUT Architecture

- Review physical infrastructure
- Examine firewalls
- Review connectivity
- Identify load balancing
- Review encryption



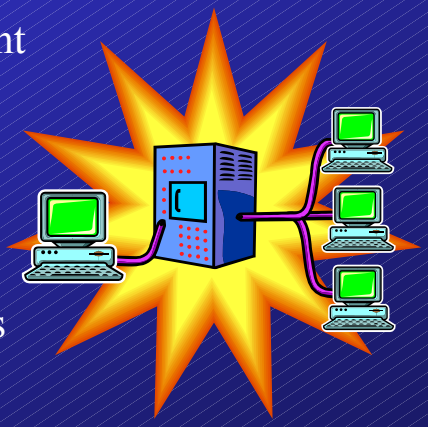
Plan - Test Data

- Define representative set of data
- Define appropriate volume of stored data
- Develop test database
- Plan backup and restoration
- Establish data verification points



Plan - Test Environment

- Obtain dedicated environment
- Typify production
  - Hardware
  - Networks
  - Databases
- Perform manual dry run tests
  - Identify concurrency risks
  - Confirm application behavior



Case Study #3

- Web-based ad-hoc reporting tool
- Development environment limited
  - Server processor speed
  - Number of processors
- Tests in development had no performance issues
- Early test in near-production environment uncovered fatal concurrency problems
- Repeatable with just one user
- Lesson: use near-production environment

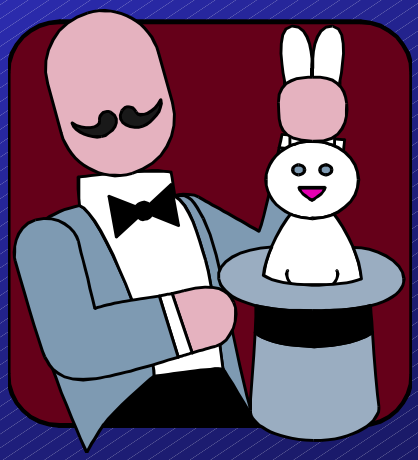
Plan - Test Metrics

- Response times – what are the AUT's targets?
- Session abandonment – when will users stop waiting and leave the site?
- Server utilization – are there target levels defined?
- Network load – how much before capacity is strained?

- Correlate test metrics to production monitoring – capture in test what you can measure in production

Plan - Toolkit

- Automated test tools
  - virtual users
  - GUI users
- Monitoring methods
  - Server performance
  - Network load
  - Client / virtual user driver
- Logs and log parsers
- Synchronize the measurements!



Create Virtual Users

- Record user actions
- Define wait / think times
- Add transactions
- Add verification checks
- Parameterize data

Create Scenarios

- Establish mix of virtual users
- Ensure a varied, representative workflow
- Establish monitoring points

Perform Dry Runs

- “Test the test”
- Use full logging
- Identify unexpected conditions
- Review instrumentation and monitors
- Validate parameterized data
- Revise test scripts

Case Study #4

- Web-based customer service application
- Web server interpreted response from application server as success
- Application server returned errors in the response
- Examined return data to identify “true” success
- Lesson: HTTP 200 is not always success! Validate responses against expected results

Perform Tests

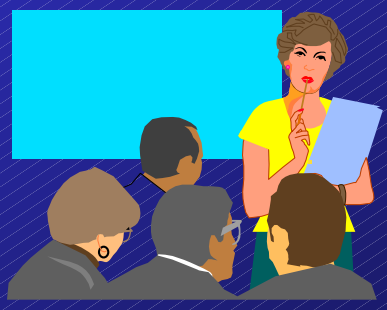
- Reduce logging to production levels
- Ramp-up virtual users
- LAN versus WAN tests
- Identify and troubleshoot bottlenecks
- Track all changes

Analyze Results

- User response times
  - Averages
  - 75th percentile or higher
- Memory utilization
- CPU utilization
- Server configuration
- Database and SQL tuning
- Code tuning

Report the Results

- Report components
  - Executive summary
  - Report body
  - Appendices
- Identify the audience
- Mirror the test plan
- Acknowledge contributions



Rinse and Repeat

- Emphasize reuse
- Develop plan templates
- Create report templates
- Develop application matrix
- Develop reusable test components
- Define standard measurements
- Define and share toolkit

Conclusion

- Automated testing is development
- Planning is essential
- Plan for reuse
- Profile the users
- Learn the technologies
- Understand the infrastructure

Questions?

Thank You!

Nancy Landau

Nancy.Landau@ALLTEL.com

ALLTEL INFORMATION SERVICES

QW2002 Paper SB1

Dr. Chris Overton
(Keynote Systems, Inc.)

Comparative Strategies for Measuring the Internet: The Whole is More than the Sum of the P

Key Points

- Here's how to make commercial implications of Web Performance directly tangible
- Integration of measurement modalities is tricky but highly rewarding
- Blind sports and complementarities of the measurement modalities

Presentation Abstract

One may classify commercially feasible measurements of Internet performance into three broad categories, which may be placed along a continuum: behind-the-firewall, synthetic, and end-user. As representatives of industry leaders in these measurements approaches, we set out to compare benefits and blind spots of the approaches and to see how they complement each other. THIS IS NOT A MARKETING TALK, but rather an in-depth, balanced discussion of success stories and open questions, both from technical and business perspectives. We work our way up from statistical prerequisites to a surprising ability to correlate traditional technical data (such as downtime, other performance problems, and design issues) with traditional Marketing metrics (such as ROI estimates.) We use an interactive bird's-eye view of multiple data sources that allows an unaccustomed ability to spot relationships. Understanding the full spectrum of measurement integrations will also help prioritize within less ambitious performance management strategies. In addition, this clarifies desired future progress.

About the Author

Chris Overton is Keynote Statistician & Quantitative Architect. Over the last decade, he has consulted as a statistician in industry and in academic biomedicine, as a software architect and developer, and in business model development. He founded Crazy Tulip Corp. to build knowledge modeling software systems. His responsibilities at Keynote include algorithm and tool design, data analysis & interpretation, internal & external education, and serving as academic liaison. He architected Keynote's SLA reporting engine and has helped several large companies build SLAs both on the provider side and on the customer side. Chris is the principal architect of Keynote's streaming media metrics and has published and lectured on related topics. He got his pure math PhD from Stanford in 1996 and has taught there and at the University of San Francisco.

Chris Bubinas is webHancer's lead Performance Assessment and Customer Impact Analyst. For over four years he has consulted in network deployment and in bringing companies? business operations to the Net. He joined webHancer near inception, sharing the vision that online businesses require and deserve a better understanding of the Web performance their customers experience. His contributions to webHancer involve data mining and analysis, building strategies to measure performance impact on audience behavior, and custom performance measurement projects with some of the Internet's largest players. Mr Bubinas resides in Ottawa, Canada from where he avidly launches the sounds of his cyber-industrial musical projects into cyberspace.

James W. MacIntyre, IV is cofounder and chief executive officer of Visual Sciences, Inc., which offers industry-leading visualization and interactivity for several data domains, especially for various types of Internet measurements. He is a member of the General Partner at NextPoint Partners, L.P., a seed and early-stage venture capital fund. A serial entrepreneur, Mr. MacIntyre also served as cofounder, chief executive officer, or general manager of a number of companies, including OneSoft Corporation, an e-commerce and managed services company; TGF Technologies, Inc., a software development company; Together Networks, a regional Internet service provider; Computer and Communications International, a

Venezuelan Internet infrastructure company, and Comprehensive Data Systems, a software applications company. Mr. MacIntyre holds a B.A. in philosophy and economics from the University of Vermont.

Comparative Strategies for Measuring the Internet:

The Whole is More than the Sum of the Parts

Chris Overton, Ph.D.
Keynote Systems
Chris Bubinas
Webhancer
James MacIntyre
CEO, Visual Sciences

Agenda

- Introduction
- Two schools of Internet-related measurement
- Discussion of the "three measurement modalities":
  - "Behind-the-firewall"
  - Synthetic measurement
  - End-user measurement
- Important gaps to fill; important synergies
- Possible data integrations already starting to happen in industry

Introduction

Too many talks in this industry say something like:

- In 2001, $Y Billion was spent on this aspect of generating and delivering Internet content
- This is projected to grow to $Z Billion by the year 200X
- So therefore, you have to pay attention to this particular optimization or service that I'm offering…

Introduction (II)

- But what fraction of these large expenditures is understood quantitatively in the following terms?
  - What value it brings
  - What value is lost for "dumb" reasons
    - e. g. availability and performance problems
  - Much harder: what value is lost due to omissions
    - e. g. what *potential* value is lost due to what is *not* offered

Introduction (II)

- But what fraction of these large expenditures is understood quantitatively in the following terms?
  - What value it brings
  - What value is lost for "dumb" reasons
    - e. g. availability and performance problems
  - Much harder: what value is lost due to omissions
    - e. g. what *potential* value is lost due to what is *not* offered
- This speaker would assess current understanding in these areas as pathetic

Introduction (III)

- *Ad hoc studies* by consulting agencies may make varying degrees of progress in answering these questions, but:
  - Not in much of a *standardized*, *general*, or *scalable* way
  - Usually not reaching conclusions earlier than months after gathering data
- More automated conclusions closer to real-time have to derive from measurement data

Introduction (IV)

- In this talk, we'll show how each class of measurement strategy fails to provide comprehensive understanding
- However, we'll also demonstrate how their appropriate combination can solve the puzzle of understanding market impact of Internet presence, *including missed potential value*
- One may also use these techniques to quantify "softer" areas such as site design and navigability
- Leading companies are beginning to figure this out, suggesting this will soon become a competitive necessity

Agenda

- Introduction
- **Two schools of Internet-related measurement**
- Discussion of the "three measurement modalities":
  - "Behind-the-firewall"
  - Synthetic measurement
  - End-user measurement
- Important gaps to fill; important synergies
- Possible data integrations already starting to happen in industry

The traditional challenges in managing a web presence

1. Operations: understand **in real time** what is going on so you can fix/manage it
2. Product Management/Marketing: ("PMM") understand the **financial impact** of what you are doing and not doing
   - At a planning stage (months in advance)
   - Understand how to spin the current situation
     - In this context, "current" usually means a time scale of weeks to months

Ops vs. PMM goals

- Different time scales lead to different approaches and different data
- However, market impact of smaller time scale occurrences call for marketing interest in understanding them and in developing/automating response strategies

Ops goals

- Ops tends to get saddled with responsibilities for products prioritized and built elsewhere; focus is on day-to-day (minute-to-minute) management
- Metrics tend to be some proxies for availability and maybe even performance
- Market for ops-level measurement seems relatively mature, but discontinuous with PMM metrics

Product management goals

- Plan for execution that is delayed by development and deployment times
- The goal is to understand market impact, but errors arise due to time lag, prediction uncertainty, and "softer" subject area of human behavior
- Thus metrics aspire to predict market impact ("predicted profit"), but lack trustworthiness and rigor
- Real-time knowledge is not "actionable" and so is less relevant

What is the market for quantifying Internet-related performance for PMM clients?

- Due to perceived budgetary control by PMM, many measurement companies and services aspire to serve their needs
- However, most measurement services seem to serve ops clients
- Fancier reporting and research for PMM clients appears less scalable

We will argue for a kind of integration that not only addresses both client sets, but benefits them in the new way of getting them to speak the same language

Agenda

- Introduction
- Two schools of Internet-related measurement
- **Discussion of the "three measurement modalities":**
  - "Behind-the-firewall"
  - Synthetic measurement
  - End-user measurement
- Important gaps to fill; important synergies
- Possible data integrations already starting to happen in industry

The three measurement modalities

We now talk about what actually can be measured…

This is important preparation for what can be gained by combining modalities, so we'll need to be thorough

We define the modalities and then compare both what they *do* measure (capabilities) and what they *do not* (limitations, difficulties)

The three measurement modalities

- Behind-the-firewall ("BTF")
- Synthetic measurement, including (part of) Internet
- End-user measurement

The three measurement modalities

Basic definitions:

- Behind-the-firewall
  - Anything derivable from within a company's web site
- Synthetic measurement over Internet
  - Anything measurable by artificial web traffic
- End-user measurement
  - Anything derivable from measurement of real users (generally acting voluntarily, since non-voluntary behavior is much cheaper to run synthetically)

The three measurement modalities

Main experience of authors' companies

- Behind-the-firewall
  - Visual Science
- Synthetic measurement over Internet
  - Keynote
- End-user measurement
  - Webhancer

Behind-the-firewall capabilities

- Access to internal corporate information, such as purchasing behavior
- Proprietary user behavior within domain, before encryption
- View deeper inside applications, allowing more granular understanding of performance
- Substantial partial insight into end-user experience is possible by analyzing packet transit
- Low performance and bandwidth overhead for measurement

Synthetic measurement capabilities

- More realistic than BTF for response time and especially for availability, given transit over Internet
  - Especially for for load testing
- Easily standardized sampling for simpler comparability ("benchmarking")
- Customizable measurement is able to focus on specific functionalities, useful for SLA's (such as intra-company)
- Visibility into optimizations due to distributed server locations (CDN's)

End-user measurement capabilities

- Most accurate measurement of actual user experience
- More complete view of voluntary user behavior than possible from behind the firewall:
  - What competitors does click-away go to?
  - User-side unavailability offers truer picture
  - More holistic picture of user traffic
- Subject to privacy limitations, view of unencrypted data at user end
- Flux and distribution of user load gives a way to prioritize and weight synthetic measurements

Behind-the-firewall difficulties

- If a user abandons, you don't fully understand when or why, and you certainly don't understand what they do instead
- Without measuring over the Internet, you're missing a big enough piece of the puzzle that this clouds your conclusions, especially for distributed content
  - You may not know how users are transiting the Internet, making it difficult to detect and repair Internet problems
- Lack of visibility into competitors
- Most fundamentally: lack of visibility into what you are "missing"

Synthetic measurement difficulties

- Need server-side or user-side data to know you are measuring the right thing
  - Bias introduced unless data weighted by user traffic!
- You generally sample a small fraction of possible pages/behaviors
- You necessarily sample from a small fraction of possible locations
- Either the faster and more important backbone transit gets lost in last-mile noise, or else you sample only a small fraction of transit paths
  - As measured in the metric of "**packet seconds**": where packets spend their transit time
- Measurement bandwidth cost is very much of a limiting factor, so intelligent sampling optimization is crucial

End-user measurement difficulties

- This richest data set is also the noisiest – confounded e.g. by only partially known user hardware specs, or by performance degradation due to simultaneously running software
- This richest data set is the hardest to mine; best practices are still evolving substantially, even at industry leaders
- User-panel generally suffers self-selection bias: most lucrative users are probably less likely to let you snoop their behavior; biases are time-varying!
- Trust issues generally imply anonymization – a limitation as compared to behind-the-firewall
- Low-traffic sites may not allow granular conclusions
- Without synthetic overhead (such as tracert), data are less actionable

Summary of measurement modalities

Modality	End-user (panel)	Synthetic (across Internet)	Server-side (behind firewall – "BTF")
Pros	real performance & behavior; maybe follow user click-away; high data depth	easy to schedule; understandable Internet latency; standardized	managed by host, so no privacy concern; granular view inside applications
Cons	self-selection bias; last-mile noise unwieldy data; privacy constraints; limited visibility into smaller sites	artificial traffic; limited scope of behaviors; Partial view of Internet; measurement cost	must infer much of Internet latency, including user loc. limited view of user click-away, competitors

Agenda

- Introduction
- Two schools of Internet-related measurement
- Discussion of the "three measurement modalities":
  - "Behind-the-firewall"
  - Synthetic measurement
  - End-user measurement
- **Important gaps to fill; important synergies**
- Possible data integrations already starting to happen in industry

Important gaps to fill; important synergies

- Before we showcase the "dream team", let's suggest strategies for filling gaps arising from use of a single modality
- For this section, we'll assume a reasonable single-modality measurement strategy is already in place
  - A huge assumption, by the way!
- We'll always assume some routine behind-the-firewall measurement

Additions to synthetic measurement

- A reasonable starting point would include regular synthetic measurement having alarming
- Knowing how to weight measurements depends critically on an understanding of one's user base, available either through end-user of BTF
- Substantial savings may be realized in choice of measurement locations based on this knowledge
- Intelligent alarming can benefit from synthetic + BTF by using a BTF process of elimination to build evidence for specific Internet problems
- Suppose you're a synthetic measurement company trying to decide what to measure (i.e. without BTF knowledge.) This is a great use for end-user measurement!
  "The boomerang effect"
- Correlate with BTF to obtain a (weak) predictor of lost value based on synthetic measurement; use this to justify PMM decisions

Additions to BTF measurement

- Use synthetic & end-user to manage outside the firewall
- Use synthetic over long term (sparsely sampled) to choose providers and understand changing performance
  - Beware long-term shift in synthetic
- Synthetic useful to help quantify BTF improvements
- Use end-user to quantify competitive threats
- Use synthetic to understand competitive performance
  - But needed in much less granularity than one's own!
- End-user can give most timely problem indicator: a rapidly falling user load

Additions to end-user measurement

Most promising area for development:

1. Disentangle shifting biases by correlation with synthetic
2. QUANTIFY DOLLAR VALUE OF PERFORMANCE/AVAILABILITY/OTHER VARIATION BY INTEGRATION WITH BTF
3. THEN, USE THIS TO INFER ADDITIONAL LOST VALUE DUE TO USER ABANDONMENT IN FAVOR OF COMPETITORS

Agenda

- Introduction
- Two schools of Internet-related measurement
- Discussion of the "three measurement modalities":
- "Behind-the-firewall"
  - Synthetic measurement
  - End-user measurement
- Important gaps to fill; important synergies
- **Possible data integrations already starting to happen in industry**

The solution to all the world's problems (in measuring value of web investments)

- We have already seen that measurement strategies starting from one modality lead naturally to include the others
- A full combination may be cost-prohibitive for smaller companies (and infeasible for companies whose web traffic is too small for end-user to be sufficiently granular)
- The approach here is still usually not fully realized
- However, some of the largest companies are beginning to put these pieces together, suggesting the knowledge will "trickle down"

Step one: dollar-based metrics in near-real time

- Any metrics in units of dollars seem more suspect than metrics in units of time, because they are either operationally irrelevant (occurring over large time intervals or else being "bursty"), or mere marketing approximations
- However, assuming financial events can be monitored and aggregated in near-real time, the results can be quite dramatic

Step one: dollar-based metrics in near-real time

<demo>: "conversion" events for a financial company

Step two: correlation of web structure with financial metric

- By identifying which web pages lead to financial events, very dramatic consequences of web site design may become apparent
- This is still possible without use of synthetic or end-user measurements

Step three: correlation of BTF with end-user and/or synthetic

- It seems enough of a challenge to correlate only performance numbers
- However, if end-user measurements are assumed to reflect the experience of the entire set of users, this can be correlated with financial metrics (as calculated BTF, possibly with other predictors) to estimate actual value of performance
- When investments are then considered, specified performance improvements can then be cost-justified and perhaps even contracted for

Step four: build in competitive abandonment to quantify loss to competitors

- Accurate knowledge on this front requires end-user data
- Usually, one will start with unjustifiable simplifying assumptions to make a first approximation
  - For example, all user visits are independent
- This allows first-pass estimation of competitive landscape and its sensitivity to web experience

Step five (advanced): more accurate end-user + synthetic measurement integration

- Since each modality has its biases, it follows that more accurate results are possible by appropriate analysis of results
- This greater accuracy is possible without additional measurement cost, but benefits most from some knowledge of user geography
- This is an area of continuing development, but adds more to *accuracy* (and thus to cost-benefit) as opposed to *functionality*

Conclusions

- It is now possible for reasonably accurate dollar-unit metrics to be used near real-time and in an operations context
- At the very least, these are useful in an ops context to prioritize problem solving
- Gradually, real-time dollar metrics can narrow the gap between ops and PMM
- Appropriate measurement integration is vital to this process

Author Contact Info

- Synthetic measurement over Internet
  - Chris Overton, Keynote Statistician
  - coverton@crazytulip.com, coverton@keynote.com
- "Behind-the-firewall"
  - Jim MacIntyre, CEO, Visual Sciences
  - Jim@visualsciences.net
- End-user measurement
  - Chris Bubinas, Webhancer
  - cbubinas@Webhancer.com

QW2002 Paper 7I1

Mr. Philip Joung
(Caw Networks)

Get Real! The Importance of Realism in Web Site Capacity Assessment

Key Points

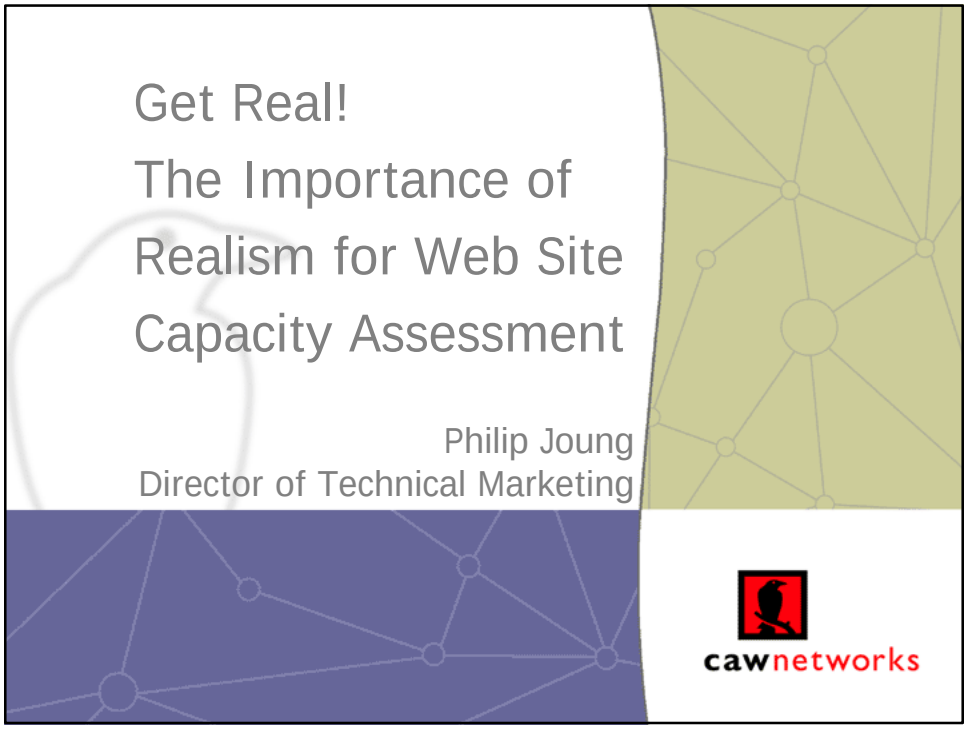
- Achieving reliable Web site load testing results means generating realistic traffic
- It must model both user behavior and network behavior, currently uncommon in Web testing
- Network issues(packet loss, link latency, jitter, etc) significantly affect performance

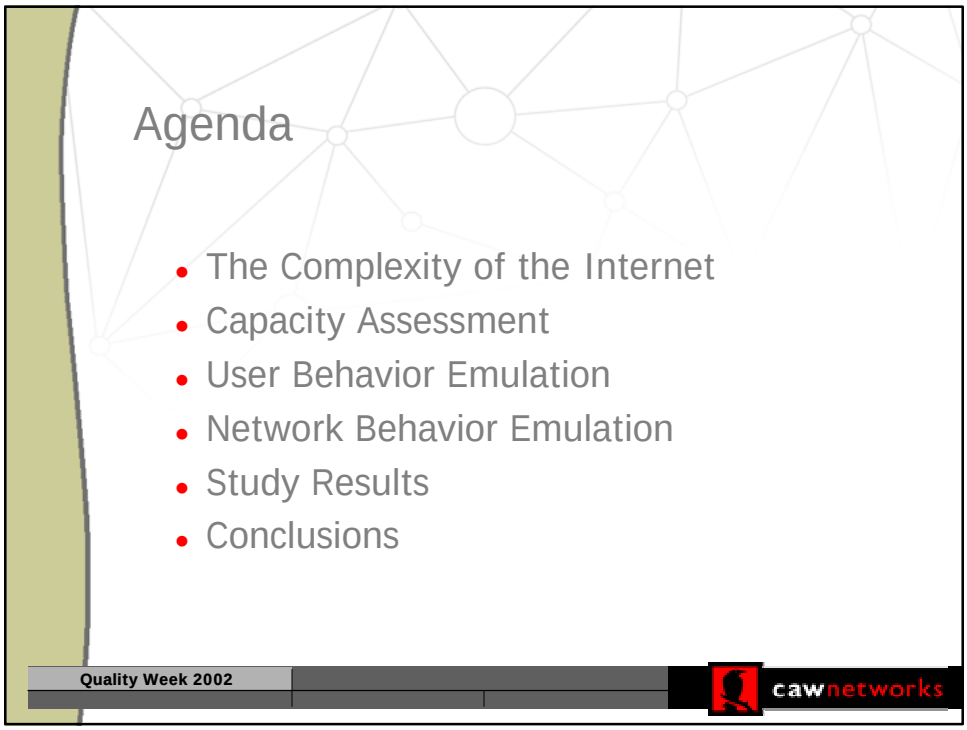
Presentation Abstract

What is real world capacity assessment? It is carefully combining the aspects of not only user realism, but network realism in order to achieve reliable, accurate results during Web site and network performance testing. User and network realism includes important measures such as packet loss, link speeds, millions of IP addresses, browser emulation, SSL, and other factors that can cause significant performance issues. The Internet has all these issues, and testing with these values are an important part of gaining a clear picture of system performance from the start and in the safety of the test lab. In-house and industry studies have shown dramatic performance issues with even small amounts of network realism thrown in: NASA has shown that only 3% packet loss causes a 50% drop in FTP performance.

About the Author

Philip Joung has worked in various IT related jobs since 1987. After working at a CASE tool software consultancy for two years, he helped start the company Embark.com (which still thrives today as the largest, most trafficked Web site for students going to college) in 1996. He joined as the Webmaster to run the technology side of the 7 person company. By the end of 1999, the company had 300 people, and Mr. Joung was the Chief Systems Architect. He ran Embark.com's IT systems, and in 1999, grew the network to a large $12 million network operations center with 150 servers, redundant Internet lines, 2 tons of battery backups and a large, 1-ton generator. Scaling this network required testing, but often the testing did not provide an accurate picture of the performance in production.
After four and a half years with Embark.com, Mr. Joung left to join Caw Networks, a company devoted to finding the true capacity of network systems and Web sites. He took this opportunity to help them develop a solution that would solve many of the scalability headaches at Embark.com. Mr. Joung is now helping to get the word out about this company's products and the reasons why user and network realism matters.
Philip Joung has presented at various technical seminars internal to the companies he has worked at. At Caw Networks, he's presented at various trade shows such as N+I as a demonstrator and more than a dozen Caw Networks customer workshops. He is also scheduled to speak at STAREast in May, at N+I Tokyo in July, and CMG 2002 in December.





Philip Joung, pjoung@caw.com, Caw Networks

The Internet

- Network made up of millions of components
  - Routers, caches, modems, switches, hubs, servers, wireless, etc.
- Web sites with complex, multi-tier architectures
- Millions of users
  - Personalization
  - Security
  - Complex functionality
  - Different browsers, connection speeds
  - Erratic behavior

Quality Week 2002

cawnetworks

With all the Variables, What Can be Done to Improve Performance?

- Faster
  - Equipment
  - Network
- More
  - Equipment
  - Network connections
- Redesign
  - Applications/database
  - Network
- All are costly and complex to do, and often risky

Quality Week 2002

cawnetworks

Capacity Assessment to the Rescue!

- Compares different design prototypes
- Helps tune the system
- Validates vendor performance claims
- Demonstrates performance of the final system
- Ensures ongoing performance in the system

Traditional Capacity Assessment

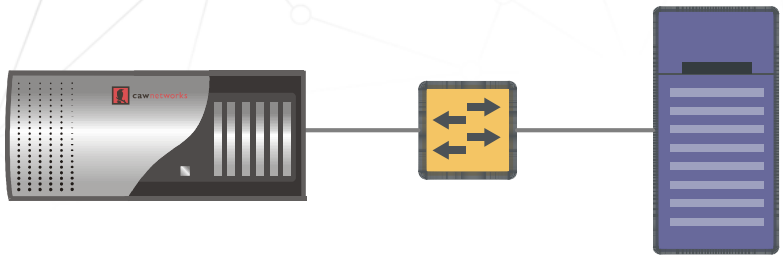
- Primarily software based
- Careful attention to modeling user behavior
  - Scripts of what the simulated user does
  - Browser version
  - SSL usage
- Increase amounts of load applied to
  - Find errors
  - Discover bottlenecks
  - Cause system failure

Philip Joung, pjoung@caw.com, Caw Networks

What's Missing?

- The network side of the equation
  - Packet loss
  - Realistic link speed
  - Others...
- Testing with traffic equal to expected loads rather than extrapolating

Test Network



WebAvalanche 1822, version 3.1
Capacity Assessment Appliance

Netgear 10/100
Ethernet Switch

850 MHz PIII
1 GB RAM
Windows 2000
IIS 5.0

Philip Joung, pjoung@caw.com, Caw Networks

Test Settings

- Test Duration: 148 seconds
- Files: 1 byte, 15 Kb, 1500 bytes, 1 byte, 1 Kb, 100 bytes
- No think time
- HTTP 1.1 protocol
- No packet loss
- No link latency (full network speed)
- Increase load to 3,100 users over 128 seconds

Quality Week 2002

What's In This Test?

- Under clean conditions, the test network exhibits no failures
- Successes, failures and timings are recorded
- One condition is altered per assessment, and the results, including changes, recorded

Quality Week 2002

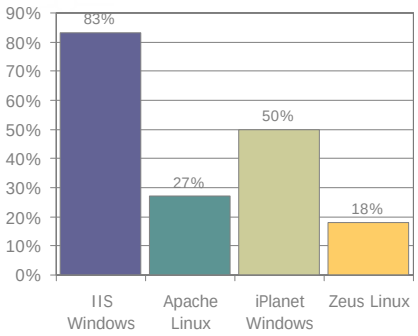
Clean Run Results

- Attempted number of transactions: 275,010
- Successful number of transactions: 275,010
- Successful transactions/sec: 1,858
- Ave. est. server processing time: 0.003 ms
- Total number of packets: 1,915,904
- Maximum average bandwidth: 58.64 Mbps

User Path and Behavior

- Dynamic content much more intensive than static HTML
- What a user does on a Web site matters
  - Searches, logins, purchases all create load

Decrease in Performance of Dynamic CGI vs. Static Files



* Source: Server's Advantage, PC Magazine, January 15, 2002

Philip Joung, pjoung@caw.com, Caw Networks

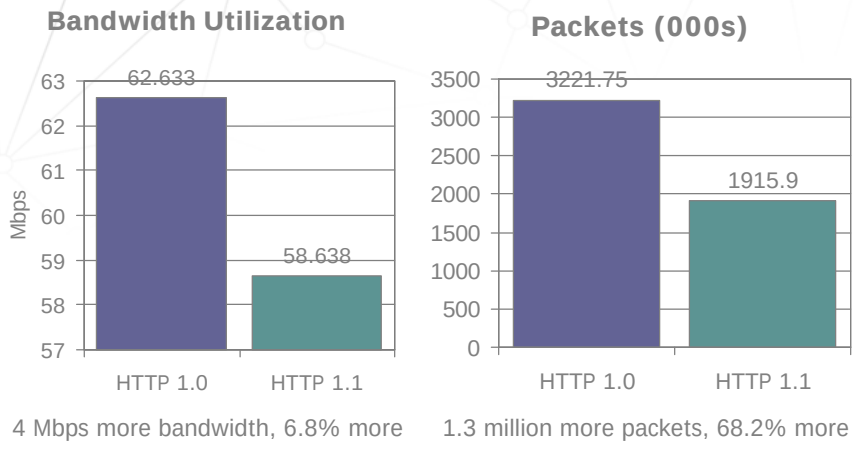
Browser Version
HTTP 1.0 and HTTP 1.1

- Browsers use particular versions of HTTP
- HTTP 1.0
  - Uses separate network connections to transfer objects (HTML page, graphic, etc)
  - Some browsers open multiple connections
- HTTP 1.1
  - Uses connection persistence, where one network connection can transfer multiple objects
  - Helps reduce overhead of setting up and tearing down network connections

Quality Week 2002

caw networks

HTTP 1.0 versus HTTP 1.1



4 Mbps more bandwidth, 6.8% more

1.3 million more packets, 68.2% more

* Source: Caw Networks Realism Study, February, 2002

Quality Week 2002

caw networks

Security

- Browsers primarily use secure socket layers, or SSL, for security
- Employs various encryption methods to improve security
- Increases processing time on both server and client
- Increases bandwidth utilization

SSL's Performance Overhead

32 times fewer

335 times more processing time

2.2 times more bandwidth

* Source: Caw Networks Realism Study, February 2002

Philip Joung, pjoung@caw.com, Caw Networks

Other User Behaviors to Consider

- Different activities
  - Streaming Media
  - FTP
  - Email
- User frustration
- Cookie usage
- User "think time"

Network Behavior

- Packet loss
- Link speeds
- More later...

Philip Joung, pjoung@caw.com, Caw Networks

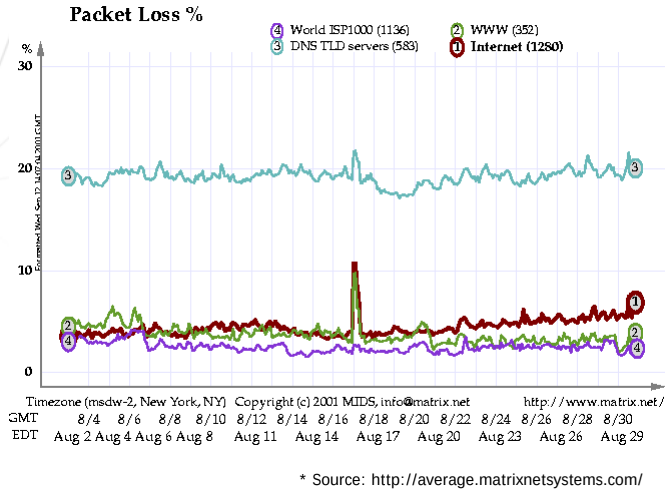
What is Packet Loss?

- The Internet carries its data in the form of packets
- Network congestion
  - Networks discard or deny packets on the network that it can't handle
- Errors in transmission

Quality Week 2002

caw networks

Packet Loss on the Internet

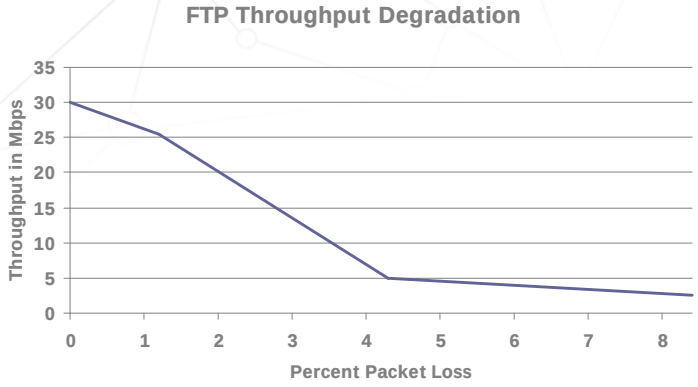


* Source: http://average.matrixnetsystems.com/

Quality Week 2002

caw networks

Performance with Packet Loss According to NASA



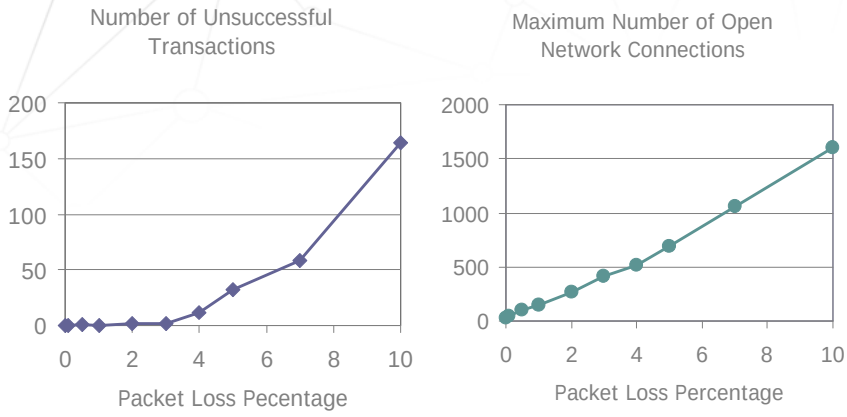
3% packet loss halves performance

* Source: The Effect of Packet Loss on TCP Application Performance, NASA Integrated Services Network, December 11, 1998

Quality Week 2002

cawnetworks

Packet Loss Effect on Web Servers

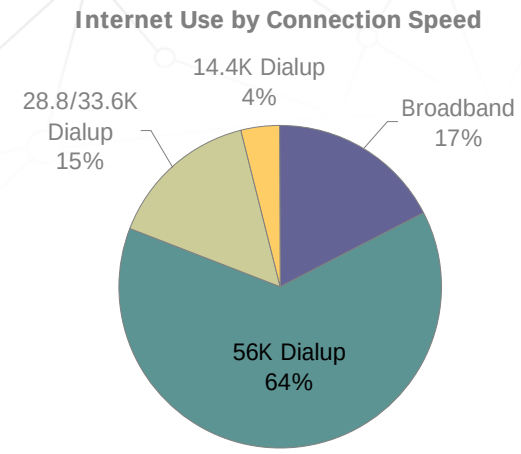


* Source: Caw Networks Realism Study, February 2002

Quality Week 2002

cawnetworks

Link Speeds



* Source: Nielsen/Netratings, July 2001

Quality Week 2002

cawnetworks

Why Does Link Speed Matter?

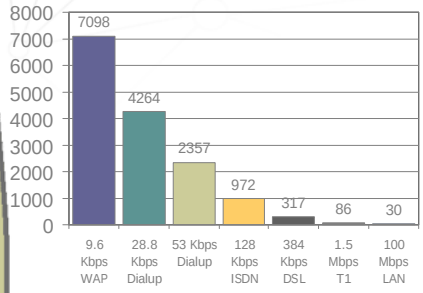
- Slow connections increase server load
- Differing link speeds exercise the network differently
- Modeling link speed allows one to accurately gauge user experience

Quality Week 2002

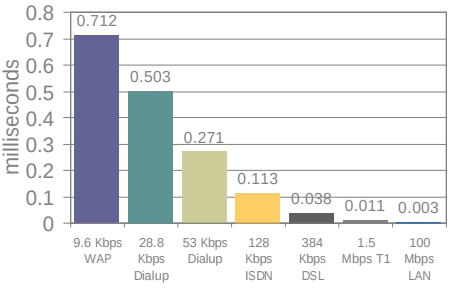
cawnetworks

The Effect of Link Speed

Maximum Open Network Connections



Average Estimated Server Processing Time



* Source: Caw Networks Realism Study, February 2002

Quality Week 2002

cawnetworks

What Other Network Issues Matter?

- IP addresses
- IP fragmentation
- Jitter
- Bursty traffic patterns

Quality Week 2002

cawnetworks

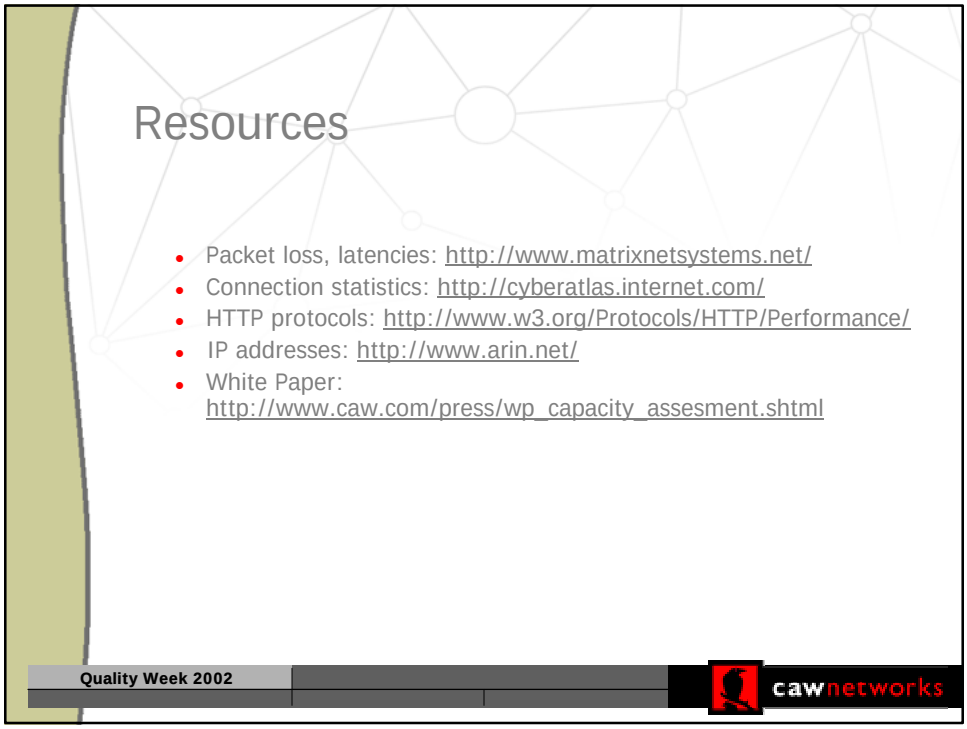
Network and Application Performance Closely Related

- What happens on the application affects the network
  - User behavior
  - File sizes
  - Security
- Network also significantly affects application
  - Memory and CPU utilization
  - Stability

Conclusions

- Capacity assessment can make designing, scaling and maintaining a system much easier and more effective
- Rigorous capacity assessment requires emulating
  - User behavior
  - Network behavior
- Network behavior can dramatically affect application performance
- The combination creates a much more reliable and accurate set of results

Philip Joung, pjoung@caw.com, Caw Networks


Resources
Packet loss, latencies: http://www.matrixnetsystems.net/
Connection statistics: http://cyberatlas.internet.com/
HTTP protocols: http://www.w3.org/Protocols/HTTP/Performance/
IP addresses: http://www.arin.net/
White Paper: http://www.caw.com/press/wp_capacity_assesment.shtml
Quality Week 2002
cawnetworks

Thank You!
pjoung@caw.com if you have questions or want to discuss this further

QW2002 Paper 7I2

Eric D. Siegel
(Keynote Systems)

Don't Get Trampled by the Crowd: Realistic Load Testing of Web Sites Across the Internet

Key Points

- There are critical differences between Web and legacy load-test environments
- The Internet's internal structure must be considered for realistic load testing
- Practical recommendations for highly-realistic load testing of Web sites

Presentation Abstract

Use of traditional tools and techniques designed for clientserver or terminal-host systems on dedicated networks will almost certainly produce misleading results when used on Web systems. During this intensive session we'll look at the reasons why traditional load tests fail to produce realistic loads for Web applications and how such flaws lead to deceptive conclusions and inaccurate business and technical decisions. We'll look at the difficulties caused by abandonment, poor session structures, diffused servers, and lack of direct control over many Web and Internet resources. We'll also examine relevant portions of the Internet's structure (DNS, peering, caching, third-party servers and content distribution networks, server system connectivity, etc.) that may impact the end-user's perception of your Web site's performance and that must be considered in designing an accurate test. We'll then give detailed recommendations about how to construct load tests that realistically depict both the true end-user load and the true end-user experience.

About the Author

Eric Siegel, Principal Internet Consultant at Keynote Systems, has been a member of the Internet community since 1978. He is the author of "Designing Quality of Service Solutions for the Enterprise" (John Wiley & Sons) and is an instructor and panelist in Internet performance and QoS at major industry conferences such as Networld+Interop, CA World, Service Networks, Quality Week, WWW Conferences, and CMG. Before joining Keynote, Mr. Siegel was a Senior Network Analyst at NetReference, Inc., where he specialized in network architectural design for Fortune 100 companies, and he was a Senior Network Architect with Tandem Computers, where he was the technical leader and coordinator for all of Tandem's data communications specialists worldwide. Mr. Siegel also worked for Network Strategies, Inc. and for the MITRE Corporation, where he specialized in computer network design and performance evaluation. Mr. Siegel received his B.S. and M.E.E. degrees in Electrical Engineering from Cornell University, where he was elected to the Electrical Engineering honor society.

(7I2) Don't Get Trampled by the Crowd:

Realistic Load Testing of Web Sites Across the Internet

Eric Siegel

Principal Internet Consultant
Keynote Systems
777 Mariners Island Blvd
San Mateo, CA 94404

eric.siegel@keynote.com
http://www.keynote.com

KEYNOTE The Internet Performance Authority

Agenda

- **Performance Challenges on the Web**
- **Web Architectural Review**
- **Web Load Testing**
- **Internet Statistics**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Performance Is Important!

- **"Twenty-eight percent of shoppers who have suffered failed performance attempts said they stopped shopping at the Web site where they had problems, and six percent said they stopped buying at that particular company's off-line store."** (Boston Consulting Group, quoted in Infoworld / Computerworld 3/00)
- **"[One site] found the bailout rate fell immediately from 30% to 6-8% just because of one tiny second of load time!"** (Zona Research 4/99)
- **"Surfers say that slow-loading Web sites are the biggest cause of irritation ... Seven percent of respondents say they hit their equipment ... 2% say they've become so upset they've hit the person who sits next to them."** (Market & Opinion Research International 2/02)
- **"Perhaps as much as $4.35 billion in e-commerce sales in the U.S. may be lost each year due to unacceptable download speeds and resulting user bailout behaviors."** (Zona Research 4/99)

KEYNOTE The Internet Performance Authority

A Definition of Performance

- **Web e-commerce performance measures the user's experience interacting with your Web site, *not your in-house experience or the experience inside your Web hosting center.***
  - **Download time**
    - The dissatisfaction and abandonment points are different for different types of pages and different classes of users
  - **Transaction Time**
    - banking, stock trading, purchasing
  - **Availability**
  - **Errors**
    - Failed connection attempts
    - Missing or incorrect pages
    - Missing page components
    - Broken links
    - Transaction failure

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Performance Challenges

- 24x7 availability and geographic distribution; expectation of universal access
- A shared network resource
- No control over customers' environment
- Multiple servers and caches, which may be geographically distributed, participate in a single user interaction
- Poor support for session structures; difficult or impossible to detect transaction abandonment
- Potentially massive peak volumes ("flash" loads)

The Web and Internet are very different from legacy client:server and terminal:mainframe environments!

KEYNOTE The Internet Performance Authority

Agenda

- Performance Challenges on the Web
- Web Architectural Review
  - Overall Web and Internet Architecture
  - Servers
  - Internet Backbones and Peering
  - Caching
  - Content Distribution Networks
- Web Load Testing
- Internet Statistics

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

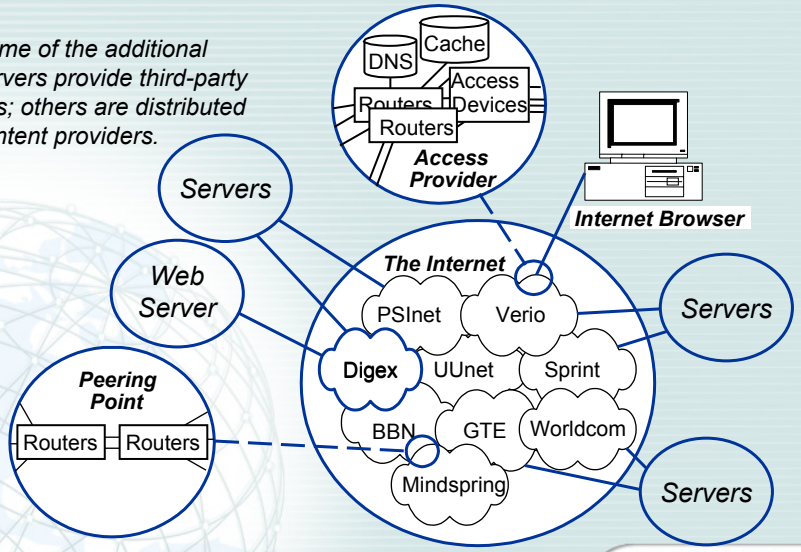
Web Page Technology Overview

- The Domain Name System (DNS), a worldwide hierarchy of directories, translates fang.dog.com into 10.9.23.22.
- TCP/IP carries the data between your browser and 10.9.23.22; it detects errors and corrects them by retransmitting.
- The data consists of HTTP, HTML, and the page's information.
- HTTP (Hypertext Transfer Protocol) carries the Hypertext Markup Language (HTML) and provides the basic Web page commands:
  - GET
  - Query String (e.g., GET fang.dog.com/filename?fur=matted)
  - POST
- HTML describes the page:
  - Formatting
  - Content, and the servers/files from which that content can be downloaded (e.g., pix.fang.dog.com/gifs/picture1.gif)
  - Links

KEYNOTE The Internet Performance Authority

Internet Architecture Overview

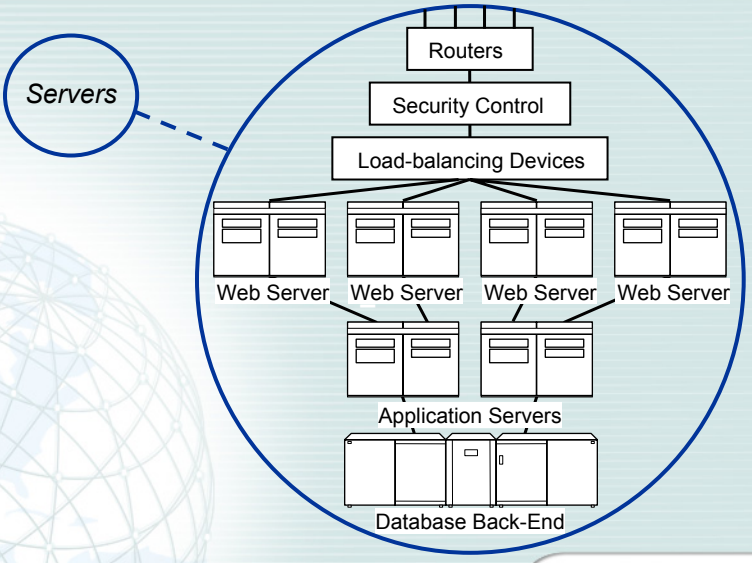
Some of the additional servers provide third-party ads; others are distributed content providers.



KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Server Farm Architecture



KEYNOTE The Internet Performance Authority

Servers: Load Balancing

- An attempt to distribute load reasonably across local resources
  - Often looks at server performance before allocating incoming load
  - May need to maintain transaction context (next slides)
- Flash load handling:
  - Sends overflow to overflow servers with special content
  - Automatically replicates frequently-requested files onto overflow servers

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Servers:
Web Transactions – 1

- ***Scaling transactions is much more difficult than scaling simple Web page delivery!***

- **Need to maintain transaction context between Web pages, associating a user with a transaction**
  - **Use remote IP address?**
    - Different users of one ISP can have same address
    - Users can switch IP addresses in mid-transaction
  - **Use a cookie?**
    - Set cookie when user first appears, or after login
    - Cookie remembers transaction ID, etc.
    - User can set browser to refuse cookies
    - Load balancing devices can't handle encrypted cookies unless they can decrypt SSL

KEYNOTE The Internet Performance Authority

Servers:
Web Transactions – 2

  - **Embed user, state information within each page and link?**
    - Requires dynamic page generation
    - Page is generated with each link modified to contain user and state information:

```
<A HREF="dog.htm?userID=123abc">
```

      or with custom-built POST form data:

```
<FORM NAME=FANG METHOD=POST ACTION=dog.htm>
<INPUT TYPE=HIDDEN NAME="userID" VALUE="123abc">
...
</FORM>
```

    - Load balancing devices can't read a URL or page data unless they can decrypt SSL

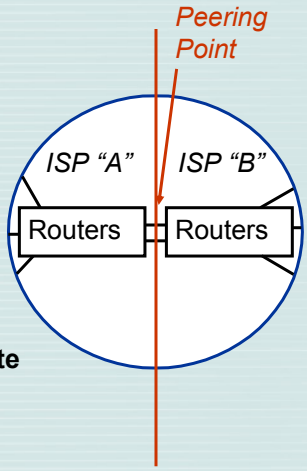
- **Need to recover resources from abandoned transactions**
  - **Web transactions usually do not tell server when browser has abandoned transaction**
  - **Timeout is a reasonable technique**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

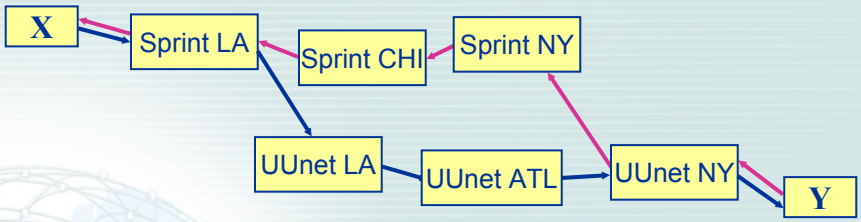
Internet Routing Between ISPs (Peering)

- Internet Service Providers enter into legal contracts to carry each other's traffic.
  - Traffic transfer between ISPs occurs at peering points
  - Peering philosophies differ among ISPs
- Congestion may occur at peering points, especially public ones.
  - The primary inter-ISP routing protocol usually does *not* look at congestion
- The end-to-end route in one direction is usually different from the end-to-end route in the other direction!
  - Depends on legal and financial arrangements between ISPs, etc.



KEYNOTE The Internet Performance Authority

Asymmetric Internet Routing



- In "hot potato routing," each ISP tries to hand off the packet as quickly as possible to the next ISP, to avoid the expense of carrying it.
  - The route is *asymmetric*.
  - For example, the ACKs for a file transfer may travel over a very different (longer?) route than the file data itself — but delayed ACKs will delay the transfer!
- Some ISPs act differently, to maintain control over the packet.

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

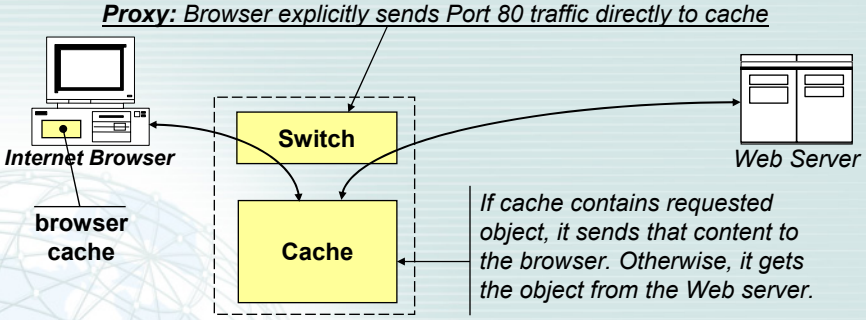
Caching

- **Caching systems are usually located between server and browser**
  - **They temporarily store commonly-requested, unchanging objects (images, streaming media files, etc.)**
  - **Caching usually improves performance**
    - Decreases the load on the server and/or communications links
    - Decreases download time seen by the browser
  - **There may be a "cache tax" — a slight delay for all objects**
  - **Caches may be used to assist traffic distribution by mirroring some server content at a remote location**
  - **Some caches may be pre-loaded to handle flash loads**
- **Three types of caches**
  - **Inside the Browser**
  - **Client-Side**
  - **Server-Side**

KEYNOTE The Internet Performance Authority

Basic Web Cache Operation

Transparent: *Port 80 traffic goes through cache regardless of original destination*
Proxy: *Browser explicitly sends Port 80 traffic directly to cache*



- **Traffic is diverted by external or internal switch or router**
- **Cache may handle http, ftp, JavaScript files, streaming media, etc.**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Content Distribution Networks

- Content Distribution Networks (CDNs) have constructed worldwide systems of caching, geographic traffic distribution, and content distribution management devices.
  - Usually in partnership with local ISPs
  - CDNs specialize in static content (.jpg, .gif, etc.); they may also be able to handle authentication, streaming media and some dynamically-generated Web pages.
  - End-user must still go to a hosting facility to obtain base HTML and dynamic content
  - True transactions must be handled by hosting facility
- Distribution can be over the Internet or by alternate paths (satellite, dedicated link)
  - Content can be pre-positioned before major events

KEYNOTE The Internet Performance Authority

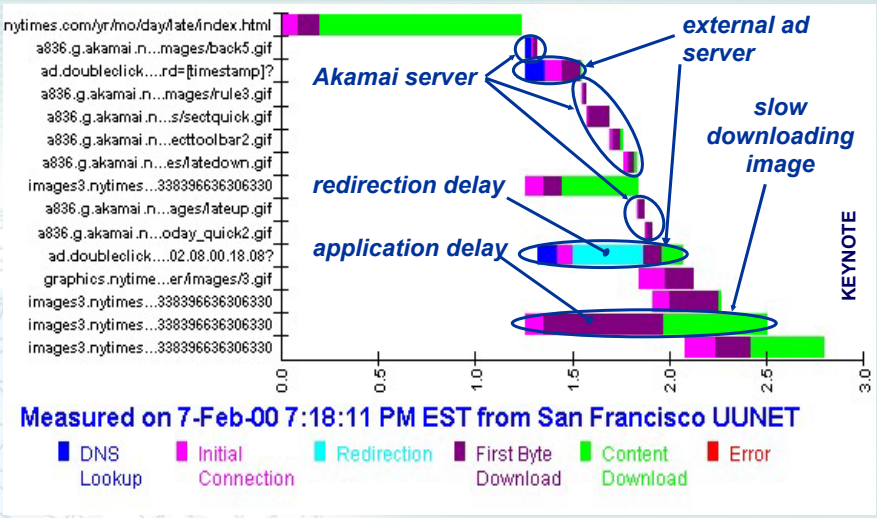
Summary of Web Performance Factors

- The Web page seen by the browser is often generated from a number of different sources:
  - Ad servers
  - Geographically-distributed content servers
  - Caches
- Download performance is affected by:
  - Protocol behavior
  - Number of concurrent download connections; persistence
  - Geographic location of the browser
  - Congestion and latency between servers and browser
  - Performance of load-distribution and load-distribution schemes
  - Performance of the servers and their back-end databases
- For example...

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Web Page Download Time Components



This page includes "Akamaized" content distribution and DoubleClick ad servers

KEYNOTE The Internet Performance Authority

Agenda

- Performance Challenges on the Web
- Web Architectural Review
- Web Load Testing
  - Web Testing vs. Legacy Testing
  - The "Concurrent Users" Fallacy
  - Load Metrics
  - Testing Types
  - Test Tools and Scripting for Realism
- Internet Statistics

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Web *vs.* Legacy Differences – 1

("Legacy" is client:server or terminal:host over a private network; e.g., IBM 3270 SNA)

- **Abandonment**
  - **Legacy operators do not abandon ("click away") if response time is poor; they will complete the transaction**
  - **Abandonment decreases the load (especially on subsequent pages) as performance degrades**
- **Session Termination Signaling**
  - **Legacy systems usually give a clear indication of session termination or abandonment; *Web protocols usually do not***
  - **Abandoned Web sessions use resources until they're forced to close by the server after timeout, etc.**
  - **"Concurrent sessions" statistics may be misleading**
    - It may include abandoned sessions that have not yet been forced to close by timeout, etc.

KEYNOTE The Internet Performance Authority

Web *vs.* Legacy Differences – 2

- **Variable Loading**
  - **Legacy loads are externally buffered**
    - Computer load can be very uniform, because queuing is in the call center's telephone system or at the computer system's input port
    - Computer load cannot exceed the number of call center operators or input ports
    - Diverted customers are still handled by someone or something
  - **Web loads are extremely variable and hit the system directly**
    - Steady state "concurrent users" load test is very unrealistic
    - Extreme ("Flash") load may crash system, deny service
    - Possible customer dissatisfaction crisis
- **Skill Uniformity**
  - **Web customer skills and behaviors are much more varied than legacy operator skills**
  - **Multiple skill profiles are necessary to model wide variations in skills and behaviors**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Web *vs.* Legacy Differences – 3

- **Network Complexity**
  - **Web networks are much more geographically distributed and varied than legacy networks**
    - Important customers may have poor connectivity, but you must still provide acceptable service
    - Connection latency and bandwidth have a large effect on the time that the browser connection and server resources are held
  - **Web networks contain elements that affect performance of the enterprise site, but are under only indirect control of the enterprise. They must nevertheless be evaluated.**
    - DNS
    - ISP backbones and peering
    - Geographically distributed servers; Caching; CDNs
- **Client Complexity**
  - **Web client ("browser") is usually more complex, and varied, than a legacy terminal or client**
    - Browser cache; Cookies
    - Javascript, plug-ins, etc.

KEYNOTE The Internet Performance Authority

Web Load Testing Must Include:

- **Variable loading, for realism and to test flash loads**
- **Satisfaction measures, based on response time**
- **Abandonment behavior, for realism and to evaluate resource recovery mechanisms**
- **Distributed loading, to test the Internet connections, peering, caching, and content distribution networks**
- **Multiple geographic profiles, to represent the wide variations in Web access technology and location**
- **Multiple user connectivity profiles, because latency and bandwidth affect the load seen by the servers**
- **Multiple user profiles, to model the wide variation in user skills present in the Web community**
- **Script variability, of both data and paths, to avoid creating unrealistic hot spots that are handled by caching**
- **True browser emulation, of varied browsers, including browser caching, cookies, Javascript, SSL, etc.**

KEYNOTE The Internet Performance Authority

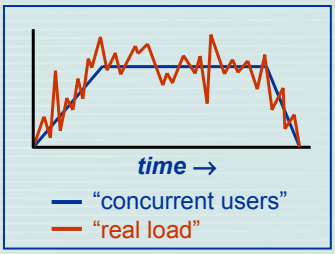
Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

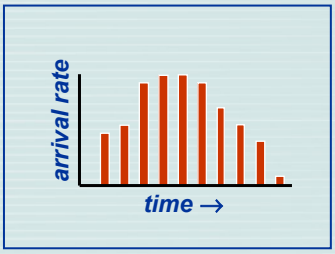
a note about:

"Concurrent Users"

Presented Load ≠ Concurrent Users

- **Because of abandonment behavior, "concurrent users" is an unrealistic substitute for "load."**
  - It's really an *output*, not an *input*.
  - If system capacity increases, the number of concurrent users may decrease, because they're completing their work quickly.
- **There's no standard definition or measurement of concurrency**
  - When does a session end? Most Web connections are stateless.
- **And *real Web loads are not uniform***
- **Therefore, use:**
  - User session starts, page view starts, or hit starts per unit time
  - Arrival Rate distribution



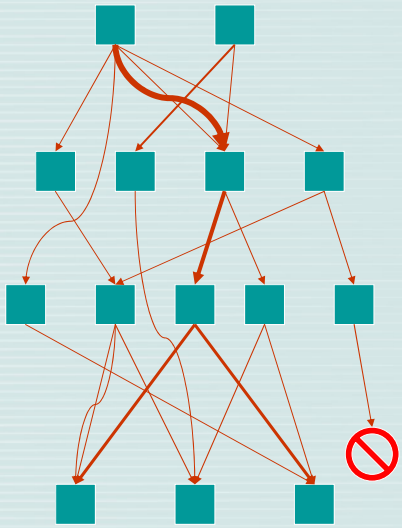


KEYNOTE The Internet Performance Authority

a note about:

Path Variation

- **Web user paths are more variable (and sometimes "random") than legacy user paths**
- **Web system caching in browsers, the network, and the servers can create extremely inaccurate results if test variation is too low**



KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Load Characterization and Measurement Tools

- Most tools work by examining server log files off-line
- Different tools usually give different results.
- If the site is not yet live:
  - Statistics from similar sites that are publicly available
  - In-house trials
  - "Intuition"
  - ***Anything*** **is better than ignoring all the variables and creating a load characterization that's identical for each user and ignores think time, abandonment, etc.**

KEYNOTE The Internet Performance Authority

Load Metrics

- Load metrics are affected by user class:
  - New users
  - Experienced users during normal conditions
  - Experienced users during unusual conditions
- Typical metrics:
  - Connection speed, location (geography, access ISP)
  - Browser configuration
  - Counts and averages for standard metrics:
    - User Sessions (may be difficult to measure),
    - Page Views (number of full pages requested), Average Page Size
    - Hits (number of HTTP GETs and POSTs), Average Hit Size
      - *Note: Caches and CDNs may absorb many of these!*
  - Major transaction paths and histogram of requested pages
    - Percentage of major paths that resulted in purchases
  - Think time and tolerance for wait time

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Types of Testing – 1

- ***Functional / Regression testing*** **– does the site work at all? – is critically important, but it's not enough.**
  - **These tests find missing elements, broken links, errors.**
  - **They're exactly repeatable, so software testing is simpler.**
  - **They can be scripted by simple record/replay tools.**
  - **They succeed even if they have to wait a long time; most real Web users have abandoned the site by then.**
- ***Load testing*** **measures the response of the site to a specified load.**
  - **The load in a "load test" doesn't need to stress the system; many load tests are designed to emulate a normal load.**
  - **The load test parameters (characterization, load levels, scripts) can be validated by comparison to real loads.**
  - **This type of testing can be used to measure the effects of changes to the Web system.**

KEYNOTE The Internet Performance Authority

Types of Testing – 2

- ***Stress testing*** **finds the instantaneous breaking points.**
  - **Under what load level, and what type of load, does the system fail or provide unacceptable response times?**
  - **Will load-distribution devices fail?**
  - **Will database replies time-out and result in empty pages?**
- ***Endurance testing*** **measures system performance after a sustained high load.**
  - **System performance may degrade after a large number of users**
    - Poor re-use of system resources
    - Poor handling of abandoned sessions
  - **Some systems may break entirely after a sustained high load.**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Testing Tools

- **Most testing is done within the server site.**
  - **Functional / Regression Testing (if all page components are within the site)**
  - **Initial stress testing**
  - **Testing of Web server and back-end database integration**
- **Final testing should be done across the network.**
  - **Find problems with geographically-distributed systems**
    - Distributed servers and Content Distribution Networks
    - DNS difficulties
    - Traffic distribution technology
  - **Find problems in Internet connectivity**
    - ISP connectivity
    - Network aggregation bottlenecks (routers, etc.)
    - Caching problems in caches that you don't control
    - Peering to ISPs that are used by customers

KEYNOTE The Internet Performance Authority

Statistics, Scripting, and Realism

- **Get meaningful statistics for each page, not just an arithmetic average for the entire transaction.**
  - **Arithmetic means can be misleading; extreme delays that may cause abandonment will be hidden. It must be possible to obtain abandonment and dissatisfaction rates.**
  - **"Total" transaction time hides poor single-page responses.**
- **Get meaningful error statistics, page by page.**
  - **Include dissatisfaction and abandonment**
- **Create different classes of users with different characteristics (familiarity with site, think time, tolerance for delay, etc.)**
- **Insert enough randomness to avoid creating unrealistic hot spots or caching**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

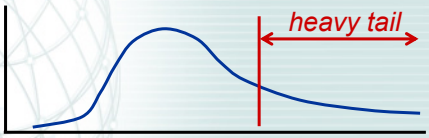
Agenda

- Performance Challenges on the Web
- Web Architectural Review
- Web Load Testing
- Internet Statistics

KEYNOTE The Internet Performance Authority

Statistical Behavior of Internet Traffic

- Internet traffic is self-similar and heavy-tailed
  - Result of mixing long files, small files, ACKs, compressed video, human think time, etc.
  - (Connection arrival rate is still normally distributed)
- Self-similar traffic is very bursty
  - No natural length to the bursts
  - Bursts are not quickly smoothed by larger time scales
  - Capacity problems can appear at 50% utilization!
- Heavy-tailed (right tail) traffic can have an arbitrarily large amount of very large values



KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

Performance Statistics for Heavy-Tailed Traffic

- **With heavy-tailed traffic:**
  - **Arithmetic Mean may be misleading.**
    - Individual long measurements can distort the mean, especially if the typical measurement is very short (e.g., DNS, TCP Connect)
    - This is more troublesome for SLAs and long-term trending than for problem diagnosis, which involves detailed examination of datapoints
    - If you have 10,000 measurements of 0.02 seconds, plus TEN measurements of 20 seconds, the arithmetic mean is 0.04 seconds — a *doubling* of the mean because of only *ten* measurements!
    - Geometric Mean or Median are better, more stable indicators of typical performance as seen by users
  - **Standard Deviation is *very* misleading in measuring heavy tail data**
    - Can be massively distorted by a single large measurement!
    - Geometric Deviation or 85th percentile are better, more stable indicators of variance

KEYNOTE The Internet Performance Authority

Computation of Performance Statistics

- **Arithmetic Mean is easy and useful for "quick looks"**
  - **Arithmetic means can be aggregated (e.g., DNS + TCP Connect time) by simple addition; other statistics cannot.**
    - (85th Percentile of DNS) + (85th Percentile of TCP Connect) is NOT the 85th Percentile of (DNS + TCP Connect)
  - **"Trimming above *n* seconds" is a computationally-simple method for improving the stability of the arithmetic mean, but the number of trimmed values must also be considered.**
- **Other measures require more manipulation**
  - **Percentiles give a reasonable, but coarse, estimate**
    - Median is the 50th percentile
  - **Geometric Mean is the best measure**
    - It's the nth root of the product of the n measurements.
- **Geometric Deviation is a good measure of variance**
  - **"Geometric Deviation" is a *factor* = 10^(std deviation{log(x_i)})**
- ***See Keynote website "Resources/White Papers" for statistics articles***

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

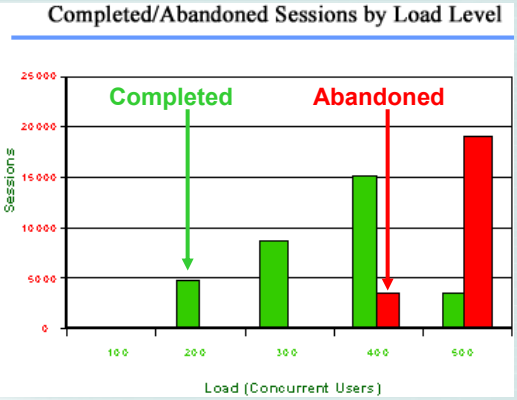
the inevitable ad. . .

LoadPro Load Testing

Network Computing EDITOR'S CHOICE

***LoadPro*™ realistic, distributed load testing service for Web sites**

- **A full *service*, not just a box of software**
- **Geographically distributed, coordinated, realistic loading across the Internet**
- **Service includes *fast* development of custom user profiles that include abandonment behavior, geography, types of connections used, path variation, etc.**
- ***massive* capacity**



Uses accurate statistics (geometric mean, per-page measures, etc.); provides statistics on user dissatisfaction, lost revenue, etc.

Keynote also offers self-service functional testing, etc.

KEYNOTE The Internet Performance Authority

Keynote Systems (www.keynote.com), "*The Internet Performance Authority®,*" **is the world's leading supplier of Internet performance measurement, diagnostic, load testing, and consulting services to companies with e-commerce Web sites.** Keynote® captures over 20 million performance measurements daily, using Keynote's global infrastructure of over 1500 measurement computers connected to the major Internet backbones from over 100 statistically selected Internet access locations in 50+ metropolitan areas worldwide. Internet performance and availability data are collected at Keynote's sophisticated operations center and are instantly available to customers through any Web browser, by real-time XML transfer, or by FTP. **Keynote currently measures individual Web pages as well as transactions, streaming media, and wireless. Keynote also supplies highly-accurate, distributed Web load testing services.**

Eric Siegel is Principal Internet Consultant with Keynote Systems, Inc. and is the author of *Designing Quality of Service Solutions for the Enterprise* (John Wiley & Sons, 1999). Before joining Keynote Systems, Mr. Siegel was a Senior Network Analyst at NetReference, Inc., which specializes in network architectural design and strategic planning, and he was a Senior Network Architect with Tandem Computers, where he was the technical leader and coordinator for all of Tandem's data communications specialists worldwide. Mr. Siegel also worked for Network Strategies, Inc. and for the MITRE Corporation, where he specialized in computer network design and performance evaluation. Mr. Siegel received both his B.S. and M.E.E. degrees in Electrical Engineering from Cornell University, and he has been a member of the Internet community since 1978.

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric D. Siegel. All Rights Reserved.

QW2002 Paper 8I1

Robert A. Martin
(The MITRE Corporation)

Vulnerabilities and Developing for the Net

Key Points

- Hackers make use of recently discovered vulnerabilities to prey on commonly used software.
- CVE is an international industry, academia, & government software vulnerability effort.
- CVE allows for integration of multiple security services, products and databases.

Presentation Abstract

This presentation will discuss the CVE Initiative, an international, community-based effort from industry, government, and academia, that is creating an organizing mechanism that will make the finding and fixing of software product vulnerabilities more rapid, predictable, and efficient. The opportunities that this initiative is creating for software developers, security practitioners, and systems owners will be explored. These opportunities are in their systems, products, and services, as well as for their customers. Attendees will be shown the details of several of the most common types of vulnerabilities and their causes, with examples from recent real-world products that have the vulnerabilities. Additionally, the attendees will leave with an understanding of how the CVE Initiative is helping enterprise security management of vulnerabilities and exploits become more predictable, structured, and effective as a result of CVE-enabled information security products, services, and methodologies.

About the Author

A. Martin is the primary point of contact for CVE Compatibility efforts, a co-lead for MITREs Cyber Resource Center web-site and a Principal Engineer in MITREs Information Technologies Directorate. At the culmination of his five years of Y2K leadership and coordination efforts, Mr. Martin served as the Operations Manager of the Cyber Assurance National Information Center, a 24x7 cyber security watch center within the Presidents Y2K Information Coordination Center. Prior to these efforts, Martin developed a standardized software quality assessment process that was used to helped over 100 of MITREs Air Force, Army, and FAA customers improve their software acquisition methods as well as the quality, cost, and timeliness of their delivered software products. Today, Martin's efforts are focused on the interplay of cyber security, critical infrastructure protection, and e-Business technologies and services. Martin received a bachelors degree and a masters degree in electrical engineering from Rensselaer Polytechnic Institute and a masters of business degree from Babson College. He is a member of the ACM, AFCEA, IEEE, and the IEEE Computer Society.

Quality Week 2002

Vulnerabilities and Developing for the Net

Robert A. Martin
The MITRE Corporation

5 September 2002

The views expressed in this presentation are those of the authors and do not necessarily reflect the policies or position of The MITRE Corporation.
Editorial graphics © 1996, 1997, 1998, 1999, 2000, 2001 Martin, used with permission. © 2001 The MITRE Corporation

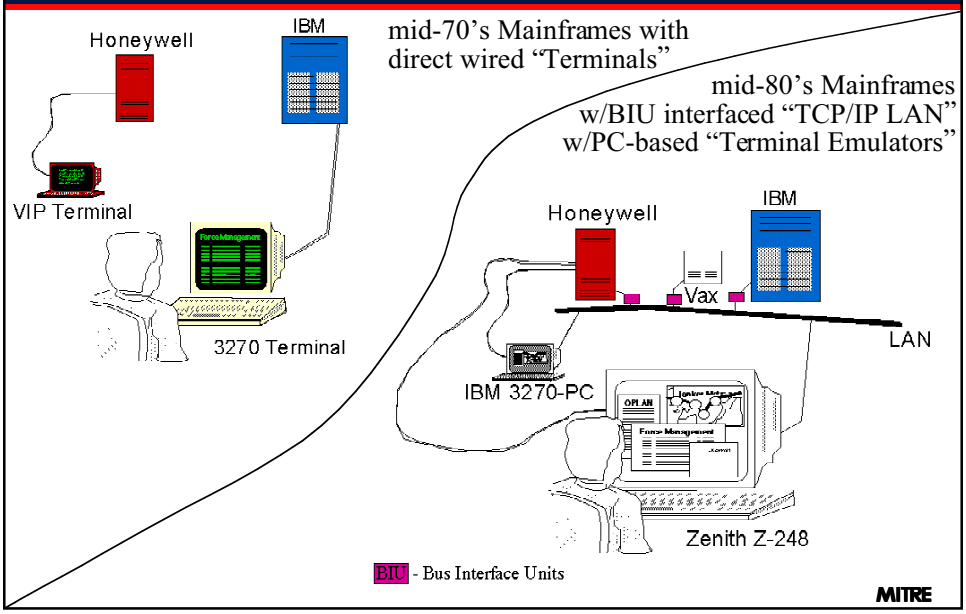
MITRE

Outline

- **Background and Motivation**
  - **Finding Out About Vulnerabilities**
  - **The Problem and a Solution - CVE**
  - **CVE Compatibility**
  - **The CVE Process**
  - **Summary**

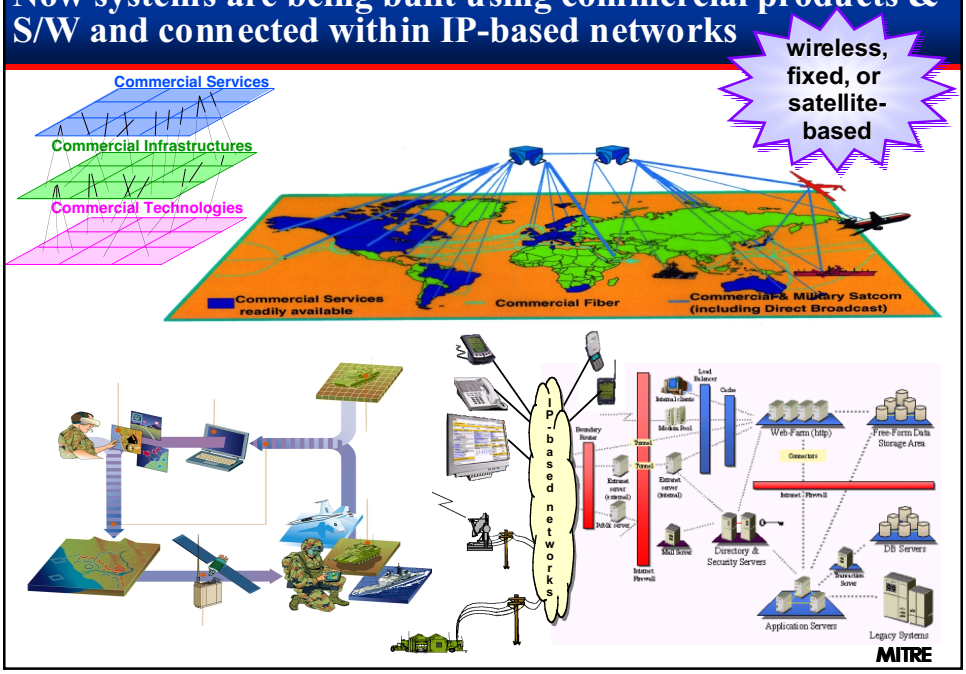
MITRE

DoD started w/stand-alone computers, terminals & custom S/W -- Then came PCs w/COTS S/W terminal emulators and TCP/IP LANs



MITRE

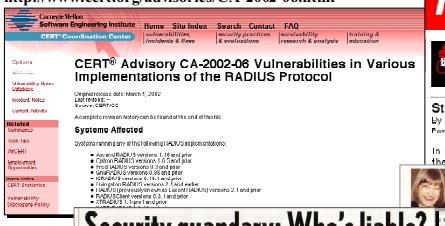
Now systems are being built using commercial products & S/W and connected within IP-based networks



MITRE

Many Motivations for Getting on top of Vulnerabilities

http://www.cert.org/advisories/CA-2002-06.html



http://www.theregister.co.uk/content/53/24244.html

http://www.eweek.com/article/0,3658,s=701&a=23193,00.asp

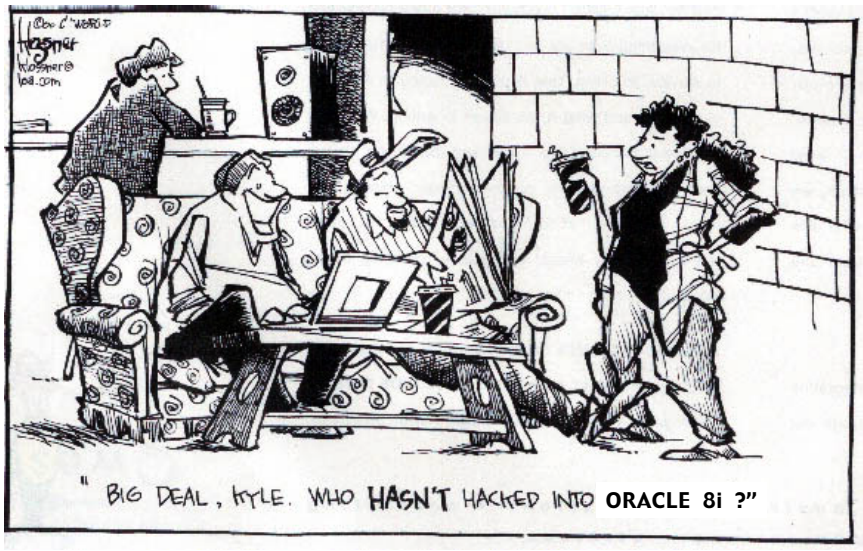
http://www.baselinemag.com/article/0,3658,s=1867&a=23195,00.asp

MITRE

Organization's Internet-visible "Faces" are being abused through Vulnerabilities in Commercial S/W

MITRE

The Wrong Publicity Can Be Bad...



MITRE

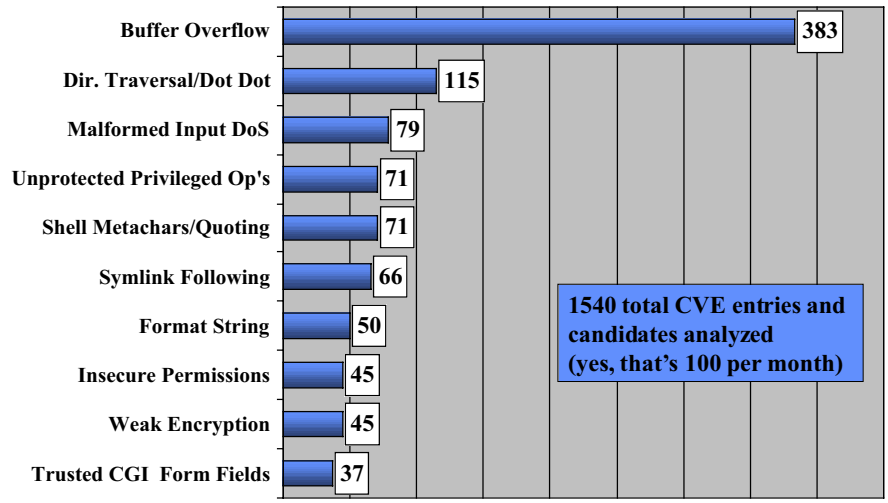
Software problems with security implications are referred to as Vulnerabilities or Exposures

- Vulnerabilities are security related software problems that could directly allow serious damage
- Examples:
  - phf, ToolTalk, Smurf, rpc.cmsd, etc.
  - Oracle XSQL servlet 1.0.3.0 and earlier allows remote attackers to execute arbitrary Java code by redirecting the XSQL server to another source via the xml-stylesheet parameter in the xslt stylesheet. *[9 Jan 01 Georgi Guninski]*

- Exposures are security related software problems that could be used as stepping stones for a successful attack
- Examples:
  - Running finger, poor logging practices, etc.

MITRE

Top Ten Vulnerability Types in CVE (Issues publicized between Jan 2000 and April 2001)



MITRE

Vulnerabilities Have Been Found in Almost Every Type of Commercial Software There Is

Mail Servers
1st Up Mail Server
All-Mail
ALMail32
Avirt Mail Server
Becky! Internet Mail
CWMail
Domino Mail Server
Exchange Server
Hotmail
Internet Anywhere Mail Server
ITHouse Mail Server
Microsoft Exchange
Pegasus Mail
Sendmail

Security Software
ACE/Server
BlackICE Agent
BlackICE Defender
Certificate Server
CProxy Server
ETrust Intrusion Detection
GateKeeper
InterScan VirusWall
Kerberos 5
Norton AntiVirus
PGP
SiteMinder
Tripwire

Web servers & tools
Domino HTTP Server
IIS
NCSA Web Server
Sawmill
WebTrends Log Analyzer

Internet
AFS
Apache
BIND
CGI
Cron
IMAP

Routers
3220-H DSL Router
650-ST ISDN Router
Ascend Routers
Cisco Routers
R-series routers

Network Applications
BackOffice
Meeting Maker
NetMeeting

DBMSs
Access
DB2 Universal Database
FileMaker Pro
MSQL
Oracle

Desktop Applications
Acrobat
Clip Art
Excel
FrameMaker
Internet Explorer
Napster client
Notes Client
Novell client
Office
Outlook
PowerPoint
Project
Quake
R5 Client
StarOffice
Timbuktu Pro
Word
Works
Workshop

Development Tools
ClearCase
ColdFusion
Flash
Frontpage
GNU Emacs
JRun
WebLogic Server
Visual Basic
Visual Studio

Operating Systems
AIX
BeOS
BSD/OS
DG/UX
FreeBSD
HP-UX
IRIX
Linux
MacOS Runtime for Java
MPE/iX
NetWare
OpenBSD
Palm OS
Red Hat
Security-Enhanced Linux
Solaris
SunOS
Ultrix
Windows 2000
Windows 95
Windows 98
Windows ME
Windows NT

Firewalls
Firewall-1
Gauntlet Firewall
PIX Firewall
Raptor Firewall
SOHO Firewall

Sample of Vulnerabilities Announced in 1999 & 2000

MITRE

Outline

- Background and Motivation
- Finding Out About Vulnerabilities
- The Problem and a Solution - CVE
- CVE Compatibility
- The CVE Process
- Summary

MITRE

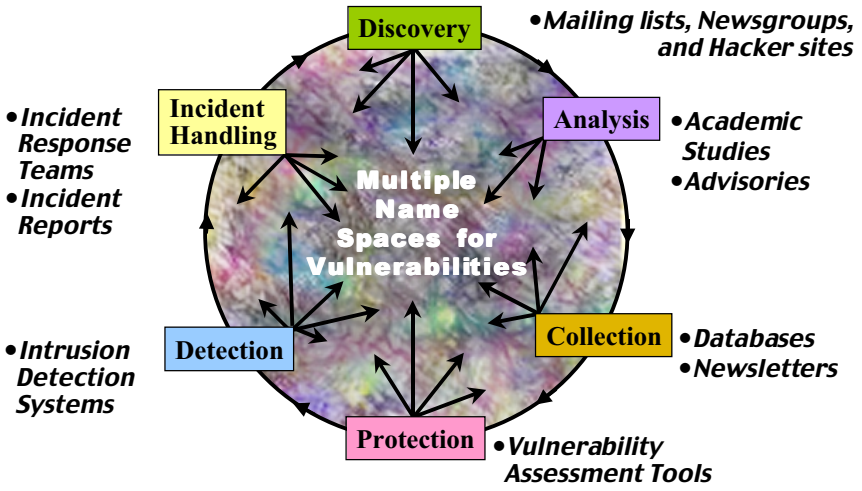
So how do you find out about commercial software vulnerabilities if the vendors aren't going to tell you?

Three groups have emerged who share that same curiosity

- Hackers
  - want to find vulnerabilities and exposures so they can exploit them to gain access to systems
- Commercial interests groups
  - want to be hired to find, or want you to buy their tools to help you find, the vulnerabilities and exposures
  - offer services to come and do an evaluation of your systems
- Philanthropists
  - include security researchers in various government, academic, and non-profit organizations, as well as unaffiliated individuals that enjoy searching for vulnerabilities and exposures
  - usually share their knowledge and tools freely

MITRE

There are Many Different types of Groups Involved in Providing Information about Vulnerabilities



The rule has been, "Whoever finds it, names it"

MITRE

Implications of multiple name spaces for information on vulnerabilities

- Difficult to correlate data across multiple organizations and tools
  - IDS and assessment tools
  - Security tools and fix information
  - Incident information
- Difficult to conduct a detailed comparison of tools or databases (Vulnerabilities are counted differently)

Vulnerability Sharing databases and web sites

Site Name	Type	Organization
arachNIDS	free IDS database	Max Vision Network Security/Whitehats
CERIAS Vulnerability Database	database	CERIAS/Purdue University
Dyodor's Playhouse	hacker web site	Insecure.Org
Online Vulnerability Database	database	Ernst & Young's eSecurityOnline.com
ICAT Metabase	free web site	NIST
Bugtraq mailing list Database	mailing list database	SecurityFocus.com
PacketStorm	hacker web site	Security, Inc.
SWAT Database	database	AXENT Technologies

Alert and Advisory Services - From Security Groups & Organizations

Service	Type	Organization
Bugtraq	e-mail list	Bugtraq
Casandra	alerts	CERIAS/Purdue University
CERT Advisories	advisory	CERT Coordination Center
CyberNotes	monthly newsletter	NIPC
Razor	advisory	Bindview Corporation
S.A.F.E.R.	monthly newsletter	The Relay Group

Protection/Detection Scanner and IDS tools & services

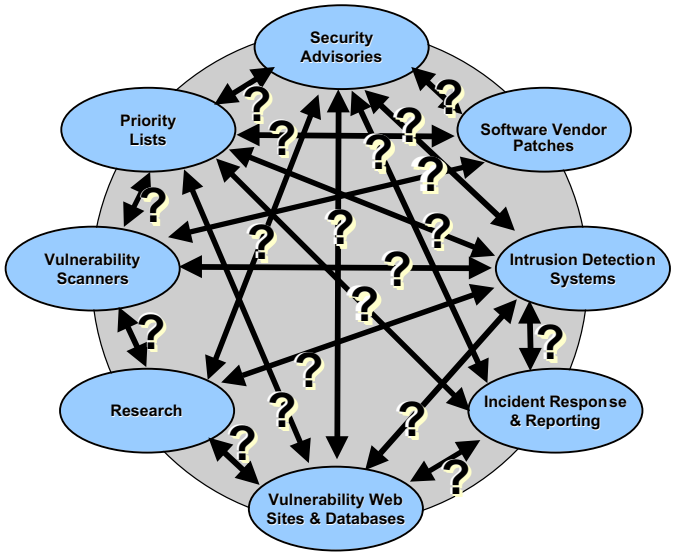
Product	Tool Type	Organization
Centrax	scanner/IDS	CyberSafe
CyberCop	scanner	Network Associates
Dragon	IDS	Network Security Wizards
HackerShield	scanner	BindView Corporation
LANPATROL	IDS	Network Security Systems
[illegible]saus	freeware scanner	Renaud Deraison & Jordan Hrycaj
[illegible]Prowler	IDS	AXENT Technologies
[illegible]ysGuard	ASP-based scanner	Qualys
[illegible]Secure	IDS	Internet Security Systems
[illegible]iever	scanner	Symantec Corporation
[illegible]NT	scanner	World Wide Digital Security
[illegible]re IDS	IDS	Cisco Systems
STAT	scanner	Harris Corporation
SWARM	scanner	Hiverworld, Inc.

Alert and Advisory Services - From Software Vendor Groups & Organizations

Service	Type	Organization
IBM ERS	advisory	IBM
Microsoft Product Security Notification Service	advisory	Microsoft Corporation
SGI Security Advisory	advisory	Silicon Graphics, Inc.
Sun-alert	alert	Sun Microsystems, Inc.

MITRE

Difficult to Integrate Information on Vulnerabilities and Exposures



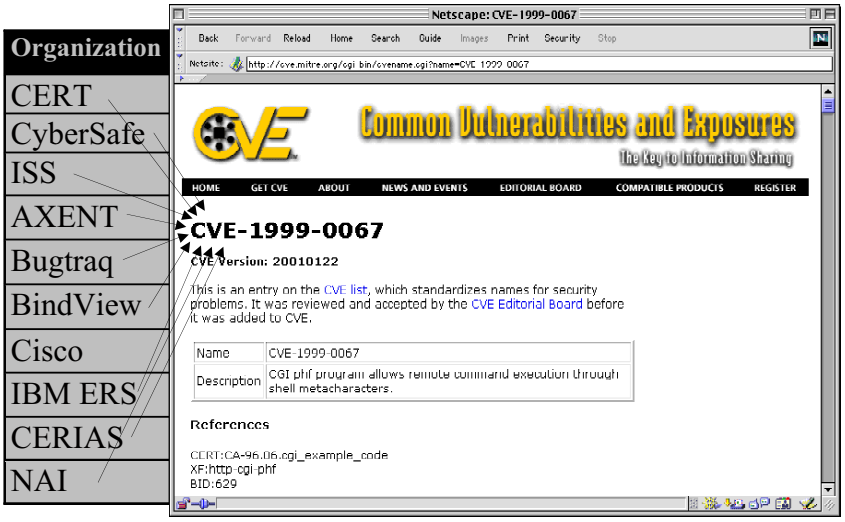
MITRE

Outline

- Background and Motivation
- Finding Out About Vulnerabilities
- The Problem and a Solution - CVE
- CVE Compatibility
- The CVE Process
- Summary

MITRE

The adoption of CVE Names by the Security Community is starting to address this problem

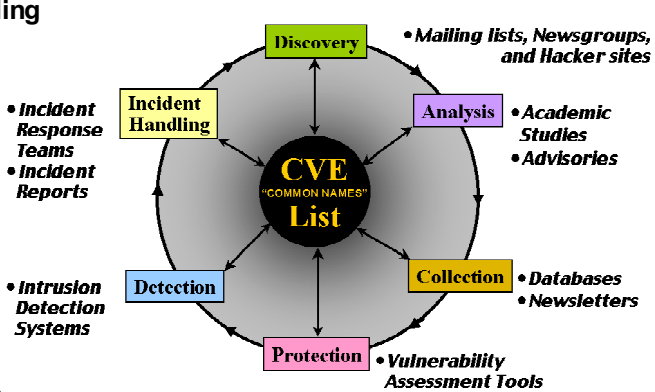


Along with the new rule, "Whoever finds it, gets a CVE name for it"

MITRE

The Vulnerability Information Sharing Process with CVE - - *"Whoever finds it, gets a CVE name for it"*

- Assigning a unique identifier to each problem
- Remaining independent of any particular perspective
  - Not just a developer's, researcher's, tester's, or analyst's view
- A community-wide effort via:
  - the CVE Editorial Board, the CVE Advisory Council, and the organizations adding CVE names into their tools, databases, web sites, & services
- Publicly open and shared
  - Will eventually list all publicly known security problems



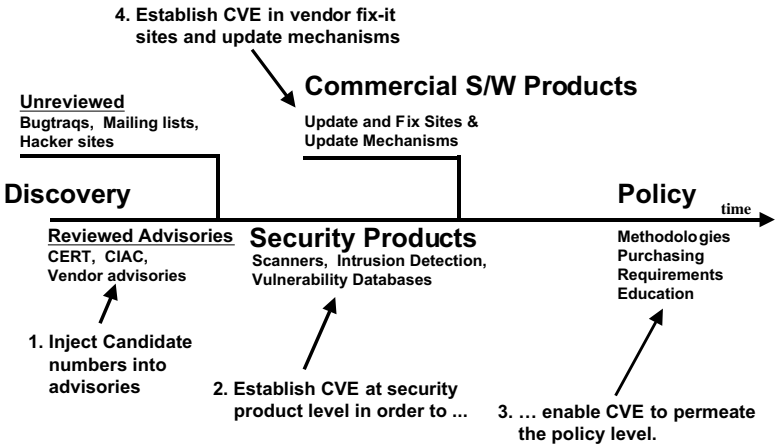
MITRE

The Common Vulnerabilities and Exposures (CVE) Initiative

- An international security community activity led by MITRE focused on developing a list that provides common names for publicly known information security vulnerabilities and exposures.
- Key tenets
  - One name for one vulnerability or exposure
  - One standardized description for each vulnerability or exposure
  - Existence as a dictionary rather than a database
  - Publicly accessible for review or download from the Internet
  - Industry participation in open forum (editorial board)
- The CVE list and information about the CVE effort are available on the CVE web site at [cve.mitre.org]

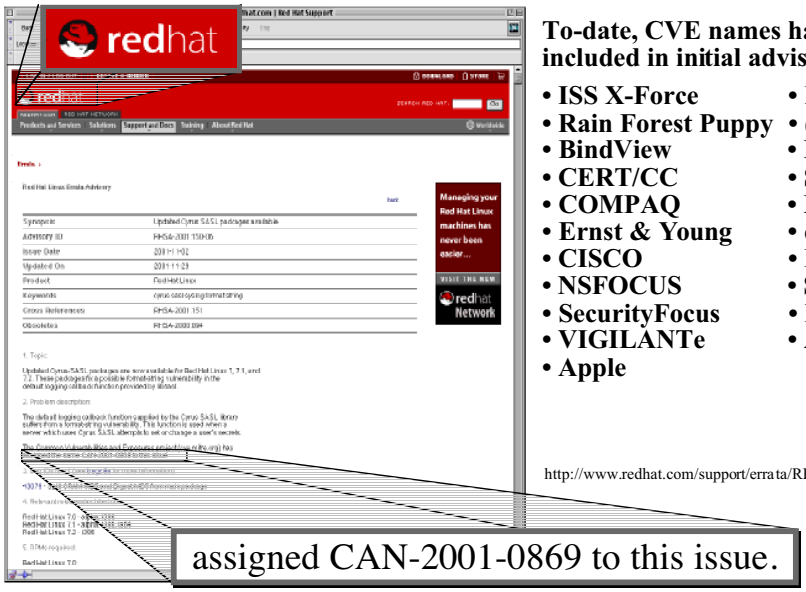
MITRE

The CVE Strategy



MITRE

Many organizations are reserving CVE names and using them in their alerts and advisories



To-date, CVE names have been included in initial advisories from:

- ISS X-Force
- Rain Forest Puppy
- BindView
- CERT/CC
- COMPAQ
- Ernst & Young
- CISCO
- NSFOCUS
- SecurityFocus
- VIGILANTe
- Apple
- IBM
- @stake
- HP
- SGI
- Microsoft
- eEye
- Rapid 7
- Sanctum
- Red Hat
- Apache

http://www.redhat.com/support/errata/RHSA-2001-150.html

MITRE

Outline

- Background and Motivation
- Finding Out About Vulnerabilities
- The Problem and a Solution - CVE
- CVE Compatibility
- The CVE Process
- Summary

MITRE

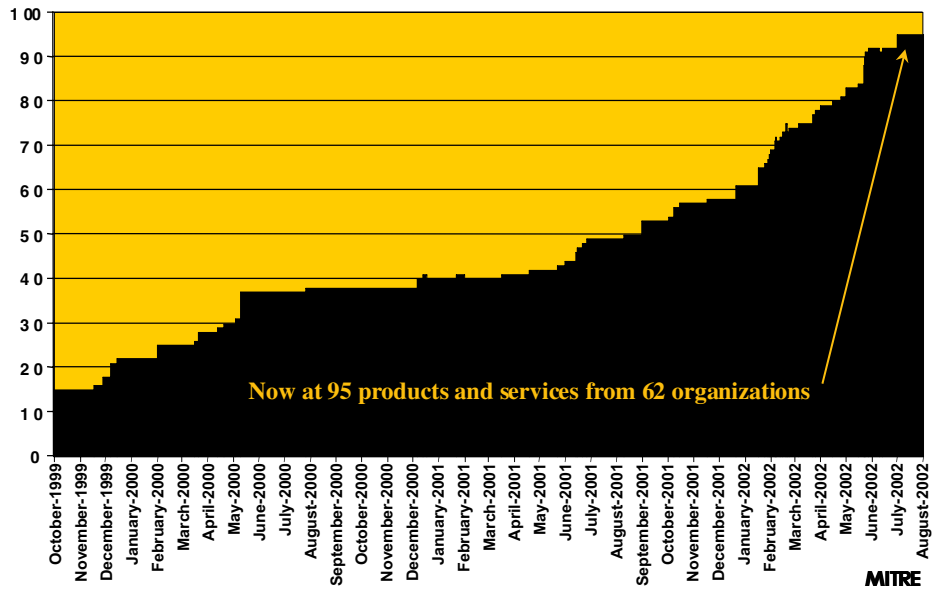
What does CVE-compatible mean?

- CVE-compatible means that a tool or database can "speak CVE" and correlate data with other CVE-compatible products
- CVE-compatible means it meets the following requirements:
  - Can find items by CVE name (CVE searchable)
  - Includes CVE name in output for each item (CVE output)
  - Provided MITRE with "vulnerability" item mappings to validate the accuracy of the product or services CVE entries
  - Makes a good faith effort to keep mappings accurate

MITRE

Timeline of CVE Compatibility Declarations

(as of 1 July 2002)



MITRE

Organizations With Products That Use CVE
(as of 1 July 2002)

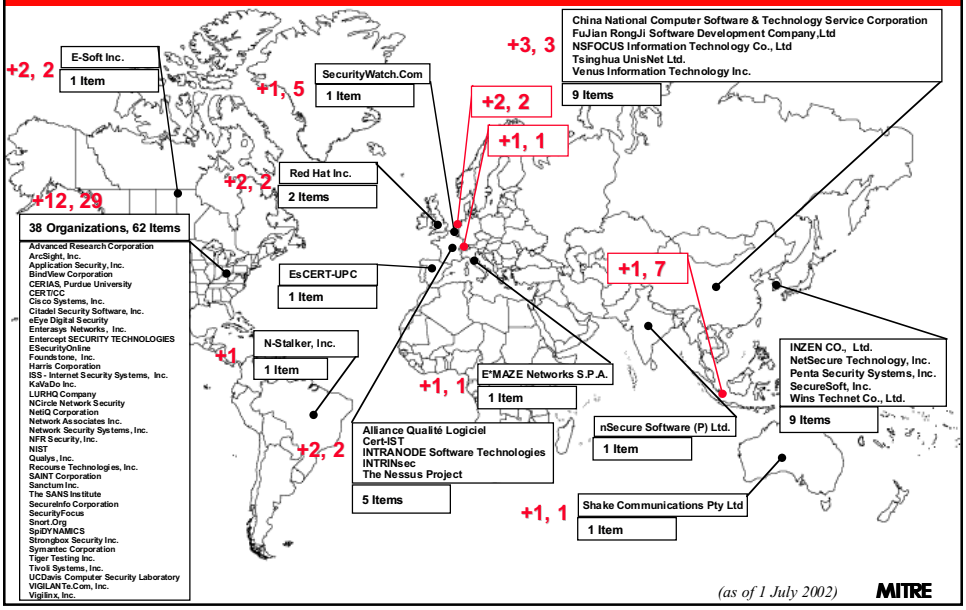
- **These (62) organizations have publicly declared that they are working on (95) CVE-compatible tools, databases, web sites, or services**

Advanced Research Corp
Alliance Qualité Logiciel
Application Security, Inc.
ArcSight, Inc.
BindView Corporation
CERIAS/Purdue University
CERT Coordination Center
Cert-IST
Cisco Systems
Citadel Security Software, Inc.
CSS *(China National Computer Software & Technology Service Corporation)*
E*MAZE Networks S.P.A
E-Soft Inc.
eEye Digital Security
Enterasys Networks *(bought Network Security Wizards)*
Entercept Security Technologies
esCERT-UPC
eSecurityOnline
Foundstone, Inc.
FuJian RongJi Software Development Company, Ltd
Harris Corporation
Internet Security Systems
Intranode
INTRINsec
Inzen
Kavado Inc.
LURHQ Corporation
nCircle *(formerly Hiverworld)*
The Nessus Project
NetIQ
NetSecure Technology, Inc.
Network Associates Inc.
Network Security Systems
NIST
NFR Security
NSFOCUS Information Technology Co., Ltd
N-Stalker, Inc.
nSecure Software (P) LTD.
Penta Security Systems
Qualys
Resource Technologies
Red Hat Inc.
SAINT Corporation *(formerly World Wide Digital Security, Inc.)*
Sanctum Inc.
SANS
SecureInfo Corporation
SecureSoft, Inc.
Security Focus, Inc.
SecurityWatch
Shake Communications Pty Ltd
Snort.org
spiDYNAMICS
Strongbox Security Inc. (SSI)
Symantec Corporation
Tiger Testing
Tivoli Systems Inc.
Tsinghua UnisNet Technology, Ltd.
UC Davis, Computer Security Lab
Venus Information Technology Inc.
VIGILANTe *(merged with Cyrano's Networks Vigilance subsidiary)*
Vigilinx, Inc.
Wins Technet Co., Ltd.

Up-to-date list at http://cve.mitre.org/compatible

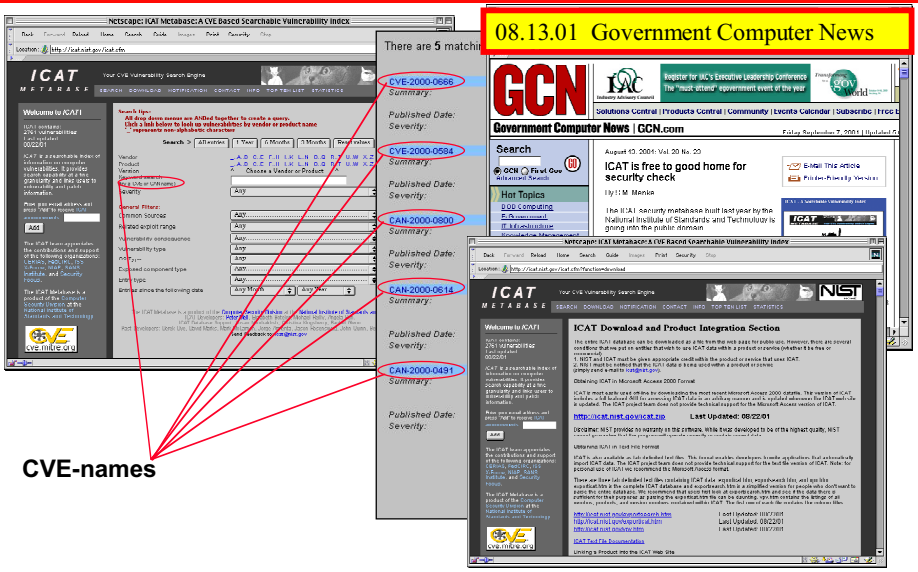
MITRE

Where CVE-compatible Items Have Come From and Where the New Ones Are Coming From



(as of 1 July 2002) MITRE

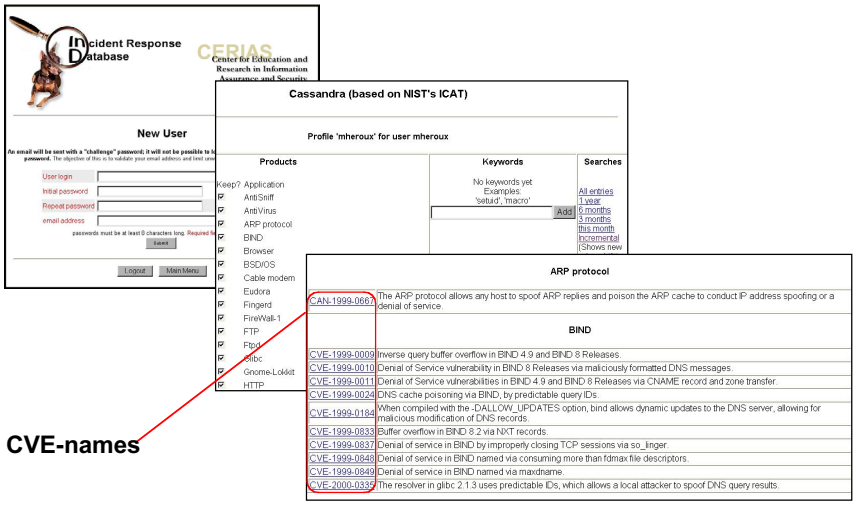
Examples of CVE-compatible items:
The ICAT Metabase



http://icat.nist.gov

MITRE

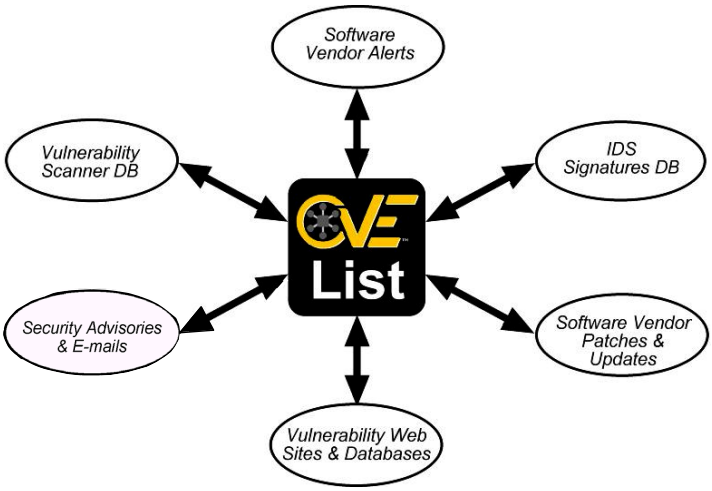
Examples continued:
Cassandra



https://cassandra.cerias.purdue.edu

MITRE

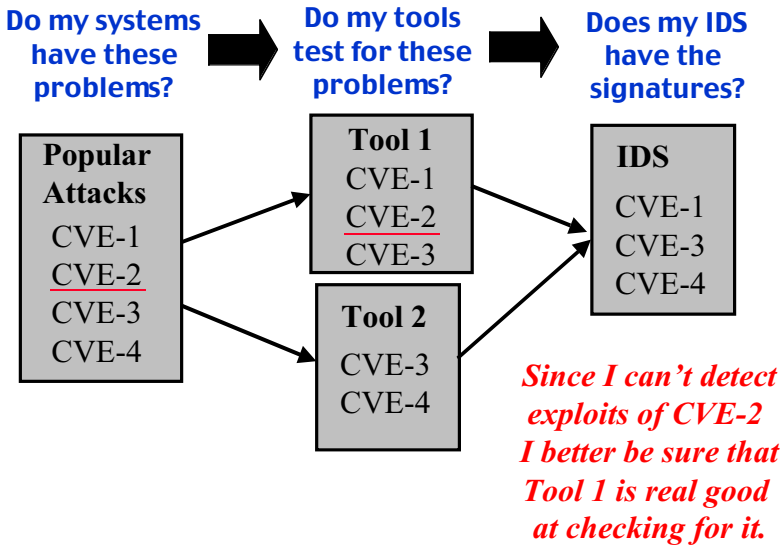
CVE compatibility provides a path for integrating information on Vulnerabilities and Exposures



CVE compatibility means that a tool or database can "speak CVE" and correlate data with other CVE-compatible products.

MITRE

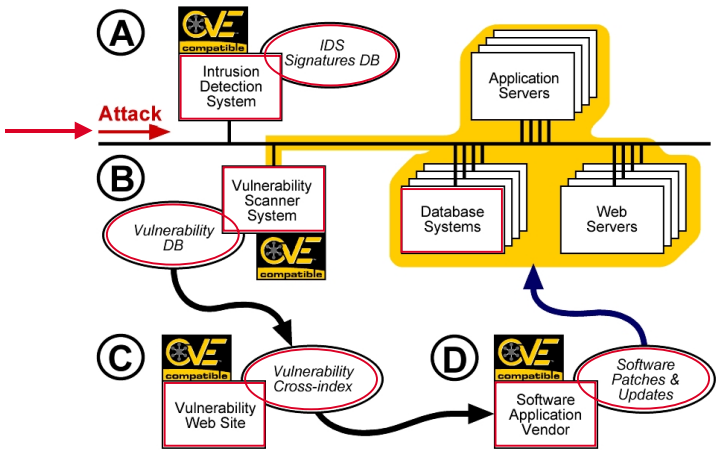
Example using CVE compatibility to go from Advisories to Vulnerability Scanners to IDSes



Since I can't detect exploits of CVE-2 I better be sure that Tool 1 is real good at checking for it.

MITRE

A CVE-Enabled Process Leverages CVE compatibility



MITRE

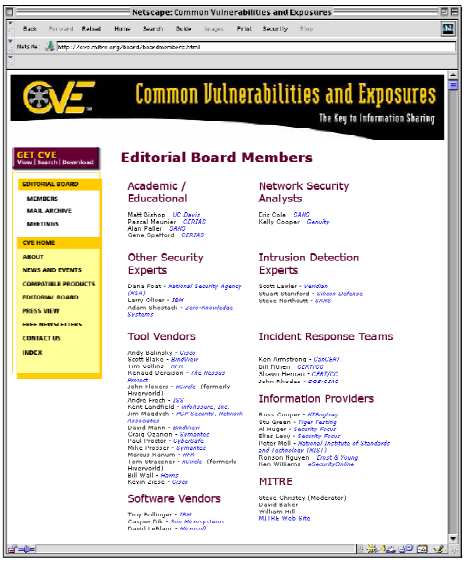
Outline

- Background and Motivation
- Finding Out About Vulnerabilities
- The Problem and a Solution - CVE
- CVE Compatibility
- The CVE Process
- Summary

MITRE

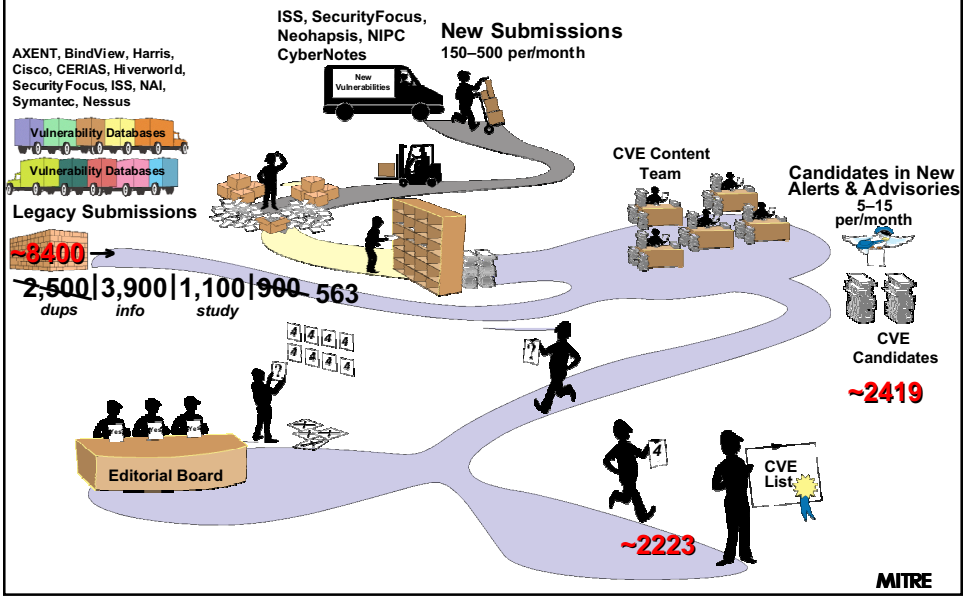
CVE Editorial Board

- Includes mostly technical representatives from 32 different organizations including researchers, tool vendors, response teams, and end users
- Reviews and approves CVE entries
- Discusses issues related to CVE maintenance
- Holds monthly meetings (face-to-face or phone)
- Maintains publicly viewable mailing list archives
  [cve.mitre.org/board/archives]



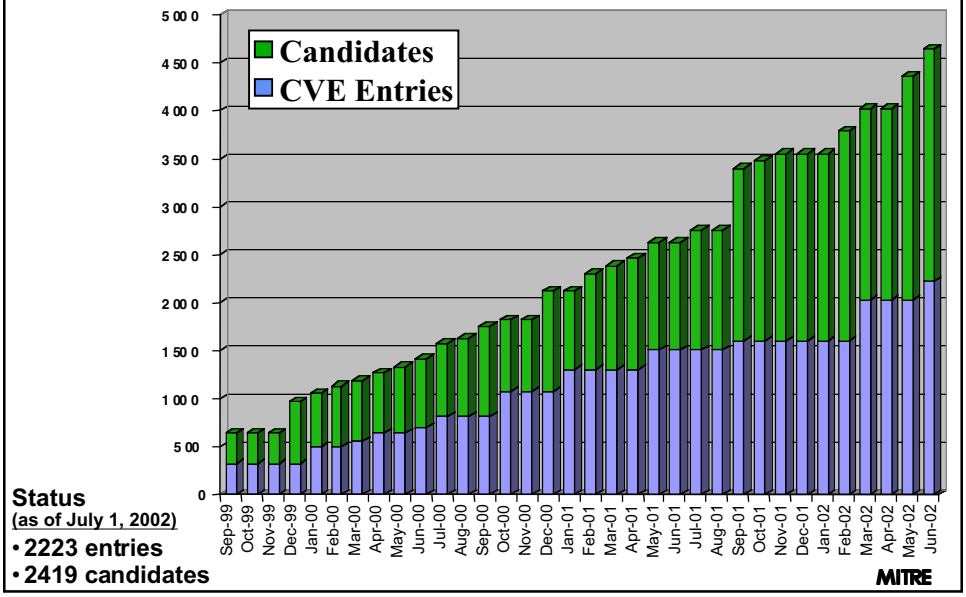
MITRE

Where the CVE List comes from



MITRE

CVE Growth

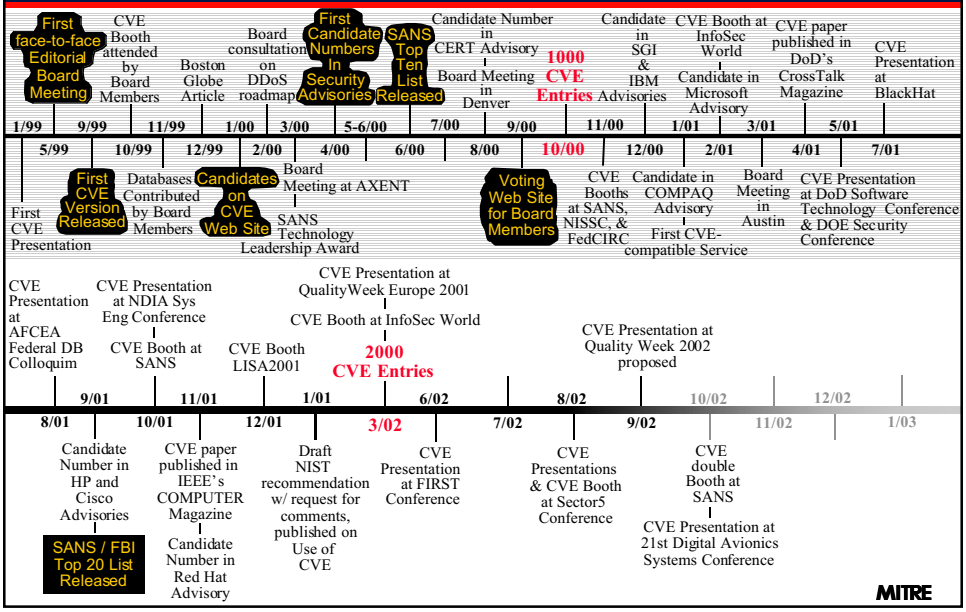


Status
(as of July 1, 2002)

- **2223 entries**
- **2419 candidates**

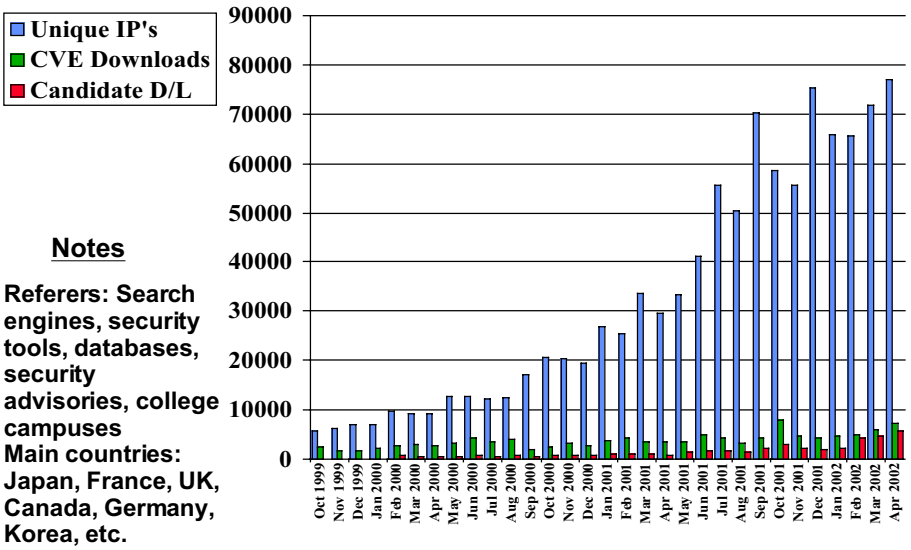
MITRE

Major CVE Milestones



MITRE

CVE Web Site Statistics

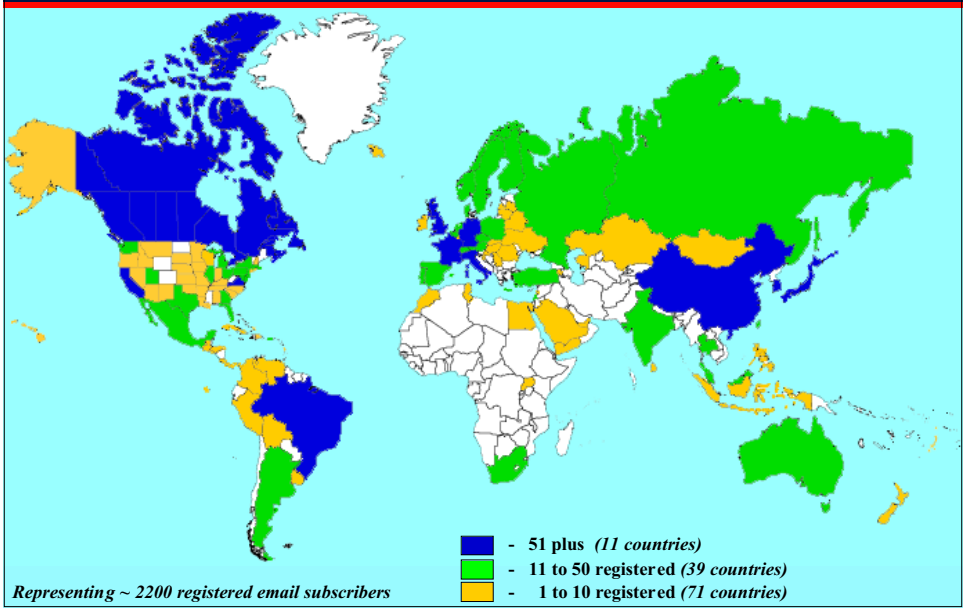


Notes

- Referers: Search engines, security tools, databases, security advisories, college campuses
- Main countries: Japan, France, UK, Canada, Germany, Korea, etc.

MITRE

CVE email Lists Have an International Readership



Representing ~ 2200 registered email subscribers

Outline

- Background and Motivation
- Finding Out About Vulnerabilities
- The Problem and a Solution - CVE
- CVE Compatibility
- The CVE Process
- Summary

MITRE

SANS Institute 2001 Top Ten uses CVE names …another step down the policy road

2. Vulnerable CGI programs and application extensions (e.g., ColdFusion) installed on web servers.

Most web servers support Common Gateway Interface (CGI) programs to provide interactivity in web pages, such as data collection and verification. Many web servers come with sample CGI programs installed by default. Unfortunately, many CGI programmers fail to consider ways in which their programs may be misused or subverted to execute malicious commands. Vulnerable CGI programs present a particularly attractive target to intruders because they are relatively easy to locate, and they operate with the privileges and power of the web server software itself. Intruders are known to have exploited vulnerable CGI programs to vandalize web pages, steal credit card information, and set up back doors to enable future intrusions, even if the CGI programs are secured. When Janet Reno's picture was replaced by that of Adolph Hitler at the Department of Justice web site, an in-depth assessment concluded that a CGI hole was the most probable avenue of compromise. Allaire's ColdFusion is a web server application package which includes vulnerable sample programs when installed. As a general rule, sample programs should always be removed from production systems.

Systems Affected:
All web servers.

CVE Entries:
** Sample CGI programs (All CGI)
Remedy:
Remove all sample CGI programs on a production server.

** CAN-1999-0736(II) 4.0, Microsoft Site Server 3.0, which is included with Microsoft Site Server 3.0 Commerce Edition, Microsoft Commercial Internet System 2.0, and Microsoft BackOffice Server 4.0 and 4.5)

(see http://www.microsoft.com/technet/security/bulletin/ms99-013.asp)
Remedy:
Apply patch at : ftp://ftp.microsoft.com/bussys/iis/iis-public/fixes/usa/Viewcode-fix/

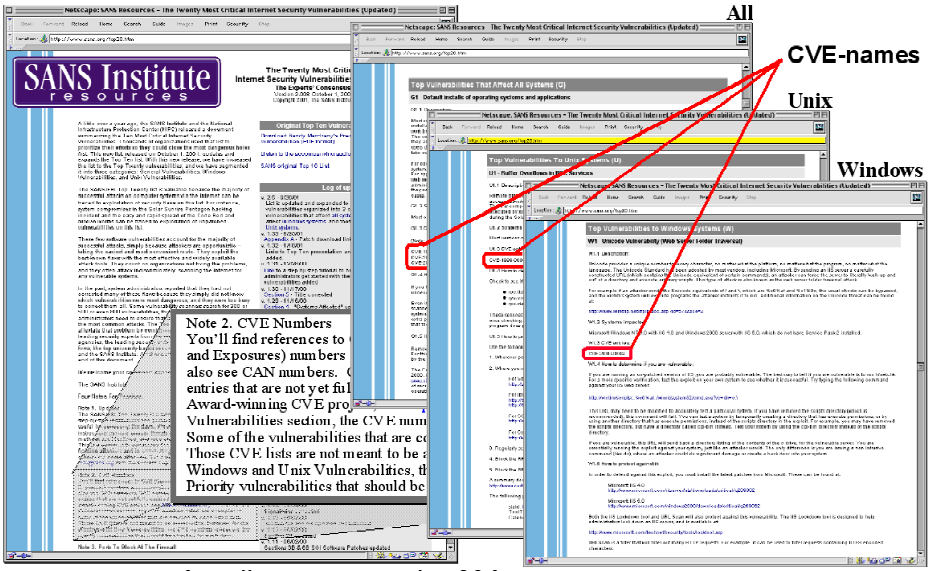
CVE-1999-0067 (phf phone book program included with older NCSA and Apache server)
CVE-1999-0068 (mylog.html' sample script shipped with the PHP/FI)
CVE-1999-0270 (IRIX 6.2, IRIX 6.3, IRIX 6.4)
CVE-1999-0346 (sample script shipped with the PHP/FI package)
CVE-2000-0207 (IRIX 6.5)

CVE-names

http://www.sans.org/topten.htm

MITRE

FBI/SANS Institute 2001 Top Twenty uses CVE names …yet another step down the policy road

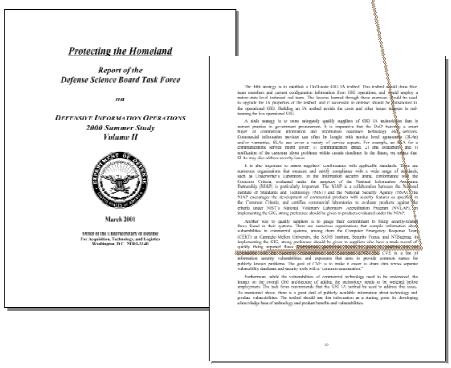


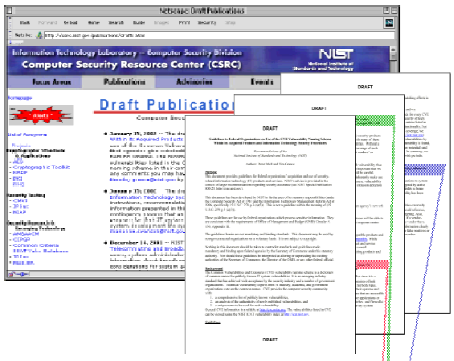
http://www.sans.org/top20.htm

MITRE

Several Parts of the Federal Government Have Called for the Use of CVE and CVE-Compatible products

Furthermore, preference should be given to products that are Compatible with the Common Vulnerabilities and Exposures (CVE) list.





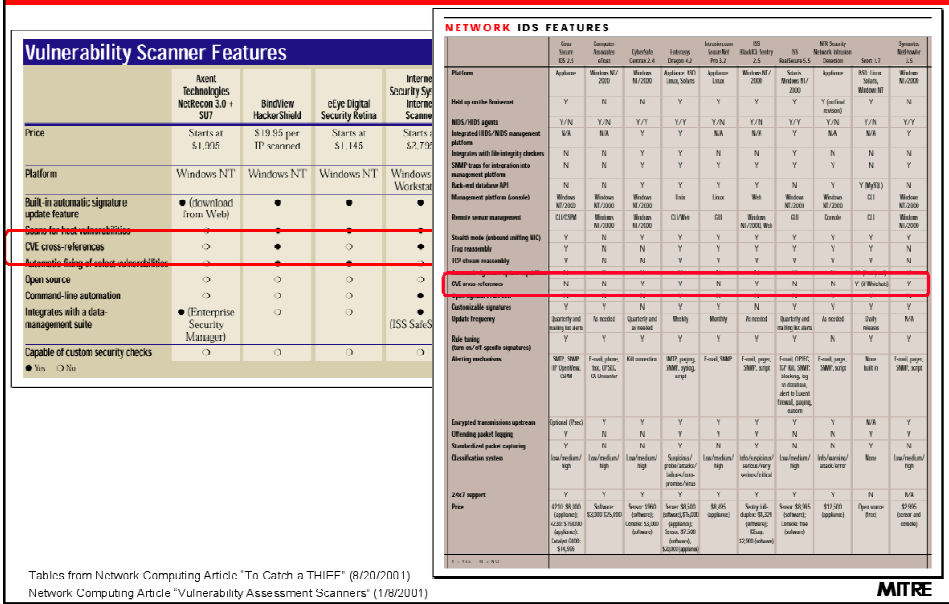
Federal departments and agencies should…

1. **give substantial consideration to … [CVE-compatible] products and services.**
2. **periodically monitor their systems for applicable vulnerabilities listed in … CVE**
3. **use [CVE] in their descriptions and communications of vulnerabilities**

http://www.acq.osd.mil/dsb/tfreports.htm
http://csrc.nist.gov/publications/drafts/Use_of_the_CVE.PDF

MITRE

Example:
CVE helping to make Detailed Product Comparisons



Vulnerability Scanner Features

	Axent Technologies NetRecon 3.0 + SU7	BindView Hacker Shield	eEye Digital Security Retina	Internet Security Sy Internet Scanne
Price	Starts at $1,995	$19.95 per IP scanned	Starts at $1,145	Starts a $2,70
Platform	Windows NT	Windows NT	Windows NT	Windows Workstat
Built-in automatic signature update feature	● (download from Web)	●	●	●
Scans for host vulnerabilities	○	●	●	●
CVE cross-references	○	●	○	●
Automatic fixing of solved vulnerabilities	○	●	●	●
Open source	○	○	○	○
Command-line automation	○	○	○	●
Integrates with a data-management suite	● (Enterprise Security Manager)	○	○	● (ISS SafeS
Capable of custom security checks	○	○	○	○

● Yes ○ No

NETWORK IDS FEATURES

	Cisco Secure IDS 2.5	Computer Associates eTrust	CyberSafe Centrax 2.4	Enterasys Dragon 4.2	Intrusion.com SecureNet Pro 3.2	ISS BlackICE Sentry 2.5	ISS RealSecure 5.5	NFR Security Network Intrusion Detection	Snort 1.7	Symantec NetProwler 3.5
Platform	Appliance	Windows NT/ 2000	Windows NT/2000	Appliance: BSD, Linux, Solaris	Appliance: Linux	Windows NT/ 2000	Solaris, Windows NT/ 2000	Appliance	BSD, Linux, Solaris, Windows NT	Windows NT/2000
Hold up on the Internet	Y	N	N	Y	Y	Y	Y	Y (optional version)	Y	N
NIDS/HIDS agents	Y/N	Y/N	Y/Y	Y/Y	Y/N	Y/N	Y/Y	Y/N	Y/N	Y/Y
Integrated IDS/NIDS management platform	N/A	N/A	Y	Y	N/A	N/A	Y	N/A	N/A	Y
Integrates with file integrity checkers	N	N	Y	Y	N	N	Y	N	N	N
SNMP traps for integration into management platform	N	N	Y	Y	Y	Y	Y	Y	N	Y
Back-end database API	N	N	Y	Y	Y	Y	N	Y	Y (MySQL)	N
Management platform (console)	Windows NT/2000	Windows NT/2000	Windows NT/2000	Unix	Linux	Web	Windows NT/2000	Windows NT/2000	CLI	Windows NT/2000
Remote sensor management	CLI/CSPM	Windows NT/2000	Windows NT/2000	CLI/Web	GUI	Windows NT/2000, Web	GUI	Console	CLI	Windows NT/2000
Stealth mode (inbound sniffing NIC)	Y	N	Y	Y	Y	Y	Y	Y	Y	Y
Frag reassemble	Y	N	N	Y	Y	Y	Y	Y	Y	N
TCP stream reassembly	Y	N	N	Y	Y	Y	Y	Y	Y	N
CVE cross-reference	N	N	Y	Y	N	Y	N	N	Y (if Whitehats)	Y
Customizable signatures	Y	Y	N	Y	Y	N	Y	Y	Y	Y
Update frequency	Quarterly and mailing list alerts	As needed	Quarterly and as needed	Weekly	Monthly	As needed	Quarterly and mailing list alerts	As needed	Daily releases	N/A
Rule tuning (turn on/off specific signatures)	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
Alerting mechanisms	SNMP, SMAP, IP QuickView, CSPM	E-mail, pager, Net, UPSEL, OS Commander	Kill connection	SMTP, paging, SNMP, syslog, script	E-mail, SNMP	E-mail, pager, SNMP, script	E-mail, OPSEC, IGF-NAI, SNMP, blocking, log as database, alert to Lucent firewall, paging, custom	E-mail, pager, SNMP, script	None built in	E-mail, pager, SNMP, script
Encrypted transmissions upstream	Optional (Pro)	Y	Y	Y	Y	Y	Y	Y	N/A	Y
Offending packet logging	Y	N	N	Y	Y	Y	N	N	Y	Y
Standardized packet capturing	Y	N	N	Y	N	Y	N	N	Y	N
Classification system	Low/medium/ high	Low/medium/ high	Low/medium/ high	Suspicious/ probe/attacks/ failures/core-promise/virus	Low/medium/ high	Info/suspicious/ serious/very serious/critical	Low/medium/ high	Info/warning/ attack/error	None	Low/medium/ high
24x7 support	Y	Y	Y	Y	Y	Y	Y	Y	N	N/A
Price	$795-$9,995 (appliance); 42.85-$160,000 (appliance); Catalyst 6000: $14,995	Software: $2,000-$25,000	Sensor: $960 (software); Console: $3,000 (software)	Sensor: $9,500 (software),$15,000 (appliance); Server, $17,500 (software), $40,000 (appliance)	$8,495 (appliance)	Sentry (all-duplex): $5,249 (enterprise); $2,000 (software); $2,995 (software)	Sensor: $8,995 (software); Console: free (software)	$17,500 (appliance)	Open source (free)	$7,995 (server and console)

Tables from Network Computing Article "To Catch a THIEF" (8/20/2001)
Network Computing Article "Vulnerability Assessment Scanners" (1/8/2001)

MITRE

CVE is Even Being Used to to Compare and Contrast products

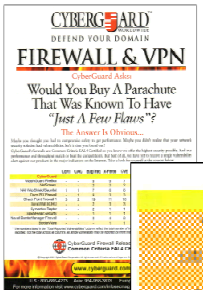
… or the vulnerabilities they do or don't find...

NETWORK IDS SIGNATURE RESULTS

Attack	CVE	No. of packets	Cisco Secure IDS 2.5	Enterasys Dragon 4.2	Intrusion.com SecureNet Pro 3.2	ISS BlackICE Sentry 2.5	ISS RealSecure 5.5	NFR Security NFR/Network Intrusion Detection	Snort 1.7	Symantec NetProwler 3.5
AMD	CVE-1999-0704	11	Y	Y	N	Y	Y	N	Y	N
NOS	CVE-1999-1011	22	Y	Y	N	Y	Y	Y	Y	Y
WU-FTP	CVE-1999-0368	44	N	Y	N	N	Y	Y	Y	N
SNMP write	CAN-1999-0517	2	N	Y	N	N	Y	Y	N	N
Guest SMB login	CAN-1999-0519	19	N	Y	N	Y	Y	N	Y	N
IMAPD	CVE-1999-0005	8	Y	Y	Y	N	Y	Y	Y	N
PHF	CVE-1999-0067	10	Y	Y	Y	Y	Y	Y	Y	Y
Unicode	CVE 2000 0884	10	Y	Y	N	Y	Y	Y	Y	N
IIS 6 ISAPI	CAN 2001 0241	11	Y	Y	N	N	N	Y	Y	N
Total (out of 9)			6	9	2	5	8	7	8	2
Detect attacks fragmented (Frag-T9)			Y	Y	Y	Y	Y	Y	Y	N

Y = YES N = NO

Tables from Network Computing Article "To Catch a THIEF" (8/20/2001)



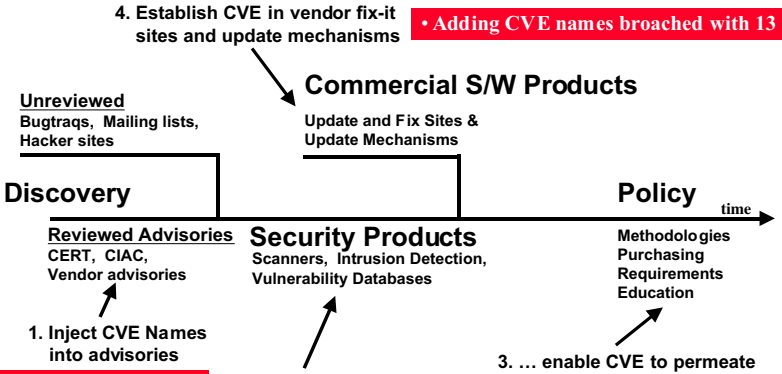
	CERT	CIAC	BugTraq	X-Force	CVE
CyberGuard	-	-	-	-	-
[illegible]	-	-	9	9	7
[illegible]	-	-	2	2	2
[illegible]	1	1	7	6	6
[illegible]	-	1	9	3	3
[illegible]	3	2	19	11	10
[illegible]	-	-	2	3	3
[illegible]	-	-	2	1	1
[illegible]	-	-	1	1	1
[illegible]	-	-	3	4	3
[illegible]	-	-	-	1	1

Ad from SC Magazine (April 2002)

by talking about the vulnerabilities they do or do not have...

MITRE

The CVE Strategy: Where are we? (as of 1 July 2002)

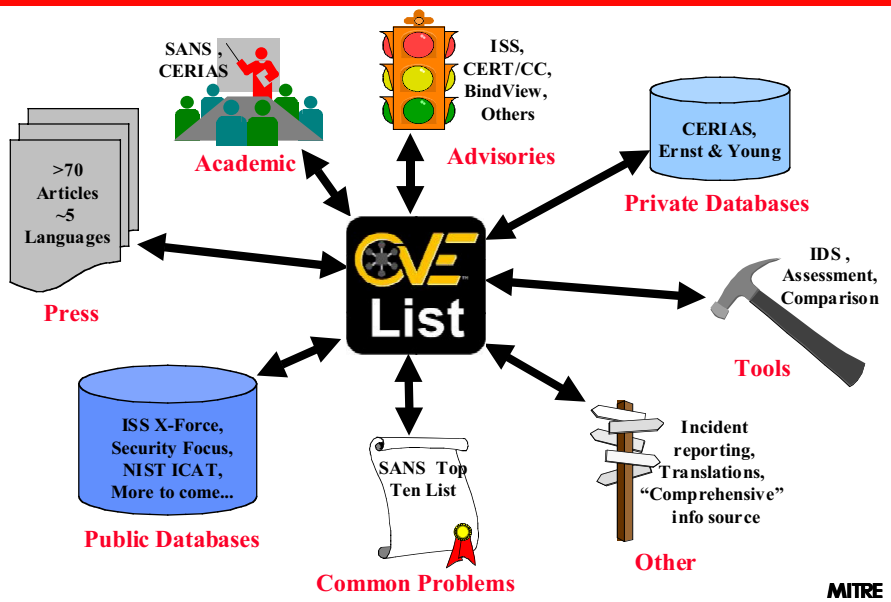


CVE names have been included in initial advisories from ISS X-Force, Rain Forest Puppy, IBM, @stake, BindView, CERT/CC, HP, SGI, COMPAQ, Microsoft, Ernst & Young, eEye, CISCO, Rapid 7, NSFOCUS, Sanctum, SecurityFocus, VIGILANTe, Red Hat, Apache, and Apple.

- 2223 CVE Entries -- 2419 Candidates.
- 95 CVE-compatible products from 62 groups.
- 53 more from 26 others in "the works".

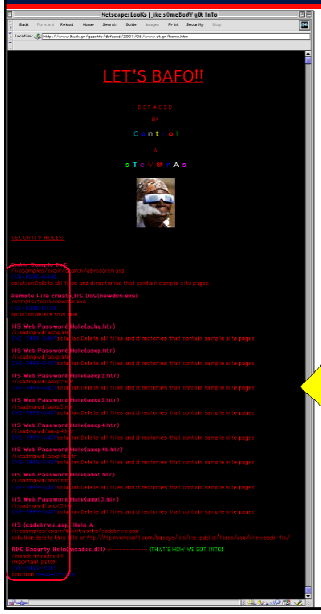
- SANS / FBI Top 20 uses CVE names
- Network Computing IDS & Scanner Comparisons included CVE
- Draft NIST Rec. calls for use of CVE
- DSB Report calls for CVE compatibility
- Network World IDS Comparison included CVE coverage

CVE is the center of many activities and efforts… ...and it's still growing



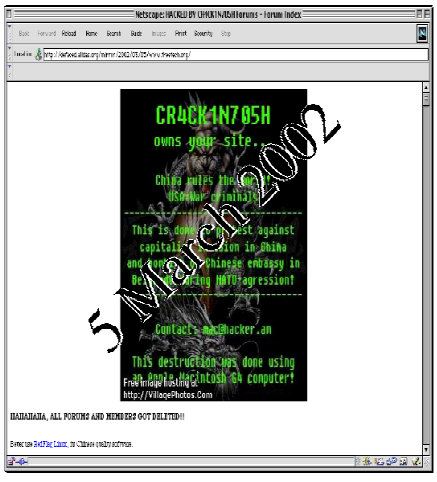
MITRE

CVE is even getting used by Hackers !



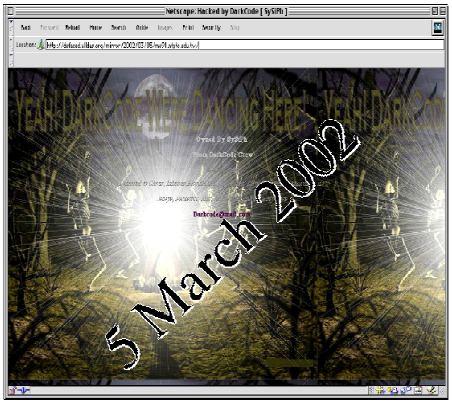
At least two hackers are now suppling CVE names for the vulnerabilities that they find in the sites they hack into.

And Yes, In Case You Wondered… …the Hacking Continues



Taiwan Education Site
http://me91.ntptc.edu.tw

Linux and Solaris Free Forums
http://www.freetech.org



MITRE

For More Information

CVE web site

http://cve.mitre.org

MITRE

The Vulnerabilities of Developing on the Net

Robert A. Martin
The MITRE Corporation

Disaster has struck. You would think that firewalls, combined with filtering routers, password protection, encryption, and disciplined use of access controls and file permissions would have been enough protection. However, an overlooked flaw in the commercial web server application allowed a hacker to use a buffer overflow attack to leverage the application's privileges into administrator-level access to the server. From there it was easy to gain access to other machines within the Intranet and replace the public Web pages with details of the hack. With the company's public site showing a live video stream of an ongoing internal, private and sensitive company meeting, it left little room for doubt as to how badly they had been hacked.

While most organizations have addressed the various aspects of implementing cyber security, many are failing to successfully address the one security area where someone can bypass all other efforts to secure the enterprise. That area is finding and fixing known security problems in the commercial software used to build the systems. There may be an answer, however, that will transform this area from a liability into a key asset in the fight to build and maintain secure systems. The answer rests in an initiative to adopt a common naming practice for describing the vulnerabilities, and the inclusion of those names within security tools and services. The initiative has been in practice for more than a year across a broad spectrum of the information security and software products community: It is called the Common Vulnerabilities and Exposures (CVE) initiative.

To Err Is Human

Every programmer knows they make mistakes when writing software, whether it be a typo, a math error, incomplete logic, or incorrect use of a function or command. Sometimes the mistake is even earlier in the development process – reflecting an oversight in the requirements guiding the design and coding of a particular function or capability of a software program. When these mistakes have security implications, those with a security bent will often refer to them as vulnerabilities and exposures.[1]

All types of software, from large complex pieces to small and focused ones, are likely to contain software mistakes with security ramifications. Large complex software like operating systems, database management systems, accounting systems, inventory management systems, as well as smaller applications like macros, applets, wizards, and servlets need to be evaluated for mistakes that can impact their security integrity. Remember that when we put these various software products together to provide an overall system, each of the software elements that make up the system could be the one that compromises it.

Things were different in the past when an organization's computer systems were stand-alone and only interacted with other systems within the same organization. Only a few systems used tapes and file passing to exchange information with outside systems. The same holds true for government and military systems, including weapons. This isolation meant that errors in commercial or developed software usually had limited impact, at least from the public's point of view. In fact, most errors, crashes, and oversights went unnoticed by the general public. At most, these problems would cause occasional troubles for an organization's closest business partners.

There Is No Hiding Now

The same is not true today. Very few of today's organizations, whether in the private sector or government, have or build self-contained systems. It is the norm for employees, customers, business partners, and the general public to have some degree of access and visibility into the minute-by-minute health and performance of an organization's software environment. Processing delay, calculation mistakes, system downtime, even response time slowdowns are noticed and often draw criticism.

Accompanying this increased visibility is an explosion in the different ways systems are accessed and used. Web and application servers have been created to help make systems interconnect and leverage Internet-based technologies. Access to web sites, purchase sites, online help systems, and software delivery sites makes the organizations that own the sites very visible. To better support business partners and employees working at remote locations, on the road, or from home, we have connected our backroom systems to the corporate Intranet and extranet. New technologies have emerged, like instant messaging, mobile code, and chat, whose functionality requires effortless access by users across organizational boundaries. The movement to highly accessible systems, driven by the need to save time and make businesses more efficient, and the reality of having to do more with less, has dramatically increased the impact of mistakes in commercial software.

While errors in self-developed software can still have a major impact on an organization's ability to function, it is the vulnerabilities and exposures in the commercial software they use to build systems that creates the bigger problem. A mistake in a commercial program can open a front or a back door into situations that most organizations strive to avoid. A mistake permitting unauthorized access can expose private information about customers and employees. It can allow hackers to change information or perform services with your systems to their own advantage. In addition, a vulnerability can allow them to shut down your internal and publicly accessed systems, sometimes, without your knowledge. In those cases where the vulnerability or exposure allows someone to make changes or bring down systems, or when the theft of services and information is eventually

noticed[2], there can be a huge impact to the organization's public image[3]. There can also be legal liability and direct operational impact.

What Can You Do?

Determining the vulnerabilities and exposures embedded in commercial software systems and networks is a critical "first step" to fixing the problems. A simple patch, upgrade, or configuration change could be sufficient to eliminate even the most serious vulnerability, if you know what you need and how to get it.

To find information about vulnerabilities in commercial software that your organization uses, you have to do some research and probably spend some money. With commercial software, the customer has little or no insight into the implementation details. At the very best you may have an understanding of the general architecture and design philosophy of a package. Companies offering commercial software treat the design details and software code as business-critical private information. In addition, since most of these companies are highly competitive, commercial software vendors are sometimes reluctant to share their problems, even with their customers.

Who Knows?

So how do you find out about commercial software vulnerabilities if the vendors are not going to tell you? During the last decade, three groups have emerged who share the same curiosity. For sake of discussion we will refer to these as the hackers[4], the commercial interests, and the philanthropists. The hackers, unfortunately, want to find vulnerabilities and exposures so they can exploit them to gain access to systems.

Those with commercial interests want to be hired to find the mistakes, or they want you to buy their tools to help you find the vulnerabilities and exposures yourself. They offer their services through consultants who will evaluate your software systems, and through tools that you can buy and run yourself. Some proffer the use of their tools as an Internet-based service. This group includes software and network security companies that provide security consulting services and vulnerability assessments, databases of vulnerabilities and exposures, and the tools for security services and vulnerability evaluations.

The philanthropists include security researchers in various government, academic, and non-profit organizations, as well as unaffiliated individuals that enjoy searching for these types of mistakes, usually sharing their knowledge and tools freely.

Each group has members focused on sharing information: among like-minded hackers, for a price in most cases for the commercial interests group, and generally for free in the philanthropists group. For all three groups the search for vulnerabilities and exposures in commercial software is challenging since the commercial marketplace is constantly developing and authoring new classes of software capabilities and new ways of using them. This mushrooming of commercial software capabilities also creates an ever-changing challenge for organizations using commercial systems. The challenge is to correctly configure and integrate the offerings of various vendors without opening additional vulnerabilities and exposures from configuration and permission mistakes.

How to Find Out

In response to the arduous task of tracking and reacting to new and changing vulnerabilities and exposures, the members of these three groups are using Web sites, news groups, software and database update services, notification services like e-mail lists, and advisory bulletins to keep their constituents informed and current.

So information on vulnerabilities in commercial software is available. That is great, right? Well, not quite. There are several problems. The biggest is that each organization (or individual) in these three groups has been pursuing their vulnerability discovery and sharing efforts as if they were *the* source of information on vulnerabilities. Each uses its own approach for quantifying, naming, describing, and sharing information about the vulnerabilities that they find. Additionally, as new types of software products and networking are introduced, whole new classes of vulnerabilities and exposures have been created that require new ways of describing and categorizing them.

Another problem is that finding the vulnerabilities and exposures within systems is just the first step. What we really want to do is to take the list of vulnerabilities and get them fixed. This is the software vendors' domain – those who create and maintain our commercial products. Unless they use the same descriptions and names as the hackers, commercial interests, and philanthropists groups, it is difficult, confusing and frustrating to get the fix for any particular problem you find.

A Closer Look at Who Knows What

The Internet is the main conduit hackers use to share information on vulnerabilities and how to exploit them. Different member organizations in the commercial interests group have their own mechanisms for sharing vulnerability information. For example, the tool vendors create vulnerability scanners that are driven by their own vulnerability databases. The intrusion detection system (IDS) vendors build different types of software systems for monitoring your network and systems for attacks. There are also scanner and IDS tools available from the philanthropists group as freeware. Both the scanner and IDS providers have to continuously update their tools with new information on what and how to look for problems. Examples of these organizations and tools are shown in Table 1.

Scanners typically include tests that compare version infor-

Table 1. *Scanner and IDS Offering Examples*

Product	Tool Type	Organization
Centrax	scanner/IDS	CyberSafe
CyberCop	scanner	Network Associates
Dragon	IDS	Network Security Wizards
HackerShield	scanner	Bindview Corporation
LANPATROL	IDS	Network Security Systems
Nessus	freeware scanner	Renaud Deraison & Jordan Hrycaj
NetProwler	IDS	Axent Technologies
QualysGuard	ASP-based scanner	Qualys
RealSecure	IDS	Internet Security Systems
Retriever	scanner	Symantec Corporation
SAINT	scanner	World Wide Digital Security
Secure IDS	IDS	Cisco Systems
STAT	scanner	Harris Corporation
SWARM	scanner	Hiverworld, Inc.

mation and configuration settings of software with an internal list of vulnerability data. They may also conduct their own scripted set of probes and penetration attempts. IDS products typically look for indications of actual attack activities, many can then be mapped to the specific vulnerabilities that these attacks could exploit. The scanner market recently developed a self-service-based capability. It uses remotely hosted vulnerability scanners on the Internet that you can hire to scan your Internet resident firewalls, routers, and hosts. The results of the scans are provided through a secure link, and you can usually run the scans whenever you want. These scans are shielded from everyone but you, including the service provider. IDS capabilities are often available as part of a managed security service, where the organization contracts out the intrusion detection and monitoring to a security services vendor.

Both IDS and scanner tool providers harvest information about vulnerabilities and exposures from public information sites, hacker sites, newsletters, and advisories. They also have their own investigative researchers who continuously look for new vulnerability information that will make their company's offering better than the competition[5], as well as providing them with the "free" advertising that comes with finding and publicly reporting new vulnerabilities and exposures. Typically, these researchers serve as consultants within the company, offering their services to evaluate an organization's systems and networks. Their parent companies also offer databases of vulnerabilities for a fee, although some also share the information openly as raw information on a Web site.

Some members of the philanthropists group also offer very sophisticated search and notification services for free, but their veracity, quality, and levels of effort vary considerably. Examples of vulnerability-sharing organizations are shown in Table 2.

Site Name	Type	Organization
arachNIDS	free IDS database	Max Vision Network Security/Whitehats
CERIAS Vulnerability Database	database	CERIAS/Purdue University
Fyodor's Playhouse	hacker web site	Insecure.Org
Online Vulnerability Database	database	Ernst & Young's eSecurityOnline.com
ICAT Metabase	free web site	NIST
Bugtraq mailing list Database	mailing list database	SecurityFocus.com
PacketStorm	hacker web site	Securify, Inc.
SWAT Database	database	Axent Technologies
Vigil@nce AQL	database	Alliance Qualité Logiciel
X-Force Database	free web site	Internet Security Systems

Table 2: *Vulnerability Sharing Examples*

In addition to freeware scanner, IDS tools, and vulnerability databases, the philanthropists group's government and academic members offer several announcement, alert, and advisory services that are widely used and highly valued. Some commercial interests group companies offer these types of free services as well. Examples are shown in Table 3.

There are numerous venues for finding out what vulnerabilities and exposures exist in your organization's commercial software systems, as well as many tools and service providers willing to help you determine which vulnerabilities and exposures you have.

The three groups we've covered -- hackers, commercial interests, and philanthropists – all address locating the vulnerabilities and exposures in the commercial software that forms the base of your live systems and networks.

Service	Type	Organization
Bugtraq	e-mail list	Bugtraq
Casandra	alerts	CERIAS/Purdue University
CERT Advisories	advisory	CERT Coordination Center
CyberNotes	monthly newsletter	NIPC
Razor	advisory	Bindview Corporation
S.A.F.E.R.	monthly newsletter	The Relay Group
SANS NewsBites	e-mail list	SANS Institute
Security Alert Consensus	e-mail list	Network Computing and SANS
SecurityFocus Newsletter	newsletter summary of Bugtraq e-mails	SecurityFocus.com
SWAT Alerts	alerts	Axent Technologies
X-Force Alert	advisory	Internet Security Systems

Table 3: *Alert and Advisory Services Examples*

We will now address finding the "fixes." The product vendors who make the software in which these vulnerabilities were found provide the solutions for vulnerabilities. Many of them have their own methods of providing their customers with software fixes and updates. Until recently, most vendors were not very proactive in distributing patches and updates outside of their normal software development cycle. This has improved considerably. Now, many major vendors provide alerts and advisories concerning security problems, fixes, and updates (See Table 4).

Service	Type	Organization
IBM ERS	advisory	IBM
Microsoft Product Security Notification Service	advisory	Microsoft Corporation
SGI Security Advisory	advisory	Silicon Graphics, Inc.
Sun-alert	alert	Sun Microsystems, Inc.

Table 4: *Vendor Alert and Advisory Services Examples*

But can these various vulnerability services, tools, and databases, along with the software vendor's update announcements effectively combine to help you assess, manage, and fix your vulnerabilities and exposures? The short answer is that it used to be very difficult, but now a way to do it seems to be at hand. So what was wrong, and what changed?

The Tower of Babel

In 1998 if you tried to use these various tools, services, and databases you were faced with a problem rooted in each ones heritage. Each had developed its own naming standards and methods for defining individual entries in their respective vulnerability data stores. Table 5 shows how the same vulnerability was referred to by 12 different names by 12 leading organizations. With such confusion, it was very hard to understand what vulnerabilities were faced, and what vulnerabilities were or were not being looked for by each tool. Then the vulnerability or exposure still had to be mapped to the software vendor's name for the problem to get a fix.

Driven by our own attempts to develop an integrated picture of what was happening in our networks and in trying to select some new tools, The MITRE Corporation[6] started to design a method for working through the confusion of vulnerability and exposure information. This method was based on the creation of a reference list of unique names that would then be mapped to the appropriate items in each tool and database. In January 1999 the first public statement of our idea for a

Organization	Name used to refer to vulnerability
AXENT	phf CGI allows remote command execution
BindView	#107 – cgi-phf
Bugtraq	PHF Attacks – Fun and games for the whole family
CERIAS	http_escshellcmd
CERT	CA-96.06.cgi_example_code
Cisco Systems	HTTP - cgi-phf
CyberSafe	Network: HTTP 'phf' Attack
DARPA	0x00000025 = HTTP PHF attack
IBM ERS	ERS-SVA-E01-1996:002.1
ISS	http - cgi-phf
Symantec	#180 HTTP Server CGI example code compromises http server
Security Focus	#629 - phf Remote Command Execution Vulnerability

Table 5: *Vulnerability in the Tower of Babel*

reference list of unique names for vulnerabilities and exposures was presented at the 2nd Workshop on Research with Security Vulnerability Databases, held at Purdue University. MITRE presented a paper [1] at this conference that outlined the basic ideas and approach for what is today called the Common Vulnerabilities and Exposures (CVE) initiative.

Our vision for CVE was to provide a mechanism for linking together vulnerability-related databases or concepts – and nothing more (See Figure 1). Rather than viewing this narrow scope as a limitation, we saw it as an advantage. By agreeing to limit the use of CVE to the role of a logical bridge, we could avoid competing with existing and future commercial efforts. This was important, since it was critical that commercial organizations concur with the CVE concept and proceed to incorporate the initiative into their various products and services.

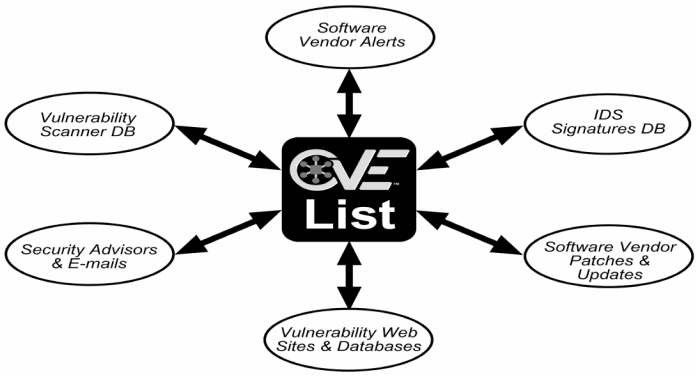


Figure 1: *CVE List as a Bridge*

By March 2001 the CVE effort had evolved into a, cross-industry effort involving more than 30 occupations in creating and maintaining a standard list of vulnerabilities and exposures. Almost half of the known vulnerabilities and exposures are either listed or under review, and presently 29 organizations are building nearly 50 products or services that use CVE names as a key element of their functionality.

How CVE Works

The CVE initiative is an international community activity focused on developing a list that provides common names for publicly known information security vulnerabilities and exposures. The CVE list and information about the CVE effort are available on the CVE Web site at cve.mitre.org/cve/.

The common names in CVE result from open and collaborative discussions of the CVE editorial board. This board, as shown in Table 6, includes members from numerous information security-related organizations around the world, including

Area of Expertise	Organizations
Academic/Educational	UC Davis, SANS, CERIAS
Network Security Analysts	Vista IT, Genuity
Other Security Experts	IBM Research, MITRE, Zero-Knowledge Systems
Intrusion Detection Experts	Silicon Defense, SANS
Tool Vendors	The Nessus Project, ISS, PGP Security-Network Associates, BindView, AXENT, CyberSafe, Symantec, NFR, Hiverworld, Harris, Cisco
Software Vendors	IBM, Sun Microsystems, Microsoft
Incident Response Teams	CanCERT, CERT/CC, DOD-CERT
Information Providers	NTBugtraq, Security Focus, National Institute of Standards and Technology (NIST), Ernst & Young, eSecurityOnline.com

Table 6: *CVE Editorial Board Composition*

commercial security tool vendors, members of academia, research institutions, government agencies, and other prominent information security experts. The board identifies which vulnerabilities or exposures will be included in CVE, then determines the common name, description, and references for each entry. The CVE name, for example CVE-1999-0067, is an encoding of the year that the name was assigned and a unique number N for the Nth name assigned that year.

MITRE maintains the CVE list and Web site, moderates editorial board discussions, and provides guidance throughout the process to ensure that CVE remains objective and continues to serve the public interest. Archives of board meetings and discussions are available for review on the CVE web site at cve.mitre.org/board/archives/. Other information security experts are invited to participate on the board on an as-needed basis, based upon recommendations from board members.

The key tenets of the CVE initiative are:

- One name for one vulnerability or exposure.
- One standardized description for each vulnerability or exposure.
- Existence as a dictionary rather than a database.
- Publicly accessible for review or download from the Internet.
- Industry-endorsed via the CVE editorial board and CVE-compatible products.

What Does CVE-Compatible Mean?

CVE-compatible is a phrase that indicates that a tool, Web site, database, or service uses CVE names in a way that allows for a cross-link with other repositories that use CVE names. To be CVE-compatible, the product, service, database, or Web site must meet the following three requirements:

- CVE Searchable: A user can search using a CVE name to find related information.
- CVE Output: Information is presented that includes the related CVE name(s).
- Mapping: The repository owner has provided a mapping relative to a specific version of CVE, and has made a good faith effort to ensure accuracy of that mapping.

Different products and repositories address different portions of the complete CVE list. For example, some might deal with UNIX, while others cover Windows NT. When looking at CVE-compatible items, you will need to evaluate them against your organization's specific needs in terms of platforms coverage and the software products that you use.

Why Use CVE-Compatible Products?

CVE compatibility allows you to use your vulnerability databases and tools together since they can "talk" to each other through shared CVE names. For example, if a report from a vulnerability scanning tool incorporates CVE names, you can quickly and accurately locate fix information in one or more of the separate CVE-compatible databases and Web sites to determine how to fix the problems identified by the vulnerability scanner. Also, with CVE-compatible tools, you will know exactly what each tool covers because the CVE list provides a baseline. Simply determine how many of the CVE entries are applicable for your platforms, operating systems, and commercial software packages, and use this subset to compare against the tool's coverage. Before the use of common names, it was extremely difficult to identify the vulnerabilities of your systems[7], or to determine whether a particular tool or set of tools covered them.

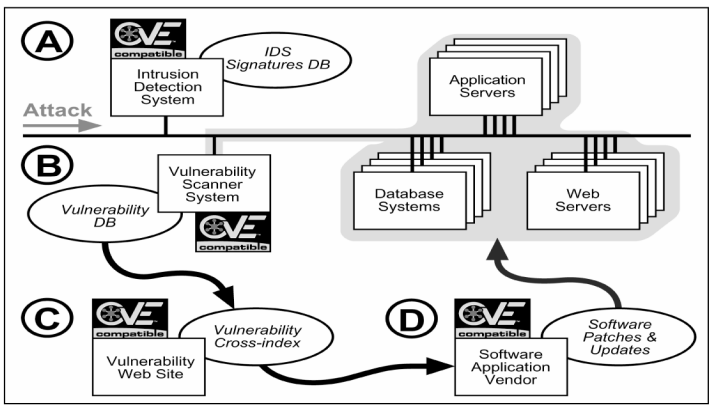
Improving the Process

The CVE effort is changing the way organizations use security tools and data sources to address their operational security posture. The example organization in Figure 2 is able to detect an ongoing attack with its CVE-compatible IDS system (A). In a CVE-compatible IDS, specific vulnerabilities that are susceptible to the detected attack are provided as part of the attack report. This information can then be compared against the latest vulnerability scan by your CVE-compatible scanner (B) to determine whether your enterprise has one of the vulnerabilities or exposures that can be exploited by the attack. If it does, you can turn to a CVE-compatible fix database at the software vendor or you can use the services of a vulnerability Web Site like ICAT Metabase[8], which lets you identify (C) the location of the fix for a CVE entry (D), if one exists.

Identifying Your Risk

Another thing you can accomplish with CVE-compatible products, that would be hard if not impossible to do before common names were adopted, is improve how your organization responds to security advisories. If the advisory is CVE-compatible it will include CVE entries. With that information you can see if your scanners check for these vulnerabilities, and determine whether your IDS has appropriate attack signatures for the alert.

Figure 2: *A CVE-enabled process*



Additionally, for systems that you build or maintain for customers, the CVE compatibility of advisories and announcements will help you directly identify any fixes from commercial software vendors in those systems (if the vendor fix site is CVE compatible). This is a much more structured and predictable process for handling advisories than most organizations currently possess.

Making Guidance Actionable

Earlier this year, a group of concerned security professionals put together a "Top 10" list [2] that outlined the most common, critical Internet security threats. The effort was orchestrated by the System Administration, Networking, and Security (SANS) Institute and brought together a consensus list from a wide variety of security experts. To help bring specificity and make the recommendations actionable, each of the top 10 suggestions had the appropriate CVE names, detailing each of the specific issue areas for a variety of platforms and products. A total of 68 CVE names were called out in the list of 10 threats.

Advanced Research Corporation	Internet Security Systems, Inc.
Alliance Qualité Logiciel	Max Vision Network Security/Whitehats
AXENT Technologies, Inc.	NIST
BindView Development	The Nessus Project
CERIAS/Purdue University	Network Security Systems
CERT Coordination Center	Network Security Wizards
Cisco Systems	NTBugtraq
Computer Security Laboratory, UC Davis	PGP Security, Network Associates
CyberSafe	Qualys
CYRANO	SANS
Ernst & Young	Security Focus, Inc.
Harris Corporation	Security Watch
Hiverworld, Inc.	Symantec
Intranode	Tivoli Systems Inc.
Intrusion.com	World Wide Digital Security

Table 7: *Organizations Developing CVE-Compatible Products*

Who Is CVE-Compatible?

While the list of organizations with CVE-compatible products is expanding, at this writing the vendors in Table 7 are those working toward compatibility. For a current list visit the CVE web site at cve.mitre.org/compatible/.

Today, there are several members of each type of tool, service, repository, and announcement capability that support CVE names. Underrepresented areas are vendor announcement and vendor fix sites; however, several vendors are actively discussing adding CVE names to their announcements. By the time you read this article there should be several vendors using CVE names in their announcements and alerts. In addition, like the CVE editorial board, the list of organizations working on or delivering CVE-compatible products has become international in scope.

Conclusion

The application of all known security fixes and patches is the complement of standard security protection mechanisms. Keeping current on fixes offers a robust method for keeping the commercial software that makes up your organization's software infrastructure healthy. Vulnerabilities and exposures will always be a part of our systems, as will the groups that find and share

information about vulnerabilities and exposures in commercial software. With common name integration and cross-referencing abilities emerging in vulnerability and exposure tools, web sites, and databases, it is becoming possible to deal with these mistakes and improve our systems' security. Handling security incidences is more systematic and predictable as CVE is supported within the commercial and academic communities. As vendors respond to user requests for CVE-compatible fix sites, the complete cycle of finding, analyzing, and fixing vulnerabilities will be addressed.◆

On-Line Resources

The on-line resources of this article contain hyperlinks to further references. For the full list please see page 32 of this on-lne version.

References

1. Mann, David E. and Christey, Steven M., Towards a Common Enumeration of Vulnerabilities, 2nd Workshop on Research with Security Vulnerability Databases, Purdue University, West Lafayette, Ind., Jan. 21-22, 1999.
2. Jackson, William, Top 10 System Security Threats Are Familiar Foes, *Government Computer News*, Jun. 12, 2000.
3. Sullivan, Bob, Hospital Confirms Hack Incident, MSNBC, Dec. 9, 2000.
4. Lemos, Robert, Power Play: Electric Company Hacked, ZDNet News, Dec. 15, 2000.
5. Mell, Peter, The ICAT Metabase, Computer Security Division at the National Institute of Standards and Technology, icat.nist.gov/icat.taf Dec. 19, 2000.

Notes

1. Vulnerability is a mistake that someone can directly use to gain access to things they are not supposed to have. An exposure is a mistake that gives that person access to information or capabilities that he or she can then use, as a stepping stone, to gain access.
2. A computer hacker broke into a hospital in the Seattle area and thousands of medical records were downloaded. The hacker's activities went unnoticed by the hospital, and when the hacker went public with his accomplishment, his claims were initially denied. The next day, the hospital confirmed the intrusion [3].
3. A Microsoft Web site was penetrated by a Dutch hacker through the Web server's "IIS Unicode" vulnerability that let him copy files, execute commands, and change files [4].
4. Unlike its original meaning that referred to a hacker as a prolific and inventive software programmer, hacking during the past few years has come to refer to the act of circumventing security mechanisms of information systems or networks. "Black-hat" hackers are those intent on doing harm, as opposed to "white-hat" hackers, who are usually working in support of organizations to help them assess and understand the vulnerabilities and exposures in their systems. Black-hat hackers are sometimes referred to as crackers.
5. As an alternative to tracking and recording each update, patch, and upgrade that gets applied to each platform in the enterprise, the use of vulnerability scanners is an attractive choice for monitoring the health of software applications. These tools are benefiting from the vigor of the market place's hunt for vulnerability information and the development of testing approaches that can turn up the presence of vulnerabilities or exposures in the "deployed" systems of an organization. However, due to "false positives," "false negatives," and incomplete coverage to date, these tools are not a panacea.
6. MITRE, working in partnership with government, is an independent, nonprofit corporation working in the public interest.
7. The CVE initiative is in the process of analyzing and categorizing all of the "legacy" vulnerabilities and exposures, and assigning them CVE numbers. Numerous members of the security vulnerabilities reporting and tracking community have donated their legacy databases to the CVE effort to support this effort.
8. The ICAT Metabase is a searchable index of computer vulnerabilities and exposures. ICAT is not itself a vulnerability and exposure database, but is instead a searchable index leading to vulnerability resources and patch information [5].

Did this article pique your interest?

Would you like to learn more about correcting vulnerabilities and exposures in commercial software that is used to develop your organizations infrastructure? Then attend the Thirteenth Annual Software Technology Conference 2001 on April 29-May 4 in Salt Lake City. Robert A. Martin will speak on this topic in Track 9 on May 2.◆

About the Author

Robert A. Martin is a co-lead for MITRE's Cyber Resource Center Web-site, and a principal engineer in MITRE's Information Technologies Directorate. At the culmination of his five years of Y2K leadership and coordination efforts, Martin served as the operations manager of the Cyber Assurance National Information Center, a 24x7 cyber security watch center within the President's Y2K Information Coordination Center. Today, Martin's efforts are focused on the interplay of cyber security, critical infrastructure protection, and e-Business technologies and services. Martin received a bachelor's degree and a master's degree in electrical engineering from Rensselaer Polytechnic Institute and a master's of business degree from Babson College. He is a member of the ACM, AFCEA, Institute of Electrical and Electronics Engineers (IEEE), and IEEE Computer Society.

Robert A. Martin
The MITRE Corporation, MS B155
202 Burlington Road
Bedford, MA 01730-1420
Voice: 781-271-3001
E-mail: ramartin@mitre.org

QW2002 Paper 8I2

Mr. Steve Whitchurch
(Network Associates, Inc.)

Implementing a Web Based Testing Framework

Key Points

- Web Based Testing Frameworks, Do they really work
- Will a project team use a Web Based Testing Framework if its provided
- What type of Information should be included/shared through a Web Based Testing Framework

Presentation Abstract

This paper will describe a Web Based Testing Framework that ties together Product Requirements, Test Plans, Test Specifications, Testing Status, and Defect Reporting. I will talk about why this framework was used in place of the more traditional Software Engineering documents and tools. I will also talk about the construction of the testing frame work and how it has evolved through use and feedback from the project team. I will talk about the interaction of the project team with the testing framework, (what they liked and what they did not like). I will also talk about future enhancements like Automatic Test Data Collection, and Test Execution.

About the Author

Steve Whitchurch has been in the Software QA arena for 18 years. During that time he has worked at Intel, Mentor Graphics, Summit Design, Tektronics, and is currently a QW Team lead at Network Associates, Inc.. Steve has also been involved in testing everything from Real Time OS Software, Video Editing and Special Effects Software, to Electronic Design Automation Software.
Steve has also been active outside of the work environment as a Speaker at PNSQC, STAR, Quality Week, and SAO Forums. Steve was also the creator and publisher of the Software QA Magazine (now known as Software Testing & Quality Engineering Magazine, published by SQE). Steve has also written for the QA/Testing Web Site, Sticky Minds.

Implementing
A
Web Bases Testing Framework

By
Steve Whitchurch
Network Associates, Inc.

The Goals of the Project

- *Goal #1:* A solution that everyone agreed added value to the project
- *Goal #2:* A way to communicate not only test status, but current defect count, defect status, build status, test statistics by build, a quick health check of the product, and any special test information that may come up.
- *Goal #3:* Easy to use. The solution needed to be implemented in such a way that anyone on the project could access the information from any type of system (i.e. Unix or Windows).
- *Goal #4:* There needed to be links between all project documents. This would help with link test items to functional specification and requirements, etc.

Overview of Web Based Testing Framework

- Quick Status web page
- Supporting web pages

Why Use a Web Based Testing Framework

The Construction of the Web Based Testing Framework

Information that should be shared Using a Web Based Testing Framework

- Project Document
- Test Data
- Defect Data
- Defect Trends
- Testing Status

How did the
Web Based Testing Framework
Evolve Over Time

How Did The Project Team
Interact with the
Web Based Testing Framework

Future Plans

- Automatic Test Data Collection
- Test Execution
- Automated Reports
- Email Notification

Conclusion

- Did we meet the project goals?
- Would I do it Again?

Implementing a Web Based Testing Framework

By
Steve Whitchurch
Network Associates, Inc.
Beaverton, OR
steve_whitchurch@NAI.com

Abstract:

This paper will describe a Web Based Testing Framework that ties together Product Requirements, Test Plans, Test Specifications, Testing Status, and Defect Reporting. I will talk about why this framework was used in place of the more traditional Software Engineering documents and tools. I will also talk about the construction of the testing framework and how it has evolved through use and feedback from the project team. I will talk about the interaction of the project team with the testing framework, (what they liked and what they did not like). I will also talk about future enhancements like Automatic Test Data Collection, and Test Execution.

About the Author:

Steve Whitchurch has been in the Software QA arena for 18 years. During that time he has worked at Intel, Mentor Graphics, Summit Design, Tektronics, and is currently the Sr. QA Team Lead Engineer for the ePO Server group at Network Associates, Inc.. Steve has been involved in testing everything from Real Time OS Software, Video Editing and Special Effects Software, to Electronic Design Automation Software. Steve has also been active outside of the work environment as a Speaker at PNSQC, STAR, Quality Week, and SAO Forums. Steve was the creator and publisher of the Software QA Magazine (now known as Software Testing & Quality Engineering Magazine, published by SQE). Steve has also written for the QA/Testing Web Site, Sticky Minds.

1. The Goals of the project, what the team was looking for and why

The problem was to solve a communications gap between QA, Development, Marketing, and Management. The solution had to be one that would work for everyone associated with the project. It also had to be something that everyone would use, and added value to the project. The top-level goals were:

- *Goal #1:* A solution that everyone agreed added value to the project

- *Goal #2:* A way to communicate not only test status, but current defect count, defect status, build status, test statistics by build, a quick health check of the product, and any special test information that may come up.

- *Goal #3:* Easy to use. The solution needed to be implemented in such a way that anyone on the project could access the information from any type of system (i.e. Unix or Windows).

- *Goal #4:* There needed to be links between all project documents. This would help with link test items to functional specification and requirements, etc.

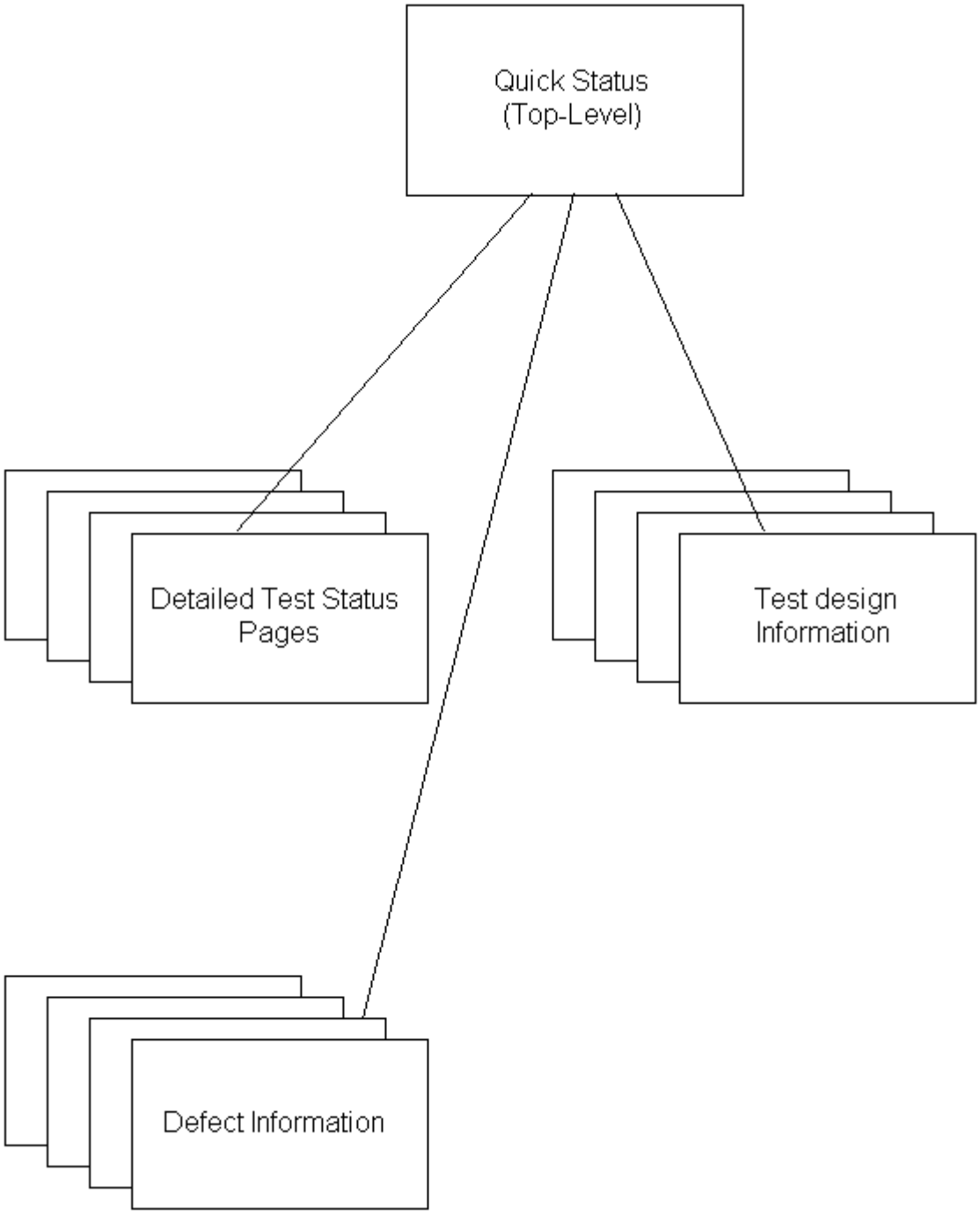
2. Overview of the Web Based Testing Framework

The Testing Framework consisted of two parts, the main web page and the supporting web pages:

- **The main web page:**
  The main or "Quick Status" web page was at the top level of the hierarchy. This web page displayed the most resent test status and defect information, as well as links to more detailed project information.

- **Supporting web pages:**
  The next web pages in the hierarchy were what I have called supporting web pages. These web pages contained the more detailed test data, test case information, defect data, links to supporting documentation (i.e. Functional Specifications, Test Plans), etc.

The web page hierarchy looked like the following:



3. Why use a Web Based Testing Framework over traditional Software Engineering documents?

The more traditional engineering documents (i.e. Functional Specifications, Test Plans, Test Specifications, etc.) mainly consist of paper. These types of documents are hard to update, and time consuming to distribute every time an update is made. Because of this, most of the time these documents are out of date. In some cases,

these documents are never updated.

The inverse of the traditional document is a Web Based document. These documents are very easy to update and very easy to distribute. Mainly because the deployment of these document is usually to one central place, a web server. There is no running around handing every project team member an updated copy of a current document. As long as everyone one the project team has access to the web server, then everyone on the project team will have access to current project documents and information.

There is still that discipline factor as with any document update. If no time is given to updating the project documents, the project documents will be out of date no matter what type of documentation system you have. The use of the web-based system just makes distribution a whole lot easier.

When it comes to status information, automated works in most cases. This type of project information can be setup to update without any time taken away from engineering tasks. The automatic updating process can also be applied to some of the project documents as well. Any time you can show a positive impact on a project schedule, you have a winning process. This is a big plus for the web-based approach.

4. How the Web Based Testing Framework was constructed. What tools, if any, were used

One of the goals was to have an information source that was easy to uses and accessible by anyone in on the project team. A web based “html” framework was selected. This made the testing framework accessible from any web browser running on any type of platform.

The tool used to create the framework web pages, was the Netscape Composer. There were those occasions, where the html file had to be opened in an editor, and changed to produce the exact result we wanted. But for the most part this tool worked out just fine.

The other tools that were used occasionally were screen capture tools. I used the native tools that came as part of the SunOS. The screen capture files where saved as gif images, allowing ease importing into the html code to be displayed as part of the web pages.

5. What information should be shared through a Web Based Testing Framework

I’m not sure if there is any information that should NOT be shared through a web-based system. Maybe the filter would be information that only needs to be seen by

certain people because of the nature of its contents. But all other project information could be shared using a web-based system.

The top-level information that my project shared was:

- Project Requirement
- Functional Specifications
- Design Level Specifications
- GUI Specifications
- Test Plans
- Test Design Specifications
- Testing Status
- Defect data
- Defect Trends

6. How did the Web Based Testing Framework evolve from the first web page? How did the project team help mold the testing framework.

The current web based framework took five iterations to finalize a look and feel that everyone on the project agreed was usable. The group also reviewed the data that was being presented.

The key to making a system like this work, is to get buy in from the complete project team. This is a very important step in the design process. If you do not get full buy in, I would be concerned that your project will eventually fail. This is not a step to skip.

I feel that the success of this project was largely do to the involvement of the project team in the review of the Web Based Testing Framework. They defined the data, they defined the look and feel, and they used the end product.

7. How does the Project Team interact with the Web Based Testing Framework?

- **What did the project team like?**

    - For the most part, the team like the complete system.
    - The team liked the linkage between the test cases and the defects filed. It was a kind of one-stop shopping. The user of the system did not need to visit one web site to review defect information. Then visit another to review or run the test case / test data that found the problem. This could all be done from one

web site.

- **What did the project team NOT like?**

  - The team would like to see more automation in the updating of the project (e.g. Test data, defect data, test status, project documentation) information.

  - Links into the source code control system. This would allow test case, defect, and code linkage. This would better document any fixes that where linked to a bug found through testing.

8. Future Plans for the Web Based Testing Framework

- **Test Execution from the Web Page**

  One of the areas of improvement that I'm very excited about is Test Execution. The plan is to have a button or menu that will allow the user of the web site to execute test cases at will. Each of the Test Specification pages will have this mechanism available. This will allow a developer or test engineer to execute any test he or she wants at anytime on any build.

  Example; Suppose a development engineer has added some new code. Could be a bug fix or new feature. He then wants to run a set of test cases that will verify that his new code has not broken other functionality. He can go to the Testing Framework web site, select the test or tests that he wants to run, and click a button to run them. The result of the test run would be a report that tells the engineer if his code is working.

  This feature would be very valuable to any project team member. It would also give the power of running test cases to anyone on the project team. What better way to get others testing.

- **Automatic Test Data collection**

  At the time this paper was written, the only automated part of the framework was some test statistics gathering. This was a large part of the data communicated on the Quick Status (top-level) web page. But a lot more automation needs to be completed.

  Example: It would be nice if test suites would run automatically after a build, then update the appropriate web pages on completion.

  This is just one example; there are lots of opportunities for automation with a system like the Web Based Testing Framework.

- **Project Team Wish List**

  The following is a list of features that the project team would like added to the Web Based Testing Framework:

  - Automated Test execution.
  - Automated reports.
  - A solid connection between the testing framework and the defect tracking system.
  - Email notification.

9. Conclusion

- **Did the Web Based Testing Framework meet the goals**

  Yes it did, in fact the project team used the Web Based Testing Framework all the time. Lets review each of the goals that were set at the start of the project:

  - *Goal #1: A solution that everyone felt added value to the project.*

    This goal was met by including everyone in the design of the tool. Everyone had input, everyone reviewed the progress of the tool, everyone uses the web site.

  - *Goal #2: A way to communicate not only test status but, current defect count, defect status, build status, test statistics by build, a quick health check of the product, and any special test information that may come up.*

    This goal was met by making sure the information communicated through the Web Based Testing Frame Work was what the team members wanted access to. Through reviews of the framework and its data, the team made the desition of what types of data should be communicated.

  - *Goal #3: Easy to use. The Testing Framework needed to be implemented in such a way that anyone on the project could access the information from any type of system (i.e. Unix or Windows).*

    The Web Based Testing Framework was written in html making it accessible from any system and any type of web browser. This made the project data available to anyone that needed it.

  - *Goal #4: There needed to be links between all the project documents.*

Each test case, reported defect, etc. had links all the way back to the original project requirements document.

- **Would I do it again**

  You bet!

  I believe that the Web Based Testing Framework had a large impact on the quality of the product. It gave every team member access to current and timely test and defect information. It gave team members one place to go to run test cases, and respond to problems found.

  The best indicator that this project was a success, was that every team member used the Web Based Testing Framework as part of their everyday work.

QW2002 Paper 9I1

Yiftach Resheff
(Antenna Software)

Assessing Risks In Wireless Solution Implementation

Key Points

- Introduction of a typical wireless solution
- Review of risks involved in implementation
- Provide set of tools to assess risks

Presentation Abstract

The presentation will review the various players in a typical enterprise wireless solution. It will then review the risks associated with the implementation and finally provide the attendants a set of tools that they can take with them and apply to their own process.

About the Author

Yiftach Resheff has more than 10 years in various aspects of software testing and quality assurance. Yiftach currently manages the QA team at Antenna Software where he is charge of the entire quality assurance life cycle from the requirements verification, to software validation and release schedule. Yiftach holds a BA in mathematics from Queens College and an MS in Operation Research from Columbia University in New York.

Assessing Risks in Wireless Solution Implementation

QW 2002 Conference

September 3-6, 2002
San Francisco, CA

Yiftach Resheff

Introduction

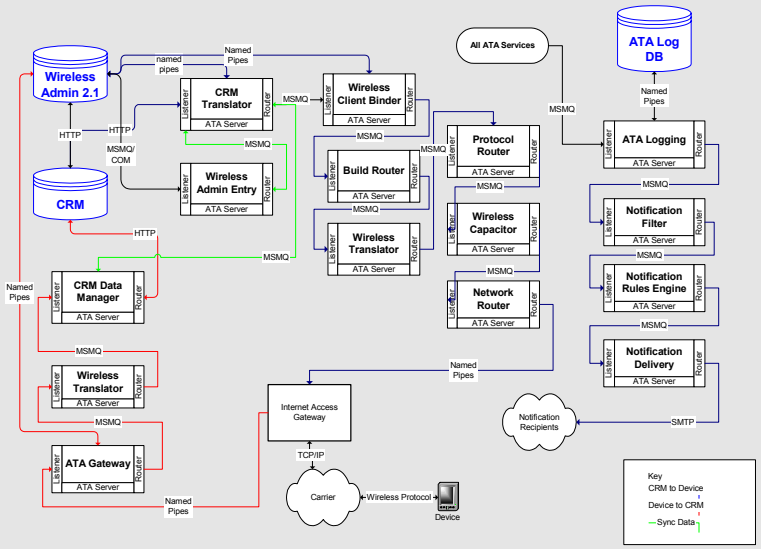
- Antenna Software develops web and wireless solutions that extend existing Client Relationship Management applications to the mobile workforce. Antenna's solutions run on an array of devices such as the Pocket PC family, Blackberry RIM and the Motorola PageWriter.
- Yiftach Resheff is the Quality Assurance Manager at Antenna Software, Inc. The Quality Assurance group at Antenna is responsible for all aspects of web and wireless testing, including verification of requirements, functional testing and performance testing.

Introduction

- The presentation has 3 main goals :
  - Overview of a typical wireless solution
  - Review of risks and challenges involved in implementation of wireless solutions
  - Tools and methods to assess risks in implementation of wireless solutions

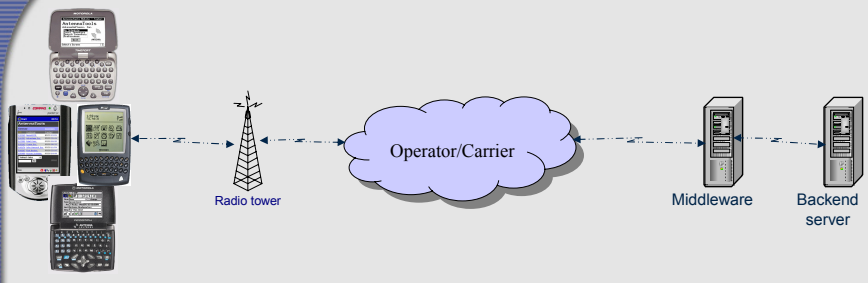
Overview of a Typical Wireless Solution

Overview of a Typical Wireless Solution



Overview of a Typical Wireless Solution

The bird's eye view



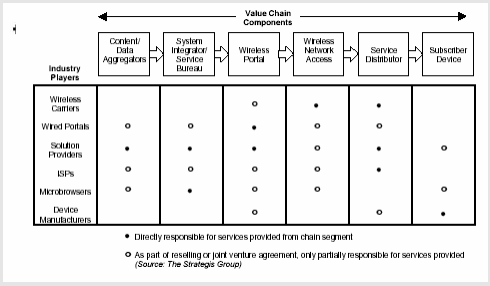
Mobile — Land Line

Device — Network — Server

Review of Risks and Challenges Involved in Implementation of Wireless Solutions

Review of Risks and Challenges

- All issues in traditional client/server and web testing still apply. Almost all wireless implementations include various web interfaces where users can view and modify their preferences and system administrators can control system configurations. Therefore issues such as browser compatibility, link checking and page response time are still valid and should be incorporated in the master test plan.
- Large number of players



- As an example the Compaq IPAC, running the POCKETPC operating system from Microsoft and using the Sierra Wireless network card can connect over AT&T Wireless CDPD network provided by GO America

Review of Risks and Challenges

Multiple networks, standards and protocols

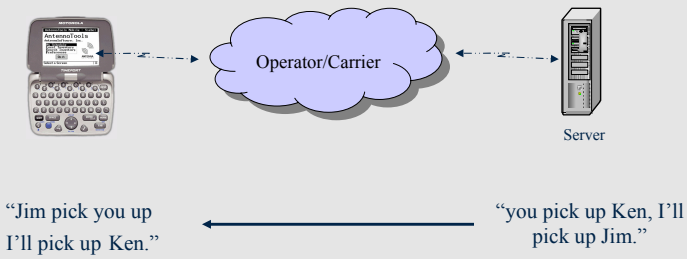
Network Technology	Lead US Carrier/Operator	Data Speed	Voice	Type	Main Usage
Paging	RIM, SkyTel	~8 kbs	No	Packet	One way, Two way, Alphanumeric, Light Data (RIM, Palm VII)
GSM	VoiceStream, Cingular	9.6/14.4 kbs	Yes	Circuit	Voice, SMS, Data (Phones)
DAMPS/TDMA	AT&T, Cingular	No	Yes	Circuit	Voice, SMS, Data (Phones)
CDMA	Sprint PCS, Verizon	14.4 kbs	Yes	Circuit	Voice, SMS, Data (Phones)
CDPD	AT&T W, Go America, OmniSky, Verizon	19.2 kbs	No	Packet	Data (Phones, Handhelds, Notebooks)
GPRS	VoiceStream, Cingular, AT&T W	28.8-128 kbs	Yes	Packet	Voice, SMS, Data, Media (3G Phones, Handhelds)
Metricom	Metricom	128 kbs	No	Packet	High Speed Data, Media (Handheld, Notebooks)
Bluetooth	NA	721 kbs	No	Packet	Data (Phones, Handhelds, Notebooks)
802.11b	NA	11,000 kbs	No	Packet	High Speed Data (Handhelds, Notebooks)

Review of Risks and Challenges

- Fault tolerance - Different networks resolve messages failure in different ways. Some will retry to send the message until it is delivered. Others will notify the sender of the failure and rely on the sender to re-send it.
- Delivery quality detection. Some network will notify the sender or the receiver when the quality of the message is low, others will not
- Latency - Predicting the latency of individual messages is very hard to achieve as it depends on a lot of different factors such as device location, weather conditions, NOC current load and signal strength.

Review of Risks and Challenges

- Sequencing- network does not always sends messages in the order in which they are received. If the application relies on the correct sequence for data integrity this issue can pose a serious risk



Review of Risks and Challenges

- Character string limitations - some networks will not deliver certain characters or character combinations. It is impossible to find out all possible combinations. Some characters might be used in the network logic for message parsing. Often the operators themselves do not have a comprehensive list of constraints.
- Message size limitation - some operators will limit the size of each individual message, either truncating or splitting it. The size usually includes spaces, non-printable characters, html tags, etc.

Review of Risks and Challenges

- Storage limitation- some operator will not keep an undelivered message for more than a set period of time.
- Volume limitation- Operators set limitations on the number of messages that can be sent simultaneously or over a set period of time. (e.g. Skytel has a limitation of 500 messages over 72 hours, Bellsouth over Mobitex has a limit of 25,000 characters a month per device)

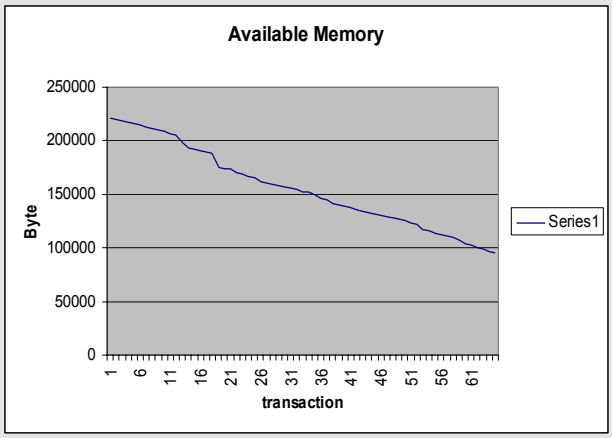
Review of Risks and Challenges

- Large number of devices- mobile phones, PDAs and pagers - using various operating systems- Symbian, Pocket PC, Palm OS, J2ME, Linux, Wisdom
- Limited Resources - battery life. The battery might not be able to support what is considered a typical work load of an end user. Also, messages might get lost if the battery 'dies' while messages are processed by the operating system or by the application.

Review of Risks and Challenges

- Limited Resources - memory. The combination of limited amount of memory and the relative immaturity of code written for wireless devices poses a serious risk.
- A memory leak is an error that occurs when usage increases the allocated memory without releasing that memory when the program is finished with it. The amount of allocated memory increases until it exceeds the physical memory; at this point, performance degrades dramatically and ultimately crashes the device.
- Some IDEs (Integrated Development Environments) provide a memory leaker tool. Some functions are memory leak prone (e.g. the PageWriter is known to leak through the consecutive beeps).

Review of Risks and Challenges



Review of Risks and Challenges

- Usability - Among the challenges in designing for wireless devices are:
  - Devices should be usable in your hand, not on a desk
  - Input is done via various means- stylus, mini keyboard, QWERTY keyboard
  - Navigation on small screen is difficult- easy to get lost . Site maps are hard to fit.
  - Since there is no standardization of functions and buttons it is hard to achieve a consistent 'look and feel' across devices

Tools and Methods to Assess Risks in Implementation of Wireless Solutions

Tools and Methods

Network Notification of successful or unsuccessful message delivery

Table of Contents

PIN Status

Page Recall

Page Recall Response for PIN: 1778954

Message number 1 of 39

From : X40:clardev2@antennasoftware.com
Submit date and time : 07/10/2002 16:23:43
Delivered date and time : 07/10/2002 16:24:28
Class : TEXT CLASS
Folder : FILED
Message :
clardev2@antennasoftware.com|[!A0][D494]b19g|1|1|

Message number 2 of 39

From : X40:clardev2@antennasoftware.com
Submit date and time : 07/10/2002 19:34:32
Delivered date and time : 07/10/2002 19:34:57
Class : TEXT CLASS
Folder : FILED
Message :
clardev2@antennasoftware.com|[!A0][D494]b1a1|1|1|

Message number 3 of 39

From : X40:clardev2@antennasoftware.com
Submit date and time : 07/10/2002 16:22:10
Delivered date and time : 07/10/2002 16:23:15
Class : TEXT CLASS
Folder : FILED
Message :

Tools and Methods

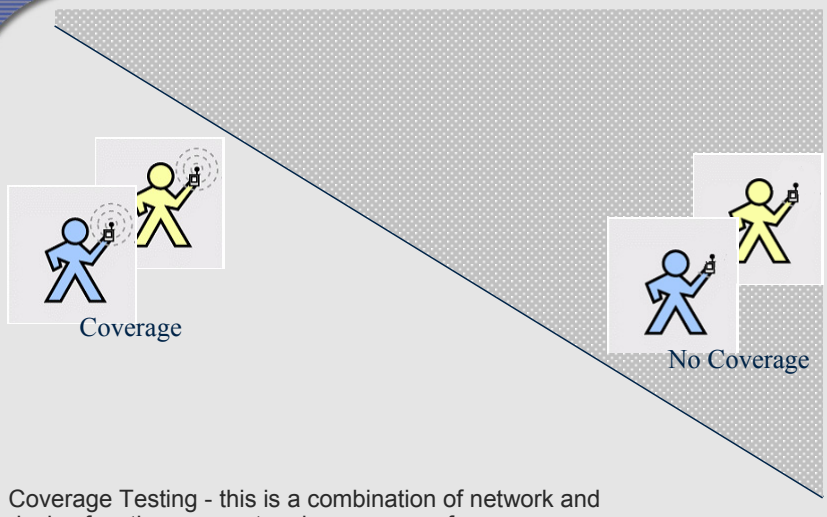
Network logs including time stamps and message body

Message Recall HELP?

Inbox for 1778954
check all uncheck all

Select	Message Text	Date & Time Submitted
	clardev2@antennasoftware.com|[!A0][D494]9kli|1|1| `AD `bc268435465`bw268549874`csgbitov`ct2002/07/12 18:15:00`bzY`cb Reassignment- Subase has been reassigned to gbitov| 	Fri 07/12/2002 06:18PM EDT
	clardev2@antennasoftware.com|[!A0][D494]9klg|1|1| `BA `aa268435465`bc268435465| 	Fri 07/12/2002 06:15PM EDT
	clardev2@antennasoftware.com|[!A0][D494]9klh|1|1| `AD `bc268435465`bw1`csjbuckley`ct2002/07/11 13:07:00`bzN`cbFix it,	Fri 07/12/2002 06:15PM EDT
	clardev2@antennasoftware.com|[!A0][D494]9klf|1|1| `AH`bc268435465`atDiego in debug - 07/11/2002 01:06PM`dj612-1`aq2002/07/11 13:07:00`ac268435461`acjbuckley`aa268436068`ag268440020`ahChris Jones`ai317-481-7292`afFederal Express`aj6648 Perimeter Road`amIndianapolis`anIN`ao46241`apUS`boDiego in debug -	Fri 07/12/2002 06:15PM EDT
	clardev2@antennasoftware.com|[!A0][D494]9kl7|1|1| `AD `bw268549869`csgbitov`ct2002/07/12 17:54:00`bzY`cb Reassignment- Subase has been reassigned to gbitov| 	Fri 07/12/2002 06:09PM EDT
	clardev2@antennasoftware.com|[!A0][D494]9kl6|1|1| `AD `bc268435465`bw1`csjbuckley`ct2002/07/11 13:07:00`bzN`cbFix it,	Fri 07/12/2002 06:05PM EDT
	clardev2@antennasoftware.com|[!A0][D494]9kl4|1|1| `AH`bc268435465`atDiego in debug - 07/11/2002 01:06PM`dj612-	Fri 07/12/2002 06:05PM EDT
	clardev2@antennasoftware.com|[!A0][D494]9kl5|1|1|	Fri 07/12/2002 06:05PM EDT

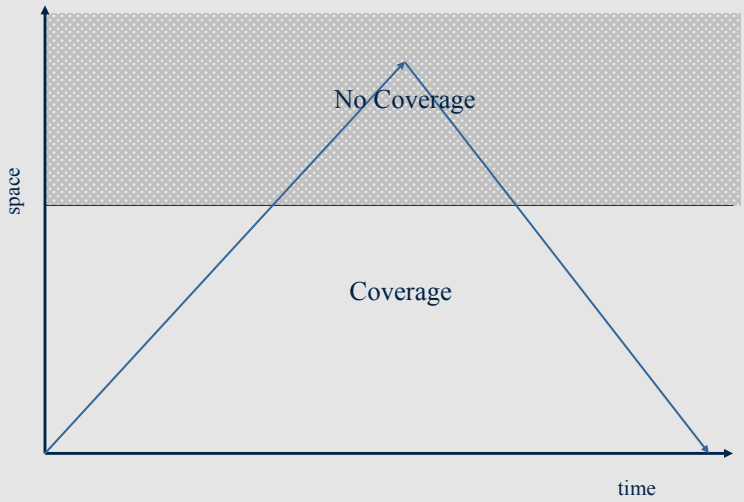
Tools and Methods



Coverage Testing - this is a combination of network and device functions. operators have a map of coverage. Behavior under poor coverage. Test at the edges- when device goes in and out of coverage.

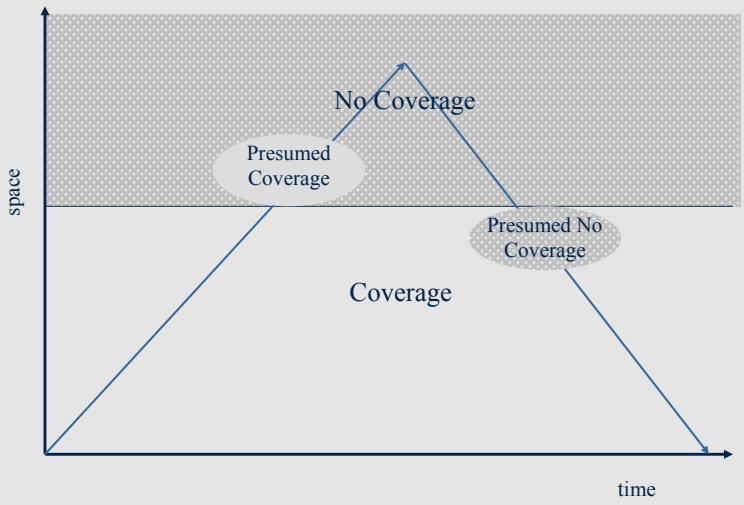
Page 21

Tools and Methods



Page 22

Tools and Methods

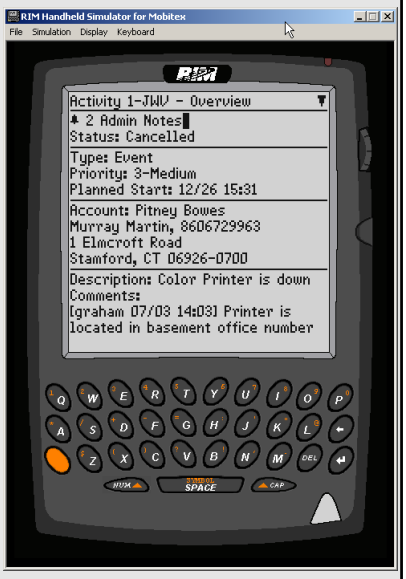


Page 23

Tools and Methods

- Simulator- The simulator is a window that appears when you execute your program in the IDE. It provides a "virtual device" that mimics the characteristics of the device for which you are developing your program. Clicking a button on the simulator keypad with your mouse generates an event almost exactly as it would if you had pushed the button on the real device.

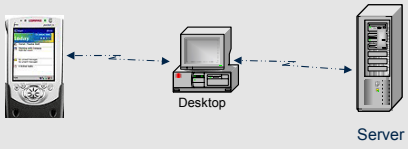
- It is strongly recommended that you deploy your application in the Simulator before downloading to your device. This could shorten the testing life cycle substantially. At the same time final testing should always be conducted on the real device as they do not always behave the same.



Page 24

Tools and Methods

- Sync Testing enables the device to communicate with a desktop computer using a cable, cradle, or infrared. You can also use your existing computer to connect to other resources through its internet connection.
- Sync testing helps isolate issues by taking the wireless aspect of messaging out of the equation.

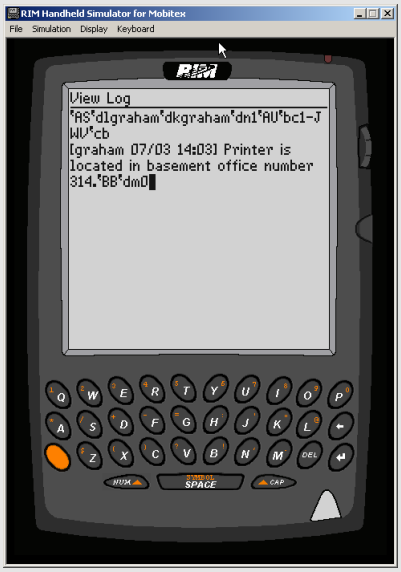


Tools and Methods

- Data loaders - Some wireless IDEs have a data loader with which you can feed data into the device without relying on wireless connection. You can use the data loader to test the application's behavior with extreme, incorrect and boundary values.

Tools and Methods

- Inbound and outbound queues and log are essential in order to make sure that messages arrive and get sent.

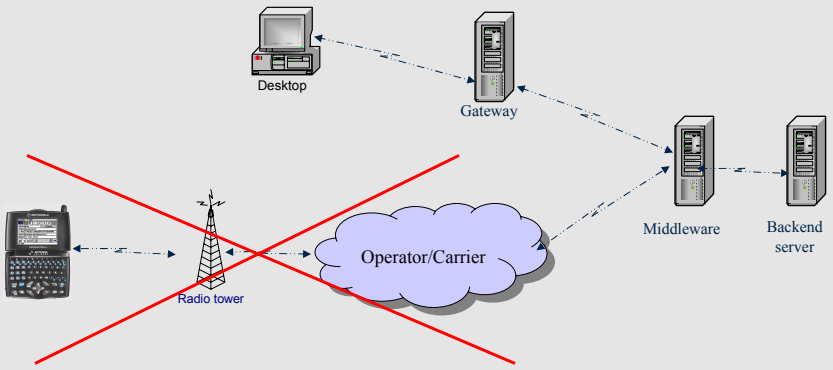


Tools and Methods

- Automated testing tools that run on the actual device are expensive. Most of them are OS specific and will not be compatible with all required devices.
- You can create a stress test scenario on the device by turning off the radio and then manually creating a large number of inbound and outbound messages. Turning the device back on will result in a burst of messages going in both directions.

Tools and Methods

- Client server automated tools can be used in order to simulate messages coming from the device to the backend server through a gateway server



Questions/Comments?

Contact: Yiftach Resheff
QA Manager
Antenna Software
Email: yresheff@antennasoftware.com

QW2002 Paper 9I2

Ibrahim K. El-Far, Roussi Roussev, and Nattawut Sridranup
(Florida Institute of Technology)

A Framework for Testing Wireless Applications

Key Points

- Key software testing challenges with WAP applications
- Model-based test automation for testing protocols and markup languages
- Setting up wireless test environments

Presentation Abstract

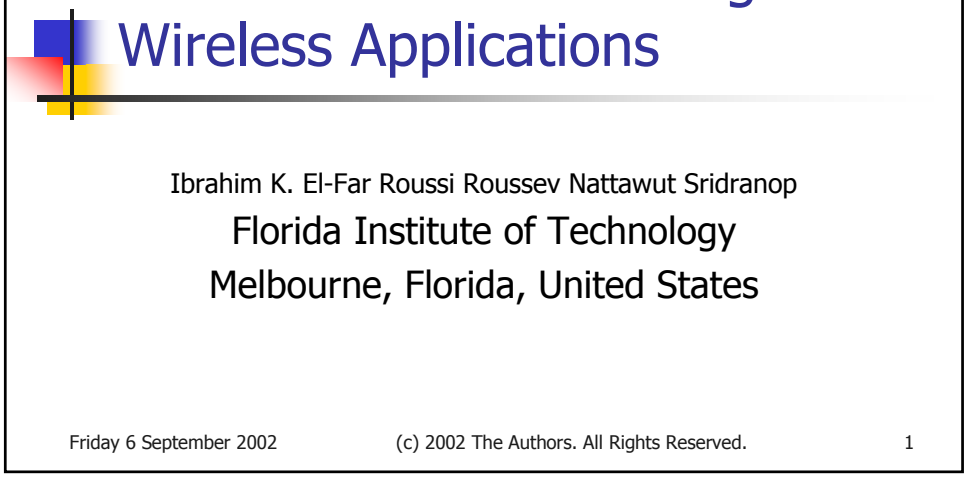
The Wireless Application Protocol or WAP is an open de facto industry wide standard for developing applications over wireless communications networks. WAP specifications present a variety of solutions analogous to those employed by the traditional Internet ex-cept they are more optimized for wireless applications. Examples include the Wireless Session Protocol for hypermedia transfer and the Wireless Markup Language which was designed to suit browsing on telephony and other mobile devices. With these new tech-nologies come testing problems and concerns, some of which we have attempted to ad-dress while testing a smart phone device. In this presentation, we give a detailed account of this experience. We show how to build a framework for testing client and server wire-less applications and how we employ models of protocols and markup languages to drive test automation. We discuss some of the difficulties in setting up the test environment and in generating and evaluating our tests.

About the Author

Ibrahim K. El-Far is a doctoral candidate in computer science under James A. Whittaker at the Florida Institute of Technology, Melbourne, Florida. He has nearly six years of ex-perience in model-based testing with state machines. Mr. El-Far is currently preparing a dissertation on testing software modeled with formal grammars and developing the G-Factory™ tool that performs much of the automated testing effort described in the pres-entation. His interests are in investigating software models, appropriateness of models to specific contexts, test automation and tools, and software testing education.

Roussi Roussev is a Master's student in computer sciences and he currently works for the Center for Software Engineering Research, Florida Institute of Technology, Melbourne, Florida. He has extensive experience in Windows programming and has worked with the Windows Kernel group at Microsoft in 2000 and 2001. He is interested in software test-ing methods, operating systems and distributed software.

Nattawut Sridranup is a doctoral student in computer science at Florida Institute of Tech-nology, Melbourne, Florida, and he develops contract test solutions at the Center for Software Engineering Research at the university. He is interested in applying formal methods in software testing.



A Framework for Testing Wireless Applications

Ibrahim K. El-Far Roussi Roussev Nattawut Sridranop
Florida Institute of Technology
Melbourne, Florida, United States

 (c) 2002 The Authors. All Rights Reserved.

Agenda

- A word about the project
- A quick background on WAP
- An introduction to model-driven testing
- A report on project activities and status
- A discussion of future work
- Summary and conclusions

 (c) 2002 The Authors. All Rights Reserved.

About the Project

- History – contract, initial problems
- Staff – abilities, skills, assignments
- Scope – extent of activities
- Requirements – deliverables, demonstrations
- Approach – planning, solutions, tools
- Status – achievements, remainder of work

(c) 2002 The Authors. All Rights Reserved.

Handheld Wireless Devices

- Low power requirements
- Low and variable bandwidth
- High latency
- Ephemeral network connectivity
- Weak processing power
- Typically small displays
- Limited user input

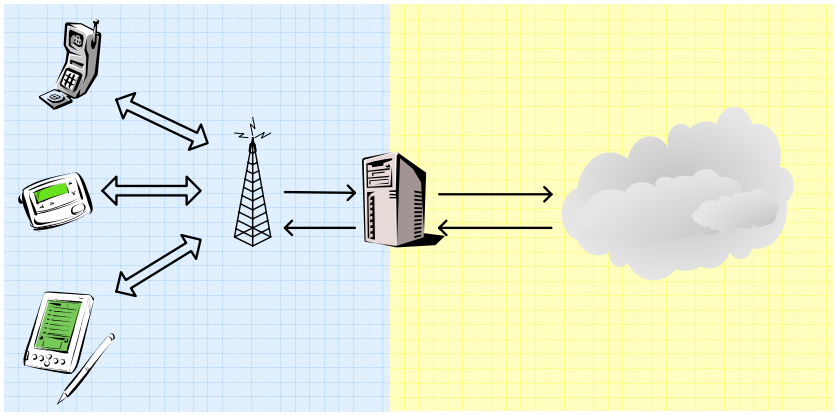
(c) 2002 The Authors. All Rights Reserved.

Wireless Application Protocol

- Consists of a collection of specifications of wireless solutions
- Solutions are claimed to address the aforementioned constraints of handheld wireless devices
- Proposed by the WAP Forum (now Open Mobile Alliance)
- Possesses backing of various major industry players

(c) 2002 The Authors. All Rights Reserved.

WAP: The Big Picture



(c) 2002 The Authors. All Rights Reserved.

Architecture: WAP vs. Internet

(c) 2002 The Authors. All Rights Reserved.

Wireless Session Protocol

- A binary hypermedia transfer protocol
- Provides both connection-oriented and connectionless modes of service
- Provides a Push mechanism

(c) 2002 The Authors. All Rights Reserved.

continued

Wireless Session Protocol

- Connection-oriented mode
  - Reliable transfer of packets
  - Transient network connectivity is addressed
- Connectionless mode
  - Unreliable transfer of packets
  - Stateless
- Provides a Push mechanism
  - Initiated by server
  - Intended for software updates and provisioning

(c) 2002 The Authors. All Rights Reserved.

Wireless Markup Language

- A markup language that extends XHTML
- Provided for backward compatibility
- Earlier versions were extensions of XML
- We were asked to consider WML 1
- WAP 2 requires XHTML Mobile Profile
- Addresses the peculiar needs of narrowband wireless devices

(c) 2002 The Authors. All Rights Reserved.

WML 1.3

- Information is organized into decks and cards
- A card specifies units of interaction
- Cards are grouped into decks
- A deck is analogous to an HTML page

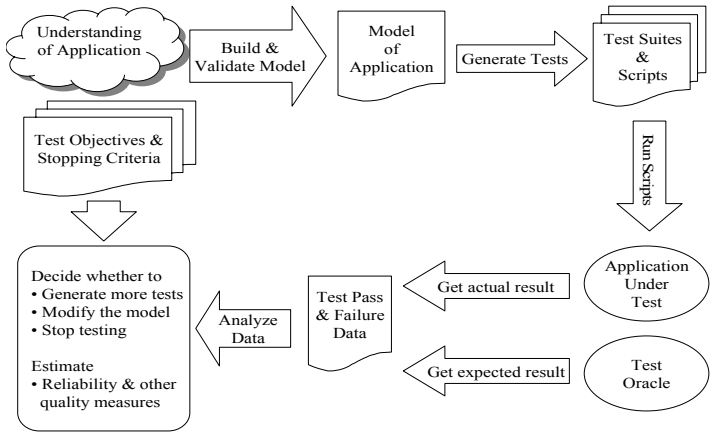
(c) 2002 The Authors. All Rights Reserved.

WML 1 Sample

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML
1.1//EN""http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
   <card id="card1" title="WML test page">
      <p>Content</p>
   </card>
</wml>
```


(c) 2002 The Authors. All Rights Reserved.

Model-Driven Testing



(c) 2002 The Authors. All Rights Reserved.

Grammars

- A grammar is a formal system of an alphabet and a number of rules
- The rules describe how to form words and or sentences based on the alphabet
- A grammar defines a language or the collection of all words and or sentences that can be formed using its rules
- ISO 14977 standard defines a notation that can be used to express grammars

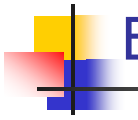
(c) 2002 The Authors. All Rights Reserved.

Grammars: Example

```
Sentence = Noun Phrase, Verb Phrase;
Noun Phrase = Proper Noun | Determiner, Common Noun;
Proper Noun = “John” | “Jill”;
Common Noun = “car” | “hamburger”
Determiner = “a” | “the”
Verb Phrase = Verb, Adverb | Verb;
Verb = “drives” | “eats”;
Adverb = “slowly” | frequently”;
```

(Example taken almost verbatim from Thomas Sudkamp’s “Languages and Machines: An Introduction to the Theory of Computer Science,” published by Addison-Wesley, 1997)

(c) 2002 The Authors. All Rights Reserved.



Example Generated Sentences

John eats slowly
The car drives frequently
Jill drives slowly
.
.
.
.

(c) 2002 The Authors. All Rights Reserved.

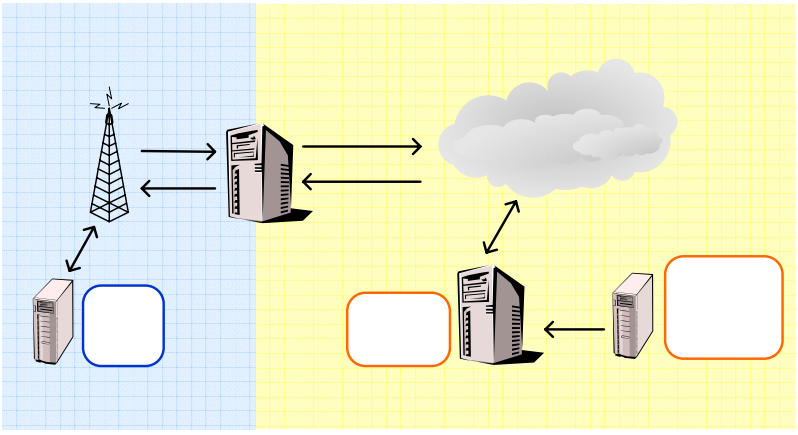
Project Deliverables

- Test design
- Test automation framework
- Test suites used
- Problem reports

(c) 2002 The Authors. All Rights Reserved.

Scenario 1

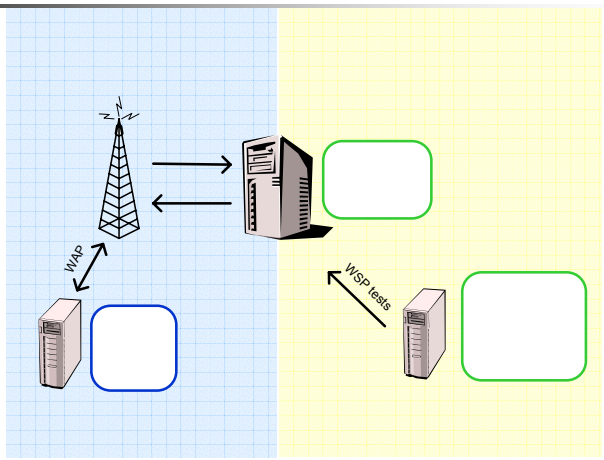
WML Test Environment Set Up



(c) 2002 The Authors. All Rights Reserved.

Scenario 2

WSP Test Environment Set Up



(c) 2002 The Authors. All Rights Reserved.

Modeling the WSP Get

- Fields in every PDU
  - TID (associates requests with replies)
  - Type (identifies PDU type; 0x40 for Get)
- Fields of the Get PDU
  - URILen (length of the URI Field)
  - URI (Universal Resource Identifier)
  - Headers (headers associated with request)

(c) 2002 The Authors. All Rights Reserved.

continued

Modeling the WSP Get

```
General PDU = TID, Specific Type PDU;
Specific Type PDU = Get PDU | … other PDUs …;

Get PDU = Get Type, URILen, URI, Headers;
Get Type = “0x40”;
URILen = uintvar; (* unsigned variable-length integer*)
URI = (* list of test URI’s *)
Headers = (* Extracted from grammar in specifications *)
```


(c) 2002 The Authors. All Rights Reserved.

Modeling WML 1.3

Verbatim from DTD:

```
<!-- flow covers "card-level" elements, such as text
  and images -->
<!ENTITY % flow "%text; | %layout; | img | anchor |a
  |table">

<!-- Task types -->
<!ENTITY % task "go | prev | noop | refresh">
```


(c) 2002 The Authors. All Rights Reserved.

continued

Modeling WML 1.3

Equivalent rules in grammar:

```
(* ENTITY flow *)
flow = text | layout | img | anchor | a | table;

(* ENTITY task *)
task = go | prev | noop | refresh;
```


(c) 2002 The Authors. All Rights Reserved.

continued

Modeling WML 1.3

Verbatim from DTD:

```
<!ELEMENT wml ( head?, template?, card+ )>
```

Rule in grammar:

```
wml element = "<wml", wmlAttr, ">", [head],
  [template], card, {card}, "</wml>";
```


(c) 2002 The Authors. All Rights Reserved.

Demo

Or What Kind of Tool is Needed to Model Applications and Generate Tests

(c) 2002 The Authors. All Rights Reserved.

Future Work

- Model other components of WAP
- Improve collecting & evaluating results
- Automate model building for certain types of systems
- Develop better techniques for generating interesting tests
- Enhance existing tools to help users better visualize models

(c) 2002 The Authors. All Rights Reserved.

Questions and Comments

(c) 2002 The Authors. All Rights Reserved.

Contact Information

You can reach Ibrahim K. El-Far by
Email: ielfar@acm.org
Web: http://www.testingresearch.com/
You can reach Roussi Roussev by
Email: rroussev@se.fit.edu
You can reach Nattawut Sridranop by
Email: bird@se.fit.edu

(c) 2002 The Authors. All Rights Reserved.

QW2002 Paper 2P1

Mr. Vijay Sikka & Mr. Anurag Khemka
(Nirixa, Inc.)

Outsourcing in QA and Testing

Key Points

- Role of outsourced services in QA/QC
- Cost savings and leverage gained by outsourced testing services
- Issues and experiences of using outsourced services

Presentation Abstract

Recent downward market trends, waning revenues and emphasis on the bottom line has prompted a resurgence in the trend of outsourcing. Nearly half of Fortune 500 companies are using outsourced services, whether they are business process management, customer service, software development, transcription or QA and testing. All the top 20 ISVs now have software development and QA labs in countries such as India, Ireland, China, Israel amongst several others. Questions range from whether outsourcing QA and testing been a productive experience for companies to whether the promise of cost savings was fulfilled.

About the Author

Vijay Sikka is the co-founder and principal of Nirixa, Inc. a company providing comprehensive QA and testing outsourced services. Vijay's more than 15 year executive career and strengths span business development, engineering, and operations. Vijay has done marketing at IBM in Madison Avenue, New York and engineering management for 7 ½ years at Intel Corporation headquarters in Santa Clara. In 1996, Vijay founded IBrain Software, Inc. and served as its CEO until its acquisition by Entigen Corp in 1998. In 1999, Vijay started Bodha.com, Inc. and served on its board of directors until its acquisition by Peregrine Systems (NASDAQ: PRGN) in 2002. Vijay has built international teams and engineering organizations with successful multi - year operations. Vijay is an advisor and a consultant to several bay area companies.

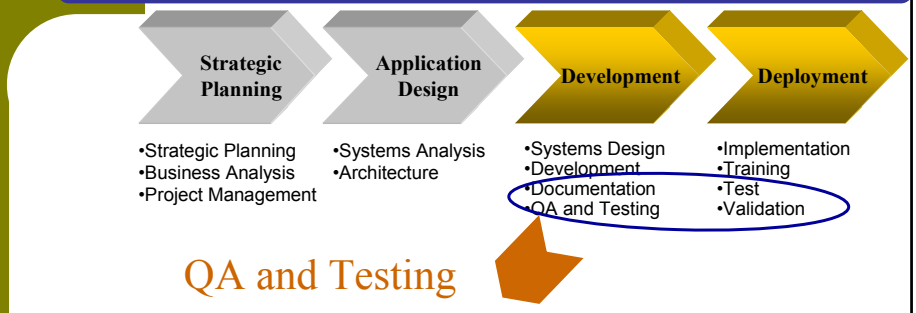
Anurag Khemka is the co-founder and principal of Nirixa, Inc. a company providing comprehensive QA and testing outsourced services. Anurag has more than fifteen years of experience in enterprise software development, product architecture, and executive management. Anurag is the founder of MarketFirst Software, a pioneer and leader in Enterprise Marketing Automation, and currently serves there as Vice President and Chief Technology Officer. Prior to founding MarketFirst in 1996, Anurag was the director of research and development at Cambio Networks, and previous to that held positions as the chief product architect and R&D manager for UB Network's enterprise-wide, client/server based network management products.

Software QA and Testing Services

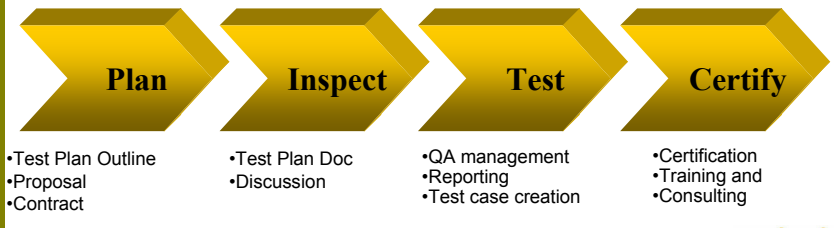
Vijay Sikka

Nirixa, Inc.

QA and Testing is a Key Component of Software Development

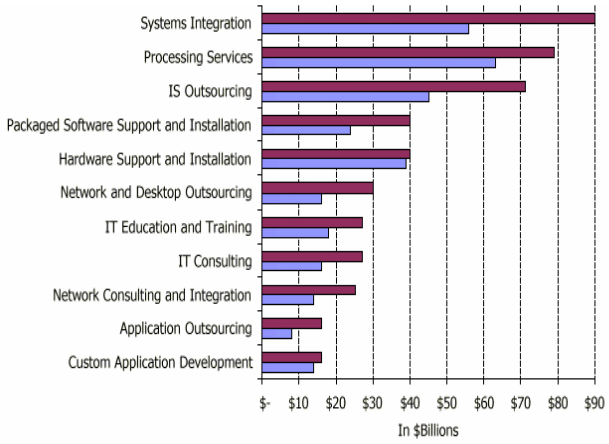


QA and Testing



Copyright © 2001-02 Nirixa, Inc.

Global IT Services Marketplace



Total Market Size

- 1999: $315 Billions
- 2003: $465 Billions
- CAGR: 10%

Source: IDC

Copyright © 2001-02 Nirixa, Inc.

nirixa

Global IT Services Marketplace

- **According to International Data Corporation**
  - Total Market size: $315B(1999) ⇒ $465B (2003)
  - Processing Services $79B (17%)
  - IS Outsourcing $72B (15%)
  - QA Testing Services is part of above two categories
- **According to Sanford Bernstein**
  - Total Market size: ~500B in 2003
  - Software QA Testing and Business Processing Services forms 15% ($75B) of it
- **QA Testing Services & Outsourcing is significant Market**

Copyright © 2001-02 Nirixa, Inc.

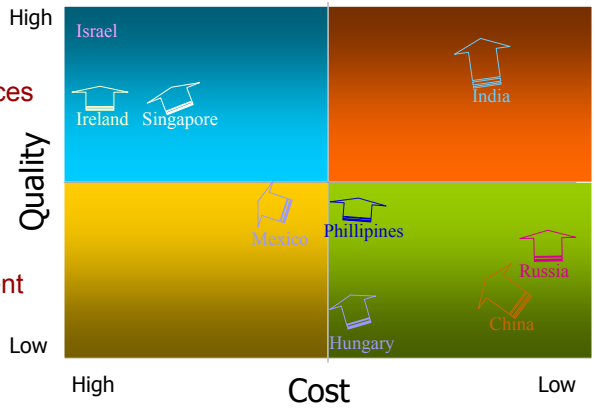
nirixa

Major Outsourcing Destinations

Critical Capabilities

- High Quality
- Lower costs
- Depth of IT resources

- Time to Market
- Specialized Skills
- Project Management



Source McKinsey & Co. 2001 and Nirixa authors research
Quality depends on resource availability, capability maturity model (Software Engineering Institute) and cultural fit.
Labor costs include taxes, markups, risks, and insurance
Trend arrows are based on Nirixa research.

Copyright © 2001-02 Nirixa, Inc.

Typical Services offered by a QA Outsourcing Company

- Test Plan Generation
- Functional and Stress QA testing
- Validation and Quality testing of product documentation
- QA management and reporting
- Process improvement, training and consulting
- Management and access to bug databases

Copyright © 2001-02 Nirixa, Inc.

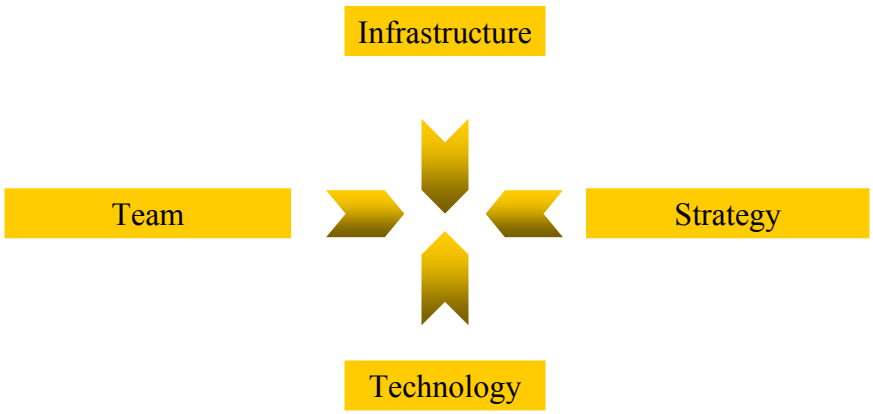
How to Evaluate Outsourcing Destination Country

- Large Software Industry
- Engineering Resource Pool
- English Speaking Skills
- Hard Working Culture
- Low Cost of Development
- High Quality
- SEI CMM and ISO Certifications
- Good Communication and Power Infrastructure

Copyright © 2001-02 Nirixa, Inc.

nirixa

How do you Evaluate a Good QA Outsourcing Company



Copyright © 2001-02 Nirixa, Inc.

nirixa

Evaluation: Team

- Key executives in US for effective management
- Experienced management team in outsourced country
  - US educated or trained for workstyle match
  - Familiarity with local government and cultural issues
- World class QA and Test engineering team
  - Computer Science graduates with diverse experience
  - Specially trained in QA methodology and techniques
  - Blend of software development experience
  - Low price point – high value

Copyright © 2001-02 Nirixa, Inc.

nirixa

Evaluation: Infrastructure

- Established development and training center
- Web based round the clock operations
- Established QA lab
- Emphasis on customer privacy and security
- Remote workgroup productivity infrastructure
  - Multiple ISPs
  - PC Video/Audio conferencing
  - VPN
  - Source control, bug tracking with remote access
- Network of specialists, educational institutes, and partners

Copyright © 2001-02 Nirixa, Inc.

nirixa

Evaluation:Strategy

- Strategic focus on QA
- Maintain low cost and Flexibility of scale
- Staff incentives for customer satisfaction
- Established Partnerships
- Target specific industries

Copyright © 2001-02 Nirixa, Inc.

nirixa

Evaluation :Technology

- Faster results through building IP
  - QA methodology
  - Best practices
  - Standards Documents
  - Test components
- Continuous emphasis on leading edge technologies
  - Multiple OS, Test tools (scripting), languages
- Value add offerings
  - Training
  - Education
  - Certification
  - Quality consulting

Copyright © 2001-02 Nirixa, Inc.

nirixa

The most asked Question

- What is the ideal ratio of Developers to Testers?
  - Microsoft [1:1] {Cusamano, Michael A. and Richard W. Selby, Microsoft Secrets, Simon and Schuster, Inc., New York, 1995.}
  - Ecommerce projects [4:1] {SQATester.com}

✓We recommend:

Well defined requirements, APIs, peer reviews & unit testing early on with a ratio of [5:2 or better] {Software Program Manager's Network, *Best Practices*, 1999, on www.spmn.com; McConnell, Steve. Rapid Development, Microsoft Press, Redmond WA, 1996.}

Copyright © 2001-02 Nirixa, Inc.

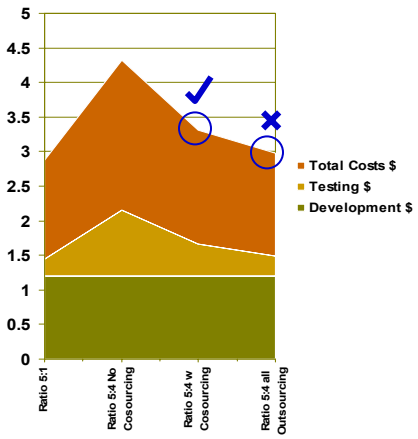
nirixa

Co-sourced QA and Test Model

Cost advantages comparison

- Monthly cost per developer and tester = $10,000.00. One year project duration
- Average team size = 10 developers and variable testers
- Assumes good software engineering, well defined APIs, and well defined requirements

✓ **Co-sourced QA and Test offers cost advantages and has a higher likelihood of success than all outsourced**



Copyright © 2001-02 Nirixa, Inc.

nirixa

Cost savings using a co-sourced model

	Ratio 5:1	Ratio 5:4 No Co sourcing	Ratio 5:4 w Co sourcing	Ratio 5:4 all Outsourcing
Development mm$	1.2	1.2	1.2	1.2
Testing mm$	0.24	0.96	0.456	0.288
Total Costs mm$	1.44	2.16	1.656	1.488

Copyright © 2001-02 Nirixa, Inc.

nirixa

Summary

- Outsourced QA and Testing is here to stay
- Check references from existing customers
- Look for focus on QA and Testing
- Look for English language skills
- High quality --- best value
- Use Co-sourced instead of Outsourced model

Copyright © 2001-02 Nirixa, Inc.

nirixa

Outsourcing in QA and Testing

Authors: Vijay Sikka and Anurag Khemka

Table of Contents

OUTSOURCING IN QA AND TESTING

ABSTRACT

ABOUT THE AUTHORS

VIJAY SIKKA – PRINCIPAL, NIRIXA, VSIKKA@NIRIXA.COM
ANURAG KHEMKA – PRINCIPAL, NIRIXA, ANURAGK@NIRIXA.COM

MARKET SIZE AND GLOBAL OUTSOURCING TRENDS

MOST FREQUENTLY ASKED QUESTIONS

A. HAS OUTSOURCING QA AND TESTING BEEN A PRODUCTIVE EXPERIENCE?
B. DID YOU SAVE COSTS WHILE MAINTAINING THE TIMELINESS OF YOUR PRODUCT DELIVERABLES?
C. WHICH COMPANIES YOU FOUND DID A GOOD JOB FOR YOU?
D. WHICH COUNTRY DID YOU OUTSOURCE TO? IRELAND, CANADA, ISRAEL, INDIA, CHINA OR OTHER?
E. WAS THE TIME DIFFERENCE A HURDLE FOR TEAM COMMUNICATION?
F. WAS LANGUAGE AN ISSUE IN COMMUNICATION?
G. WHAT TYPE OF TESTING DID YOU SUCCESSFULLY OUTSOURCE? BLACKBOX? UNIT? FUNCTIONAL? LOAD? REGRESSION?
H. DID YOU COMPLETELY REPLACE YOUR QA DEPARTMENT OR DID YOU USE OUTSOURCING AS AN EXTENSION?
I. DID YOU REQUIRE THE QA AND TESTING PEOPLE TO HAVE COMPUTER SCIENCE OR SOFTWARE BACKGROUND?
J. WHICH STANDARDS BODIES DID YOU FIND MOST RELEVANT TO YOUR QA AND TESTING? ISO? ASQC?

THE "CO-SOURCED" QA AND TESTING MODEL

WHAT IS THE IDEAL RATIO OF DEVELOPERS TO TESTERS?
COST SAVINGS USING A CO-SOURCED MODEL WHILE MAINTAINING SUITABLE DEVELOPER TO TESTER RATIO

SUMMARY

303 Almaden Boulevard, 6th Floor
San Jose, CA – 95110
408.998.7844 (Phone)
408.998.7845 (Fax)

www.nirixa.com
info@nirixa.com

Abstract

Recent downward market trends, waning revenues and emphasis on the bottom line have prompted resurgence in the trend of outsourcing. Nearly half of Fortune 500 companies are using outsourced services, whether they are business process management, customer service, software development, transcription or QA and testing. All the top 20 ISVs now have software development and QA labs in countries such as India, Ireland, China, and Israel amongst several others. According to IDC, global information systems outsourcing market is expected to pass $72B by 2003.

Today's more complex software systems pose increasingly difficult challenge for software vendors when it comes to QA, testing, and certification of these software systems. Parallel processing, distributed multi-tiered components, multitude of operating systems, databases, Internet platforms etc., make it very difficult for a software vendor to build in-house infrastructure to QA and timely release their products with quality. Outsourced services, where hardware and software infrastructure as well as expert resources are available with ability to timely scale based on the need, provide solution to this challenge. As companies across the board are beginning to consider outsourcing essential services, the information technology professional is faced with a daunting set of questions about outsourcing. This paper discusses some of these questions and experiences gathered by authors and other IT professionals, customers, and providers during last few years of outsourced services growth.

Questions range from whether outsourcing QA and testing been a productive experience for companies to whether the promise of cost savings was fulfilled. Specific experiences of dealing with communication, time zone and cultural differences are discussed. Background requirements of people who work on QA and testing as well as the types of testing outsourcing lends itself best to will be identified. Finally, the authors discuss the models, processes and QA standards that make outsourcing experience the most effective for business executives and information systems professionals.

About the Authors

Vijay Sikka – Principal, Nirixa, vsikka@nirixa.com

Vijay's more than 15 year executive career and strengths span business development, engineering and operations. Vijay has done marketing at IBM in Madison Avenue, New York and engineering management for 7 ½ years at Intel Corporation headquarters in Santa Clara. In 1996, Vijay founded IBrain Software, Inc. and served as its CEO until its acquisition by Entigen Corp in 1998. In 1999, Vijay started Bodha.com, Inc. and served on its board of directors until its acquisition by Peregrine Systems (NASDAQ: PRGN) in 2002. Vijay has built international teams and engineering organizations with successful multi - year operations. Vijay is an advisor and a consultant to several bay area companies. Vijay holds a Masters degree in Artificial Intelligence from Syracuse University New York, and dropped out of Stanford Ph. D program to start his first company.

Anurag Khemka – Principal, Nirixa, anuragk@nirixa.com

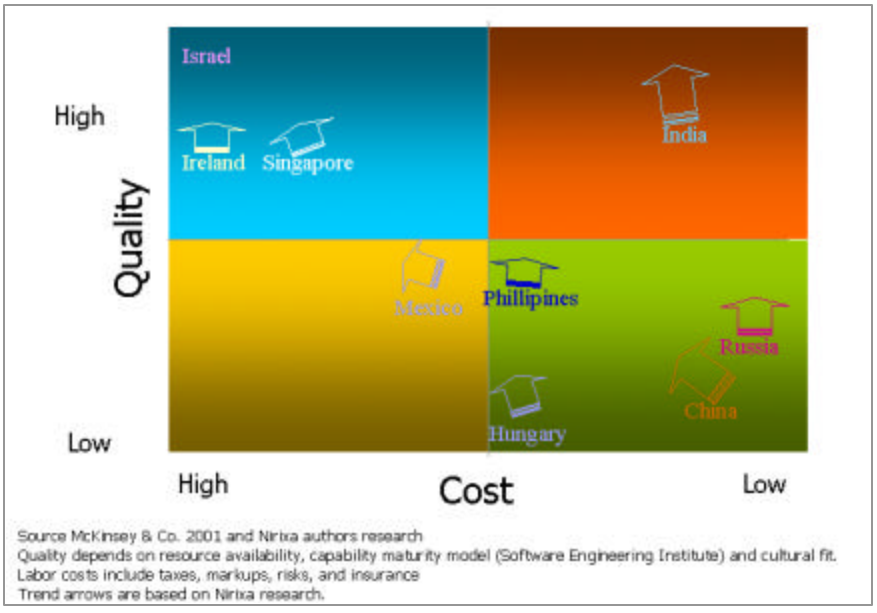
Anurag Khemka has more than fifteen years of experience in enterprise software development, product architecture, and executive management. Anurag is the founder of MarketFirst Software, a pioneer and leader in Enterprise

Marketing Automation. Anurag founded MarketFirst in 1996 and managed all aspects of the company for the first two years as President and CEO. After growing the company to about 40 people, Anurag attracted senior executives (from companies such as Oracle, Netscape, Informix etc.) to manage the company growth while he focused on the product and vision as CTO. Prior to founding MarketFirst, Anurag was the director of research and development at Cambio Networks, where he developed enterprise management software. Before Cambio Networks, he served as the chief product architect and R&D manager for UB Network's enterprise-wide, client/server based network management products. Anurag holds a B.S. in Electrical Engineering from the Indian Institute of Technology, Delhi, India and an M.S. in Computer Science from the University of Louisiana, Lafayette.

Market and global outsourcing trends

Software IT Services is a big market, all analysts agree that the size of the total IT Services market in US will be close to $500B in 2003. According to IDC QA Testing services is part of Business Processing and IS outsourcing which is a combined $150B market[1]. According to Sanford Bernstein, in 2003 Software QA Testing and Business Processing Services will be $75B or 15% of the total IT Services Market. Gartner Dataquest have also forecasted that the market of global business process outsourcing (BPO), will grow to $543 billion in 2004, at a compound annual growth rate of 21 percent. Though no analyst specifically calls out QA outsourcing as a separate category, even a 5% share of the corresponding major category gives it a significant market size.

Gartner recently forecasted that by December'02 more than 80 percent of multinationals will use IT outsourcing to save money, overcome skills shortages or increase flexibility. Gartner's research indicates that 200 of the Fortune 500 companies used offshore application outsourcing[2] in the year 2000. Meta group, in 2002, recommends "Global 2000 organizations should rigorously evaluate offshore outsourcing possibilities for IT and related services and projects for possible inclusion in their overall portfolio of sourcing options". During the economic downturn of the past 12 months, efforts by Global 2000 organizations to reduce IT costs have added a counter-recessionary impetus to the long-term trend toward increased use of offshore outsourcing service providers.



Source McKinsey & Co. 2001 and Nirixa authors research
Quality depends on resource availability, capability maturity model (Software Engineering Institute) and cultural fit.
Labor costs include taxes, markups, risks, and insurance
Trend arrows are based on Nirixa research.

A McKinsey report, on topmost software outsourcer countries, places India all alone in the quadrant that boasts high quality and the lowest costs, making it a very attractive high quality outsourcing destination[3]. Nirixa research indicated that countries including

[1] According to IDC Processing Services will be $79B and IS Outsourcing Services will be $72B in 2003
[2] Offshore outsourcing is mainstream for Fortune 500 Enterprises, Author R. Terdiman. Gartner Research September, 7 2001
[3] Programmers abroad: A primer in offshore software; Author Inigo Amoribieta et al. McKinsey Quarterly 2001 Number 2

China, Russia and others through extensive government incentives and infrastructure development are rapidly improving the quality of software outsourced.

We believe that the QA and Testing are very suitable for leveraged outsourced offshore services as it provides easily measurable benefits, and quantifiable ROI. Most IT and development organizations find it difficult to have adequate QA resources and lab infrastructure, and end up shipping substandard products under time to market pressures. In the current outsourced model followed by many companies including Nirixa, not only these companies can easily get highly trained untainted QA resources, but can also rent large QA lab infrastructure for expanded quality testing at a fraction of the cost of doing themselves.

We talked to several CTOs, Directors of QA and VP Engineering colleagues and customers regarding their experiences with outsourced QA and testing. Our questions and the general answers we got from them regarding their experiences are listed below.

Most frequently asked questions

a. Has outsourcing QA and Testing been a productive experience?

Most of the colleagues believed that pure outsourcing models didn't work as intended. They had no proximity with the QA team and closed loop interaction. They felt that outsourced QA work resulted in the following common feedback:

- Outsourcer didn't understand our product
- Outsourcer didn't find a lot of bugs they should have found
- Outsourcer didn't understand what they were supposed to do
- We didn't get good reporting

Several of them had more pleasant experiences if they had internal QA managers interacting and controlling the outsourcing team.

b. Did you save costs while maintaining the timeliness of your product deliverables?

Responses on this ranged from some very successful and well-managed projects and product deliverables resulting in cost savings to unhappy campers who shipped products that didn't work and spent more than planned. Another key aspect that emerged was that cost should never be the only consideration when outsourcing QA. Cost of not doing QA or doing poor QA would be increased exponentially if a product were released with bugs that prevent users from using it.

c. Which companies you found did a good job for you?

Most of those surveyed preferred to keep the names of the companies they worked with anonymous. All of the colleagues believed that asking the companies for current customer references and checking them for satisfaction worked well. Several outsourced QA and test companies maintain a satisfaction percentage metric of their customers through established interviews and questionnaires. It is a good idea to ask the outsourcing provider for a report on this metric.

d. Which country did you outsource to? Ireland, Canada, Israel, India, China or other?

Most respondents had worked with Indian outsourcing vendors. Most cited proficiency in English and quality of work from Indian companies as better than other outsourcing countries. Some cited experience working with Indian companies that maintained strong management presence in America as well as QA labs in India as better suited to their working style. A few didn't enjoy working with outsourcers from any other country.

e. Was the time difference a hurdle for team communication?

All agreed that the time difference in team communication could become a hurdle in productivity if the outsourcing company with a good methodology and process didn't address it. It was observed also that the addressing time difference issue with good process led to better handling of tighter build/regression schedules as release dates came closer. Most of the colleagues observed the following cycle to be very effective in proactively handling the time difference to an advantage.

1. Developers deposited latest build and release in the source control at night before leaving for the day.
2. The QA and test experts in the outsourcing company would run the test cases and generate reports on the release and upload the bug tracking system for the developers
3. Developers would review the bugs and test result reports and work on fixing the bugs.

This cycle reportedly worked very effectively with outsourcing QA and test companies in India that were 12.5 hours ahead in time.

Some outsourcing companies reportedly worked in multiple shifts to accommodate overlaps with the developers in USA for conferencing and one on one communication.

f. Was language an issue in communication?

People reported issues with accented English from outsourcing companies and the fact that Americans are not trained to talk at ESL (English as Second Language) speeds. Both of these issues were addressed by companies that used outsourcing through Internet Relay Chat and teleconferencing that involved people from the outsourcing companies who were in United States. It helped to work with outsourcing companies, which maintained offices and key executives and management in United States.

g. What type of testing did you successfully outsource? Blackbox? Unit? Functional? Load? Regression?

Most of the companies that have used outsourcing reported trying functional, unit, white-box and black box testing.

h. Did you completely replace your QA department or did you use outsourcing as an extension?

All of the companies that have worked with outsourcing QA and test providers reported working with them as extensions of their existing QA departments. That was the model that worked best with them. Some companies who once had QA departments reportedly replaced them with an outsourced company. Often the outsourcing company in the beginning acted as overflow for QA departments who were overloaded. Sometimes outsourcing QA and test experts were brought in where the customers never had QA and used the outsourcing company to help them transition into having their own QA team.

i. Did you require the QA and testing people to have computer science or software background?

Most companies and colleagues we talked to wanted QA and testing people with computer science and software background. However, they sometimes lamented that QA outsourcing companies gave them people with limited or no HTML, JavaScript, DHTML, C++/Java backgrounds. Companies also warned against accepting at the face value claims by the outsourcing company that they had "Unix experts" or "Windows experts". Most recommended finding out if the QA and test people had certifications by Sun or Microsoft or some other US based training and certification organization.

An important factor in each successful outsourcing partnership was having a QA manager who is already familiar with the product and company-testing practices should be leading the offshore team -- onsite at the remote location.

j. Which standards bodies did you find most relevant to your QA and testing? ISO? ASQC?

All companies and colleagues reported that standards bodies had no relevancy. They didn't recommend ISO 9000 or any standards certifications of the outsourcing companies.

The "co-sourced" QA and testing model

All of us agreed that the model that tends to work best is a "co-sourced" model in which an outsourcing QA team is not completely replacing the in-house QA team. Instead, the outsourced QA team is working as an extension of the in-house team with a key QA manager on the in-house team maintaining full knowledge and control of the QA tasks and project plan. Also, another model that has been reported to work is one of a "expert-contributor" model. In the mentor-student model, the in-house team maintains a key *expert* and the outsourced QA team provides one or more *contributors*. This model provides a "round the clock operation" where contributors work as extensions of the expert with his/her guidance and complete most of the work in an offshore location. The keyword in successful co-sourcing model is "control". Whether all the QA and testing resources are in house, outsourced or co-sourced using the definition we have provided above, a frequently asked question in the minds of IT executive in a company is the following.

What is the ideal ratio of Developers to Testers?

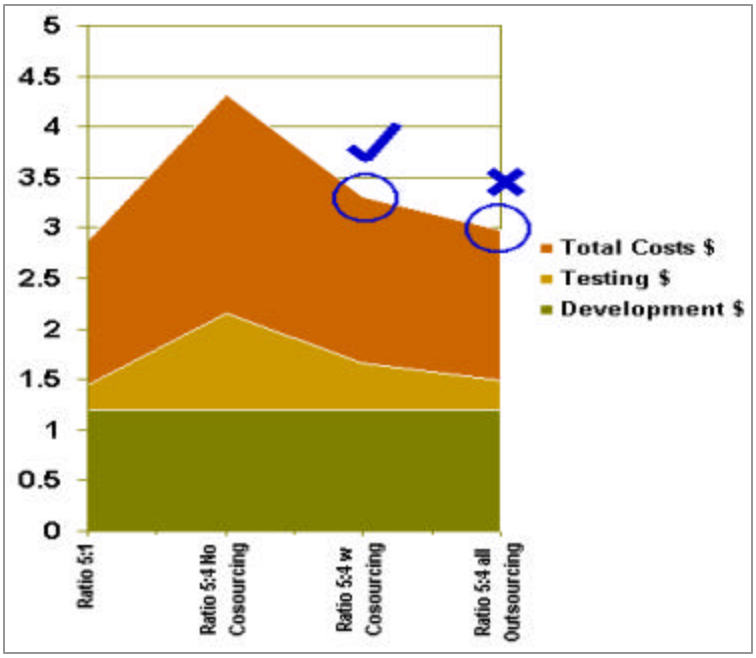
Let's start to answer this question by first pointing out Microsoft. It is known that this ratio is 1 to 1 in Microsoft.[4] Others have reported a more extreme 4 to 1 ratio for Ecommerce projects[5] and a dependence on several factors including software engineering [6], [7],[8].

Based on our experience, authors recommend a ratio of 5 to 2 or better as more appropriate for usage intensive GUI oriented enterprise software applications. Testing of wireless, mobile and more modular applications may be handled by a lesser number of testers. However, the key to success in any project regardless of the ratio of developers to testers is well-defined requirements, APIs, peer reviews & unit testing early on in the project lifecycle.

Cost savings using a co-sourced model while maintaining suitable developer to tester ratio

Authors believe that pure outsourcing while it offers price advantages in off shoring to a country such as India, Russia or China doesn't work as effectively as co-sourcing where existing QA and testing resources are supplemented by offshore teams. Let's work through some numbers to illustrate this point.

Let's imagine an enterprise software application with significant amount of user interface and ecommerce backend work involving database access and multiple threads of business usage.



The project duration is of 1 year with the first major release happening in 6 months and subsequent releases happening on a quarterly basis. For the sake of simplicity, monthly cost per developer and tester is assumed to be $10,000.00. Since we are focusing on the team size of QA and testing, we will assume a fixed development team size of 10 developers. A key underlying assumption we are using is that there is a good software engineering practice being followed by development teams including well defined

[4] Cusamano, Michael A. and Richard W. Selby, Microsoft Secrets, Simon and Schuster, Inc., New York, 1995
[5] SQATester.com
[6] It Depends: Deciding on the Correct Ratio of Developers to Testers. 2000. Johanna Rothman
[7] Software Program Manager's Network, Best Practices, 1999, on www.spmn.com
[8] McConnell, Steve. Rapid Development, Microsoft Press. Redmond WA, 1996.

303 Almaden Boulevard, 6th Floor
San Jose, CA – 95110
408.998.7844 (Phone)
408.998.7845 (Fax)

www.nirixa.com
info@nirixa.com

APIs, bug tracking systems and source control systems as well as product requirements definitions and functional requirements are well defined. We are assuming that an offshore team fully burdened costs $3000 per tester. For the one-year duration, we get the following numbers (All values are in millions).

	Ratio 5:1	**Ratio 5:4 No Co sourcing**	**Ratio 5:4 w Co sourcing**	**Ratio 5:4 all Outsourcing**
Development $	1.2	1.2	1.2	1.2
Testing $	0.24	0.96	0.456	0.288
Total Costs $	1.44	2.16	1.656	1.488

As mentioned earlier, Nirixa recommends developers to testers ratio of 5:2 or better. In this case study we are assuming if there are 10 developers and 2 testers then the testing staff will be overwhelmed. A ratio of 5:4 with no co-sourcing becomes very expensive. On the other hand, if the 2 testers who started with the project are supplemented by 6 testers from an off-shore company in a co-sourced model, the chances of project success and delivering high quality products on time is significantly increased. The outsourcing model where all testing staff is outsourced is not very effective because QA and testing is highly communication-oriented activity. With newer trends in software development including emphasis on web services, QA and testing is becoming more intertwined with early development. Consequently, it is more effective to retain some in-house QA resources and supplement them with co-sourced offshore QA and test resources.

Summary

The outsourcing QA and testing trend is here to stay in the current market conditions and need for better product quality while controlling the costs. Business executives and information systems professionals are better off carefully considering all the criteria and available choices before taking a decision to outsource to an offshore entity. What country are they planning to outsource to, the quality, culture, communication and English speaking skills all should be taken into account. References of other customers that have been serviced by the vendor they are considering should be obtained and checked early on. Preference should be to the co-sourced model than a complete outsourced one. The offshore vendor should be used to supplement existing QA and test resources rather than replace them en mass. Those companies that maintain offices and key executives in USA should be given consideration. It will provide an important communication and problem-solving intermediary with the offshore team

QW2002 Paper 2P2

Mr. Eric Patel
(RapidSQA)

Creating Quality From Scratch: How To Build a Test Organization

Key Points

- A fully-functional test department takes time
- Get ready to simultaneously take on multiple roles and responsibilities
- Make sure the quantity of your tasks does not adversely affect the quality of your tasks

Presentation Abstract

With more and more companies realizing the need for testing throughout the product development process, there's a growing demand for bigger, better QA teams. If you find yourself the first member of a newly formed department, it's likely you'll be asked to do the testing and build the team simultaneously. This presentation presents a strategy for meeting the day-to-day testing challenges, while planning for the future of the department. It tells you how you can lay the foundation and build the house ? all at the same time.

About the Author

Eric Patel is Chief Quality Officer for RapidSQA, a Software Quality Service Provider (SQSP) of training and consulting solutions. He is co-founder of the Nokia Quality Forum (NQF) Boston and the QAI Greater Boston Software Quality Assurance Association, and also is creator of the RapidSQA methodology for Web testing. Eric has 11+ years of experience in software testing, test management, and software quality assurance.

Eric is a frequent presenter at software quality conferences and meetings. He has been certified by ASQ as a quality manager and software quality engineer, and by QAI as a software test engineer. As Deputy Regional Councilor for the ASQ Software Division Region 1, Eric maintains active memberships in ASQ and IEEE. Published in Software Quality Professional (SQP) and STQE, he also serves as a reviewer for SQP and The Journal of Software Testing Professionals. In addition, Eric is co-founder and instructor with Northeastern University's new Certificate Program in Software Quality Assurance. He holds a BSEE from the University of Vermont.

QualityWeek 2002

International Internet & Software Quality Week

September 4, 2002

Creating Quality From Scratch

How to Build a Test Organization

Eric Patel
CQM, CSQE, CSTE
Chief Quality Officer
RapidSQA
epatel@rapidsqa.com
www.rapidsqa.com

Creating Quality From Scratch Copyright © 2001-2002, Eric Patel.

RapidSQA

Agenda

- The Situation
  - Dealing with Perceptions and Expectations
  - Day-to-Day Testing Challenges
  - Managing the Effects of "Solitary Confinement"
- Test Planning and Time Management Techniques
- Mistakes to Avoid
- The Balancing Act
  - Laying the Foundation while Building the House
  - Planning for the Future
  - Resource Planning Matrix
- Lessons Learned
- Action Plan
- Conclusion
- Resources

Creating Quality From Scratch Copyright © 2001-2002, Eric Patel.

RapidSQA

The Situation

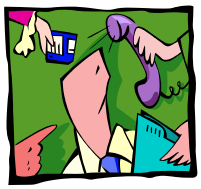
- You're the first – and only – hire
- There are no formal testing or quality assurance activities
- The product may be at any stage of development
- You have to do all the work of an entire department
- The question you find yourself asking: "What did I get myself into?"

Dealing with Perceptions and Expectations

Perception/ Expectation	What to Do
You are the "expert" and know what you're doing	• Obtain education/training
They don't know what you should be doing	• Write your job description • Educate them on testing
You are solely responsible for testing	• Document what you will and will not be testing • Consider outsourcing
You will help hire future members of your team	Manage the recruitment process
You are the acting Test Manager until someone is hired	Learn to manage yourself, others, and the test process

Day-to-Day Testing Challenges

- Hands-on, manual testing
- Bug reporting
- Bug triage meetings
- Product quality status
- Fighting the daily "fires"
- High visibility role
- Feeling of "solitary confinement"

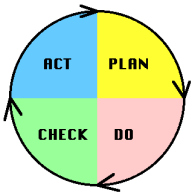


Managing the Effects of "Solitary Confinement"

- Interact and build relationships with the developers, designers, and others
- Involve and solicit testing assistance from others
- Leave the office and/or go out to lunch
- Make time to train and educate yourself
- Cultivate your professional network
- Be prepared to work longer and harder but try to work smarter
- Watch for burnout symptoms

Test Planning Techniques

- Test documentation
  - Quality Manual
    - Overview of the quality principles policies, procedures, and guidelines
  - Test Plan
    - Description of what will (and will not) be tested
- Resource planning
  - Team members
  - Equipment
  - Time



Time Management Techniques

- Effectiveness- doing the right things
- Efficiency- doing things right
- 7 Habits of Highly Effective People by Stephen Covey

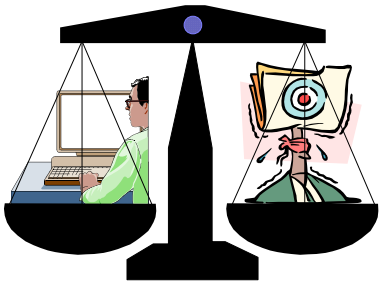
Urgent and important	Not urgent but important
Urgent but not important	Not urgent and not important

Mistakes to Avoid

- Unrealistic and overly-optimistic expectations or assumptions
- Not getting buy-in and support
- Not documenting your testing efforts or creating a testing schedule
- Implementing test automation prior to manual testing
- Not having a coach or mentor
- Loosing sight of the end goal
- Not learning from your mistakes

The Balancing Act

- Test Engineer
  - Test execution
  - Defect management
  - Test reporting
- Test Manager
  - Recruiting/HR
  - Test planning
  - Test management

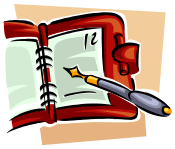


Laying the Foundation while Building the House

- Juggle hands-on contributor and test manager responsibilities
  - "Holding down the fort" as you "wait for reinforcements"
- Establish yourself as a knowledgeable and competent tester
  - Capitalize on your strengths
  - Convert weaknesses to strengths
- Demonstrate test management skills
  - Show that you can do the job
  - Inform management of your career goals/intentions

Planning for the Future

- Initially, the majority of your tasks will be of the test engineer
- Over time, more of your responsibilities may include test management
- Demonstrate your leadership capabilities
- Take your career path into consideration
- Don't forget about corporate culture



Resource Planning Matrix

Area	*Short term, Low cost*	*Long term, Increased cost*
Bug tracking	**Spreadsheet**	**Defect management system**
Project management	**Spreadsheet**	**Project management tool**
Test automation	**Scripting**	**Automated test software**
Test environment	**Workstation**	**Test lab**

Lessons Learned

- A fully-functional test department takes time
- Mentally prepare yourself for the long haul
- Get ready to simultaneously take on multiple roles and responsibilities
- A written and approved test plan keeps everyone honest
- Make sure the quantity of your tasks does not adversely affect the quality of your tasks
- Educate others on features and benefits of testing
- Diligent recruiting pays off in the long run
- You may be asked to "step up to the plate"

Action Plan

- Assess the present situation
  - Business plan
  - Corporate goals and objectives
  - What is the desired quality level?
  - Promises made to the customer
- Plan the current testing effort
  - Risk Analysis
  - Quality Manual
  - Test Plan

Action Plan (cont.)

- Perform daily testing activities
  - Manual testing
  - Bug reporting
- Plan for long-term test improvement
  - Roadmap all testing activities and resources
  - Acquire additional resources
  - Obtain upper management buy-in on the plan

Conclusion

- "Creating quality from scratch" involves "building the house" while simultaneously "laying the foundation"
- Building a testing organization involves a balancing act between individual contributor and team lead/management roles
- Tackle the most important issues adequately… simultaneously building for today… while also planning for tomorrow

Resources

- ASQ Quality Management Division. (2001). The Certified Quality Manager Handbook. ASQ Quality Press. 0-87389-487-1.
- Black, Rex. (1999). Managing The Testing Process. Microsoft Press. 073560584X.
- Frank, Barbara; Marriott, Phil; and Warzusen, Chett. (2000). The Software Quality Engineer Primer. Quality Council of Indiana.
- Humphrey, Watts S. (1989). Managing the Software Process. Addison-Wesley. 0-201-18095-2.
- Kit, Edward. (1995). Software Testing in the Real World. Addison-Wesley. 0-201-87756-2.
- Pressman, Roger S. (1997). Software Engineering: A Practitioner's Approach. McGraw-Hill. 0-07-052182-4.

Resources (cont.)

- American Society for Quality (ASQ): www.asq.org
- QA Forums: www.qaforums.com
- Quality Assurance Institute (QAI): www.qaiusa.com
- RapidSQA: www.rapidsqa.com
- Software Quality Engineering: www.sqe.com
- Software Quality Page: swquality.com/users/pustaver/index.shtml
- SQA Tester: sqatester.com
- StickyMinds: www.stickyminds.com

QW2002 Paper 3P1

Dr. Huey-Der Chu & Mr. Jin-Chuan Lu
(National Defense Management College)

Applied Six Sigma to Software Process Improvement

Key Points

- Software Process Improvement by Six Sigma
- Load Testing of a Multi-Tier Client/Server Application
- Data Analysis by The Kolmogorove-Smirov statistical means

Presentation Abstract

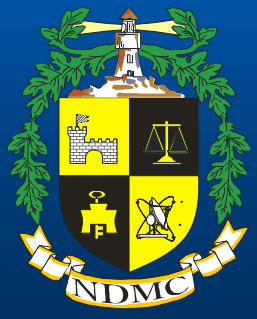
This paper applied the Six Sigma quality cycle: Define, Measure, Analyze, Improve and Control (DMAIC) to improve software process. We plotted user complaints against response time (define), drew up a test plan and executed the load testing (measure), collected testing data from different client sites for analysis (analyze), found where the bottleneck was and fixed it (improve) and monitored the status to satisfy users' requirement (control).

About the Author

Dr. Huey-Der Chu is currently the director of Computer Centre at the National Defense Management College in Taiwan as well as the member of Committee of Software Quality at the Chinese Society for Quality. His research currently interests in Software Process Improvement, Knowledge Management and Quality Management. Dr. Chu received a Ph.D. (1998) in the Centre for Software Reliability at the University of Newcastle upon Tyne, England, funded by the National Science Council in Taiwan.

Jin-Chuan Lu is studying Management of Information Science at Institute of Information Research, National Defense Management and he will get his master science degree in July 2002. His research interests include software quality and testing process issues. Before this he worked at DCGS for Communications Electronics & Information, Ministry of National Defense, R.O.C. and specialized Defense Information System development and project management over 10 years.

Applied Six Sigma to Software Process Improvement



6σ SPI

National Defense Management College
Huey-Der Chu

Outline

- Introduction
- An Overview of Six Sigma
- Performance Testing
- Case Study
- Conclusion

Introduction

- **Zona Research concluded that a majority of users would tend to abandon a web site if the web page cannot be accessed in 8 seconds.**

- **Research by Robert Miller and Jakob Nielsen further indicated that most users who browse the Internet tend to spend no more than 10 seconds concentrating on an interactive web page .**

第3頁

Introduction

- Internet base Applications
  - No geological constrain
  - The Number of users
  - Real-time interaction
- From customer view
  - do not care how the problem got there
  - just know that Web site "do not work"
- Getting the deliverable right the first time around

第4頁

Introduction

- Six Sigma
  - **a quality concept made popular by Motorola's quality Improvement during 1988-1989**
  - **applied successfully in many service-based organizations**
- Planning Six Sigma into an IT project

第 5頁

Introduction

- We applied the Six Sigma quality cycle: Define, Measure, Analyze, Improve and Control (DMAIC) to resolve this problem.
  - plotted user complaints against response time (define),
  - drew up a test plan and executed the load testing (measure),
  - collected testing data from different client sites for analysis (analyze),
  - found where the bottleneck was and fixed it (improve)
  - monitored the status to satisfy users' requirement (control).

第 6頁

An Overview of Six Sigma

- What is Six Sigma?

-- Six Sigma is the management philosophy that is sweeping the world by storm. Created first by Motorola in the 1980s, then popularized by AlliedSignal and General Electric (GE) in the 1990s.

--Six Sigma has more than proven its worth to organizations attempting to improve their productivity and profitability.

第 7頁

An Overview of Six Sigma

-- It is first and foremost a business process that enables companies to increase profits dramatically by streamlining operations, improving quality and eliminating defects or mistakes in everything a company does.

-- It provides specific methods to re-create the process itself so that defects are never produced in the first place.

第 8頁

An Overview of Six Sigma

-- Most companies operate at a three- to four-sigma level, where the cost of defects is roughly 20 to 30 percent of revenues.

-- By approaching Six Sigma – 3.4 defects per million opportunities – the cost of quality drops to less than 1 percent of sales.

-- When GE reduced its costs from 20 percent to less than 10 percent, it saved a billion dollars in just two years—money that goes directly to the bottom line.

第 9頁

An Overview of Six Sigma

The method to improve processes is DMAIC:

- **Define.** Defining the team to work on improvement, defining the customers of the process, their needs and requirements and creating a map of the process to be improved.
- **Measure.** Identifying key measures of effectiveness and efficiency and translating them into the concept of sigma.
- **Analyze.** Through analysis, the team can determine the causes of the problem that needs improvement.
- **Improve.** The sum of activities that relate to generating, selecting and implementing solutions.
- **Control.** Ensuring that improvement sustains over time.

第 10頁

The Issues of Performance Testing

- The key indicator in performance testing, a process in which the observation is made to the changes of a web site's response time under a designed load, has been the user's average response time.
- a performance testing is best be implemented in the early stage of a system development

第 11頁

The Issues of Performance Testing

-- Given simulating the user's behavior pattern has been a crucial prerequisite in performance testing, most performance testing tools often come with capture and replay functions that allow pre-record a user's access operations, which are then loaded onto the system through the replay function.

-- This not only allows the loading status be incrementally increased to simulate logon by multiple users, but can best simulate a more realistic online access via the built-in dialog modification recording file in simulating a user's access information.

第 12頁

The Issues of Performance Testing

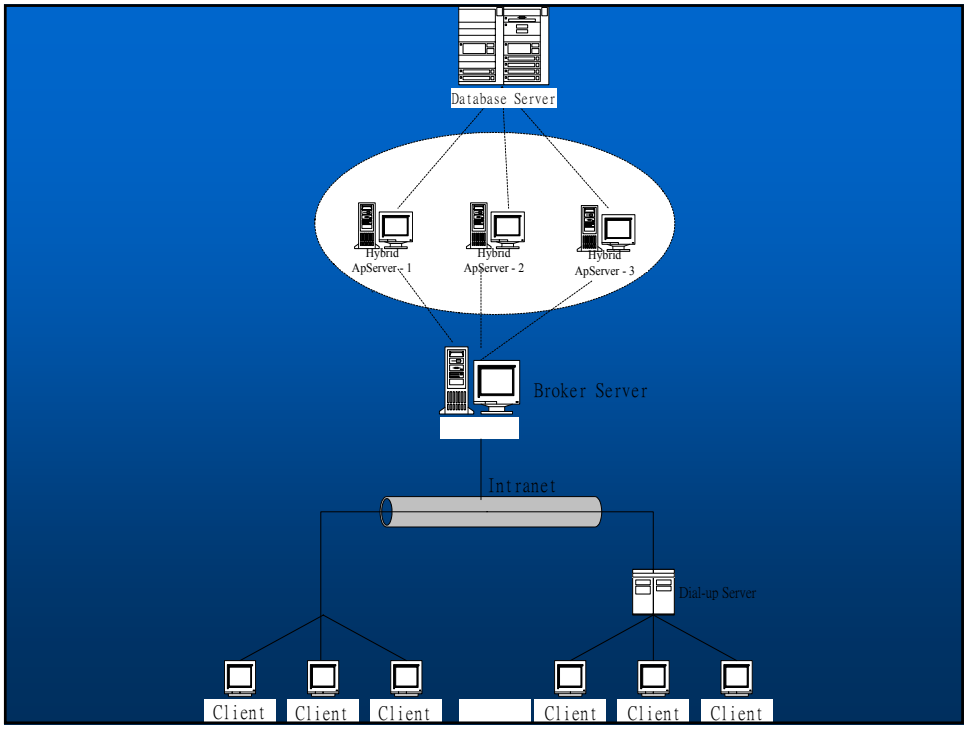
-- the monitoring functions provided by the system hardware and software also offer valuable data and an array of analysis reports that would help the tester understand the system's operating criteria and response at the time.

第 13頁

Case Study

The selected company has an island-wide presence, with a total of 1,022 user points at the present time, where the merchandise of all outlets are requisitioned through the company's merchandising distribution information system, which has been streamlined toward the end of 2001.

第 14頁



Case Study

Define Phase:

While company assessment estimating the request interval to business volume puts the maximum roof on simultaneous company-wide online requests at 168 users, the system loading capacity should be sufficient to accommodate the demand of 168 users simultaneously executing their online commands

Case Study

there is no way of knowing how the system will respond under such loading demand since the system has not yet encountered such loading demand, let alone that it may be difficult to grasp when such type of similar loading requirement will occur in the actual processing, hence the study has aimed to utilize effective testing to examine the system's response vs. its loading performance.

第 17頁

Case Study

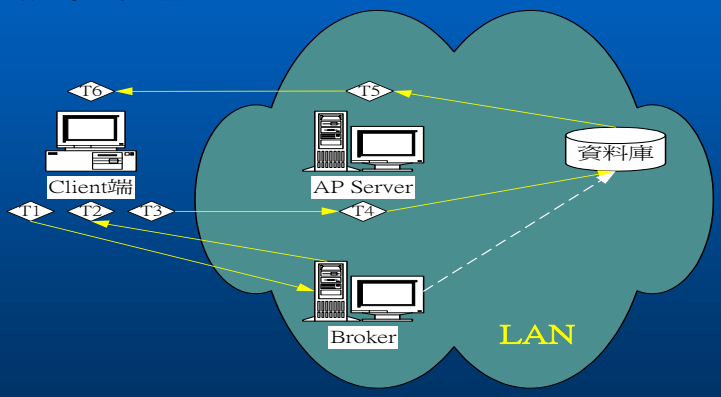
Measure Phase:

The project rates the system's overall response efficiency as the objectivity of the assessment, together with predefined test requirements, to come up with 100 test accounts (Test001 – Test100), in which a multi-sequencing test program has been devised by project researchers to sequentially activate the test program according to different account numbers, whereby a designated test hits are loaded to the server end, and the database is checked for inquiry update in every test cycle .

第 18頁

Case Study

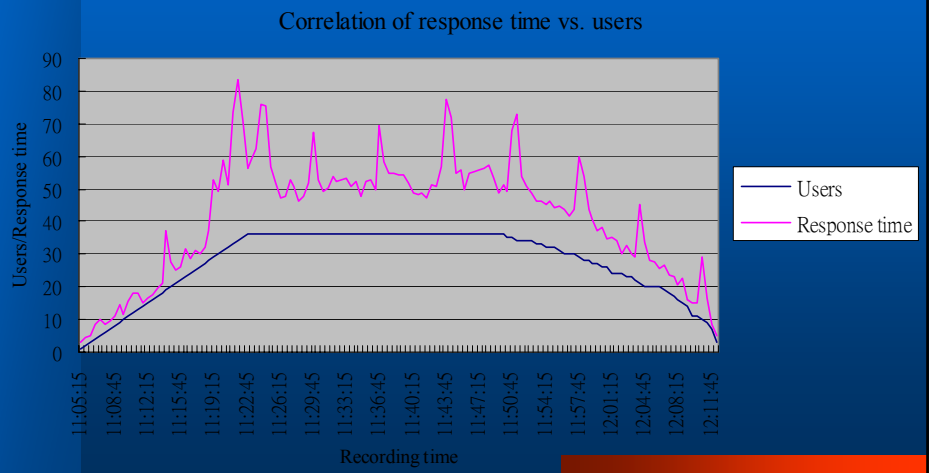
Test Framework



第 19頁

Case Study

Analyze Phase:



第 20頁

Case Study

The Kolmogorove-Smirov statistical means concluded from the test data include a rating of 0.158189 as shown in below

第 21頁

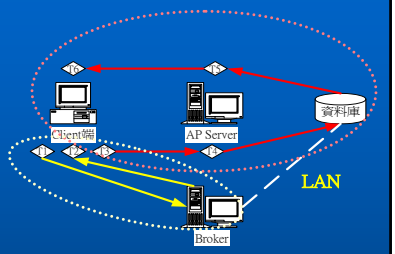
Class Groups of values	$F_{actual}(x)$ Actual value	$F_{expected}(x)$ Expected value	$\lvert F_{actual}(x) - F_{expected}(x) \rvert$
0~2	0.000000	0.000000	0.000000
3~4	0.014815	0.000016	0.014799
5~6	0.029630	0.000239	0.029391
7~8	0.096296	0.001974	0.094332
9~10	0.133333	0.010306	0.123027
11~12	0.192593	0.037473	0.155120
13~14	0.251852	0.101467	0.150385
15~16	0.311111	0.215386	0.095725
17~18	0.370370	0.373954	0.003584
19~20	0.451852	0.551197	0.099345
21~22	0.555556	0.713745	**0.158189**
23~24	0.711111	0.838229	0.127118
25~26	0.807407	0.919024	0.111617
27~28	0.881481	0.964025	0.134978
29~30	0.933333	0.985767	0.052434
31~32	0.992593	0.994965	0.002372
32~34	0.992593	0.998401	0.005808
34~	1.000000	1.000000	0.000000

第 22頁

Case Study

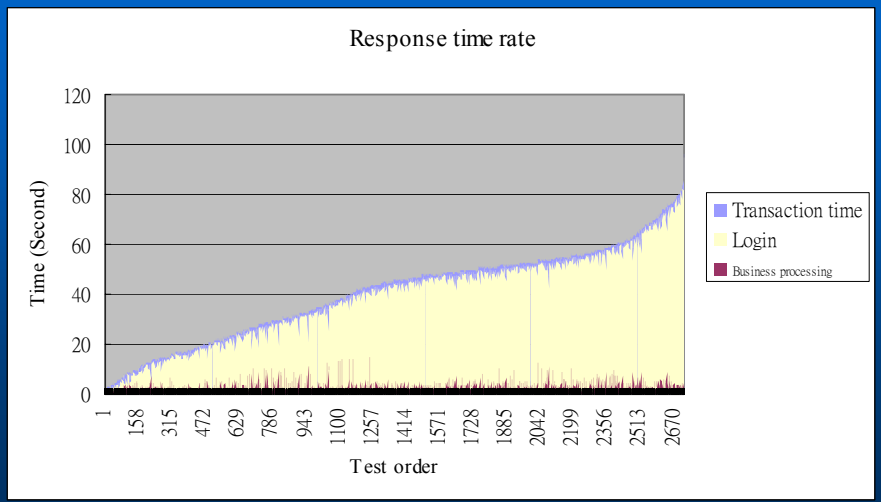
Analysis the test data

- Login in stage(T1-T2)：
  - More test data，more the time for login
- Business process stage (T3-T6)
  - More test data, no more time to process



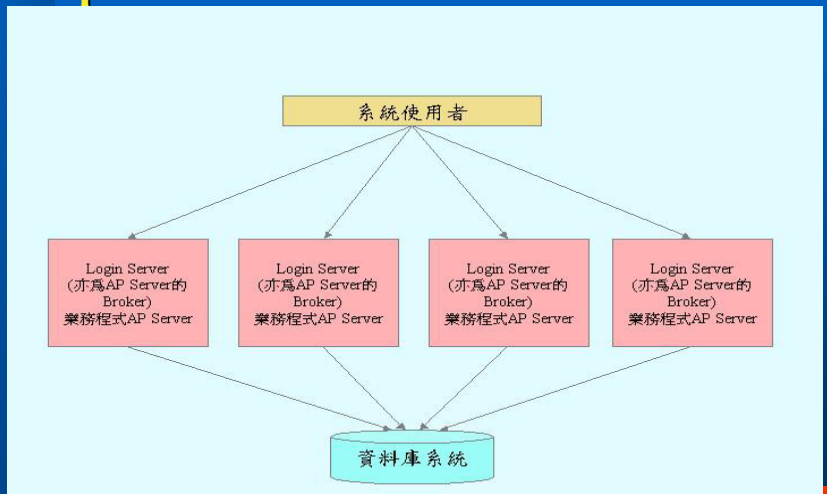
第 23頁

Case Study



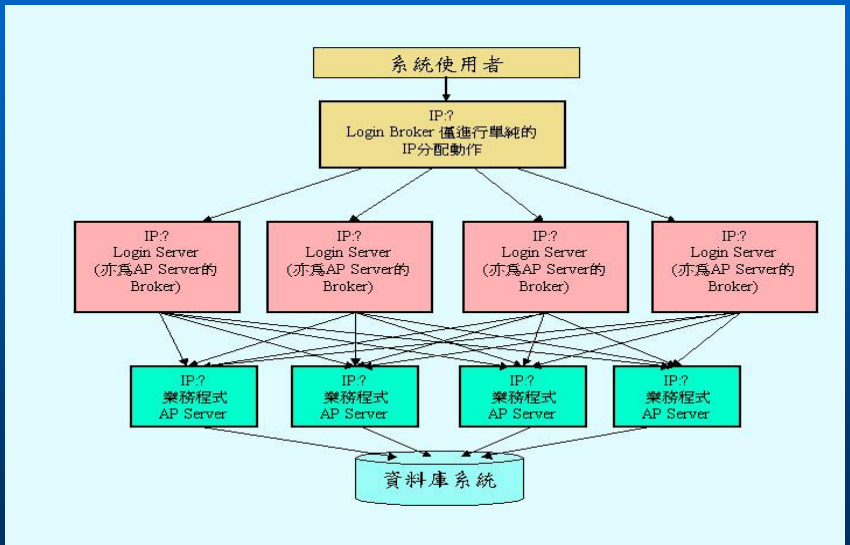
Case Study

Improve Phase:

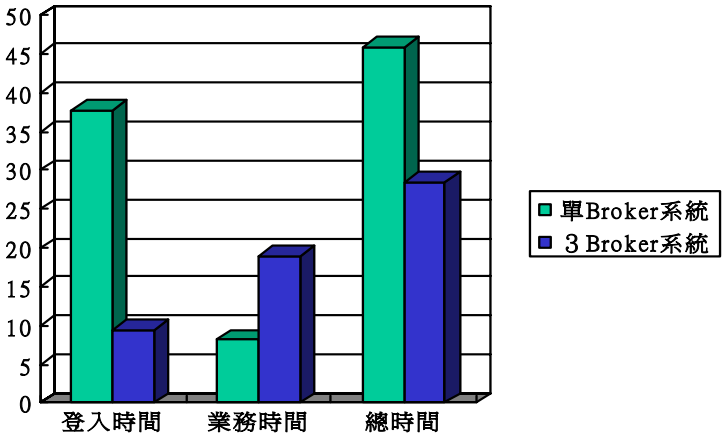


第 25頁

Case Study

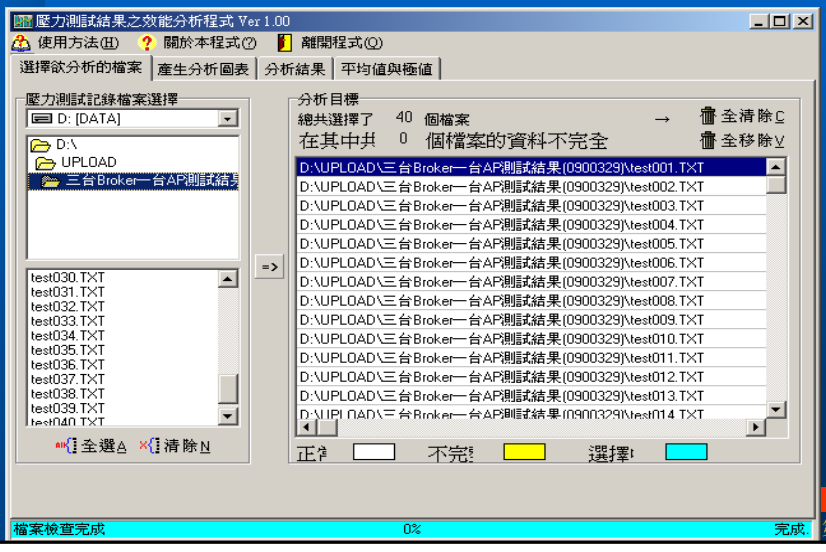


Response Time Comparison



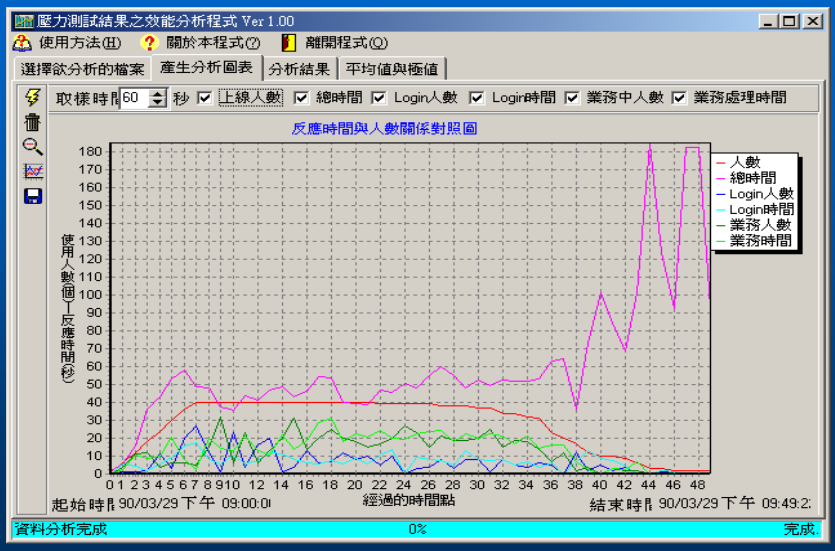
Case Study

Control Phase:



第 28頁

Case Study



Case Study

Conclusion

-- Management problems in software development
-- Six Sigma is the most powerful breakthrough management tool
-- The method of Six Sigma mapped the software development life cycle for client/server applications against the Six Sigma quality cycle (DMAIC)
-- and then aligned those cycles against the quality management life cycle (PDCA)

第 31頁

Conclusion

-- The case is shown in this paper to demonstrate that this method not only assisted the development team to focus on reducing development variation, but also gave the rest of the company an understanding of the IT involvement necessary for success.

第 32頁

Applied Six Sigma to Software Process Improvement

Huey-Der Chu Jin-Chuan Lu
Graduate School of Defense Information
National Defense Management College, Taiwan

Abstract

To achieve high quality software, it is essential to prevent errors from occurring and to test the software sufficiently before the product is delivered. The aim must be to ensure work is done "right first time" at every stage of the software development process. An experience of software process improvement by Six Sigma quality concept is illustrated. The application in the selected company is a multi-tier framework system and there are 1,022 client sites distributed in Taiwan area, one broker server site and three application server sites, which concurrently access and store data from a center database server. When delivering application to the Intranet, users dissatisfaction was at an all time high. We applied the Six Sigma quality cycle: Define, Measure, Analyze, Improve and Control (DMAIC) to resolve this problem. We plotted user complaints against response time (define), drew up a test plan and executed the load testing (measure), collected testing data from different client sites for analysis (analyze), found where the bottleneck was and fixed it (improve) and monitored the status to satisfy users' requirement (control).
Keywords: Six Sigma, Software Process Improvement, Client/Server Applications

1. Introduction

Zona Research concluded that a majority of users would tend to abandon a web site if the web page cannot be accessed in 8 seconds. Research by Robert Miller and Jakob Nielsen further indicated that most users who browse the Internet tend to spend no more than 10 seconds concentrating on an interactive web page [7]. While the profitability of commercial web sites is inextricably tied to the traffic, an under-performing system can spell losses for it could lead to reduced profit at the very least, or a compromised business reputation at the worse scenario.

While there is virtually no geological constrain to the user thanks to the rapid development of the World Wide Web Internet as the user can be anywhere thanks to the Internet access, the number of users also quadrupled when compared with the conventional client/Server applications, in which the predominant emphasis of a real-time interaction has also made system performance a critical issue. Since the customer is next in line, there is no room for error. When customer use that application, they do not care how the problem got there, they just know that the Web site "does not work." The last thing a company wants in this seamless and competitive environment is to tarnish its image and efforts with inferior Web quality. However, one bottleneck in the information technology development remains: "Getting the deliverable right the first time around."

Six Sigma is a quality concept made popular by Motorola's quality improvement during the 1988-1989 timeframe and, since that time, has been applied successfully in many

service-based organizations [2, 4]. When Web development and Six Sigma are used together, the result is Internet excellence [4]. Therefore, the purpose of this research is planning Six Sigma quality into an information technology project adds the benefit of a shared vision toward excellence and smooth transactional service.

2. An Overview of Six Sigma

Six Sigma is the management philosophy that is sweeping the world by storm. Created first by Motorola in the 1980s, then popularized by AlliedSignal and General Electric (GE) in the 1990s, Six Sigma has more than proven its worth to organizations attempting to improve their productivity and profitability [2,5]. It is first and foremost a business process that enables companies to increase profits dramatically by streamlining operations, improving quality and eliminating defects or mistakes in everything a company does, from filling out purchase orders to manufacturing airplane engines. While traditional quality programs have focused on detecting and correcting defects, Six Sigma encompasses something broader: It provides specific methods to re-create the process itself so that defects are never produced in the first place.

Most companies operate at a three- to four-sigma level, where the cost of defects is roughly 20 to 30 percent of revenues [5]. By approaching Six Sigma – 3.4 defects per million opportunities – the cost of quality drops to less than 1 percent of sales. This is because the highest quality also results in the lowest costs. When GE reduced its costs from 20 percent to less than 10 percent, it saved a billion dollars in just two years—money that goes directly to the bottom line.

The method GE and several other organizations use to improve processes is summarized by the initials DMAIC as following [2]:

(1) Define. Defining the team to work on improvement, defining the customers of the process, their needs and requirements and creating a map of the process to be improved.
(2) Measure. Identifying key measures of effectiveness and efficiency and translating them into the concept of sigma.
(3) Analyze. Through analysis, the team can determine the causes of the problem that needs improvement.
(4) Improve. The sum of activities that relate to generating, selecting and implementing solutions.
(5) Control. Ensuring that improvement sustains over time.

3. The Issues of Performance Testing

Although time and money consuming, performance testing remains highly regarded as a cost efficient operation of its potential yield being that it acts to uncover potential system glitch, with which system resources can be reshuffled or more precision computation can be found to address the demand.

The key indicator in performance testing, a process in which the observation is made to the changes of a web site's response time under a designed load, has been the user's average response time. And what make the testing findings crucial are how they can help us understand the functionality provided by the combination of hardware and software allocated. In general, a performance testing is best be implemented in the early stage of a system development, where a variety of feasible hardware and software combinations can be simulated to help an organization decide whether to purchase an expensive hardware that saves the development time, or investment in optimal software developing technology [1].

Given simulating the user's behavior pattern has been a crucial prerequisite in performance testing, most performance testing tools often come with capture and replay functions that allow pre-record a user's access operations, which are then loaded onto the system through the replay function. This not only allows the loading status be incrementally increased to simulate logon by multiple users, but can best simulate a more realistic online access via the built-in dialog modification recording file in simulating a user's access information, such as the account number, modification on purchasing quality, modification on logon interval, print out and save file features. Also, the monitoring functions provided by the system hardware and software also offer valuable data and an array of analysis reports that would help the tester understand the system's operating criteria and response at the time.

This paper attempts to apply the DMAIC method from Six Sigma to Software Process Improvement. A case study will demonstrate the techniques used by one company to combine Six Sigma quality with the performance testing effort.

4. Case Study

The selected company has an island-wide presence, with a total of 1,022 user points at the present time, where the merchandise of all outlets are requisitioned through the company's merchandising distribution information system, which has been streamlined toward the end of 2001. Despite an array of operating functions, the user units and company executives remain skeptic of the system's loading response performance. Bound by the pressing development timetable and the lack of auditing manpower, the absence of a complete software validation prior to the system's induction speaks of a potential concern in most system development processes. Although the chances of the firm's nationwide branches coming online simultaneously are rather slim, it is nevertheless prudent to seek remedy by rechecking the execution efficiency of the merchandising distribution information system before severe consequences may result. This has spawned the desire to have the system's loading capacity validated through professional testing before the system is fully streamlined.

The system has a three-tier framework design, divided into the users on the client end; web server, broker server and application server on the application end; database server on the server end. Currently, there are 1,022 users on the client end, one broker server and three application servers on the server end. And the user goes online through

the company's Intranet or the dial-up service. The system's hierarchical framework is shown in Fig. 1.

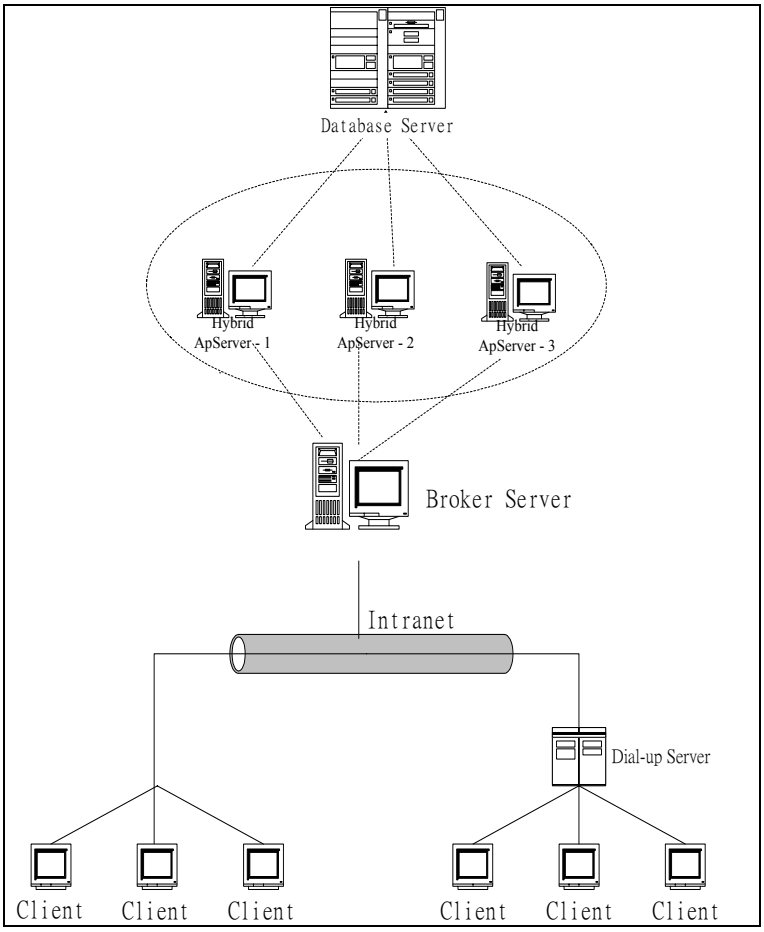


Fig. 1 Diagram of the Merchandising Distribution Information System

The system operates on the WinSock protocol, instead of the more popular HTTP protocol. The user logs on to a broker server via an exclusive operating program, where the broker server routes the user's account number and password to compare with the user's account number data retrieved from the database, and the validation signals are routed to one of the application server, where an operating program executes the request and inquiry. While the web page server supports business data inquiry and downloads operating programs for update purposes.

(1) Define phase:

While company assessment estimating the request interval to business volume puts the maximum roof on simultaneous company-wide online requests at 168 users, the system loading capacity should be sufficient to accommodate the demand of 168 users simultaneously executing their online commands. Nevertheless, there is no way of knowing how the system will respond under such loading demand since the system has not yet encountered such loading demand, let alone that it may be difficult to grasp when

such type of similar loading requirement will occur in the actual processing, hence the study has aimed to utilize effective testing to examine the system's response vs. its loading performance. Moreover, the company also wishes to determine if the system's design framework has been sufficient, and if there is any improvement needed to clear of any potential bottleneck to its anticipated efficiency.

Taken into account the system settings, several difficulties stand to confront the test:

- Despite there are many effective testing tools for the three-tier system out in the software market today, the system's WinSock protocol makes it difficult to locate a simpler version of the testing tools that can completely support the required test items.
- The system is best not be interrupted by the testing operations being that an already streamlined operating system needs to support service demands coming from the users across the island on a daily basis.
- Unlike what most automated testing tools taken to the capture and replay mode in executing the test, which allows a small amount of input for duplicating a large amount of test data needed to simulate a multiple number of virtual users operating online for conducting peak load testing, there are factual difficulties in adopting automated testing tools being that the system's users are assigned with fixed account number under strict clearance, and that the operating sequence being executed are limited to what has been authorized.
- It may be infeasible to measure the system's actual loading response unless the required users come online at the same time, and there are many questions to be overcome in terms of preparing the test setting, coordination of the operation, manpower and budget, before the method can be applied.

In light of the above, the test team's assessment leads to surveying for valid samples before the system's performance variation tendency can be concluded via statistical analysis, which may then be used for estimating the system's response time under a designated load; following depicts the final implementation sought.

(2) Measure phase:

The project rates the system's overall response efficiency as the objectivity of the assessment, together with predefined test requirements, to come up with 100 test accounts (Test001 – Test100), in which a multi-sequencing test program has been devised by project researchers to sequentially activate the test program according to different account numbers, whereby a designated test hits are loaded to the server end, and the database is checked for inquiry update in every test cycle; its designed functions are described as follows,

a. Time synchronization is executed during the initial online linkup, designed to synchronize the timing sequence between the client end and the server end.
b. The test procedure is then executed automatically at the designated time according to the number of hits programmed.
c. Actual commands of insert, update, delete and select are simulated to the merchandising distribution information system.

d. A testing diagram, as shown in Fig. 2, has six time sequencing recording points (T1 – T6) designed according to the system framework, where data on each time sequencing point occurred from T1 – T6 in each cycle are recorded in the log file.

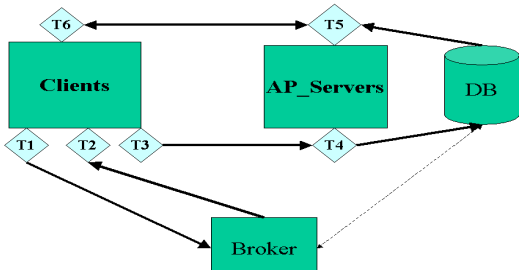


Fig. 2 Testing Framework

e. Each execution program's startup time and related parameters are encoded for identification purposes. And once the program has been executed in full, pertinent data on the timing sequencing of the program ID, startup time, end time and error are reloaded to the log file, where the log files are identified by session. The logging process is as follows,

(a) Log files are created on the client end's test computer, where timing synchronization and startup time are recorded.
(b) With the client end's startup time command defined as T1, the program is then linked to the broker server for identification validation, where the broker server will link to the back-end database to carry out identification validation; a correct identification will allow the broker server to assign an application server to the client end's program, and a time sequencing point T2 is recorded.
(c) The client end's program retrieves the business program at the application server, and a time sequencing point T3 is recorded.
(d) The application server then executes a business program containing SQL statements to retrieve data from the database, where the starting time being T4 and the end time being T5.
(e) The application server reverts the retrieved data to the client end's test program, and a sequencing time T6 is recorded.
(f) As soon as the data transmission is completed, the client end's program will call out a business program from the application server and reverts the application server's execution timing of T4 to T5. Where a multiple number of SQL statements contained in a group of programming will require the timing of each SQL statement be recorded, meaning that there will be corresponding time of T3 and T6 being recorded.
(g) As each transaction is completed, the response time of T1 to T6 and related data are recorded on the client end log files, readying for subsequent analysis.

The testing procedure has been conducted on the existing operating environment, consisting of,

a. Database: The existing database server has been used.

Operating system: Solaris UNIX.
Database: Oracle Database Server 8.

b. Broker Server:
Operating System (OS): Microsoft NT 4.0.

c. Application Server:
Operating System (OS): Microsoft NT 4.0.

d. Web Server:
Operating System (OS): Red Hat Linux 6.3.
Web Server; Apache Server.

e. Client Setting:
Operating System (OS): Microsoft Windows 98.
Browser: Microsoft IE 5.0.

For security, a complete backup of the database server, including the operating systems, database and so forth, is run eight hours prior to the testing is to being.

In support of testing the test data segments on the database server in order to run the application server under the existing system framework, the service programming contained on the application server are modified without jeopardizing the normal operations so that both the automated test program accessing the designated test data segments and the normal users accessing online data can both log on and access the system. To ensure the accuracy of testing, the client ends' hardware and software equipment are kept uniform, where the database server, broker server, application server and web server are tested using the existing servers and conducted during normal office hours of between 8am and 17pm, seeking to conclude dependable ratings on the loading performance of the servers.

Given that time delay invariably occurs in every step of the program execution to and from the client server via the networking, most of the delays are created by the router's storage and transmission characteristics if the retrieval is conducted over the Internet or an Intranet, a factor that is largely dependent on the number of routers between the server and user [4]. To best avoid the impact of such delays, the test sites are selected on those nearest the server end, together with the client ends' six test computer hardware and software setup kept uniform, to cut down the performance variations on each of the equipment which may affect the accuracy of test data concluded.

The testing procedure, sequentially carried out by the test team, has the test procedure perfected through repeated data testing and experience accumulation. While some of the test values had not been as ideal for the lack of system comprehensive in the initial stage; for example, the excessive repetitive user hits or the extended interval in the increasing user load would invariably lead to a surge in system response time to drag out the test schedule and result in having to abandon a test intermittently. After repeated experiments, an optimal repetitive execution on every test account has been concluded with cumulative experience indicating that an average performance timing at 2.251 second per hit being the best in executing a cycle of the test program.

To obtain a large sample of statistics without dragging on the test schedule due to interruption of networking or interfering the normal operations, timing interval that suffices to screen out 30 sets or more has been set under the increased system loading to allow the execution time be completed within no more than two hours as the number of users decreases. We have adopted the design of Poisson distribution model as means for evaluation a user-loading model [5]. Eventually a 30-second interval has been used for testing the surge user model, in which six computers are used to execute the test and each test computer is to sustain six execution programming, starting from one and incrementally increased to up to thirty-six test programming as shown in Table 1, where the entire test schedule has been manipulated to run within a 2-hour span.

Table-1 Summary of Test Account Placement

Test Computer	No.1	No.2	No.3	No.4	No.5	No.6
Test Account	Test001 To Test006	Test007 To Test012	Test013 To Test018	Test019 To Test024	Test024 To Test030	Test031 To Test036

(3) Analyze Phase

In the actual testing process as the number of users increases, the response time will also go as indicated in Fig. 3 for a correlation of response time vs. user base, indicating how detailed changes occur when the system is loaded. Despite there may not be clear indication of a maximum system loading capacity, we are somehow able to establish a response tendency of a loaded system under the test model.

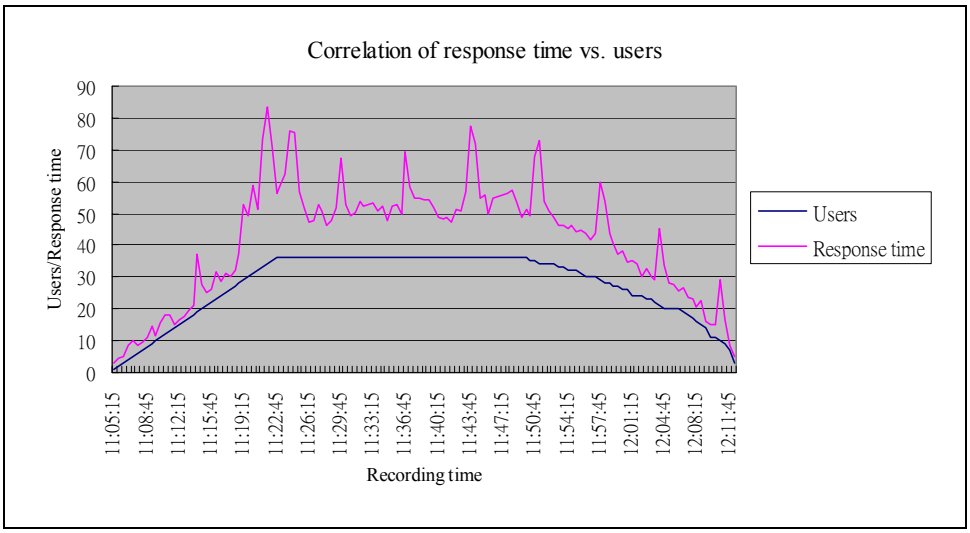


Fig. 3 Comparison of Response Time vs. User Base

The test load is increased at a 30-second interval, using every 30 seconds for grouping the time sequencing data created, starting from the first test account until all test accounts are run through, which took a total of 4,056 seconds. From which, a total of

135 time segments can be derived. The Kolmogorove-Smirov statistical means concluded from the test data include a rating of 0.158189 as shown in below,

Class Groups of values	$F_{actual}(x)$ Actual value	$F_{expected}(x)$ Expected value	$\lvert F_{actual}(x) - F_{expected}(x) \rvert$
0~2	0.000000	0.000000	0.000000
3~4	0.014815	0.000016	0.014799
5~6	0.029630	0.000239	0.029391
7~8	0.096296	0.001974	0.094332
9~10	0.133333	0.010306	0.123027
11~12	0.192593	0.037473	0.155120
13~14	0.251852	0.101467	0.150385
15~16	0.311111	0.215386	0.095725
17~18	0.370370	0.373954	0.003584
19~20	0.451852	0.551197	0.099345
21~22	0.555556	0.713745	**0.158189**
23~24	0.711111	0.838229	0.127118
25~26	0.807407	0.919024	0.111617
27~28	0.881481	0.964025	0.134978
29~30	0.933333	0.985767	0.052434
31~32	0.992593	0.994965	0.002372
32~34	0.992593	0.998401	0.005808
34~	1.000000	1.000000	0.000000

At a significant level where α equals to no more than 0.005, and the abandoned area being $C = \{d_{135} \mid d_{135} \geq 0.16081\}$ as located from the chart, The K-S statistical means of 0.158189 that has not fall within the abandoned area of 0.16081 will not rule out the data's experience distribution as a Poisson Distribution based on the theoretic model that,

$$f(x) = \frac{e^{\lambda} \cdot \lambda^{x}}{x!} \quad , \quad x = 0,1,2,3\text{K}$$

What can be concluded from the theoretic model are,

$$f(x) = \frac{(20.089)^{x} e^{-20.089}}{x!} \quad , \quad x = 0,1,2,3\text{K}$$

The model's average means, λ, has been concluded at 20.089 persons per hit as the test rating, which indicates that the system is able to process 20.089 persons/hit on the online login-logout command; in other words, the average timing of online login-logout per person can be concluded at 30/20.089, or 1.4933 seconds.

Given that the average response time of 1.4933 seconds has been concluded from the test, in conjunction with the company's estimated loading capacity of 168 persons to go online simultaneously, the average response time per person will roughly be at 4.18 minutes when the system is processing a 168 person load. As to whether this response

time falls within a normal range, it will require the company executives to further look into the normal processing time afforded by the system design.

A further examination of the test record files indicate that no excessive variations have been found in the business processing stages of between T3 and T6. To decipher the state of changes in processing time, we further divide the transaction time into two segments – one being the log in phase, meaning that covering T1 to T2, and the other being the business processing phase, meaning T3 to T6, where the sum of both will provide the transaction time of a given test. While plotting a total of 2,715 of the processing time and total transaction time concluded from the two stages as shown in Fig. 4 indicates that the longer the transaction is dragged on, the more ominous the login time, the yellow zone, will become, but variations in the business processing time, the red zone, remains unchanged. This clearly indicates that the system's login processing sequence is ominously reduced of its processing speed when the load increases, an eventual bottle neck when the system is fully streamlined and an area that calls for system developers to further examine it for viable solutions.

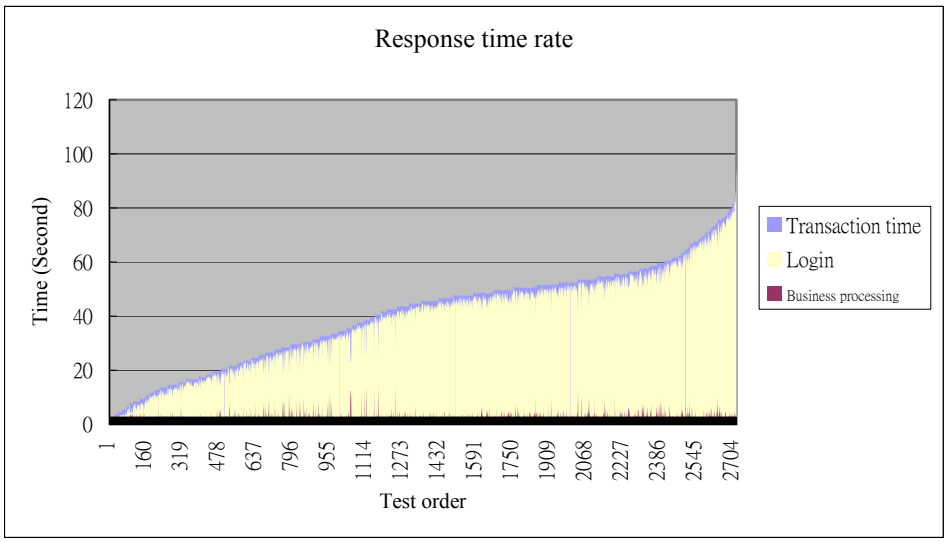


Fig. 4 Proportional ratings of the system's processing time

(4) Improve phase:

The example that a loading test can be completed without being aided by any of the automated testing tools available on the market, however the test objectives were achieved, does allow company executives to review estimated ratings of a response time, while the bottleneck created by the system's login time, between T1 and T2, remains the focal point of improvement in this phase. The system developer has already begun to modify the system framework by adding the number of broker servers, modifying the login module on the client end for an interactive selection of broker servers, switching the

account number validation function to under the application server for execution, to address concerns of bottleneck in system login time.

(5) Control phase:

In this case study the team wanted to be sure that the improvements, once implemented, held value and did not revert to error-riddled baseline. The team modified the test framework to be a monitor to watch the application performance whether or not it satisfies users requirement. The tool for monitoring the application is as shown in Fig 5.

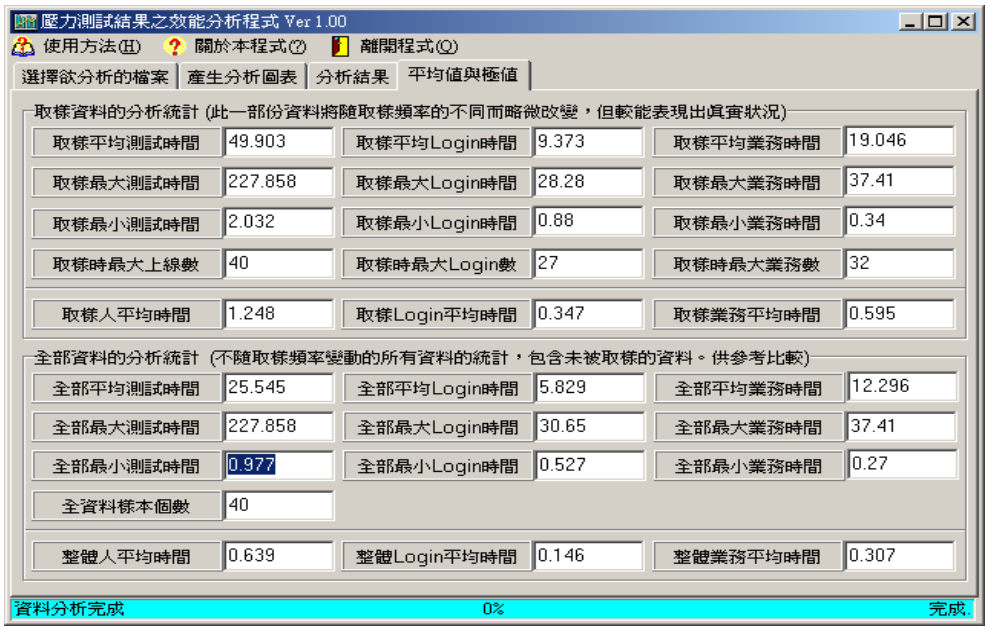


Figure 5: Tool for monitoring the application

5. Conclusion

Management problems in the development of software have been addressed over the last years by a strong focus on the improvement of the development processes. Six Sigma is the most powerful breakthrough management tool ever devised, promising increased market share, cost reductions and dramatic improvements in bottom-line profitability for companies of any size. The method of Six Sigma mapped the software development life cycle for client/server applications against the Six Sigma quality cycle (DMAIC): Define, Measure, Analyze, Improve and Control and then aligned those cycles against the quality management life cycle (PDCA): Plan, Do, Check and Action. The case is shown in this paper to demonstrate that this method not only assisted the development team to focus on reducing development variation, but also gave the rest of the company an understanding of the IT involvement necessary for success.

Acknowledgement

This research is supported by National Science Council, Taiwan, R.O.C. under contract NSC91-2416-H-123-002.

References

1. Chang, S.K. (2001), *Introduction to Function Test on E-commerce Web Sites*, Quality Monthly.
2. Eckes, G. (2001), Making Six Sigma Last—Managing the Balance Between Cultural and Technical Change, John Wiley & Sons, Inc..
3. Fang, S.R. (1999), *Fundamental Statistics*, Hua Tai Publishing, Taiwan.
4. Harrington, H.J. and McNellis, T. (2001), *Six Sigma for Internet Application Development*, Software Quality Professional, 4(1).
5. Harry, M and Schroeder, R. (2000), *Six Sigma—The Breakthrough Management Strategy Revolutionizing the World's Top Corporations,* Random House, Inc..
6. Nguyen, H.Q., (2000), *Testing Application on the Web*, John Wiley & Sons, Inc..
7. Spolverini, M. (2000), *Measuring and Improving the Quality of Web Sites*, 4th International Software Quality Week Europe (QWE2000).

QW2002 Paper 3P2

Mr. Raja Mohapatra & Mr. Bibhash Saha
(Infosys Technologies Ltd)

Effort Estimation for QA Projects Statistical Approaches & Challenges

Key Points

- Typical SDLC model
- Effort estimation of QA projects
- Statistical approach

Presentation Abstract

Effective effort estimation is one of the most challenging and important activities in software project lifecycle. Though there are too many variables human, technical, environmental, political which can affect the ultimate effort applied to develop it, software project estimation can be transformed from a black art to a series of systematic steps that provide estimate with acceptable risk.

There are many popular options available for estimations of projects, which follows SDLC. However in case of projects involving only testing, there is not much of insight to evolve with a proper estimation model. Traditionally estimation of effort for testing has been more of a ballpark percentage of the rest of the development life cycle stages. This approach to estimation cannot be applied for projects, which do not have much insight into the other phases of the SDLC. Also there is no estimation model for projects doing only test planning or scripting for the customers. Hence overall and LC wise estimation for testing related projects is still a challenge.

In this paper we will discuss the step by step approach that we followed at Infosys in evolving a model for effort estimation of such projects using statistical multivariate regression model. We will discuss the typical SDLC for QA projects, challenges faced in estimation, different parameters affecting the estimation, test of significance of each and every parameter and finally finding out the partial regression co-efficient.

This method has been deployed in some projects at Infosys and we will also discuss the outcome.

About the Author

Raja Bibhash has a Bachelor's degree in Electrical Engineering from Calcutta University. Prior to joining Infosys he was working with Larsen and Tourbo Ltd, India as an Executive and involved into Engineering, project planning & Quality Assurance activities for Manufacturing and process industry. He was also involved there in ISO 9001 initiative and SAP implementation. In Infosys Bibhash is working as a Software Quality Advisor and extensively involved in Process deployment, Metric Analysis, Statistical Process Control and various improvement initiatives in Infosys.

Raja Mohapatra has a Master's degree in Mechanical Engineering from University of Roorkee. Prior to joining Infosys he was working with Reliance Industries Ltd, India as an Asst Manager- Inspection, involved in Quality Assurance activities, subcontracting management, maintenance planning and monitoring in various plants like power, process and fibre. He was also involved there in ISO 9001 initiative and SAP implementation. In Infosys Raja is working as a Software Quality Advisor and extensively involved in Process deployment, Metric Analysis, Statistical Process Control and various improvement initiatives in Infosys.

Effort Estimation for QA (Testing) projects Statistical Approaches & Challenges

By
Bibhash Saha & Raja Mohapatra
(Infosys Tech Ltd., India)

Why Talk about this subject ??

- Estimation of effort required is first step to any project and software projects are no exception.
- Effort estimation in software projects is most challenging as it depends on so many variables which are different every time.

Traditionally how effort estimation is done?

Top down Approach

- First the size of the software is arrived at.
  - Counting FP and Use Cases are some ways
- Then the total project effort is calculated.
  - Past productivity data is used to arrive at this figure

Bottom up Approach

- Effort required for build is calculated first.
  - Estimating the number of programs and giving them some weight-age.

- Efforts required for different stages are then arrived at.
  - "past effort distribution" data is used for this.

What is a QA (Testing) project ?

- The projects which does only the system, integration or acceptance testing on behalf of the client.
- The code is developed or enhanced by some other vendors.

What is the difficulty in estimating these kind of projects ?

- The total size of the software is not known.
- Effort spent in other LC stages is not known.
- As never estimated past data was not available.

What to do then ?

Following approach was taken…

- Understanding and analyzing the exact Life Cycle stages.
- Understanding the different scenarios and problems faced.
- Defining the metrics to be collected as a basis for estimation.
- Analyzing the collected metrics.
- Deriving an estimation model.
- Validating and fine-tuning the model with further data points.

Life cycle of typical QA project

The QA (Testing) projects under consideration typically follow the following LC stages.

- Preparing master test strategy.
- Test case scripting.
- Manual testing
  - Sanity testing or Regression testing
- Preparing test result summary

Life cycle of typical QA project

The project gets following three kinds of requests depending on the customer requirements and contract.

- Test planning which involves preparation of test scripts only.
- Test execution where customer supplies test scripts.
- Test planning and test execution.

As majority of the requests received are of first two kinds, for the sake of uniformity the third kind of request were considered as combination of **"Test Planning"** and **"Test Execution"** requests.

Challenges in Estimation

- There are many different platforms under which the project has to work which affects the consistency of productivity.
- Movement of manpower across platforms also contributes to inconsistent productivity.
- High attrition of man power as team size is very large. Hence human factor is a variable.
- Effort requirement in test execution is largely dependent upon the interdependency of the test cases and success of the program being tested.

Definition and Collection of Metrics

- After extensive brain storming some metrics were decided to be collected along with the actual effort taken for any request.
- Metrics were grouped into three categories
  - Generic
  - Application specific
  - People related

Example

- Number of scripts to be prepared
- Number of impacted screens
- Number of validations
- Number of impacted programs
- Skill level of the team members
- Type of database update
- Clarity of the test requirement specification

Data Segregation

- Some of the metrics defined were not available for all requests and hence discarded for further analysis.
- There are two types of variables
  - Quantitative
  - Qualitative

Example

- Number of scripts to be prepared
- Number of impacted screens
- Number of validations
- Number of impacted programs

Example

- Skill level of the team members
- Type of database update
- Clarity of the test requirement specification

The quantitative variables were used to fit the regression model where the qualitative ones were used to fine tune the model to minimize the error further.

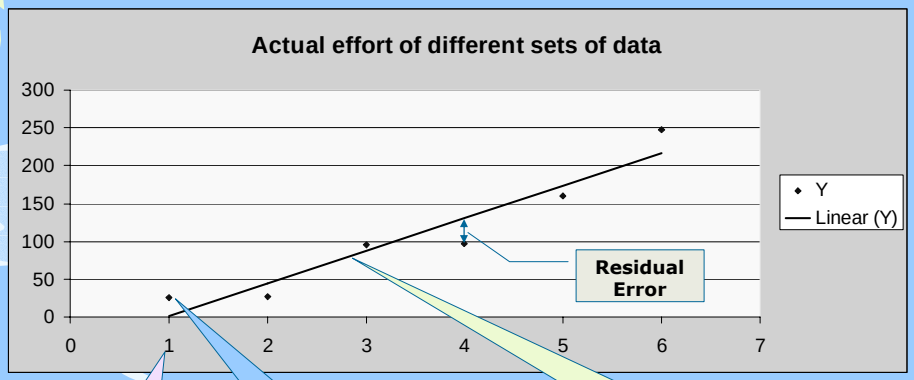
Typical Collected Metrics

Set No	m	n	p	q	Y (p.Hrs)
1	11	12	330	39	247
2	1	11	4	11	96
3	1	13	3	9	97
4	16	12	50	37	160
5	1	14	12	2	27
6	2	13	3	3	26
7	-	-	-	-	-

Where

m = Number of scripts to be prepared
n = Number of impacted screens
p = Number of validations
q = Number of impacted programs
Y = Actual Effort

Scatter Diagram for different sets of Data



One set of data (m1, n1, p1, q1)

Corresponding Actual effort

Best fit line equation of which will be found out from Multivariate Regression Method

Evolution of the Multivariate Regression Model (1)

How the regression equation looks like ?

$$\text{Estimated Effort (Y)}_{\text{unadjusted}} = \beta_0 + \beta_1 m + \beta_2 n + \beta_3 p + \beta_4 q + e$$

Where

- m = Number of scripts to be prepared
- n = Number of impacted screens
- p = Number of validations
- q = Number of impacted programs
- Y = Actual Effort

Y-intercept

Partial regression co-efficient of variable 'n'

Residual error
(Distributed normally with mean at zero and standard deviation σ^2)

Evolution of the Multivariate Regression Model (2)

To best fit the estimation:

- So we should have **(k+1)** data points where **k** is the no. of variables.
- The least square method will be adopted to minimize the residual error.
- Partial differentiation of the sum of the squared residual error Σe^2 with respect to each partial regression coefficients $\beta_0, \beta_1, \beta_2 \ldots$ will be equated to zero.
- Thus **(k+1)** number of linear equations will be obtained where **k** is the no. of variables.
- Then the linear equations are to be solved simultaneously to get individual partial regression coefficients and the y-intercept

Evolution of the Multivariate Regression Model (3)

Going back to the example we are following:

The solved regression equation looks like

$$Y = (- 74.0 - 13.2\ m + 7.14\ n + 0.0023\ p + 9.74\ q)$$

Not to worry !!!! There are so many tools available in market to do this whole job on just a click of mouse.

What next ??

It is important now to understand the effect and strength of each and every predictor variable on the response variable.

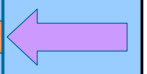
How to do this ?

- Compare the partial regression coefficients.
- Compare the ratio (t-value) of partial regression coefficient and standard error with the pre-selected alpha level.

If the selected confidence limit is 95% then the alpha level will be (1-0.95)=0.05. The t-value (without signs) should be more than the alpha level

Example

Predictor Variable	Partial Regression coefficient	't'-value
Constant	-73.96	-0.92
m	-13.24	-4.84
n	7.14	1.20
P	0.00229	0.03
q	9.743	6.56



The final formula

- The **'t' value** for predictor variable **'p'** is less than **0.05** (selected 'α' level of confidence). Also the co-efficient of regression is very insignificant as compared to others. Hence variable **'p'** was discarded from the final formula.
- The equation was **solved again** after discarding the variable 'p' and the new formula is as follows

$$Y = (- 75.8 - 13.3\ m + 7.27\ n + 9.79\ q)$$

- The **co-efficient of determination (R2)** is as high as 99.8 % showing the equation is a good fit.

Can any software estimation follow any such formula?

Other such questions and doubts

My project is totally different and the kind of requests we execute is different every time.

Do you mean to say project manager does not have any flexibility to decide?

The team changes so frequently that any formula cannot accommodate that.

A lot depends on the clarity of the requirement. Otherwise there are so many issues and they take all the time to get resolved.

Giving flexibility and taking care of Qualitative Variables

- Hence to take care of the effect of so many qualitative variables the effort thus calculated from the regression equation can adjusted as demonstrated below.

Y (adjusted) = Y (from regression equation) x C

Where

C = C1 x C2 x C3 x …

C1, C2, C3…= Weightage given to different qualitative variables in a scale of 0 to 2 (exclusive of the boundary values)

What the projects reaped out of this ?

- It helped to understand the productivity of the team and thus the resource loading. This leads to clear resource allocation methodology.
- Made us able to estimate defects in test plans and thus in controlling the review process.
- And finally to track and report the progress.

The road map ahead

- The regression model may be refitted as and when required depending on the changing scenario.
- The estimation for test execution has a drawback in case of regression testing when the effort for regression is appreciable. Re-estimating every time at the beginning of regression testing may be a solution but an overhead.

Thanks A Lot for Patient Listening

Effort Estimation for QA (Testing) projects
– Statistical Approaches & Challenges

By Bibhash Saha & Raja Mohapatra (Infosys Technologies Limited, India)

1.0 INTRODUCTION:

Effective effort estimation is one of the most challenging and important activities in software project lifecycle. Though there are too many variables – human, technical, environmental, political which can affect the ultimate effort required to develop it, software project estimation can be transformed from a black art to a series of systematic steps that provide estimate with acceptable risk and deviation. On time delivery cannot be achieved without a proper effort estimate.

There are many popular options available for estimations of projects, which follows any Software Development Life Cycle. However in case of projects involving only testing, there is not much insight to evolve with a proper estimation model. Traditionally estimation of effort for testing has been more of a ballpark percentage of the rest of the development life cycle stages. This approach to estimation cannot be applied for projects, which do not have much idea about the other phases of the Life Cycle. Also there is no estimation model for projects doing only test planning or scripting for the customers. Hence overall and life cycle wise estimation for testing related projects is still a challenge.

2.0 APPROACH:

The following step-by-step approach needs to be followed in developing the estimation model for QA (Testing) projects.

- Understanding and analyzing the exact Life Cycle stages
- Understanding the different scenarios and problems faced
- Defining the metrics to be collected as a basis for estimation
- Analyzing the collected metrics
- Deriving an estimation model
- Validating and fine-tuning the model with further data points.

2.1 PROJECT OVERVIEW:

2.1.1 SDLC Overview

The QA (Testing) projects under consideration typically follow the following LC stages.

1. Preparing master test strategy.
2. Test case scripting.
3. Manual testing
    a. Sanity testing or Regression testing
4. Preparing test result summary

2.1.2 Different Kinds of Request

The project gets following three kinds of requests depending on the customer requirements and contract.

1. Test planning which involves preparation of test scripts only (step 2 of LC only).
2. Test execution where customer supplies test scripts (step 3 and 4)
3. Test planning and test execution (all steps mentioned above)

As majority of the requests received are of first two kinds, for the sake of uniformity the third kind of request were considered as combination of “Test Planning” and “Test Execution” requests.

2.2 PROJECT CHALLENGES:

Following are some of the challenges in estimation keeping in view the nature of request, project cross section, target set by customers and variety of platforms etc.

- As the project either gets or breaks up the requests into two separate requests as mentioned above, there was a need for separate estimation for test planning and test execution.
- There are many different platforms under which the project has to work which affects the consistency of productivity.

- Movement of manpower across platforms also contributes to inconsistent productivity.
- High attrition of man power as team size is very large.
- Effort requirement in test execution is largely dependent upon the interdependency of the test cases and success of the program being tested.

2.3 MEASUREMENTS AND ANALYSIS:

After extensive brain storming with various cross sections of people, the project team collected various metrics, though at that stage the impact of various variables on the actual effort was not very clear.

Metrics were grouped into three categories like generic, application specific and people related. Separate metrics were defined for test planning and test execution type of requests. Typical metrics defined are number of test scripts, number of screens, type of database update, skill set of people etc.

The data was collected for 20 varied kinds of requests and analysis was carried out to understand the dependency of each parameter on the actual effort. Following are the observations.

- Some of the metrics defined were not available for all requests and hence discarded for further analysis.
- There are two types of variables; quantitative and qualitative. The quantitative variables were used to fit the regression model where the qualitative ones were used to fine tune the model to minimize the error further.

2.4 EVOLUTION OF THE MODEL:

2.4.1 Regression Model

In this kind of scenario, multivariate regression model was chosen to be used. The regression equation can be written as follows.

$$Y = ß_0 + ß_1X_1 + ß_2X_2 + \ldots\ldots\ldots + ß_kX_k + e$$

Where

Y=the response variable i.e. the estimated effort
$X_1, X_2, \ldots X_k$= Predictor variables.
$ß_0$= Constant Y intercept of the best fitted line
$ß_1, ß_2, \ldots ß_k$ = partial regression co-efficient of each predictor variable
e= Residual error distributed normally with mean zero and variance s^2

To minimize the residual error, the partial differentiation of **e** w.r.t $ß_0, ß_1, ß_2 \ldots$ etc was taken and the addition was equated with zero. Number of composite samples was taken to solve the equation thus formed.

After this it was important to understand the effect and strength of each and every predictor variable on the response variable. To test the effect, significance test of each variable was done by using null hypothesis and the strength was compared by comparing the partial regression co-efficients and looking at the variance of regression.

One more way to check the significance of each variable is to compare the t-values (ratio of partial regression co-efficient and the standard error in the co-efficient) with the pre-decided confidence limit.

Based on the above tests variables having less effect and strength were discarded and the regression curve was refitted.

All these are basic principles and one need not dig into theories for solving the equation traditionally as there are so many tools available in the market to help.

The same exercise was done both for test planning and test execution to come out with different effort regression equations.

2.4.2 Fine tuning the model

Once the regression equation is evolved using quantitative parameters, it was required to fine tune the model further using the various qualitative parameters like nature of job, skill set of people, database update etc. Numerical weightage to various variables were given based on past experience and interviewing people across various cross sections. A typical example of fine tuning is as follows

$$Y \text{ (adjusted)} = Y \text{ (from regression equation)} \times C$$

Where
C = c1 x c2 x c3……
c1, c2, c3 … = Constant depending on qualitative variables.

3.0 EXAMPLE:

As an example to further explain the concept, for a typical test planning kind of request the selected quantitative predictor variables were as follows

- Number of scripts to be prepared **(m)**
- Number of impacted screens **(n)**
- Number of validations **(p)**
- Number of impacted programs **(q)**

Now fitting the regression equation for the unadjusted value of effort

$$\text{Estimated Effort } (Y)_{unadjusted} = ß_0 + ß_1 m + ß_2 n + ß_3 p + ß_4 q + e$$

The sample input set of data from various requests are as follows

m	n	p	q	Y (p.Hrs)
11	12	330	39	247
1	11	4	11	96
1	13	3	9	97
16	12	50	37	160
1	14	12	2	27
2	13	3	3	26
-	-	-	-	-

(N.B.: These are only indicative data for explanation purpose)

Now solving the above equation as described in Sec 2.4.1 aiming for minimum residual error we got the following regression equation.

$$Y = (- 74.0 - 13.2\ m + 7.14\ n + 0.0023\ p + 9.74\ q)$$

Then test of significance was done for each of the variables and the 't' values for each variable is as follows.

Predictor	Coef of Regression	T
Constant	-73.96	-0.92
m	-13.244	-4.84
n	7.141	1.20
p	**0.00229**	**0.03**
q	9.743	6.56

As evident from the above data, the 't' value for predictor variable 'p' is less than 0.05 (selected 'a' level of confidence). Also the co-efficient of regression is very insignificant as compared to others. Hence variable 'p' was discarded from the final formula.

The equation was solved again after discarding the variable 'p' and the new formula is as follows

$$Y = (- 75.8 - 13.3\ m + 7.27\ n + 9.79\ q)$$

The **co-efficient of determination (R^2)** is as high as 99.8 % showing the equation is a good fit.

Some of the qualitative variables that also affect the estimated effort are as follows

- Skill level of the team members
- Type of database update
- Clarity of the test requirement specification

Hence to take care of the above effect the effort thus calculated from the regression equation was adjusted as demonstrated below.

$$Y\ (adjusted) = Y\ (from\ regression\ equation) \times C$$

Where **C = C1 x C2 x C3** and

Constant	Constant depending On	High	Medium	Low
C1	Skill level of team	1.05	1.00	0.95
C2	Type of database update	1.05 (Batch + Online update)	1.00 (Batch Update)	0.95 (Online update)
C3	Clarity of requirement spec	1.05	1.00	0.95

These qualitative parameters will adjust the estimated effort within ± 15 %.

4.0 BENEFITS:

Earlier the project was unable to estimate the effort and there was no choice other than going by the customer's schedule leading to the following problems

- Unable to understand the productivity of the team and thus the resource loading. This leads to unclear resource allocation methodology.
- Unable to estimate defects in test plans and controlling the review process.
- Unable to track the progress etc.

Now with the evolved model, above problems were solved. The project could estimate its effort with an accuracy of ± 20 %.

5.0 ROAD MAP AHEAD:

- The regression model may be refitted as and when required depending on the changing scenario.
- The estimation for test execution has a drawback in case of regression testing when the effort for regression is appreciable. Re-estimating every time at the beginning of regression testing may be a solution but an overhead.

6.0 CONCLUSION:

The model holds good for any project and the parameters can be very well standardized across the organization and constants can be fine tuned to suit the project specific needs. The effort estimation based on multivariable regression model is more convincing than all other conventional models of giving weightage to various parameters and finally arriving with the effort. Same model can be extended to other life cycle stages of software development also.

Author's Profile

Bibhash Saha

Bibhash has a Bachelor's degree in Electrical Engineering from Calcutta University. Prior to joining Infosys he was working with Larsen and Tourbo Ltd, India as an Executive and involved into Engineering, project planning & Quality Assurance activities for Manufacturing and process industry. He was also involved there in ISO 9001 initiative and SAP implementation. In Infosys Bibhash is working as a Software Quality Advisor and extensively involved in Process deployment, Metric Analysis, Statistical Process Control and various improvement initiatives in Infosys.

Raja Mohapatra

Raja has a Master's degree in Mechanical Engineering from University of Roorkee. Prior to joining Infosys he was working with Reliance Industries Ltd, India as an Asst Manager-Inspection, involved in Quality Assurance activities, subcontracting management, maintenance planning and monitoring in various plants like power, process and fibre. He was also involved there in ISO 9001 initiative and SAP implementation. In Infosys Raja is working as a Software Quality Advisor and extensively involved in Process deployment, Metric Analysis, Statistical Process Control and various improvement initiatives in Infosys

-- EOD --

This document was created with Win2PDF available at http://www.daneprairie.com.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.

QW2002 Paper 4P1

Joan Bosch Sole
(NTE, s a.)

Big And Complex Projects: Beyond Extreme Programming

Key Points

- We should find a development process suitable to our environment.
- Extreme Programming offers always good points to think about.
- Some XP techniques can be harmful in large projects.

Presentation Abstract

This paper explains the software development process chosen by NTE in case of large projects and compares it with the so-called Extreme Programming. The main conclusion is that the rules of the Extreme Programming are also valid in large projects.

About the Author

Joan Bosch was born in Barcelona (Spain) in 1971. He has a degree in Physics but he has been working in Computer Science since 1988. He has mainly worked in Civil Engineering and Medical Device applications. He has actively collaborated with the Universitat Central de Barcelona as a researcher, applying neural networks to economy. From the University he moved to NTE (New Space Technologies, develops Software for Space and Medical application); he entered as SW senior engineer, right now he is the responsible of the Software Quality Construction Department.

Large projects: beyond Extreme Programming

NTE, s.a. / Joan Bosch
joanb@nte.es

Prejudice about XP

- Extreme Programming may sound to you as:
  - Fast development.
  - Neither analysis nor design.
  - Small projects.
  - Type code first... And later we'll figure out what we have to do with it!

Who is NTE?

- NTE started as a company devoted to the aerospatial market.
- Her main customers were NASA and the European Space Agency.
- Later on a SW division was started to develop applications for medical devices.

The NTE SW target process

- The NTE development process must be compatible with:
  - Reliability and robustness
  - Maintainability
  - Flexibility in front of changes in requirements.

The TOP project

- In 1999 NTE was asked to develop the SW to control a high throughput coagulation analyzer.
- The starting point was an existing HW and thousands of requirements written as atomic sentences.
- Two years later, the SW is ready for beta testing without important flaws.

NTE & XP

- The following of the presentation tries to answer the questions:
  - How is the development process/practices followed in this project?
  - Is this process compatible with XP?

Requirements analysis practices (I).

- Identify user typologies.
- Write Use Cases to understand what the application is supposed to do.
- Usability test on a prototype for the most used areas.
- Complete conceptual model of the user's world.
- Data dictionary.
- Screen dictionary.

Requirements analysis practices (and II)

- After the functional analysis, reword all the requirements in traceable atomic sentences.
- Set up a Change Control Board and select a Product Manager.
- "The developer proposes, the Product Manager chooses"

Design practices

- Tight mapping between user concepts and implementation classes.
- Public reviews of design.
- Classes packaged in subsystems.
- Early definition of all interfaces between subsystems.

Development practices (I)

- Monitoring the project status according to implemented feature.
- Unit test at subsystem level.
- Continuous integration.
- Use of coverage tools.

Development practices (II) – Unit Test

- All subsystems have unit test.
- The unit test must be:
  - Automatic.
  - Regressive.
  - Incremental.
  - White Box.

Development practices (III) – Unit Test

- Unit Test benefits:
  - Early bug detection.
  - Avoid introducing bugs in old functionality.
  - Developers are fully independent.
  - Easier the transfer of responsibility of a subsystem between developers.
  - Easier the use of code coverage tools.

Development practices (IV) – Continuous Integration

- The application must compile, link, and pass a basic functional test, at any time.
- Benefits:
  - Early detection of integration bugs.
  - Allow the developers to test the implemented functionality in an informal way at any time.
  - Allows the PM to monitor the status of the project at any time.

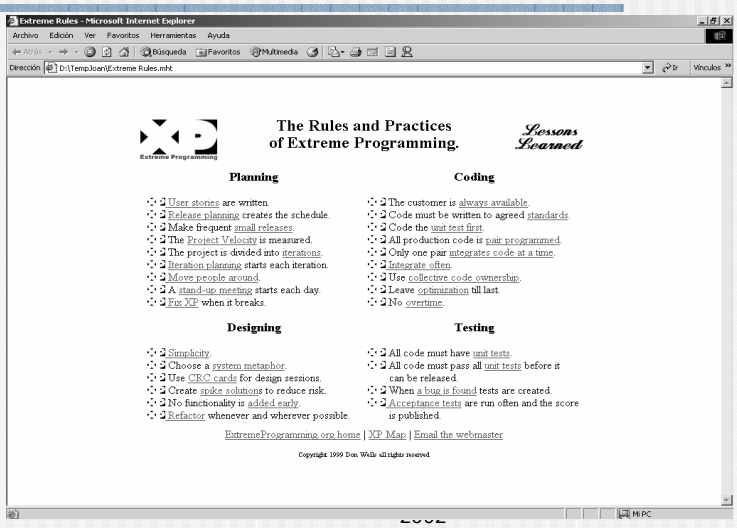
Development practices (and V) – Code Coverage Tools

- A coverage tool allows to know which lines of code have been exercised by the Unit Test.
- Benefits:
  - Provides an idea of the exhaustiveness of the Unit Test.
  - Detection of dead code.
  - Detection of a requirements misinterpretations when some code impossible to be exercised is found.
  - Detection of developers overwork.

QA Practices

- An independent QA team reviewed:
  - Specifications.
  - Design.
  - Code.
- Automatic system test.
- Continuous qualification test.

The XP rules



NTE vs. XP: shared rules (I)

- Make frequent small releases.
- The project is divided into iterations.
- Iteration planning starts each iteration.
- Symplicity.
- Create spike solutions to reduce risk.
- Refactor whenever and wherever possible.

NTE vs. XP: shared rules (and II)

- Code must be written to agreed standards.
- Leave optimization till last.
- All code must have unit tests.
- All code must pass all unit tests before it can be released.
- When a bug is found tests are created.
- Acceptance test are run often and score is published.

NTE vs. XP: compatible rules (I)

- User stories are written.
- Release planning creates the schedule.
- The Project Velocity is measured.
- A stand-up meeting starts each day.
- Fix XP when it breaks.
- Choose a system metaphor.
- Use CRC cards for design sessions.

NTE vs. XP: compatible rules (and II)

- No functionality is added early.
- Code the unit test first.
- Only one pair integrates code at a time.
- No overtime.

NTE vs. XP: non-compatible rules

- Move people around.
- All production code is pair programmed.
- Use collective code ownership.

Non-compatible rules analysis.

- The high level of specialization of the NTE projects works against:
  - Move people around.
  - Use collective code ownership.
- Pair programming has never been tried, at least in a formal way, due to the NTE company culture.

Conclusions

- NTE and XP practices are highly compatible.
- There are some differences due to:
  - The high level of specialization of her projects.
  - NTE company culture.

QW2002 Paper 4P2

Mr. Jim Hazen
(SysTest Labs)

'Excel'erating Test Status Reporting

Key Points

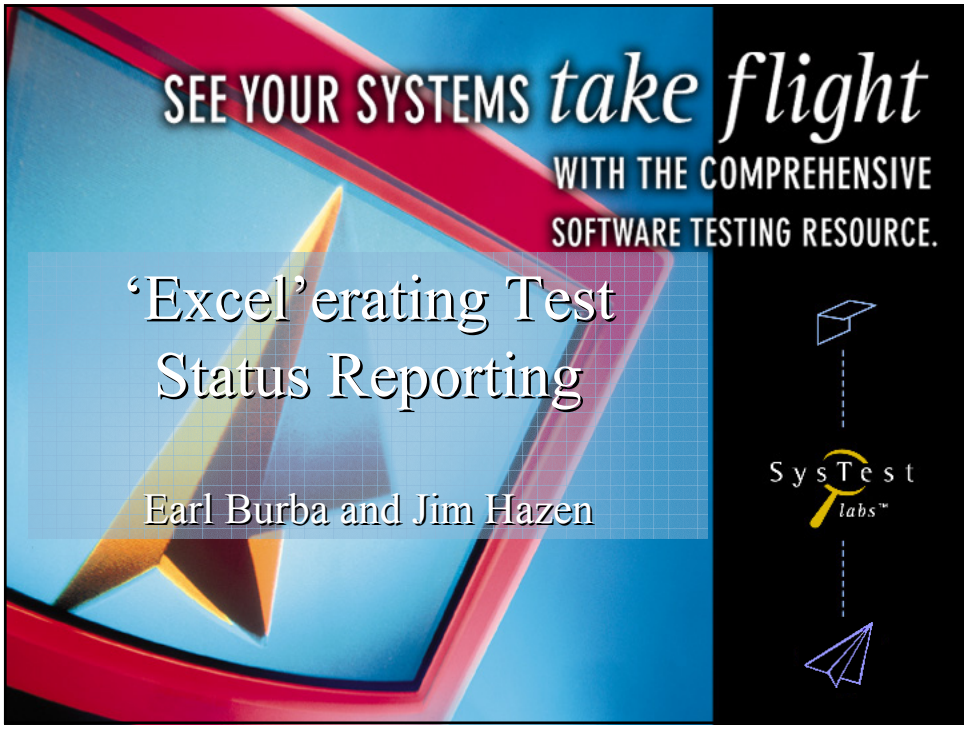
- How to use Excel as a tool to manage and report test status
- How to build the worksheets and reports
- Benefits of using the tool for test status reporting

Presentation Abstract

One tough question to answer as a tester is when will testing be done. A tool is needed to report on status. An effective alternate solution to a commercial tool is available using Microsoft Excel. Using specific functions and linked worksheets a test status a report tool can be built.

About the Author

Jim Hazen has 15 years experience in software development and testing. Jim is a Test Engineer with SysTest Labs in Denver, Colorado. Jim has worked in various lead positions in his career and has worked with a wide variety of applications on DOS, OS/2, Windows, and Web environments.



SysTest labs™

Introduction

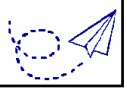
- Earl Burba
  - Sr. Project Manager
  - Twenty years of software development and testing experience
  - Experience with Windows, Web, Unix, & Embedded environments and applications
  - Software Test Evaluation Process Tool Patent
- Jim Hazen, CSTE
  - Test Engineer
  - Fifteen years of software testing experience
  - Experience with DOS, OS/2, Windows, & Web environments and applications
  - Presenter at STARWest 2001

Introduction cont.

- SysTest Labs, LLC
  - Independent 3rd Party Test Lab
  - Founded in 1990, QA Test lab opened in 1996
  - Projects range from Desktop applications to Client/Server to Web to WAP
  - FEC/NASED accredited Independent Test Authority (ITA) for voting systems
  - Clients range from Fortune 100 to small development companies
  - Strategic partners
    - IBM and Microsoft
    - Compuware, Segue,RadView, and Rational

Objectives

- Focus of Presentation
  - How to use Excel to develop a test status reporting tool
- Items to be discussed
  - Why build a tool with Excel
  - Features and functions used in Excel
  - Test system architecture
  - Test Case & Status worksheet layouts
  - Report Generation
  - Benefits of system
  - Lessons learned from development and use

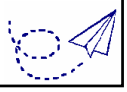


Why use Excel

- Readily available, part of MS-Office
  - Can be tied into other tools (Word)
- Relatively inexpensive to implement
  - Alternative to expensive commercial tools
- Flexible
  - Can be customized to fit needs

Features of Excel

- Formula functions and logic constructs
  - Example: Countif and IF
    - = COUNTIF(D13:D38,"x") – Count steps passed in test case
    - =IF(E7 = 0,"",IF(G7>0,"Failed",IF(D7 = F7, "Passed", "Pending"))) – determine if test case passed on status worksheet
- Linking cells and worksheets/workbooks
  - Allows for consolidation of data for analysis and reporting
- Statistical data collection
- Presentation graphs and charts

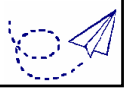


When & How to Implement the Tool

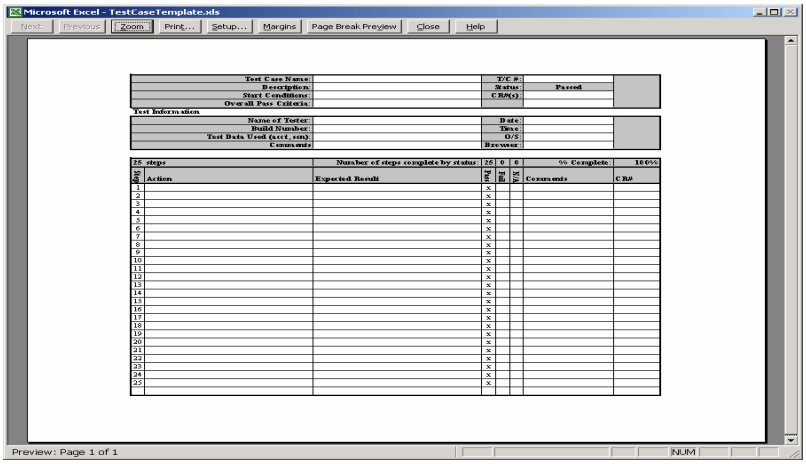
- Why implement the tool
  - Allow for ease of status reporting
  - Allow for higher accuracy in reporting
- When to implement the tool
  - As early in project lifecycle as possible, at least by time the test plan is completed.
- Initial effort to implement
  - Up front time can be high. Have to build the worksheets and formulas first then link them together. Reports and charts can be built later.

Test Status Reporting System Architecture

- Components of a testing system
  - Test Cases (core component)
    - Test Case data
    - Test Case history
  - Test Case Status Report (core component)
    - Test Case Summary Report (Functional effort)
    - Master Summary Report (Overall effort)
  - Requirements Traceability Matrix (RTM)
    - Can be added in if time allows



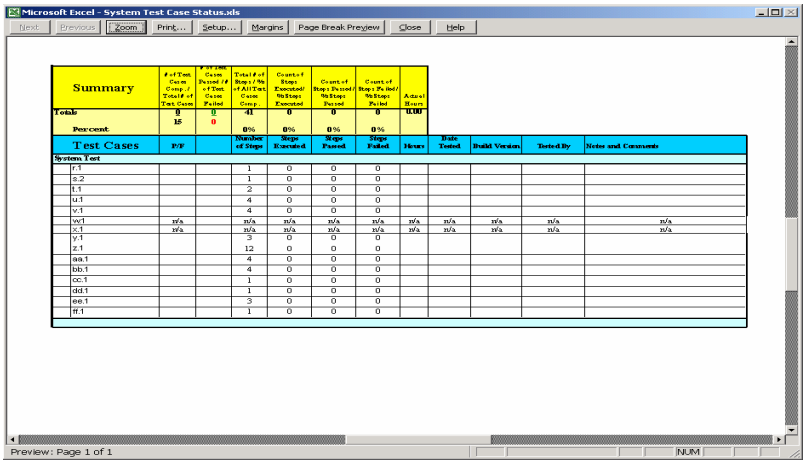
Test Case Worksheet Layout



Test Case Worksheet Formulas

- Example calculated field formulas are:
  - **Status**: = IF(D11 + E11 + F11 = 0, "Not Started", IF(A11 = D11 + F11, "Passed", IF(A11 <> D11 + E11 + F11, "Not Complete", IF(E11 > 0, "Failed"))))
  - **# of Steps**: = COUNTA(A13:A38)
  - **Number of steps complete by status**: = COUNTIF(D13:D38,"x") for Pass, = COUNTIF(E13:E38,"x") for Fail, and = COUNTIF(F13:F38,"x") for N/A
  - **% Complete**: = (D11 + E11 + F11) / A11

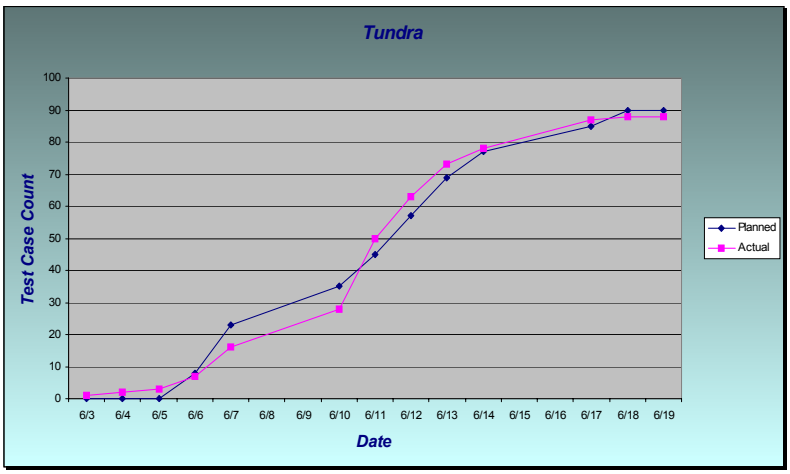
Status Worksheet Layout



Status Worksheet Formulas

- Example calculated field formulas are:
  - **# of Test Cases Comp.**: = COUNTIF(C7:C9,"Passed") +COUNTIF(C6:C9,"Failed")
  - **Total # of Test Cases**: =COUNTA(A7:A7)+COUNTA(A8:A8)
  - **# of Test Cases Passed**: =COUNTIF(C7:C8,"Passed")
  - **Passed/Failed**: =IF(E7 = 0,"",IF(G7>0,"Failed",IF(D7 = F7, "Passed", "Pending")))
- Example test case linked field formulas are:
  - **Steps Executed**: ='[TestCaseTemplate.xls]IE5.x'!$D$11 + '[TestCaseTemplate.xls]IE5.x'!$E$11 + '[TestCaseTemplate.xls]IE5.x'!$F$11

Status Report Chart



Tying it All Together

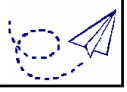
- Links can be formed between spreadsheets to share information
- Statistics are formulated automatically from the available data
- No active participation from the tester to provide information to the status tracking system

Report Generation

- Daily / overall project status
  - Cut and paste from the Status Reporting System
  - Link into other applications or documents via OLE.
- Trend analysis and reporting
  - Master Status Report can provide trending view
  - Different levels of detail can be presented based on management needs

Benefits

- How it saves time and money
  - Most businesses use Microsoft Office or have access to spreadsheets with similar functionality
  - For very little capital expenditure a working test case status reporting system can be implemented
- Higher accuracy of reporting test status
  - No direct tester interaction, automatically generated from test cases
  - Automatically generated reports, graphs etc.
  - Individual or project progress can be tracked
  - Easily incorporated into daily and project reports
  - Status of testing is readily apparent and estimates of completion can be made

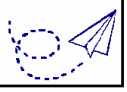


Working Model Presentation

- Demonstration of working model
  - Test Cases
    - Test Case format
    - Test Case formulas
  - Test Case Status Report
    - Test Case Summary Report
      - Format and formulas
    - Master Summary Report
      - Format and formulas

Lessons Learned

- The system does take some time to initially set up
- Cells that are critical for statistical analysis must not be moved or changed (formulas and position are static)



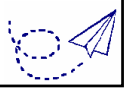
Summary

- Up to the minute status aids in management knowledge of testing progress
- A low cost tool can be easily created to track testing status
  - Particularly during the crunch time of testing

Contact Information

SysTest Labs
1630 Welton Street, Suite 500
Denver CO, 80202
(303) 575-6881
www.SysTest.com

Earl Burba - eburba@systest.com
Jim Hazen - jhazen@systest.com



‘Excel’erating Test Status Reporting

Jim Hazen & Earl Burba, SysTest Labs
303.575.6881 jhazen@systest.com eburba@systest.com

Extended Abstract:

This presentation will show how to implement a test status reporting tool and strategy using Microsoft Excel. The presentation will discuss how to use Excel as a tool to develop, manage, and report on testing status for the testing effort during a project. It will show how to build worksheets for test cases, summary reports, and status reports. It will show which of the features, and how to use them, in Excel to develop the test status reporting tool. The presentation will discuss the benefits of effective test status reporting and how Excel supports it. Also, when the tool needs to be built and implemented will be discussed. Attendees will come away with a working model for their use.

1.0 Introduction

As a tester you are often asked how far along is the testing effort and when will it be done. This is one of the tougher questions to answer, and usually the most nerve racking. Especially when the testing effort for a project is just starting up, or close to being finished. A process is needed to help gather information and effectively report on this item. The problem is that a lot of companies cannot afford a complex commercial tool due to financial reasons, or time constraints to evaluate and implement the tool.

A solution is available using commercial spreadsheet products, specifically Microsoft Excel. Using the logic and formula functions along with a combination of linked worksheets an easy to implement and use test status report tool can be built. This paper will discuss the following:

- Microsoft Excel and the features / functions used for system
- When and how to implement the tool / system
- Test Status reporting system architecture
- Test Case worksheet layout and formulas
- Test Status worksheet layout and formulas
- Report generation
- Benefits of using the tool for test status reporting
- Time needed to build and maintain the system
- Lessons learned using the tool

1.1 Microsoft Excel features and functions

Microsoft Excel is a very robust spreadsheet application. Its numerous features and functions allow the user to build simple to complex calculation & query formulas within it. This allows a user to gather and analyze data from anywhere. These functions and features are the building blocks for the test reporting system.

The function used most often is **COUNTIF**. It counts the number of cells within a range that meet a given criteria. The format is: COUNTIF(D13:D38,"x"); where D13:D38 is the range and "x" is the criteria to count. In the Test Case worksheet, which we will discuss in detail later, the COUNTIF function is used to count the number of test steps that have a specific status (PASS, FAIL, N/A). This in turn feeds into another calculation to determine if the Test Case itself is complete or incomplete, and if it passed or failed. This information is then fed into another worksheet / workbook that calculates and reports on the test status progress.

Another function used extensively is the **IF** logic construct. IF is used to conduct conditional tests on values and formulas. Returns one value if a condition you specify evaluates to TRUE and another value if it evaluates to FALSE. The format is: IF(logical_test,value_if_true,value_if_false). An example of the statement is: IF(D11 + E11 + F11 = 0, "Not Started", IF(A11 = D11 + F11,"Passed", IF(A11 <> D11 + E11 + F11, "Not Complete", IF(E11 > 0,"Failed")))). This compound IF statement is used in the Test Case worksheet to determine the status of the test (Passed, Failed, Not Started, and Not Complete). Again this information feeds into the report analysis / generation worksheet.

The method used to tie the worksheets together is '**linking**'. The formula that is created uses the link reference to the data to be collected. This formula will contain a reference to the workbook (.xls file), the worksheet, and cell. An example of the statement is: [TestCaseTemplate.xls]IE5.x!$A$11; where "[TestCaseTemplate.xls]" is the workbook file, "IE5.x" is the worksheet name, and "$A$11" is the cell to reference. The exclamation point (!) separates the worksheet name from the cell range referred to. Excel can automatically build this link for you in just a few steps (look up "Create a formula to calculate data on another worksheet or workbook" in Help for the procedure).

Finally, the typical mathematical operators (*, +, -, /) along with the other functions (SUM, MIN, MAX, etc.) used are to calculate the various statistics for reports.

Basically it is only a few functions used in Excel that make the whole thing work. Sounds pretty simple and straight forward? It is, and that is the beauty of the system.

2.0 When and how to implement the system

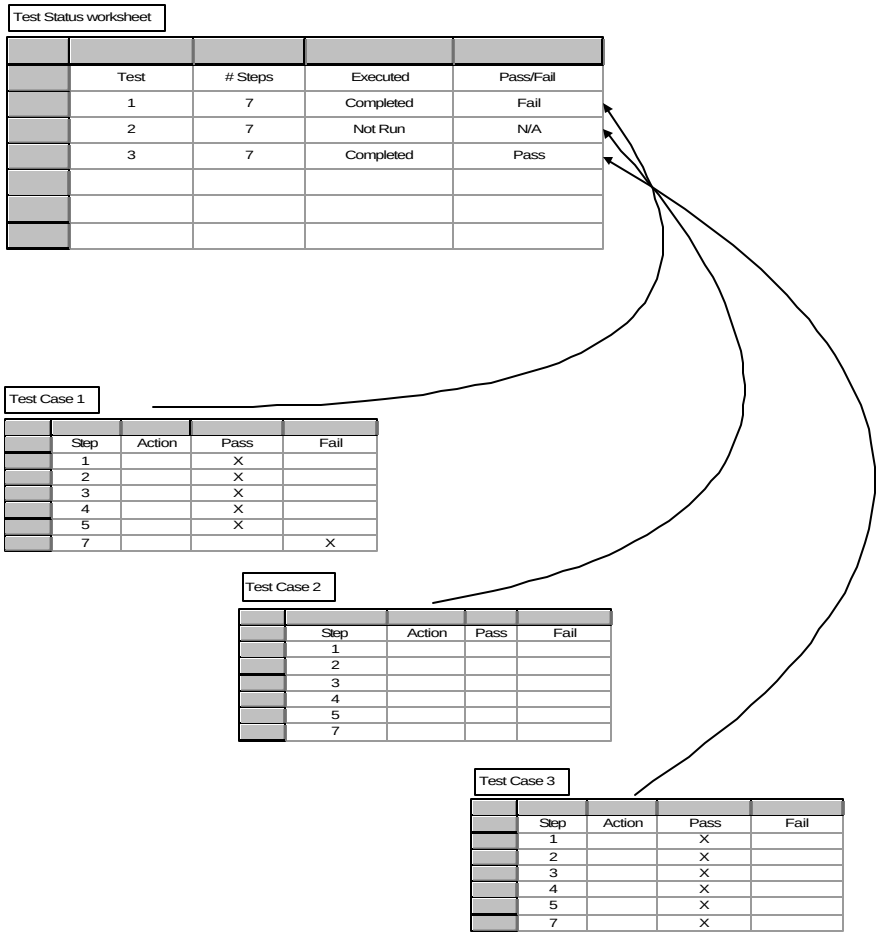
Ideally implementation should be as early in the project lifecycle as possible. The template files can be created with out all of the testing task information (requirements and functionality to test, and associated test cases). If the templates are built early on it helps in the tracking and reporting of status of test case creation. This is a bonus for both the test team and project management.

Reality occurs though and the system typically is implemented right as the test plan is being completed.

How to implement the system is very straightforward. Create a repository on the network, this is just a directory/folder that everyone has access to. All interested parties will need access to the information. First, build the template files. The first one is the Test Case template (this includes the format and formulas in it). The next one will be the Test Status template (this includes the format and formulas in it). Second, link the worksheet templates together. The test cases, and their data, will need to be associated to the status worksheet. Finally, begin building actual test cases and update/add them into the test status worksheet. Again, straightforward.

2.1 Test Status system architecture

The architecture of the system is simple. Each Test Case worksheet is linked into the Test Status worksheet. The data that is recorded and calculated in the test case is pulled into the test status worksheet where it is again crunched for project level statistics for reporting. Once the links are established data can be collected and reported on in any manner.



2.2 Test Case worksheet

Test Case Name:		**T/C #:**		
Description:		**Status:**	**Not Started**	
Start Conditions:		**CR#(s):**		
Overall Pass Criteria:				

Test Information

Name of Tester:		**Date:**		
Build Number:		**Time:**		
Test Data Used (acct, ssn):		**O/S:**		
Comments		**Browser:**		

25 steps		**Number of steps complete by status:**	**0**	**0**	**0**	**% Complete:**	**0%**
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**	**CR#**
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
	End of Test Case						

The test case worksheet itself is basic in its format and content. Specific fields are calculated (Status, % Complete, and Num. of steps complete by status) and are pulled into the test status worksheet for further analysis and reporting. Other static fields (Test Case Name, T/C #, number of Steps, Name of Tester, Build Number, etc.) can also be pulled into the test status worksheet if desired.

The calculated fields formulas are:

Status: = IF(D11 + E11 + F11 = 0, "Not Started", IF(A11 = D11 + F11, "Passed", IF(A11 <> D11 + E11 + F11, "Not Complete", IF(E11 > 0, "Failed"))))

# of Steps : = COUNTA(A13:A38)

Number of steps complete by status: = COUNTIF(D13:D38,"x") for Pass, = COUNTIF(E13:E38,"x") for Fail, and = COUNTIF(F13:F38,"x") for N/A

% Complete: = (D11 + E11 + F11) / A11

2.3 Test Status worksheet

Summary	# of Test Cases Comp. / Total # of Test Cases	# of Test Cases Passed / # of Test Cases Failed		Total # of Steps /Count of Steps Executed	Count of Steps Passed/ % Steps Passed	Count of Steps Failed/ % Steps Failed	
Totals	1 1	1 0		25	25	0	
Percent	100%				100%	0%	
Test Cases	**P/F**	**Number of Steps**	**Steps Executed**	**Steps Passed**	**Steps Failed**	**Date Tested**	**Bld Ver.**
Test1 Basic Test	Passed	25	25	25	0	1/0/00	0

The test status worksheet has numerous calculated fields (# of Test Cases Comp. / Total # of Test Cases, # of Test Cases Passed / # of Test Cases Failed, Total # of Steps, Count of Steps Passed, Count of Steps Failed, % Test Cases, % Steps Passed, % Steps Failed, and P/F) and others that contain information pulled from the test case worksheet (Number of Steps, Steps Executed, Steps Passed, Steps Failed, Date Tested, Bld Ver.). Additional information can be pulled in from test case worksheet (Tested By). Whatever is important to status reporting.

The calculated fields formulas are:

# of Test Cases Comp. : = COUNTIF(C7:C9,"Passed")+COUNTIF(C6:C9,"Failed")
Total # of Test Cases: =COUNTA(A7:A7)+COUNTA(A8:A8)
# of Test Cases Passed: =COUNTIF(C7:C8,"Passed")
# of Test Cases Failed: =COUNTIF(C7:C8,"Failed")
Total # of Steps/Count of Steps Executed: =SUM(D7:D8)
Count of Steps Passed: =SUM(F7:F8)
Count of Steps Failed: =SUM(G7:G8)
Percent # of Test Cases Comp. / Total # of Test Cases: =C2/C3
Percent of Steps Passed: =IF(F2 > 0,G2/F2,0)
Percent of Steps Failed: =IF(F2 > 0,H2/F2,0)
Passed/Failed: =IF(E7 = 0,"",IF(G7>0,"Failed",IF(D7 = F7, "Passed", "Pending")))

The test case linked fields formulas are:

Number of Steps : ='[TestCaseTemplate.xls]IE5.x'!$A$11
Steps Executed: ='[TestCaseTemplate.xls]IE5.x'!$D$11 + '[TestCaseTemplate.xls]IE5.x'!$E$11 + '[TestCaseTemplate.xls]IE5.x'!$F$11
Steps Passed: ='[TestCaseTemplate.xls]IE5.x'!$D$11
Steps Failed: ='[TestCaseTemplate.xls]IE5.x'!$E$11
Date Tested: =IF(E7 <> 0,'[TestCaseTemplate.xls]IE5.x'!$G$6, "")
Build Version (Bld. Ver.): =IF(E7 <> 0,'[TestCaseTemplate.xls]IE5.x'!$C$7, "")

In each of the calculations (local to the sheet or the link to the test case worksheet) the cell ranges (i.e., C7:C8) can vary in number (range), it depends on how the spreadsheets are built. Again, the "[TestCaseTemplate.xls]" is the workbook file, "IE5.x" is the worksheet name, and "$A$11" is the cell to reference. Notice how the IF statement is used to either get data from the test case worksheet (as in Build Version) or calculate a value based on other cells in the status worksheet.

Presented here is the core functionality, and it can be expanded upon based on need.

This information can also be pulled into another worksheet. As in the case of compatibility testing another worksheet can show which platforms (Software version, Operating System, Browser version/type, etc.) have been tested and their status. To do this duplicate the test case worksheets for each platform (i.e., browser version and type) in that test cases workbook. Then in the test status workbook create summary worksheets for each platform. Finally, create a summary worksheet that pulls its information from the platform worksheets. As an example,

Test Case Status of Primary Platforms		
	# of Test Cases Comp. / Total # of Test Cases	# of Test Cases Passed / # of Test Cases Failed
Totals	2 2	2 0
Percent	**100%**	**100%**
Completion	**IE5.5**	**NS4.7**
Test1	**Passed**	**Passed**

where each column is the platform being tested.

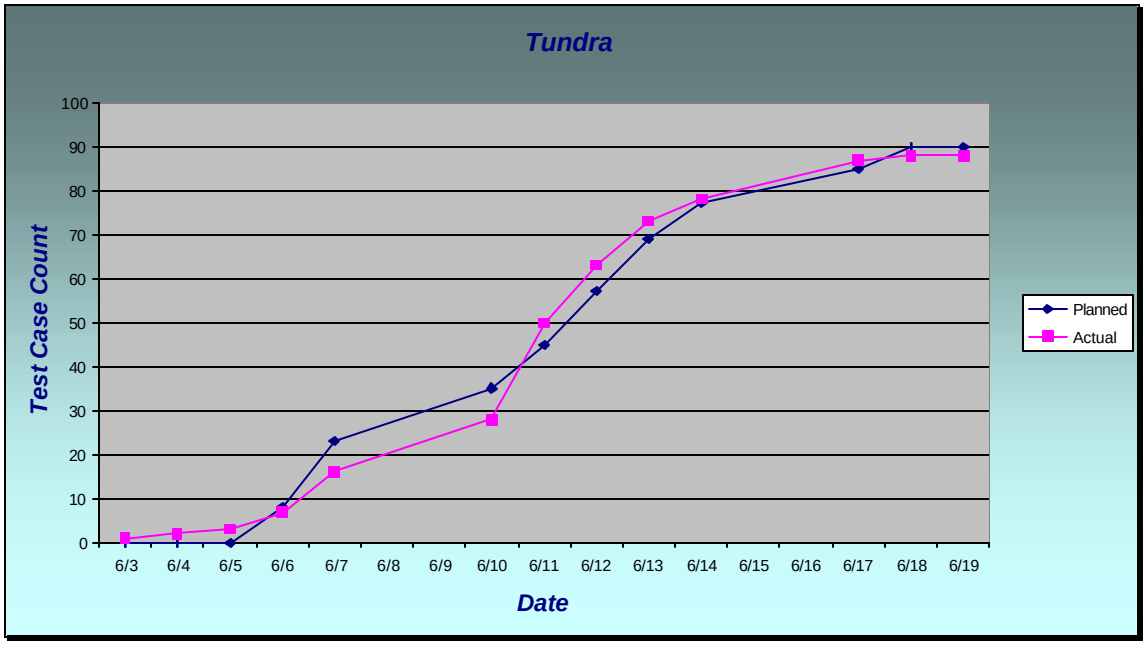
From here any type of report can be generated (tabular or chart).

3.0 Report Generation

The test summary spreadsheets themselves are used as the standard reports. Most people (management and other groups outside of Test/QA) want to know how many tests passed and failed, how many tests are executed, and how many are left to execute. They may also want to know who executed them or who is assigned to execute them. This last item helps in keeping tests assigned to staff and ensuring that the tests are being executed in a timely fashion. There is nothing worse than at the end of a project realizing an important test was not executed because it fell through the cracks, or that an important platform was not tested.

The reports can also be converted into charts for a visual view. Bar, Line, and Pie charts can be used to show status. Line charts show progress over time. This can include tests executed per week, or tests passed vs. failed per week, or number of test cases written vs. total to be written (this last one is done early in the project lifecycle and is beneficial in that it shows readiness of the Test group). Pie charts can be used to show status for coverage. This can include percent test cases executed vs. not executed, or percent platforms tested vs. not tested. Either of these can then be combined with defect status reports to show effectiveness of testing (number of new defects found per week that are attributable to test case execution, meaning that any ad-hoc testing defects are left out and just the ones found by the test cases are considered). Or it can be used to show system

stability (more defects are closed than found in a week for the amount of test cases executed). An example line chart is shown below.



4.0 Benefits of using the tool

- Higher accuracy of reporting testing status. Because the tool is automated and the format is standardized the chance for miscalculation is minimized, no direct interaction from tester.
- Earlier warning if project / test effort is in trouble.
- Individual or project progress can be tracked. Allows for better estimates of completion to be made.
- Can be incorporated into daily and project reports in other documents. This saves time and effort on the part of the test manager, and other interested parties.
- The data is in a centralized repository and is more accessible.
- It saves time and money. Because Excel is part of Microsoft Office it is readily available and the cost of the system is in the time spent to build it. No other software is needed. There isn't any ramp up time to learn or develop a 3rd party tool.

5.0 Summary

Using Excel as an alternative to a third party tool is a viable solution for companies who do not have the time or the budget to implement other tools. Utilizing its built in functions and formulas, and a little bit of inspiration, a customized and flexible test status reporting system can be built. With the information presented here anyone can jumpstart the implementation process.

We have found, out of necessity, that proper test management and status reporting are vital to the success of a project. By using the test status system we have saved ourselves time and money, increased our reporting accuracy, and been able to keep our testing projects under control.

Author(s) Biography:

Jim Hazen - Jim has 15 years experience in software development and testing. Jim is a Test Engineer with SysTest Labs in Denver, Colorado. Jim is a Certified Software Test Engineer through the Quality Assurance Institute.

Jim has worked in various capacities as a test lead / project manager, automation developer, and department manager. He has worked with a wide variety of applications on DOS, OS/2, Windows, and Web environments.

Earl Burba – Earl has over 20 years experience in software development and testing. Earl is a Sr. Project Manager with SysTest Labs in Denver, Colorado.

Earl has worked in various positions within industry. Earl's experience includes flight systems, telephony, database, business and embedded operating systems. Earl is an award winning author, and holds patents on software testing tools & methodologies.

QW2002 Paper 6P1

Dr. Nancy Eickelmann
(Motorola)

Optimizing Test Productivity to Maximize Product Quality

Key Points

- Product quality results from the right mix of test activities
- Six Sigma can help me measure what will work best
- Six Sigma makes it possible to achieve a target quality for my software

Presentation Abstract

This paper will present case study findings from a Six Sigma Black Belt Project. Every Black Belt Project has a charter that defines the customer focus and the goals of the project. This project is designed to identify the key factors that impact test effectiveness for static and dynamic test technologies. Empirical data is collected and simulation models of the generic processes are created. The models that are created abstract away unnecessary details of the process and provide a test-bed to evaluate the two test technologies relative to their effectiveness, cost in effort, time required (duration), and complexity of the activity. The project results quantify the relative productivity of static analysis techniques versus dynamic automated test technologies.

About the Author

Dr. Nancy Eickelmann is currently a research scientist for Motorola Labs and is leading the Motorola software and system test process measurement and evaluation research initiative. Prior to joining Motorola she was program manager at the NASA/WVU Software Research Laboratory, located at the NASA Independent Verification and Validation research facility. Before joining NASA she was a member of the Advanced Programs Research Group at MCC where she developed a measurement framework for guiding the decision-making process in product line development. Dr. Eickelmann began her research career as a member of the technical staff at Hughes Research Laboratory (HRL) in Malibu, California while completing her doctorate at the University of California, Irvine. She was named a Hughes Doctoral Fellow while working at HRL and received several research awards while working with Dr. Debra Richardson's Formal Methods and Software Testing Group at UCI. Dr. Eickelmann has collaborated internationally on research projects for defense systems, space station applications, space shuttle and global software development. Dr. Eickelmann holds a B.S. Finance, M.B.A., M.S. and Ph.D. Computer Science.

MOTOROLA LABS

Optimizing Test Productivity to Maximize Product Quality: A Six Sigma Case Study

Nancy S. Eickelmann, Ph.D.
Motorola Labs
1303 E. Algonquin Rd.
Schaumburg, IL, 60196, USA
(847) 538-0745
Nancy.Eickelmann@motorola.com

September 5, 2002

Outline

MOTOROLA LABS

- What is Six Sigma?
- Six Sigma Methodology
- 8 Steps of Experimental Design
- The Study
- The Results
- Conclusion

What is Six Sigma?

"Six Sigma is a 4-step high performance system to execute business strategy." Matt Barney, Motorola Inc.

1. Align executives to the right objectives and targets
2. Mobilize improvement teams
3. Accelerate results
4. Govern sustained improvement

3

DMAIC: Six Sigma Methodology

- ***Define***: requires a project charter which must include a business case with (financial impact), problem statement, goal statement, project scope, team member roles, milestones/deliverables and support required.
- ***Measure***: builds a factual understanding of existing process conditions and problems. A critical part of the measure phase is a Gage R&R report that isolates the amount of variation introduced through the data collection process versus the technical or business process to be studied.
- ***Analysis***: applie a design of experiments for the Six Sigma project.
- ***Improve***: consists of implementing the process changes isolated in the analysis phase.
- ***Control***: monitors and ensures that the new process is being followed.

4

Six Sigma for Software

Only 16.2% of software projects are completed on-time and on-budget.

In larger companies, a meager 9% of technology projects come in on-time and on-budget. In addition, about one third of all projects will be canceled before they ever get completed.

Further results indicate 53% of projects will cost an average of 189% of their original estimates. In financial terms this analysis revealed that over $100 billion in cancellations and $60 billion in budget over runs occur in the Software Sector annually. *The numbers just don't lie.**

Source: http://www.bmgi.com/course_SS_Software.htm

5

8 Steps of Experimental Design

1. **State the Problem with Clarity**
2. **Select the Output or Response Variables**
3. **Identify the Process Variables**
4. **Select the Factor Levels and Ranges of Factor Settings**
5. **Select the Appropriate Experimental Design**
6. **Plan the Experiment**
7. **Execute the Experiment**
8. **Analyze and Interpret the Results**

6

Step 1:State the Problem with Clarity

MOTOROLA LABS

What are the impacts of changing the inspection process?

Are the effects limited to the inspection process or are there larger impacts across the lifecycle?

7

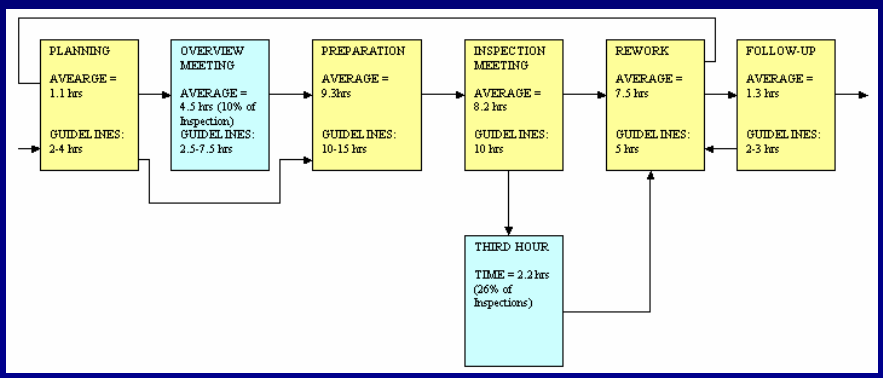
Step 2: Select the Output or Response Variables

MOTOROLA LABS

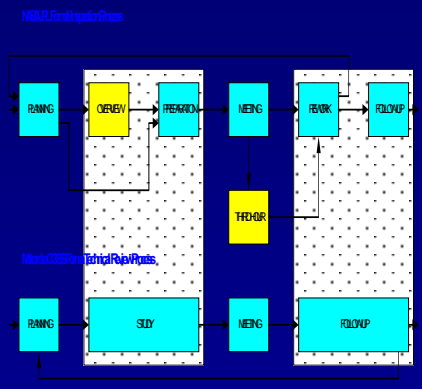
- The output variables selected were:
  - staff hours of effort by activity type
  - total staff hours of effort
  - total cycle time
  - defects found
  - fault density

8

Data Analysis: Average Effort per Process Step



Step 3: Identify the Process Variables



- Number of staff per inspection team
- Number of process steps
- Overview effort
- Planning effort
- Meeting effort
- Preparation effort

Step 4: Select the Factor Levels and Ranges of Factor Settings

MOTOROLA LABS

Table 1. High Low Conditions by Experimental Factor			
Simulation Run/Condition	Factor1 Team Size	Factor 2 # SLOC/Pages	Factor 3 # Defects
1	3	100	5
2	7	600	5
3	3	100	5
4	7	600	5
5	3	100	50
6	3	600	50
7	3	100	50
8	7	600	50

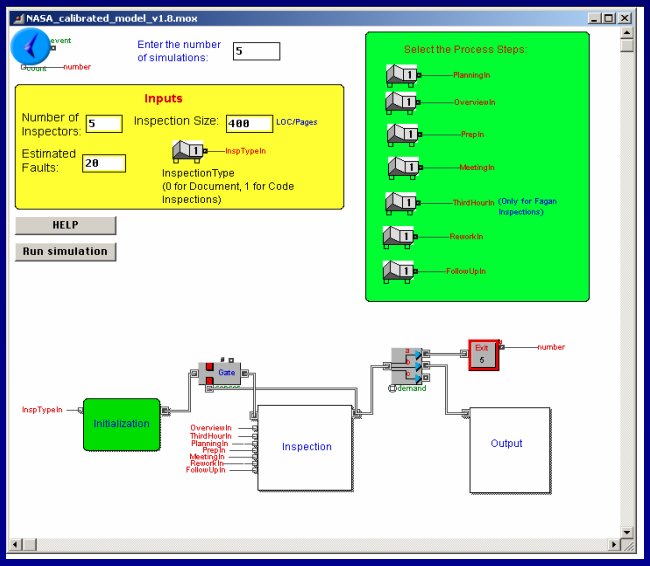
Step 5: Select the Appropriate Experimental Design

MOTOROLA LABS

- **The design selected was a 2X2X2 Full Factorial that evaluates the 3 input factors for inspections.**
- **Two of these factors team size and product size inspected are both measurable and controllable by management.**
- **The third factor or number of defects present is considered uncontrolled but measurable and a key factor.**

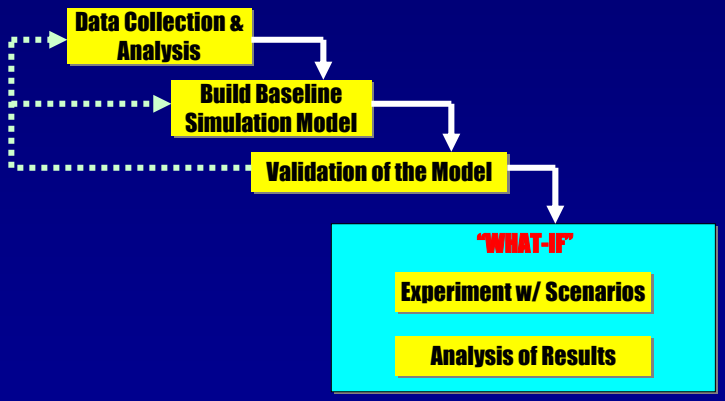
Step 6 &7: Plan and Execute the Experiment

MOTOROLA LABS

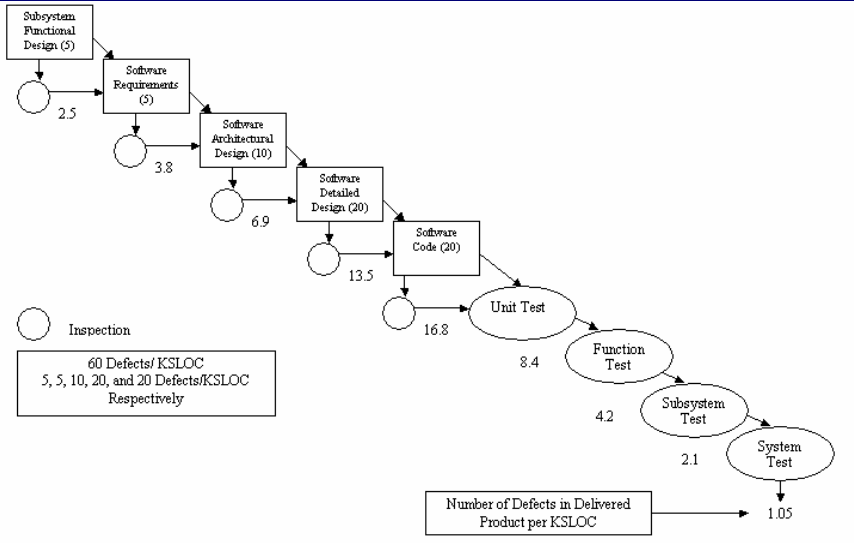


Simulation Modeling Steps

MOTOROLA LABS



Data Analysis: Inspection Effectiveness

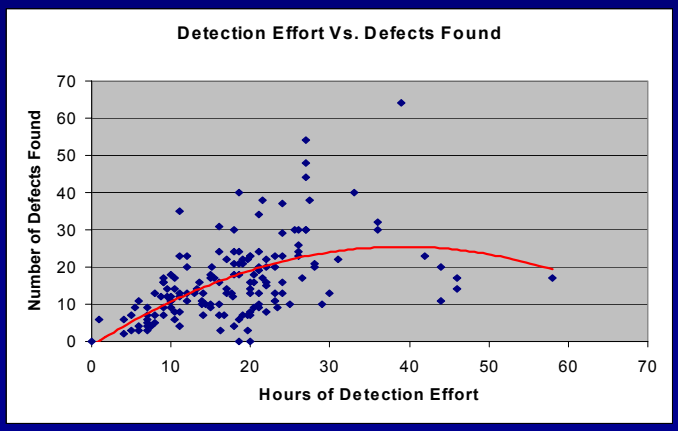


Step 8: Analyze and Interpret the Results

- time plots of responses
- residual plots
- pareto charts
- normal probability plot of effects
- P-values of effects
- main effects plots
- interaction plots
- cube plots

Data Analysis: Defects Vs. Detection Effort

- Detection Effort = Overview Effort + Preparation Effort + Inspection Meeting Effort
- After about 35 hours, Effectiveness decreases



17

Step 8: Analyze and Interpret the Results

Effect Tests

Source	N	DF	Sum of Squares	F Ratio	Prob > F
No. of Inspectors	1	1	63.28125	2392.486	0.0130
Estimated Faults	1	1	708.00845	26767.81	0.0039
Size	1	1	52.73645	1993.817	0.0143
No. of Inspectors *Estimated Faults	1	1	37.93205	1434.104	0.0168
No. of Inspectors*Size	1	1	0.08405	3.1777	0.3255
Estimated Faults *Size	1	1	33.04845	1249.469	0.0180

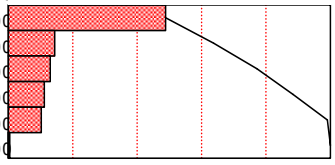
18

Step 8: Analyze and Interpret the Results

MOTOROLA LABS

Pareto Plot of Transformed Estimates

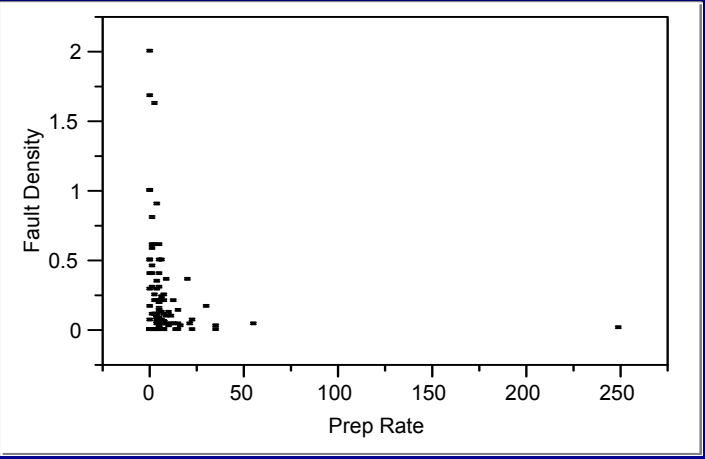
Term	Orthog Estimat
Estimated Faults[5]	-9.407500
No. of Inspectors[3]	-2.812500
Size[100]	-2.567500
No. of Inspectors[3]*Estimated Faults[5]	2.177500
Estimated Faults[5]*Size[100]	2.032500
No. of Inspectors[3]*Size[100]	-0.102500



19

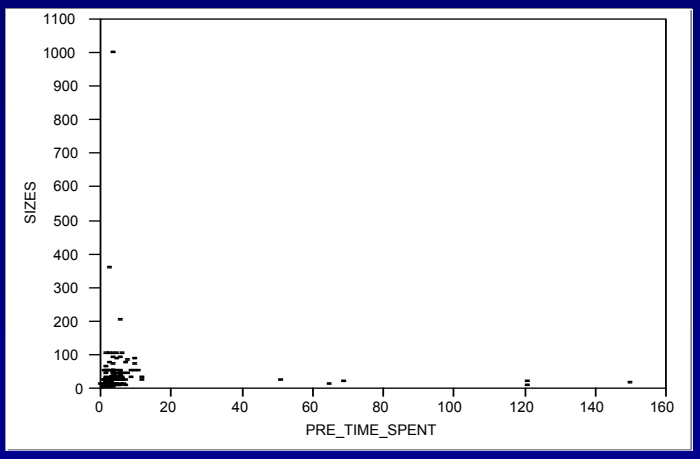
Fault Density vs. Prep Rate

MOTOROLA LABS



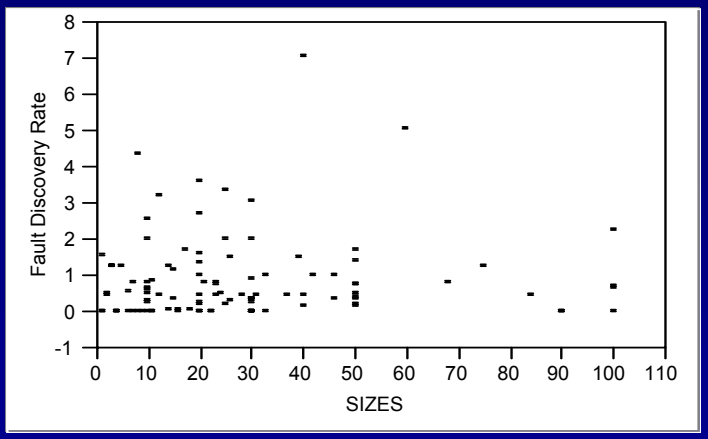
20

Size vs. Pre_Time_Spent



21

Fault Discovery Rate vs. Size



22

Productivity… Inspections

MOTOROLA LABS

…Summary of measures

- Number of Staff/Inspection ⬆
- Staff effort expended in preparation and meeting time ⬆
- Preparation Rate per Page ⬆
- Preparation Rate per LOC ⬆
- Average Defects Found ⬇

Lifecycle productivity is not measured by these measures….

Lifecycle Productivity Inspection & Test

MOTOROLA LABS

- Defects Found
  - Inspections: 65 %
  - Tests: 35 %
- Inspection Effort to Test Effort
  - 20 % (Inspections)
  - 80 % (Tests)
- Productivity of Inspections
  - 65% Defects/20% Effort = 3.25 Defects/Unit Effort
- Productivity of Tests
  - 35% Defects/80% Effort = .44 Defectss/Unit Effort

- **Productivity** Inspection to Test
  - 3.25/.44 = 7.4 Times More Productive

Lifecycle Productivity Inspection & Test

- Defects Found
  - Inspection : 5%
  - Test: 95%
- Inspection Effort to Test Effort
  - 10 % (Inspection) to 90 % (Tests)
- Productivity of Inspection
  - 5% Errors/10% Effort = 0.5 Defects/Unit Effort
- Productivity of Tests
  - 95% Defects/90% Effort = 1.06 Defects/Unit Effort

- **Productivity** Inspection to Test
  - **0.5/1.06 = 0.4 as Productive as Tests**

Summary

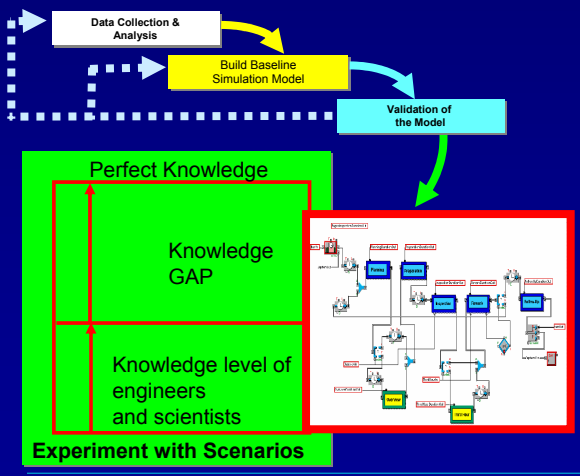
- ***Six Sigma*** *driven Quantitative Management* of…
  - Strategic Planning
  - Process Improvement
  - Technology Adoption

- *Software Problems*:
  - ***Resource optimization*** of distributed development is not well supported
  - High maturity teams ***break their process when adopting changes***.
  - Pilot programs often yield ***inconclusive evidence of technology benefits*** due to process interactions.

Conclusion

MOTOROLA LABS

Mot Labs Process Simulation Modeling is the *only* technology that provides an experimental test bed for Six Sigma based software improvement!



- Predict Schedule
- Predict Effort
- Predict Quality
- Quantify Productivity
- Quantify Cycle Time

Optimizing Test Productivity to Maximize Product Quality: A Six Sigma Case Study

Nancy S. Eickelmann, Ph.D.
SSERL, Motorola Labs
1303 E. Algonquin Rd.
Schaumburg, IL, 60196, USA
(847) 538-0745
Nancy.Eickelmann@motorola.com

ABSTRACT

This paper will present case study findings from a Six Sigma Black Belt Project. Every Black Belt Project has a charter that defines the customer focus and the goals of the project. This project is designed to identify the key factors that impact test effectiveness for static and dynamic test technologies. Empirical data is collected and simulation models of the generic processes are created. The models that are created abstract away unnecessary details of the process and provide a test-bed to evaluate the two test technologies relative to their effectiveness, cost in effort, time required (duration), and complexity of the activity. The project results quantify the relative productivity of static analysis techniques versus dynamic automated test technologies.

Keywords: Process Improvement, Defect Prevention, Process Characterization and Simulation.

1. INTRODUCTION

Extensive studies of the inspection process have been conducted at IBM, NASA, JPL and now Motorola to evaluate the normative effort and duration levels per process step and to evaluate and predict defect detection capability for the process. Several studies have been conducted over the last few decades concerning the effectiveness of inspections and the following data has been published concerning their efficacy.

- Errors Found:
  Fagan Inspections: 65 %
  Tests: 35 %
- Inspection Effort to Test Effort
  Fagan Inspections: 20 %
  Tests: 80 %

- Productivity of Inspections

65% Errors/20% Effort = 3.25 Errors/Unit Effort

- Productivity of Tests

35% Errors/80% Effort = .44 Errors/Unit Effort

- Productivity Ratio Comparison of Inspections to Test
  3.25/.44 = Fagan Inspections are 7.4 Times More Productive than Test

In addition, normative values for effort, duration, number of people required (and their respective roles) and defect detection have been documented and validated. Figure 2 provides a diagram of the Fagan Inspection process.

The primary factor correlated to defect detection for inspections was the number of hours spent in the overall process as well as individual process steps. In our study we

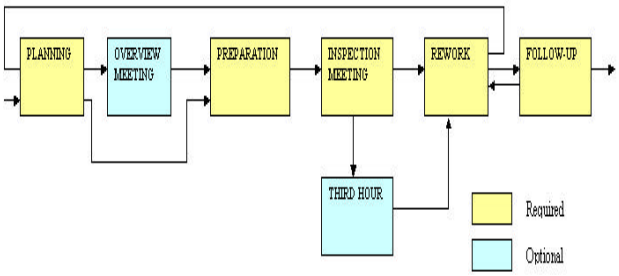


Figure 1. Fagan Inspection process steps.

wanted to determine if modifying the Fagan Inspection process would have an impact on overall product quality as well as test efficiency across the life cycle. Specifically, if the early life cycle static analysis techniques were to become less effective what would be the impact for integration test and system test?

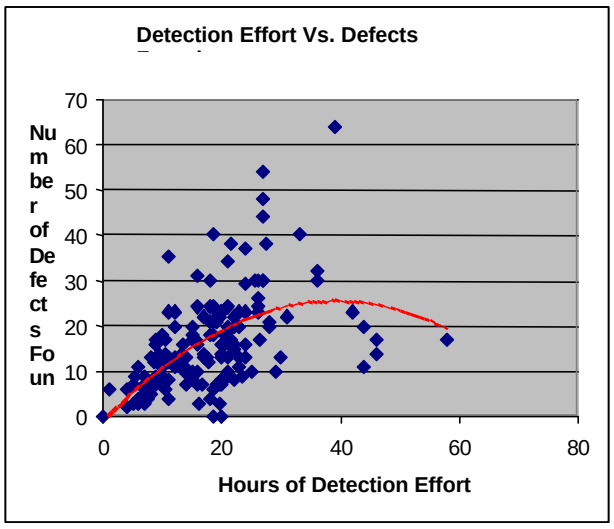


Figure 2. Number of Defects Found per Hours of Effort Spent.

The graph in Figure 5, is adapted from a study reported by NASA/JPL with 171 inspections. The total number of hours and number of defects increase in a non-linear relationship up to 28-32 hours of effort and then additional hours spent may reduce or increase with equal likelihood the number of defects found. However, at low levels of effort8hours or less a more direct relationship exists between number of defects found and number of hours of effort expended.

This study investigates the relationships between key factors in inspections and the direct impact on inspection effectiveness and the resulting impact on overall lifecycle productivity and quality. The paper is organized as follows. Section 2 describes our approach. Section 3 describes the actual experiment and modeling and simulation method used in the experiment. Section 4 discusses the results. Section 5 provides our conclusions.

2. Six Sigma Case Study Approach

Six Sigma is a quantitative approach to strategic management through analytical and experimental initiatives. The defined, rigorous and disciplined approach to process improvement is encapsulated in one of two five phase processes, DMAIC or DMADV. This paper will focus on DMAIC, Define, Measure, Analyze, Improve, Control. The define phase requires a project charter which must include a business case with (financial impact), problem statement, goal statement, project scope, team member roles, milestones/deliverables and support required. The measure phase goal is to build a factual understanding of existing process conditions and problems. A critical part of the measure phase is a Gage R&R report that isolates the amount of variation introduced through the data collection process versus the technical or business process to be studied. The analysis phase is applied using a full factorial design of experiments for the Six Sigma project. The improve phase consists of implementing the process changes isolated in the analysis phase. The control phase is applied to monitor and ensure that the new process is being followed. This paper will focus on the analysis phase and the use of process modeling

and simulation to conduct full factorial design of experiments.

The experimental approach is to create a simulation model of the organisation's current static test process. The simulation models can be executed to determine the impact of inserting a new technology with respect to productivity, schedule, quality, reliability and/or cost. A prototype tool is developed that provides an objective decision support mechanism that a manger can use to evaluate the impact of changing the current process or technology.

To conduct these experiments we follow a scientific approach to experimental design. Consisting of essentially 8 steps.

1.) The initial step is to state the problem with enough detail and clarity to communicate the intended objective of the experiment and thus the purpose of the model. A requirements document must be created. Executable simulation models (like most software) require that multiple views of the model be documented in text or graphic form. The required views include data view, decision-gate view, process view, resource usage view, and user view. The data view must describe the flow of information (possibly in the form of data flow diagrams) and formally specify data formats and data bindings. The second view is to document the critical decision gates, the decision mechanism used, and the decision criteria applied. The canonical decision example is when do we stop testing? The process view documents the process steps; the entry/exit criteria, process step inputs and outputs, and required resources for an instance of that step. The resource usage view describes the sequence and magnitude of resources required, and defines resources groups that must occur at critical stages of the process. The user view consists of use-case scenarios that describe the actual activities that a test engineer or test manager would engage in to perform specific test functions.

2.) Next the researcher must select the response set or output variables of interest. In industry based models the response variable is generally cycle time, effort, and/or number of defects detected and removed.

3.) Third, the process variables must be identified. The selection of the process variables includes the design parameters (control factors) and the noise variables. In addition the interactions among these variables must be isolated and quantified. Noise factors are uncontrolled variables that may or may not have significant impact on the response variable. The experimenter or researcher may modify the simulation model to mitigate any untoward effects resulting from noise variables. This is accomplished by altering the variable values of the noise and design parameters for various conditional levels.

4.) This step requires that a set of factor levels and ranges of factor settings (values) be determined and documented. The ranges of values for the model factors may be context sensitive and therefore the assumptions that the researcher makes when creating the model are very important to the accuracy and validity of the results. A design of experiments is a tool used in a Six Sigma project to evaluate the cause and effect relationships between numerous process variables (independent variables) and an output variable or (dependent variable). The design of experiments is an effective and efficient approach to isolating the key factors that are the significant sources of process variation. By isolating the key variables and their relationships, it enables the prediction of the impact measured in gains or losses from changes in process conditions or contexts. A full factorial design of experiments permits changing several factors or variables simultaneously versus one at a time. The full

factorial design begins with only two conditions per factor, requiring 8 simulation runs overall, see table 1. The two factors are assigned high and low values for the two conditions however, a full factorial can consider all combinations of factor conditions and in this instance a normative condition per factor is suggested. The resulting number of simulation runs with three conditions, 3X3X3 = 27 simulation runs.

Table 1. High Low Conditions by Experimental Factor			
Simulation Run/Condition	Factor1 Team Size	Factor 2 # SLOC	Factor 3 # Defects
1	-	-	-
2	+	-	-
3	-	+	-
4	+	+	-
5	-	-	+
6	+	-	+
7	-	+	+
8	+	+	+

All factor combinations can be tested for their degree of variability and each factor can be isolated to document the degree of common-cause variation versus special cause variation. In addition, process variation may result from main effects or from interaction effects. To calculate main affects the average of all observations at the low level are subtracted from the average of all observations at the high level. Interaction effects occur when the effect one factor has on a response variable is not the same for each conditional level of another factor. For example, Factor 1 and Factor 2 interaction is equal to the value of the Effect of Factor 1 for a low value of Factor 2 subtracted from the Effect of Factor 1 for a high value of Factor 2, which is then divided by the number of factors. Calculating a P-value, creating a Pareto chart of the effects, or constructing a probability plot of the effects achieves deciding which effects are statistically significant.

5.) Choice of appropriate experimental design, full factorial and reduced fractional including screening, Plackett-Burman designs, fractional factorial and response surface. Each type of design is chosen based on how much you already know about your process and how many factors you want to test.

6. & 7.) Experimental planning and experimental execution.

8.) Experimental data analysis and interpretation. This is the final step of the process and requires the data generated by the experiment be analyzed to support the researcher in drawing conclusions, verifying that those conclusions are accurate and valid, and then generating a report and presentation of the results. Implicit in this step is recommending actions to the project sponsor. In reporting to the sponsor the researcher must identify the potential causes of variation isolated, justify why the study focused on specific aspects and factors, what data was collected to support the conclusions and how that data was interpreted.

3. THE EXPERIMENT

Step 1. State the problem: What are the impacts of changing the inspection process? Are the effects limited to the inspection process or are there larger impacts across the lifecycle? Specific questions we wanted to answer:

- What is the impact on inspection effectiveness and productivity of reducing the number of preparation effort hours expended?
- What is the impact on inspection effectiveness and productivity of eliminating or combining steps in the inspection process to save time or effort?
- Is there an impact on test effectiveness if inspection effectiveness is diminished?
- How can I measure that my full life cycle test and analysis activities are

optimized to deliver the product with certifiable degree of quality?

Step 2. Select the output variables: The output variables selected were staff hours of effort by activity type and total staff hours of effort; total cycle time; defects found and fault density.

Step 3. Identify the process variables: The process changes that had been made included: reducing the number of staff per inspection team; combining rework and follow-up process steps; eliminating overviews; and reducing planning effort. The diagram in figure 3 contrasts the five required and two optional steps for Fagan Inspections and the four steps of the Formal Technical Review.

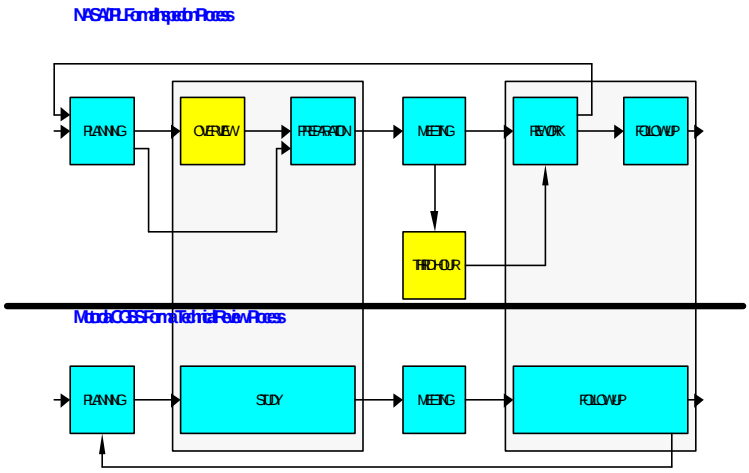


Figure 3. Overlay of two process diagrams, Fagan Inspections (top portion) and Formal Technical Reviews (bottom portion that represents a Modified Fagan process).

Step 4. Set the factor levels and ranges of factor settings: The 4 general assumptions behind the Inspection model are the following:

1) The nominal effort for an inspection is 30.5 staff-hours on average and the nominal cycle time for an inspection is 3 weeks. The nominal size for an inspection is 400-600 LOC for a 2-hour duration inspection meeting.

2) The nominal effort for the inspection occurs when the estimated fault density is between 50 – 75 faults/KLOC. For fault densities greater than 75 faults/KLOC, the inspection effort and cycle time are higher while for fault densities less than 50 faults/KLOC, the inspection effort and cycle time are on average lower than the nominal.

3) If the inspection is a Document Inspection with size in Pages, the model converts the size to LOC using the following relationship:

For Size Less than 20 pages

Size in LOC = Size in pages * 20

For Size Between 20 and 40 pages

Size in LOC = 10 * Size in pages + 200

For Size Greater than 40 pages

Size in LOC = Size in pages * 15

4) The number of faults found during an inspection depends on the defect detection effort and the estimated faults. The defect detection effort is the sum of the overview effort, preparation effort, and inspection meeting effort. Increasing the defect detection effort increases the number of faults found.

Using the model assumptions, results of the NASA studies, and the empirical data collected for this study we populated the factor table with high (+) and low (-) values. The factor ranges were derived from empirical data and are not shown.

Table 1. High Low Conditions by Experimental Factor			
Simulation Run/Condition	Factor1 Team Size	Factor 2 # SLOC/Pages	Factor 3 # Defects
1	3	100	5
2	7	600	5
3	3	100	5
4	7	600	5
5	3	100	50
6	3	600	50
7	3	100	50
8	7	600	50

Step 5. Choice of Experimental Design: The design selected was a 2X2X2 Full Factorial that evaluates the three input factors for inspections. Two of these factors team size and product size inspected are both measurable and controllable by management. The third factor or number of defects present is considered uncontrolled but measurable and a key factor.

Step 6 & 7: Experimental planning and execution. The plan was to use an abstract model of the inspection process. The model needed to support representing the Fagan Inspection process as well as support a re-configured model to reflect the changes that were implemented and resulted in the formal technical review process. The model was built using the Extend software modeling and simulation environment and incorporated the data values and factor relationships and interactions identified through our analysis of the empirical data.

The simulation model was developed using the Extend software modeling tool. The tool provides for the creation of a graphical user interfaces for the models and the simulation experiments, initialization modules to set initial values for model factors and process variables, and separate constructs for process fragments such as reviews or test execution modules, and finally reporting modules that output the data from the experiment. In this section, we will describe each part of the model and the resulting data from the experimental simulation.

The executable model has a manual input and selection screen that allows the researcher to set the factor values for the study. The input screen has several key control aspects that are used to design the experiment and are described in order of placement top left to right see Figure 4.

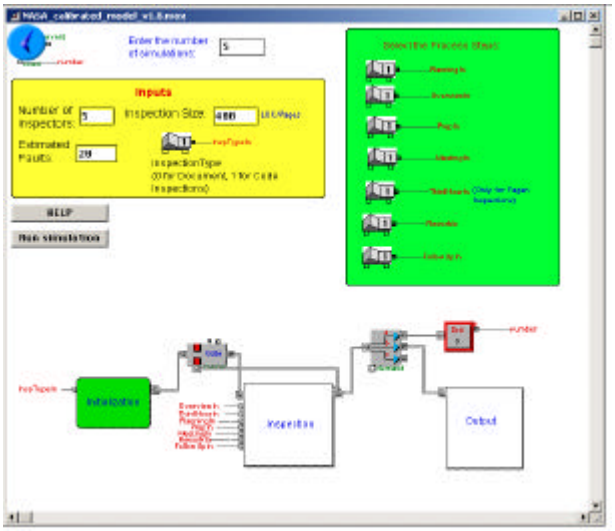


Figure 4. Extend Model input screen for design set-up and factor values.

The first object in the screen is the blue circle in the topmost left-hand corner. This object is used to select discrete event or continuous modeling as the time mechanism for the simulation. The yellow box directly below it is the manual input slots for the model factor values of number of team size, estimated number of defects, size of product (pages or SLOC) The switch is applied to provide the units of measurement for the size of the product based on the format of the product (pages versus SLOC).

The green box to the right of the screen has 7 switches that are essentially on/off devices to include or exclude a process step from a set of static techniques including Fagan inspections, reviews, walkthroughs, peer reviews and other manual analysis techniques. In Figure 3 the

Fagan Inspection process steps include five mandatory steps and two optional steps. The mandatory steps are shown in Figure 3 in yellow; planning, preparation, inspection meeting, rework, and follow-up. There are 2 optional steps; an overview meeting and a third hour or extension to the inspection meeting.

The on/off switches essentially provide the ability to reconfigure the model of the process including or excluding process steps and providing various ranges of input values for each factor.

3.1 Model Input Initialization

The Extend model requires an initialization block that provides the initial values for the model input factors. The Initialization block is shown in Figure 5. The first DE Equation on the left is used to convert from Pages to LOC, if the Inspection is a Document Inspection. The Set A(5) block is used to set the Team Size and the Estimated Faults. There are 5 Stop blocks that are used to stop the simulation if the input values are out of range. For example, the simulation stops if the inspection size is greater than 800 LOC. The Initialization block includes a Ratio block, which calculates a ratio, based on the estimated fault density. This ratio is then used to increase or reduce the estimated effort and cycle time based on the estimated fault density. For example, if the estimated fault density is between 75 and 100 faults per KLOC, the Ratio is between 0.93 and 1.18.

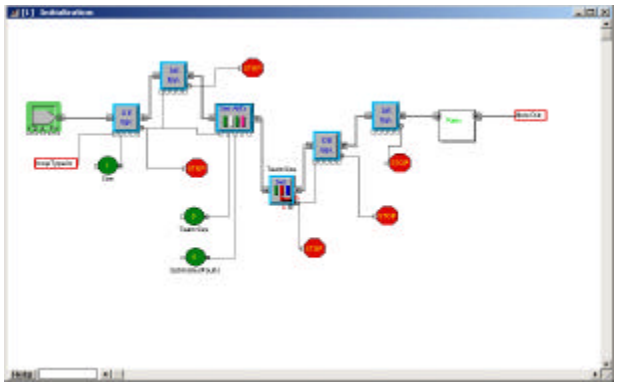


Figure 5. Extend Model initialization block.

3.2 Inspection Process Component

The principle component of the model is the inspection process itself. This component of the model represents the activities associated with the process steps and creates the linkages for the relationships between team size, product size, effort in staff hours and defect detection effectiveness. The inspection process module is shown across 3 figures

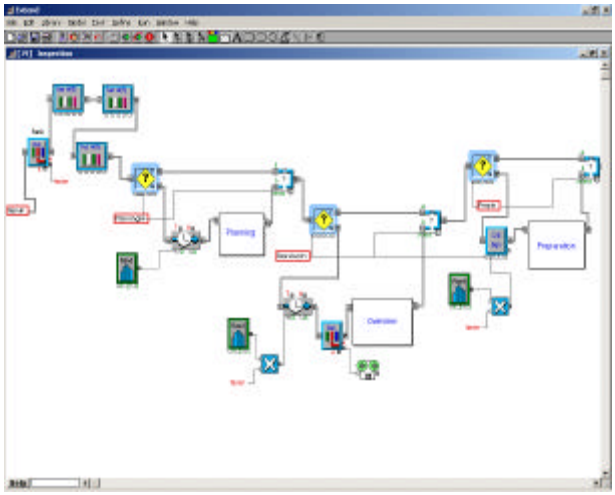


Figure 6. Inspection process module (Part1)

The Inspection block is made up of the 7 inspection process steps: Planning, Overview, Preparation, Meeting, Third Hour, Rework and Follow Up. Figure 6 shows the portion of the Inspection block from Planning to Preparation. The Inspection block has a number of activity delay blocks to calculate the cycle time for each process step. The activity delay blocks (these blocks are readily identifiable by the Timer on them. One such block is the block just to the left of Planning) are connected to a Random Distribution block that gives a value to the cycle time based on the triangular distribution. The Planning, Overview and Preparation blocks are used to calculate the effort spent in these process steps.

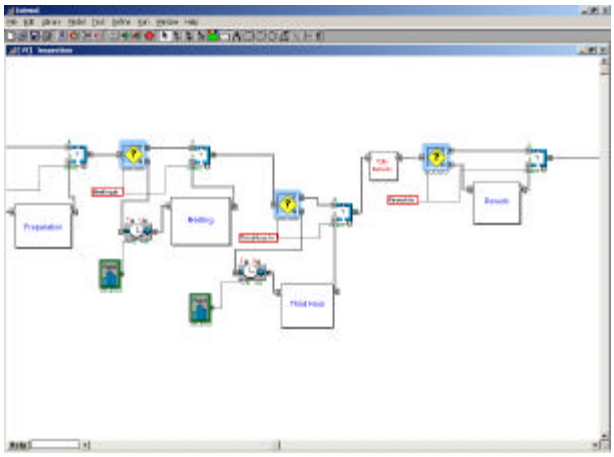


Figure 7. Inspection process module (Part 2)

Figure 7 shows the portion of the Inspection block from Preparation to Rework. There are 2 activity delay blocks to calculate Meeting Cycle Time and Third Hour Cycle Time. The Preparation, Meeting, Third Hour and Rework blocks are used to calculate the effort spent in these process steps. The Calc. Defects block is used to calculate the number of defects found in the inspection, based on the defect detection effort and the estimated faults in the inspection item. The Meeting Duration is calculated based on the Inspection Size and the Inspection Rate. The Inspection Rate is a triangular distribution. The Meeting Effort is calculated by multiplying the team size with the meeting duration.

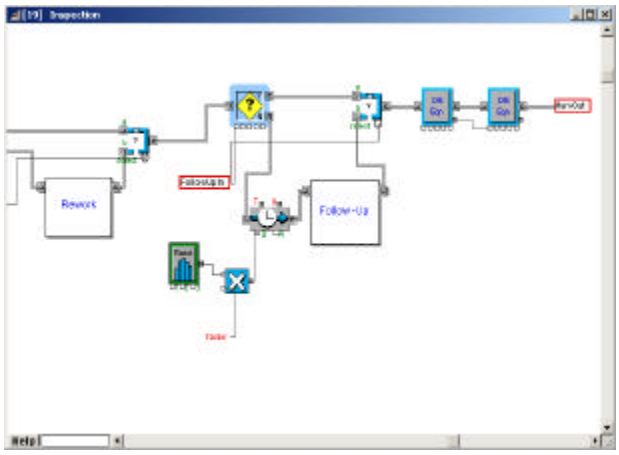


Figure 8. Inspection process module (Part 3)

STEP 8.) Experimental data analysis and interpretation. The statistical analysis required to evaluate the results of the experiment include time plots of responses, residual plots, pareto charts, normal probability plot of effects, P-values of effects, main effects plots, interaction plots and cube plots.

The time plots are to evaluate whether or not factors are associated with trends or cycles associated with time. The significance of effects is evaluated using pareto charts, normal probability plot of effects and P-values of effects.

Effect Tests

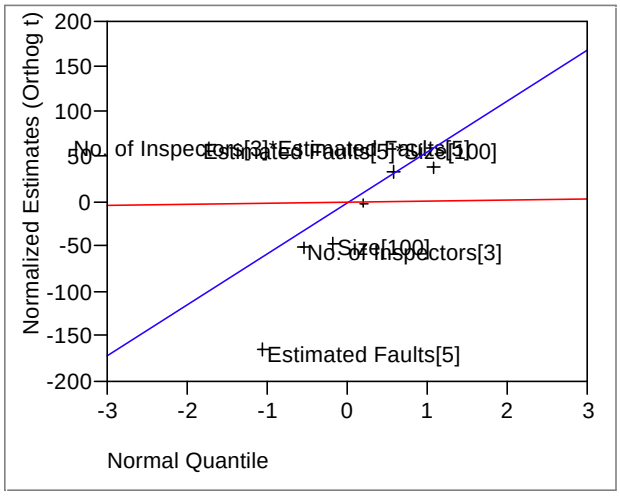
Source	N	DF	Sum of Squares	F Ratio	Prob > F
No. of Inspectors	1	1	63.28125	2392.486	0.0130
Estimated Faults	1	1	708.00845	26767.81	0.0039
Size	1	1	52.73645	1993.817	0.0143
No. of Inspectors *Estimated Faults	1	1	37.93205	1434.104	0.0168
No. of Inspectors*Size	1	1	0.08405	3.1777	0.3255
Estimated Faults *Size	1	1	33.04845	1249.469	0.0180

Effect Screening

The parameter estimates are not correlated.
The parameter estimates have equal variances.

	Lenth PSE
t-Test Scale	56.804348
Coded Scale	3.26625

Normal Plot



Blue line is Lenth's PSE, from the estimates population.
Red line is RMSE, Root Mean Squared Error from the residual.

The effects analysis provides P-Values for main effects and P-Values for interaction effects. The main effects are low values for number of inspectors (3), low values for number of defects (5) and low values for product size (100 SLOC). There were significant interaction effects number of inspectors (3) and defects (5), and number of defects (5) and product size (100 SLOC). There was no interaction between number of inspectors (3) and product size (100 SLOC).

4. CONCLUSION

A great deal of study has been focused on evaluating the inspection process as applied in software engineering. Inspections are a static analysis technique that brings a team of software professionals together to manually inspect the evolving software products from requirements, design, code, and tests. A formalized structured approach to inspections was introduced in 1974 at IBM, and was championed by Michael Fagan. Although there is a substantial published empirical evidence of the defect reduction, defect prevention, and lifecycle cost reduction associated with Fagan Inspections, industry is prone to "modify" the inspection process as a quick cost cutting measure.

A reduction of the number of preparation effort hours expended resulted in a drop in the number of defects detected per inspection, thus impairing inspection effectiveness and productivity. In addition, eliminating or combining steps in the inspection process to save time or effort resulted in further decreases of defect detection.

To evaluate the trade off of changing one part of the process the researcher must look at the full life cycle to evaluate true impacts from a lifecycle view, that is from requirements to test, am I truly improving or just shifting resources. In other words, how can full life cycle test and analysis activities be optimized to deliver the product with a certifiable degree of quality?

The calculation for full life cycle evaluation if total amount of effort to each activity and the defects detected by that activity.

Six Sigma uses process modeling and simulation as a powerful tool to achieve the development of high quality products through quantitative control of process improvements, technology adoption strategies, defect prevention efforts and automation strategies. It has been successfully applied in diverse areas such as manufacturing, chemical production, performance analysis, and more recently in software process evaluation and analysis. A rigorous method for empirical studies of the software engineering domain should prove a valuable tool in advancing knowledge in the profession and contributing to establishing objective evidence of best practices for industry.

5. REFERENCES

1. Basili, V., "A Plan for Empirical Studies", In the Proceedings of the 1st Workshop on Empirical Studies of Programmers, Washington, D.C. June 5-6, 1986.
2. Curtis, B., "By the Way, Did Anyone Study Real Programmers," In the Proceedings of the 1st Workshop on Empirical Studies of Programmers, Washington, D.C. June 5-6, 1986.
3. Eickelmann, Nancy S., "Measuring and Evaluating the Software Test Process." European Software Measurement Conference, FESMA `98, Antwerp, Belgium, May 6-8, 1998.
4. Eickelmann, Nancy S. and Richardson, Debra J., "Leveraging the Cost of Software

Testing with Measurable Process Improvement," In the Proceedings of the Computing in Engineering Conference, ETCE-ASME `97, Houston, Texas, January 28-30, 1997.

5. Eickelmann, Nancy S., Evaluating Investments in Emerging Test Technologies. Seventeenth International Conference on Testing Computer Software. Bethesda, MD, June 12-16, 2000.

QW2002 Paper 6P2

Mr. Ralph Dalebout
(IBM Corporation)

Beta Testing -- Boot Camp Basics and Beyond

Key Points

- Testing needs and value related to beta testing.
- Review of a successful program refined and honed over a number of years.
- Extensions which can be made to beta test programs.

Presentation Abstract

Today's competitive world demands rapid product development and deployment of new products. While the time-to-market window is shrinking, the sheer number and complexities of product and solution tests are increasing. Beta test environments encompassing complex, high-availability, worldwide networked environments are very often required to test end-to-end hardware and software solutions.

The key to successful beta testing is adapting to new challenges. The shrinking "time-to-market" schedules dictate shorter development and test cycles. There are increased product complexities with extensive hardware and software interdependencies. It is increasingly difficult to mirror the "real world" customer environments within internal test labs. Beta testing can be a very time and cost effective means of augmenting and complementing "core" testing programs.

The IBM Printing Systems Division (PSD) has implemented a comprehensive beta test program and has achieved a high level of success over a number of years. An eighteen step process has evolved, which covers beta test procedures from product inception to product introduction and release.

About the Author

Ralph Dalebout is a Beta Test Project Manager in the IBM Printing Systems Division's Development Test and Support group. He manages hardware and software Beta Tests with customers inside and outside IBM. Ralph's career with IBM spans 30 plus years. He has held a number of different positions in marketing (IBM Systems Engineer and Account Representative), product development (Product Planner), competitive analysis / market research and product test.

Ralph has a degree in Mechanical Engineering from the University of Utah and a MBA degree from the University of Washington.

Beta Testing Boot Camp Basics and Beyond

Ralph Dalebout

Phone: (303) 924-8130
E-mail: dalebout@us.ibm.com

IBM Printing Systems Division
Boulder, Colorado

Introduction

Quick IBM Printing Systems Division (PSD) Profile

- IBM has developed and manufactured printers for over 40 years.
- 5,000 employees worldwide.
- Full Spectrum of Product Development and Delivery ... Research, Development, Manufacturing, Marketing, Maintenance and Professional Services.
- Wide range of products (hardware, software and supplies) and services (maintenance, professional and custom) in 130 countries worldwide.
- Specializes in print solutions for the enterprise (large and small), delivering customized and comprehensive "output management" solutions.

Market Realities and External Forces That Keep Us From A Perfect World

- Today's competitive world demands rapid product development and deployment of new products. While the time-to-market window is shrinking, the sheer number and complexities of product and solution tests are increasing.
- The focus of testing is naturally migrating from specific hardware and/or software testing to customer "solution" testing. One of the challenges is understanding how a new product will fit into the customer's environment from a "work flow" point of view. We can no longer ask customers to adapt their operations to our new and enhanced products.

Why Beta Test?

- A number of enterprises and organizations don't Beta test products and solutions. Some have tried it and discontinued Beta testing for various reasons.
- Shrinking "time-to-market" schedules dictate shorter development and test cycles. At the same time, there are increased product complexities with extensive hardware and software interdependencies. It is increasingly difficult to mirror the "real world" customer environments within internal test labs.
- Alpha and Beta tests are the first place where the developed product or solution and the customer are introduced to one another. In order to assure delivery of a solid product or solution to the customer, Beta tests are a natural way to test in a complete customer environment, with a focus on the end-to-end customer solution. For printing products, the customer is often willing to use the product in complex environments and with production applications and systems. This customer capability inevitably finds product defects and problems that cannot be found in the test lab that has a strictly printing focus.
- The successful integration of Beta testing into your overall Quality Assurance strategy, procedures and processes can improve development time and reduce your overall development costs! In addition, it can significantly improve your confidence index relative to the acceptability of the product in real customer environments, before you formally release the product.

Major Beta Test Objectives

- Evaluate products and offerings in customer environments.
- Remove product defects and improve and correct product documentation during the Development and Test cycle.
- Provide feedback to Development, ID (Information Development) and Service prior to the general release of the product or offering.
- Strengthen the working relationship between the customer and IBM.
- Obtain early experience with new products using normal technical support channels on a controlled basis.

Beta Test Phases

- Planning Phase
- Customer Nomination Phase
- Customer Selection and Commitment Phase
- Execution Phase
- Completion and Wrap Up Phase

Beta Test Phase Details

Planning Phase

- Define and convene Beta Test core team.
- Define the functional attributes and/or applications that need to be tested.
- Define reliability, availability and serviceability attributes to be tested.
- Establish general test procedures, Beta Test metrics and test entry / exit criteria.
- Establish the number of Beta Tests (customers) needed and the duration of each.
- Develop initial Beta Test plan.
- Solicit input form the various support organizations, who will be involved in the Beta Test.
- Publish final Beta Test Plan and distribute to the implementation. Team.
- Describe customer profiles required to ensure successful Beta Tests.
- Identify the customer's technical support profile to ensure a successful Beta Test.
- Rank priorities of all Beta Test customer variables for selecting final Beta Test customers.
- Review the Beta Test Plan and selection criteria with the Customer Marketing and Service account teams.
- Create customer questionnaire.

Customer Nomination Phase

- Provide Marketing qualification criteria for Beta Test candidates.
- Have Marketing submit nominations for Beta Test candidates and provide customer profiles.
- Obtain initial approvals and legal clearances to disclose unannounced product(s) and offering(s) to customer.
- Formally disclose unannounced product(s) to customer.
- Have customer complete profile questionnaire.

Customer Selection and Commitment Phase

- Review Beta Test candidates profiles and select accounts which best fit the selection criteria.
- Develop and review a draft State of Work describing the responsibilities of both parties and the guidelines of the Beta Test with the customer.
- Finalize the Beta Test Statement of Work and have both parties sign.

Execution Phase

- Conduct Beta Test Readiness Review
- Schedule kickoff meeting or conference call for the Beta Test.
- Initiate the Beta Test.
- Conduct weekly status meetings / conference calls to monitor the Beta Test.

Completion and Wrap-up Phase

- Obtain consensus with the customer the Beta Test has completed on or about the Beta Test target completion date.
- Have customer complete Final Beta Test Survey form, including any suggestions concerning the Beta Test.
- Publish final Beta Test Report.
- Combine Beta test results with normal Test results for an overall Risk Assessment for the product.

Key Fundamentals and Rules For Beta Test

Beta Test Ground Rules

- Initially (and on a continuing basis), manage the expectation level with the customer and the customer account team. Explain in great detail what the Beta test is and what is isn't. Explain what the impact could be on the customer's resources and any potential risks.
- There is no cost or obligation for the Beta Test to the customer.
- The software and/or the hardware is provided at no charge to the customer. The understanding is the customer will return the hardware (e.g. printer) at end of the Beta Test. In the case of software, the customer agrees to delete the Beta Test software from their system(s).
- In some cases, we will even provide the required hardware (e.g. server) for a software Beta test.
- In the case of printer Beta Tests, basic supplies (e.g. ribbons and toner) will be supplied at no charge for the duration of the Beta Test.
- The Beta Test Statement of Work is intended to explain the general responsibilities of the two parties. Either party has the option of terminating the Beta Test prior to its normal completion.
- The level of expectation has to be carefully set with the customer prior to initiating the Beta Test.
- Normal product support channels are used for Beta tests. For problems, the customer call the regular 1-800 service support number and a formal problem is opened in the problem tracking data base.

Management Commitment

- Contrary to the adage "The best things in life are free!," there is no "free lunch" in test. Beta Testing offers a number of potential benefits, but the benefits come at a price. And the "cost" of Beta Tests goes beyond the Test organization. It extends to all the support organizations which have to participate in the Beta Test.

- Management in all the organizations which participate in the Beta Test programs needs to be committed to Beta Testing.

Key Requirements of Successful Beta Testing

Planning Phase

- Complete management chain buy-in and commitment.
- You need to pull together the Beta Test core team (Marketing, Software Development, Engineering, Information Development (Publications), Education, Product Test and Technical Support and Service) very early and insist on full participation for the duration of the Beta test.
- You need to initiate the planning phase very early. This will force the different organizations to focus on the specific requirements for the Beta Test.

Customer Nomination Phase

- Define in detail the customer profiles (environment, required installed software and hardware, technical support, etc.) describing the customers you are looking for. Exclude competitive situations, where the Beta Test is being used to counter competitive activities or pressures.
- Communicate request for nominations to field operations and personnel.
- Avoid problem accounts.

Customer Selection and Commitment Phase

- Assess whether the new product / offering has direct value to the customer. There needs to be "pony" in the Beta Test for the customer and a "pony" for the product / offering provider.
- Assess whether the customer is fully committed to the Beta Test from a customer management point of view.
- Assess whether the customer has the technical resources to effectively participate in the Beta Test. Many IT organizations are overcommitted and adding another series of "To Do's" to their workload isn't realistic.
- Assess whether product / offering provider has the local resources and commitment to support the Beta Test. Competent / on-site personnel supporting the Beta Test can significantly insure the success of the Beta Test.
- Developing a Beta Test customer profile / selection criteria matrix can help simplify the customer selection process. Developing a weighting scheme for the criteria will allow you to come up with a numeric value to rank the candidates with.

Execution Phase

- Closely monitoring all the Beta Test activities is key to a successful Beta Test. You need to make sure there is complete follow through and closure on all the problems, action items and To Do's which surface in the Beta Test and in a timely fashion.
- Follow-up ... Follow-up ... Follow-up.
- Flatten problems quickly.

Probability of Success and Beta Test Shortcomings

- Realistic Expectations
  - A number of Beta tests will be unsuccessful.
  - A number of Beta tests will be moderately successful.
  - A very few will be highly successful!
- Shortcomings and Exposures
  - Mergers and acquisitions impacting IT operations.
  - "Resource" actions impacting staffing of Beta test program
  - Regular "Day Job" priorities.
  - Changing management support commitments.

Beyond Traditional Beta Testing - Improving The Odds

- Development Partnership Program (DPP)
  - Select group of customers (based on previous Beta test experience).
  - Continuity - two year program and renewable.
  - Known environments, known personnel and working relationships ... predictable results.
  - Eliminate the twelve steps (and associated time) involved in the Beta Test Planning Phase.

Summary

- Beta testing has a very positive payoff for IBM.
- Beta testing augments and complements basic testing programs.
- Managing a successful Beta test program presents many challenges.
- New approaches (e.g. Development Partnership Program) can shorten time table and achieve a higher probability of success.

QW2002 Paper 7P1

Mr. Hans Buwalda
(LogiGear Corporation)

Managing Test Products, The Next Challenge

Key Points

- Issues in managing testware
- The significance of a product model
- Practical solutions for testware management

Presentation Abstract

Software development is organized around products. Code is produced in source files, data models are based on descriptions and documentation is available as documents and help files. Tests however seem to move away from that model now. More and more tests are kept as a collection of single test cases! Also automation scripts are maintained within test tools and are not treated the same way as other software. Version control and configuration management are likely victims and maintainability gets under pressure. This talk will take the audience through the main considerations that apply in managing tests and scripts as what they are, namely development products.

About the Author

Hans Buwalda, ABT Chief Architect, leads LogiGear's Action Based Testing™ (ABT) research and development, including ABT Toolset™ operations, and oversees the practice of the ABT methodology. Prior to joining LogiGear, Hans served as Project Director at CMG in The Netherlands where he was the original architect behind what has now become TestFrame™ - an integrated method for planning, managing, and deploying software testing and test automation. Hans is an internationally recognized expert specializing in test automation, test development, and test management. He speaks and presents workshops at international conferences on testing concepts such as Action Based Testing, The Three Holy Grails of Test Development, Soap Opera Testing, and Testing in the Cold. Hans authored (along with Dennis Janssen and Iris Pinkster) Integrated Test Design and Automation: Using the TestFrame Method (Addison Wesley 2001). He holds a Master of Science degree in Computer Science from Free University, Amsterdam.

LogiGear Expert Testing... Real World Solutions.

QualityWeek 2002

Presentation 7P1

Managing Test Products

Hans Buwalda

LogiGear® Corporation

(C) 2002 Logigear Corporation, all rights reserved

Key Issues in Testing

- Appropriate Test design
- Comprehensive automation architecture
  - manageable, maintainable
- Management of the test process
  - managers want to know what is going on
- Documentation
- Clear and useful reporting
  - progress, results
- Quality Assurance
  - efficient and effective involvement of stake holders, users, auditors
- Management of the tests
  - tests and test scripts are products that need to be managed

(C) 2002 Logigear Corporation, all rights reserved

What is a Product

- Dictionary: "Something produced by effort, or some mechanical or industrial process"
- In our case a product can be a combination of products, we will call that a complex product
- What is Important for a Product
  - To produce it
  - To manage it

(C) 2002 Logigear Corporation, all rights reserved

Examples of Software Products

- Source files
- Data models
- Design documents
- Manuals
- Help files
- Binaries and intermediate files

(C) 2002 Logigear Corporation, all rights reserved

Is a Test to be Considered a Product

- "No way, it is an activity to improve the quality of other products"
  - can be a relevant angle
  - at least some testers like to work this way (and are good at it)
  - see also: exploratory testing (James Bach)
- "Sure, it is the outcome of a test development effort"
  - either manual test plans or automated tests
  - for a large group more motivating to be a test developer
  - easier to audit or otherwise control

(C) 2002 Logigear Corporation, all rights reserved

Advantages of a Product Angle

- Easier to be structured and systematic
- Long term value re-use value
- Good basis for audits
- If automated, repeatable execution
- Can be developed well in advance

(C) 2002 Logigear Corporation, all rights reserved

Disadvantages of a Product Angle

- More work
- Tends to be less aggressive (tame)
- Inhibits variation
- Products have to be managed to keep their value

(C) 2002 Logigear Corporation, all rights reserved

Source Products Come in Levels

- System Source, consisting of
- Source files, consisting of
- Functions, consisting of
- Statements, declarations, etc

common level of management

(C) 2002 Logigear Corporation, all rights reserved

Levels in Test Products

- Test suite, consisting of
- Application test, System Test, etc, consisting of
- Test Plans, consisting of

typical focus for manual tests

- Test Cases

lowest level in test management tools

(C) 2002 Logigear Corporation, all rights reserved

Comparison with Software Development

- Test plans are typically documents, (more or less) like source files, designs and manuals
- Test cases could be compared to functions, but there are differences:
  - a test case is usually self contained, not related to other test cases
  - it is possible to have logical and physical test cases
  - closest similarity in software are object or components
- In more and more projects part of the test cases are implemented with automated test scripts
  - typically on case by case basis

(C) 2002 Logigear Corporation, all rights reserved

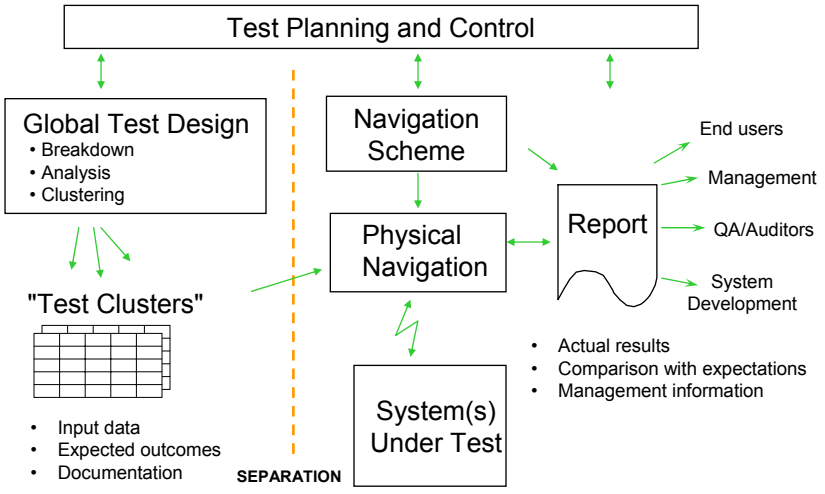
Basic Principles of Action Based Testing (ABT)

- Based on a notion that a test can be broken down in a number of consecutive actions
- Not only the test data but also the actions are in the test file, they are defined by the tester
- Instead of implementing test cases, the engineer concentrates on the programming individual actions
  - we call this person the "navigation engineer" or "navigator"
- Reporting is done at the level of the test
- Most actions are high-level
  - test oriented instead of object oriented

enter customer	Jones
instead of	
push_button	New
type_keys	Jones
push_button	Process

(C) 2002 Logigear Corporation, all rights reserved

Test Product Life Cycle



(C) 2002 Logigear Corporation, all rights reserved

Example of an ABT Test Cluster

cluster	**EXAMPLE OF A TEST CLUSTER**			
version	1.0			
author	Hans Buwalda			
section	**1. Entering clients and balances**			
	last name	first name	account nr	balance
enter client	Green	John	458473948	1500
enter client	Wood	Anna	422087596	2100
section	**2. Money Transfers**			
	from	to	sum	
transfer	458473948	422087596	500	
transfer	422087596	785793025	1201	
section	**3. Checking names and numbers**			
	account nr	last name	first name	
check name	458473948	Green	John	
check name	422087596	Wood	Anna	
	account nr	sum		
check balance	458473948	1000		
check balance	422087596	1399		

HEADER

INPUT

EXPECTED OUTCOMES

(C) 2002 Logigear Corporation, all rights reserved

ABT Reporting Example

TEST RUN SUMMARY

test cluster: Minibank business test
test sheet: demonstration scenarios
test version: 1.2
test author: Hans Buwalda, all rights reserved
test date: April, 2001

run date and time: Tue Nov 06 12:49:06 2001

SCENARIO MB01 -- Entering customers using manual numbering
In this scenario the account numbers are entered manually.

Enter Customers - Entering 5 customers in the Relation Entry screen

14, enter customer: Johnson Jean 500103381 1500

15, enter customer: Juet Christian 423137538 2100

16, enter customer: Savy Anne 848656467 1700

⋮

(C) 2002 Logigear Corporation, all rights reserved

Reporting Example Continued

:

Check Balances - Check if the balances are conform the entry and the transfers

36, check balance: 500103381 1000
check: balance
expected: 1000
recorded: 1000
result passed

37, check balance: 423137538 555
check: argument 2
expected: 555
recorded: 2600
result failed

RESULTS TEST EXECUTION

overall statistics for this test run:
number of checks: 21
number of passes: 20
number of failures: 1

fail(s) were found in these line(s):
37

(C) 2002 Logigear Corporation, all rights reserved

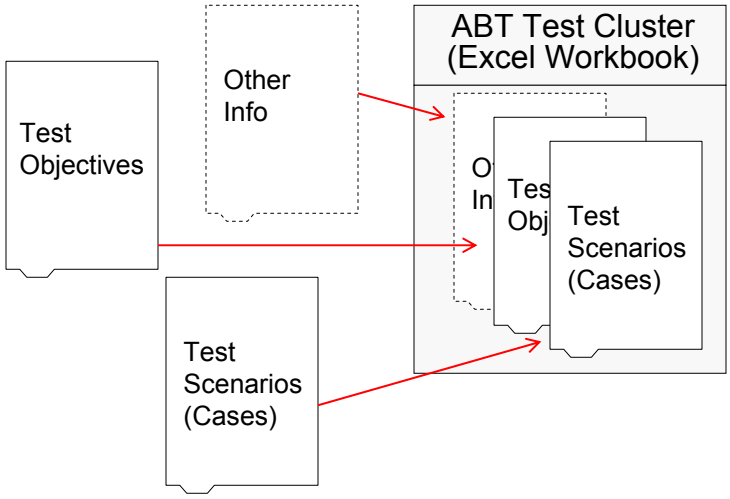
Test Objectives

...
TO-3.51 The exit date must be after the entry date
...

test objective	TO-3.51		
	name	entry date	exit date
enter employment	Bill Goodfellow	2002-10-02	2002-10-01
check error message	The exit date must be after the entry date.		

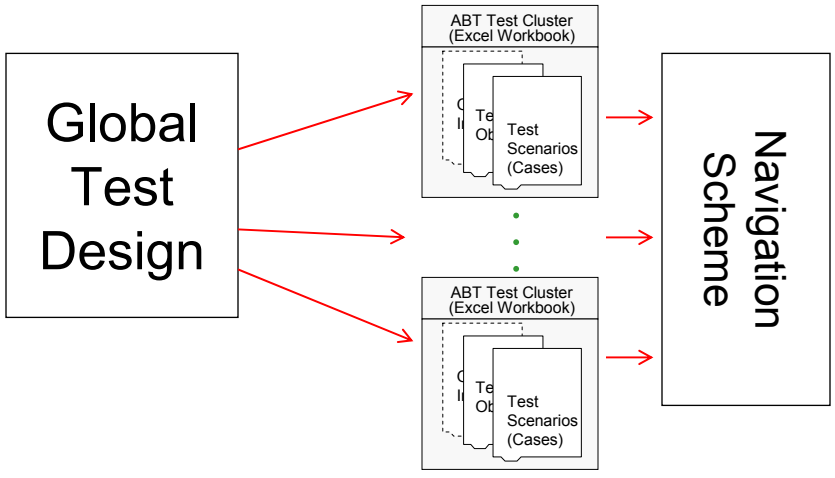
(C) 2002 Logigear Corporation, all rights reserved

Build Up of an ABT Test Cluster



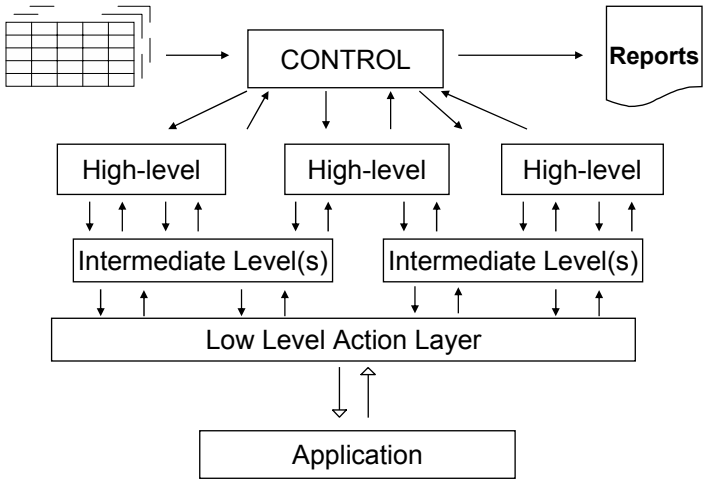
(C) 2002 Logigear Corporation, all rights reserved

ABT Test Project



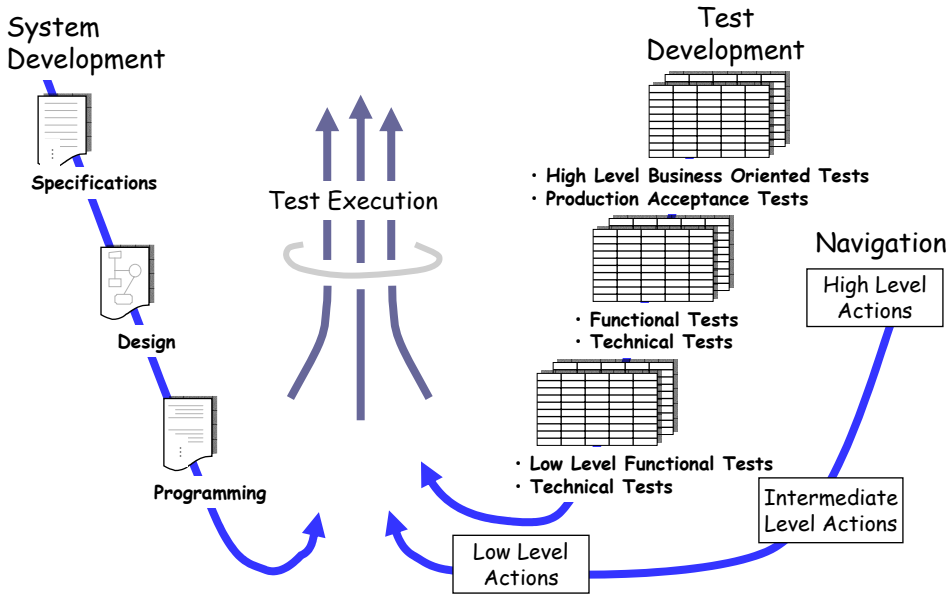
(C) 2002 Logigear Corporation, all rights reserved

Navigation Scheme



(C) 2002 Logigear Corporation, all rights reserved

Product in Perspective



(C) 2002 Logigear Corporation, all rights reserved

Our Take on Test Products

- The test cluster are treated as source files
  - interpreted by the navigation scheme
  - "compiled" into a test management tool or environment
- The test plan identifies the clusters
  - no test cases in a test plan, they go in the clusters
- Product Management is done at cluster level
  - test cases have a documentary role
- Results are also regarded as products
  - "produced by some mechanical or industrial process"
- We do not write automated scripts for test cases
  - in stead we have a navigation scheme implementing the actions (which also consists of products, not treated in this presentation)

Good "clustering" (cluster identification) is essential

(C) 2002 Logigear Corporation, all rights reserved

Clustering Recommendations

- Logical to all concerned
- Independent from other clusters
- Well differentiated and clear in scope
- Fitting the priorities and planning of the project
- Balanced in size and amount

(C) 2002 Logigear Corporation, all rights reserved

Examples of Clustering Criteria

STRAIGHTFORWARD

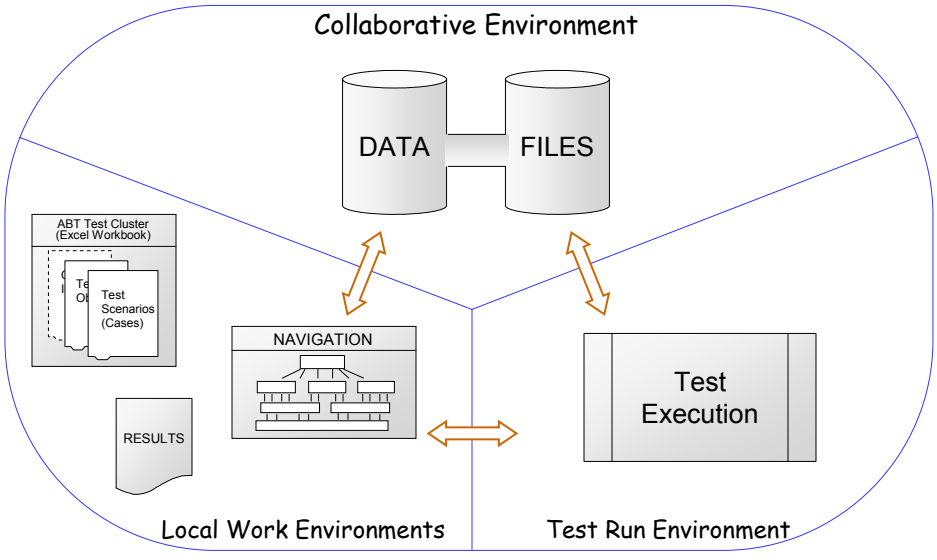
- Architecture of the system under test
- Functionality and other requirements
- Quality attributes
- Level of detail

ADDITIONAL

- Planning and control
- Level of risks involved
- Complexity of the test
- Technical aspects of test execution
- Stake holders
- Code hand-offs (Brian Marick)

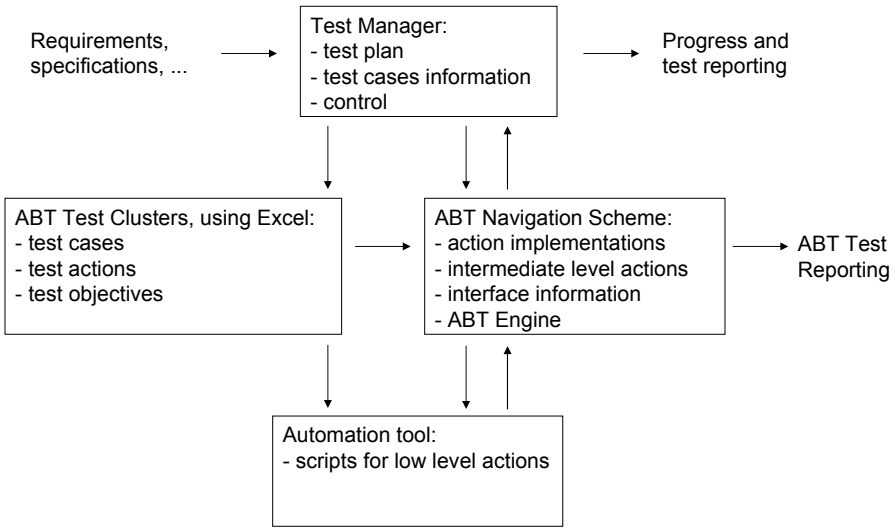
(C) 2002 Logigear Corporation, all rights reserved

Three Environments for Products



(C) 2002 Logigear Corporation, all rights reserved

Fitting in with a Test Management Tool



(C) 2002 Logigear Corporation, all rights reserved

Advantages

- The (explicit) identification of clusters has proven a good basis for control of the entire test process
  - we call this "clustering"
- Work assignment is more straightforward
  - based on developing, assessing, running and maintaining clusters
- There is a local and central environment, making it easier to organize work
  - just like with program sources
- Having a well delimited document makes it easier to communicate, manage, assess etc

(C) 2002 Logigear Corporation, all rights reserved

Summary

- Tests can be regarded as products
  - but that is not always necessary or even desirable
- A test set can be developed and managed as a structured collection of cases or as a set of documents
  - both angles have advantages
- ABT treats test development like software development,
  - test clusters as source documents
  - with a local and a collaborative environment
- When managed as test clusters, the clustering is essential

(C) 2002 Logigear Corporation, all rights reserved

Some References

- Buwalda, Hans, *Testing with Action Words, Abandoning Record and Playback*, Eurostar 1996
- Buwalda, Hans, *Testing with Action Words*, STAR 1998
- Buwalda, Hans, and Kasdorp, Maartje, *Getting Automated Testing Under Control*, STQE Magazine, November Issue 1999
- Buwalda, Hans, *Soap Opera Testing*, STAR East 2000
- Buwalda, Hans, Janssen, Dennis, and Pinkster, Iris, *Integrated Test Design & Automation Using The TestFrame Method*, Addison Wesley, 2001
- Buwalda, Hans, *The Three Holy Grails of Test Development*, Quality Week 2001
- Graham, Dorothy, and Fewster, Mark, *Automating Software Testing*, Addison Wesley, 1999
- Kaner, Cem, Bach, James, and Pettichord, Bret, *Lessons Learned in Software Testing*, 2001 (expected)
- Kaner, Cem, Nguyen, Hung Quoc, and Falk, Jack, *Testing Computer Software, 2nd Edition*, John Wiley & Sons, 1999
- Kit, Edward, *Software Testing in the Real World*, Addison Wesley Longman, 1996
- Marick, Brian, *New Models for Test Development*, www.testing.com/writings/new-models.pdf, 1999
- Nguyen, Hung, *Testing Applications on the Web*, Wiley, John & Sons, Incorporated, 2000
- Splaine, Steven, and Jaskiel, Stefan, *The Web Testing Handbook*, STQE Publishing, 2001

(C) 2002 Logigear Corporation, all rights reserved

QW2002 Paper 7P2

Mr. John Dalbey
(Cal Poly Computer Science Department)

A Climber's View of Software Quality

Key Points

- Rock climbers have some important lessons about QA to teach software developers
- 3 Lessons: Have a Process, Seek Peer Review, and Focus on Details
- Quality is ultimately an individual activity, yet cultural climate is essential.

Presentation Abstract

The sport of modern technical rock climbing has some important similarities with software development. They are both demanding technical disciplines, often performed under pressure, where even a trivial mistake may have profound consequences. This paper will examine the quality assurance activities undertaken by rock climbers and consider the implications for software developers.

About the Author

Dr. John Dalbey is an experienced rock climber and wilderness guide and leads mountaineering courses for Outward Bound. He is also on the Computer Science faculty at California Polytechnic State University in San Luis Obispo, California, where he teaches software engineering.

A Climber's View of Software Quality

Dr. John Dalbey

QualityWeek 2002

© Copyright 2002 Dr. John Dalbey

Author Biography

- Faculty member, Computer Science Dept. Cal Poly, San Luis Obispo
- Wilderness Guide, Pacific Crest Outward Bound School
- 25 years rock climbing experience

Climbing and Software?

Important similarities:

- Demanding technical disciplines.
- Performed under pressure.
- Tiny mistakes have profound consequences.

Factors in "big wall" climbing

- Multi-day climb.
- Haul all water, food, and equipment.
- Complicated logistics.
- Eating and sleeping on the cliff.
- Predominantly "direct aid" climbing (as opposed to "free" climbing).
- Gear intensive.

QA in Climbing

- Scaling vertical cliffs presents obvious hazards.
- Modern rock climbing is a very technical pursuit.
  - Sophisticated equipment.
  - Technical procedures and skills.
- QA is essential to the sport - called "safety."

What can software professionals learn from climbers?

- Many aspects of climbing "safety" are analogous to QA in the software world.
- QA for climbers must be simple, lightweight, practical, and reliable.
- Considering QA in this framework can help software professionals focus on the essentials.
- Creating a "culture" of quality.

Climbing's Three QA Practices

- Follow a process.
- Focus on details.
- Seek peer review.

Practice 1: Follow a Process

- There are technical procedures for almost all aspects of climbing safety:
  - Tying knots.
  - Anchor setup.
  - Communication protocols.
- While there may be several acceptable solutions, good climbers establish a personal process and follow it religiously.

Benefits of Process

- Avoids errors.
- Simplifies verification.
- Efficiency.
- Avoid complex decisions or judgements under pressure.

Practice 1: Anecdotes

- Stranded rappellers - First person neglected to untie knot.
- Stranded rappeller - failure to back clip, red tree ants, dropped rope.
- Salathe Wall - “redundant anchor points” saved lives.

Practice 2: Focus on Details

- Keen awareness of details is crucial as even small mistakes (e.g. forgetting to screw the gate shut on a carabiner) can be fatal.
- Three mental attitudes support the required focus:
  - Do one thing at a time.
  - Do each thing right.
  - Fix it now.

Do One Thing At A Time

- Stay focused on the task at hand so as not to be distracted at a crucial moment.
- Anecdote
  - Lynn Hill: unfinished tie-in.

Do Each Thing Right

- It's easier to maintain quality from the start than to try to repair it later in the process.
- Anecdotes
  - cleaning tool snags shoe lace.
  - sloppy rope stacking causes stuck rope.
  - shirt jams rappel device.

Fix It Now

- Complex systems are highly coupled - defects propagate quickly.
- Fix each small anomaly before it multiplies.
- It's cost effective to fix defects close to time of injection.
- Anecdote
  - unanticipated haul bag weight snarls rope system.

Practice 3: Seek Peer Review

- Climbers solicit peer review of their work.
  - Inspect tie-in before starting ascent - "How do I look?"
  - Inspect anchor setup.
  - Inspect rappel brake before descending.

Culture of Safety

- A culture of safety can strongly influence the extent to which safety is practiced.
- Numerous subtle and overt behaviors provide social cues as to safety expectations.
- Senior or lead climber can model desired standards.

Behavioral "Cues"

- Inquiring about partner's process.
- Asking for peer review.
- Articulating process steps.
  - Knot tying.
- Articulating checklists.
  - "buckle, tie-in point, knot"
- Allowing time for anomaly correction.
- Consistent use of communication protocols.

Lessons for Software Developers

- A few key QA practices can make a big difference.
- A "culture" of quality influences individual practices.
- If human lives depended on your software how would that change your practices?

Where to learn more - climbing

- *Mountaineering: The Freedom of the Hills* 1992, The Mountaineers
- Rock Climbing Instruction in Yosemite
  - Yosemite Mountaineering School www.yosemitepark.com/html/mountain.html
- For information on Wilderness Adventure Challenge courses:
  - Pacific Crest Outward Bound School www.pcobs.org

Where to learn more - quality

"Personal Software Quality"

Five day course based on W. Humphrey's *A Discipline for Software Engineering.*

Redpoint Research
www.redpoint.org

QW2002 Paper 8P1

Karen S. King
(King Consulting)

Common Problems in Tool Adoption

Key Points

- Common causes of shelfware in tool adoption
- Solutions to those problems
- Example process for integrating tools

Presentation Abstract

Many organizations acquire new tools, and six months later, find they have shelfware rather than a tool that improves probability. This paper describes eight common problems, along with examples from real-life organizations. Examples will be from a variety of different tool acquisitions, such as defect tracking systems, configuration management, code coverage tools, testing tools, as well as project management tools. It continues to provide different solutions for each problem in a format readers can integrate into their own organizations. Finally, it provides a process for introducing tools that integrates all solutions.

One of the most common challenges is the motivation for introducing a new tool. Many organizations decide to adopt a new tool because someone read an article in a journal. An individual reads about the importance of measuring code coverage as part of testing. He looks at reviews, and selects the tool with the highest rating. He brings the tool in, and asks the testing group to start using the tool. The testing group agrees, and everyone expects testing will improve because code coverage information is now being collected. The paper will explain the problems leading to this situation, as well as information on how to prevent the problems.

About the Author

Karen King, founder and principal consultant of Quality Improvement Solutions, has a special interest in implementing improvement projects. With 20 years experience in software quality improvement, she is widely known for her work in using defect metrics to direct process improvement initiatives. She has participated in the software development process in a variety of roles, including: systems programmer, QA lead, applications engineer, supplier engineer, quality manager, IV&V lead, and SEPG chair. Ms. King can be reached at karenk@alumni.rice.edu

Common Problems in Tool Adoption

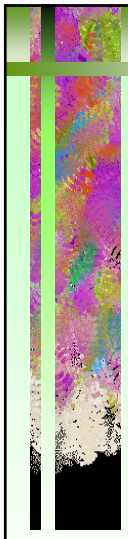
Karen S. King

Quality Improvement Solutions
Karenk@alumni.rice.edu
(503) 626-0274

Software Quality Week

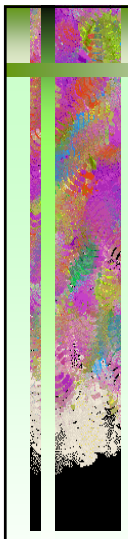
Shelfware

- How many tools are unused on your shelf?
- How many tools aren't used effectively?
- How much money and time have you wasted on products that don't work?



Stories from the Field

- Based on real-life experiences
- Describes problems leading to shelfware
- Includes information on how problems were solved



Organization Not Ready

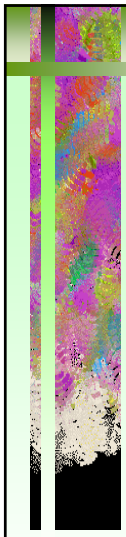
- Know why you want tool
- Automate existing process
- Collect information manually first
- What is your current capability?

Not Enough Buy-In

- Locate all stakeholders
- Within each group, find opinion leaders
- Collect requirements from all stakeholder groups
- Consider politics

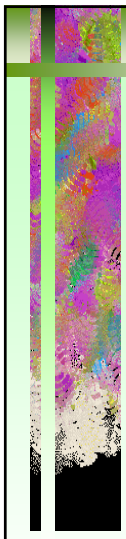
Wrong Tool Selected

- Do your research
- Rank requirements
- Get live demonstrations
- Try tool in-house



Tool Not Integrated into Process

- What processes will be affected by this tool?
- Develop high-level documentation
- Keep process documentation easily accessible



Inadequate Training

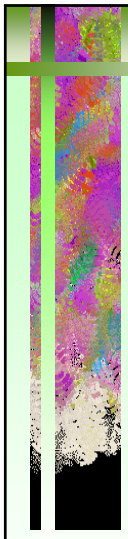
- Train users “just in time”
- Conduct hands-on training
- Develop specific training for new employees
- Have tool experts available to help

No Follow-up

- Schedule follow-up sessions with users
- Develop metrics
- Set realistic expectations at start of project

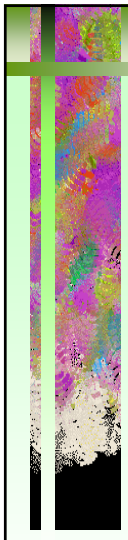
Not Enough Resources

- Provide resources for the improvement project
- Provide schedule relief for users learning new tool



No Plan for Maintenance

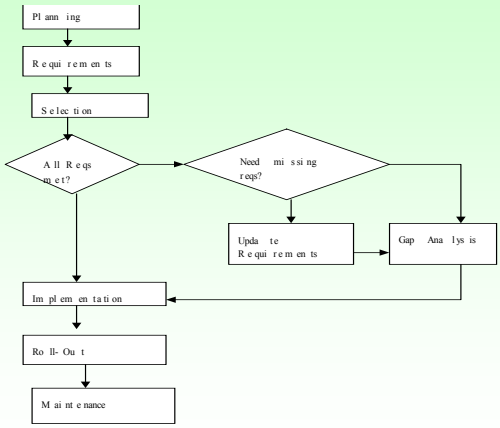
- Be prepared for follow on development
- Develop maintenance plan at project start
- Commit resources to maintaining the tool



Putting It All Together

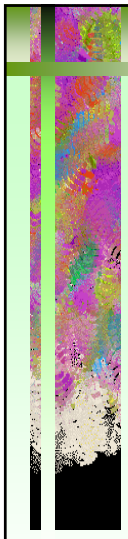
- How do you integrate the best practices?
- Develop a tool adoption process
- High-Level Process included, and can be tailored

High Level Process



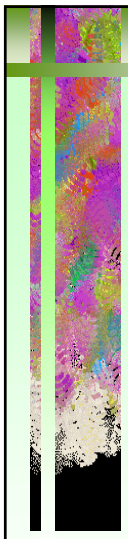
Planning

- Who?
- What?
- Where?
- When?
- Why?
- How?



Requirements

- Maintain under configuration control
- Document owner for each requirement
- Document verification method for each requirement



Selection

- Review tools against all requirements
- Verify requirements using your criteria
- It's possible no tool will meet all requirements
- If no tool meets all requirements, reevaluate your requirements

Gap Analysis

- What will it take to meet requirements?
- Can the tool vendor develop changes?
- Can you develop develop changes?
- Is the project still cost effective?
- Revise plan to reflect new development

Implementation

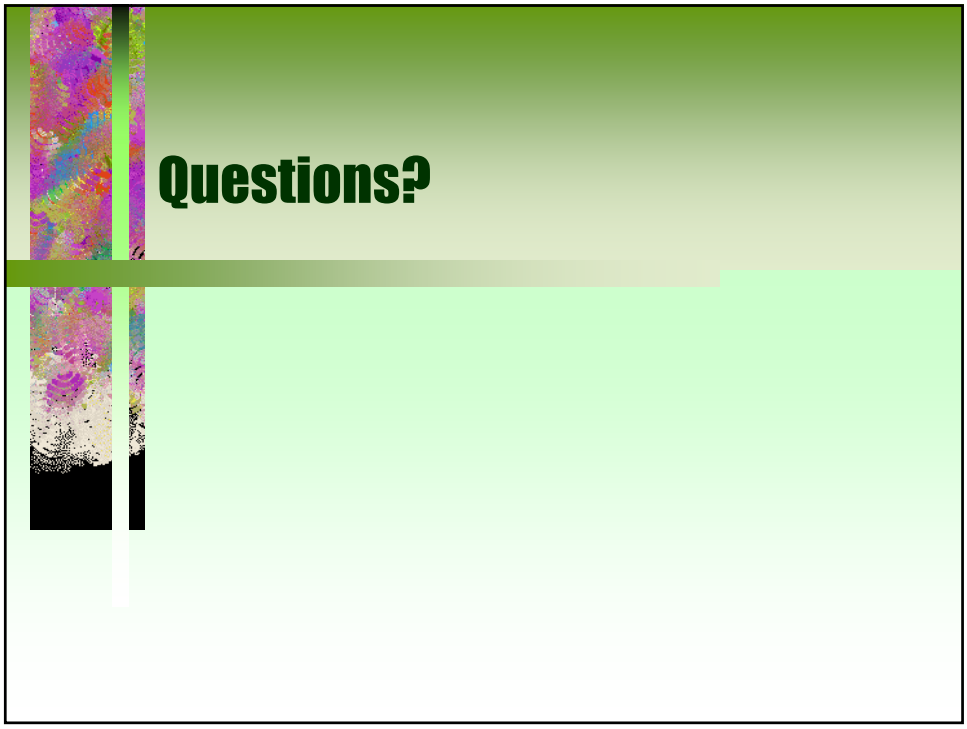
- Develop process for using the tool
- Develop training
- Create documentation
- Develop changes, if any

Rollout

- Introduce new tool to users
- Schedule based on
  - Complexity of tool
  - Number of users
  - Resources for rollout
  - Number of groups
  - Frequency of tool use

Maintenance

- Monitor acceptance
- Collect suggestions from users
- Implement changes and improvements recommended by users
- Provide support to users


Questions?

Common Problems in Tool Adoption

Karen S. King
Quality Improvement Solutions

Abstract

Many organizations acquire new tools, and six months later, find they have shelfware rather than a tool that improves productivity. This paper describes eight common problems, along with examples based on experiences from real-life organizations. Examples will be from a variety of different tool acquisitions, such as defect tracking systems, configuration management, code coverage tools, testing tools, and project management tools. It continues to provide different solutions for each problem in a format readers can integrate into their own organizations. Finally, it provides a high-level process for introducing tools and describes where you'd apply each best practice.

Karen S. King, founder and principle consultant for Quality Improvement Solutions, has a special interest in effectively implementing tools to improve the software process. With twenty years experience in software quality improvement, she is widely known for her work in using defect metrics t o direct process improvement initiatives. She has participated in the software development process in a variety of roles, including: systems programmer, QA lead, applications engineer, supplier engineer, quality manager, IV&V lead, and SEPG chair. Ms. King can be reached at karenk@alumni.rice.edu

Copyright 2001-2002, Karen S. King. All Rights Reserved

How many software tools do you have on the shelf that you do not use? How much money has your organization wasted, hoping to improve productivity? If you are like the typical company, the answer is more than you can count. If you consider tools you use, but not to the extent intended, the number soars. Someone in your organization spent significant effort justifying the purchase of each of those tools, as well as integrating them into your organization. Why don't you use them? Here are stories from eight projects at Hilanderas International, a fictitious company experiencing problems common in real-life organizations. The stories explain how tools become shelfware, along with solutions.

1. Organization not Ready

The Project Jupiter managers heard about a tool to measure code quality. They were concerned about funds spent on maintaining their code, and thought they could decrease their maintenance costs by purchasing a code quality tool. Management established goals for each quality metric, and developers slavishly modified code to meet the numbers. Project Jupiter abandoned the tool when developers complained they could not meet schedule because they were modifying code to meet the goals. The problem was that no one had researched the metrics to understand how to effectively use the measurements.

Solutions:

A) Decide why you want the tool. Explicitly document the benefit to the organization. All users must understand "what's in it for me". If they see no benefit, they will not be interested in using the product.

B) Acquire tools to automate existing processes. If you already perform the function manually, you will simply improve an existing process and the acquisition will more likely be successful.

C) When acquiring tools for new practices, simulate the tool manually to better understand your needs. I once simulated new defect tracking fields by interviewing developers to manually collect data. By collecting the information, I identified useful data fields, and was able to manually calculate metrics that measured progress against our goals. Without collecting actual data, our team would have relied on the literature for data fields and metrics, and the tool would not have met our organizational needs.

D) Consider your organization's ability to assimilate new practices, and the current capability in the practice. Many organizations purchase tools with more capabilities than necessary, figuring they will "grow into it". It would be similar to buying an Indy racecar for a beginning driver, figuring that someday he might want the additional features.

2. Not All Stakeholders Involved in Introducing Tool

Tom has been studying the Capability Maturity Model, and believes a configuration management tool would solve many of the problems on Project Neptune. He found a tool, purchased it, and worked weekends to put the project's code under configuration management. Tom believed everyone would be eager to use the new system because it would provide a rigorous method for tracking past code releases. After the tool was in

Copyright 2001-2002, Karen S. King. All Rights Reserved

place, Tom could not convince his colleagues to use it because they said it violated their personal development processes.

Solutions:

A) List job functions that could be affected by the tool. For a code coverage tool, developers might need to change their coding style, management might be involved in interpreting the metrics, system administrators must maintain the tool, and all testers must enable the tool when running tests. Ensure that you represent all functions in your tool adoption team.
B) Determine opinion leaders who understand the impact of the tool on their communities. These people are the stakeholders. If you involve opinion leaders, other people within the groups will be more likely to accept the new tool.
C) Work with stakeholders to collect requirements you have not anticipated. If you are selecting a defect tracking system, service might require customers be able to submit defect reports and receive limited information on the status of their report. If you only collect requirements from engineering, this higher-level requirement would never surface.
D) Consider the politics in your organization. If there are factions that could oppose your effort, talk to them about their concerns about adopting the new tool. Understand what it would take for them to support your effort. In Tom's case, he did not understand that his colleagues viewed their personal "code stashes" as power, and that they were resentful about Tom's efforts to consolidate the code.

3. Wrong Tool Selected

Jean was tasked to select a defect tracking system for project Apollo. She assembled a team of stakeholders and collected seventy requirements from the team. After researching products, Jean found a tool that met fifty of the requirements. She purchased that tool, but later found the tool would not support custom business rules, and users could not get email when new defects were assigned. Users would not accept the tool because it did not support their business processes.

Solutions:

A) Conduct complete research on the strengths and weaknesses of potential products. One major source of shelfware is acquiring tools at the end of the year, and making a quick decision on a tool because otherwise you'll lose funding. If you acquire a tool without doing the homework, it is almost guaranteed the tool will become shelfware. It would be better to lose funding for a risky tool, rather than have to justify buying a second tool.
B) Rank your requirements. Rather than producing a list of equally weighted requirements, work with your stakeholders to distinguish the most critical, "must-have" features from the "nice to haves". A prioritized list will prevent your selecting a product that meets 90 percent of the requirements, but does not address a critical need.

Copyright 2001-2002, Karen S. King. All Rights Reserved

C) Meet with representatives of the tool vendors and have them demonstrate their tool in your environment. For a code coverage tool, a demonstration might include running some tests in your standard work environment, and verifying the product produces the expected results.
D) Get a demo copy of the tool or a short-term license. Before bringing the tool in-house, have a plan in place on how you'll evaluate the tool, and have resources available to do the evaluation. Determine your minimum criteria to acquire the product, and do not be pressured into purchasing the product until you are satisfied the tool meets the minimum criteria.

4. Tool not Integrated into the Process

Al acquired a tool to collect code metrics. He always ran the metrics tool before he integrated his code. Mary ran metrics after she'd completed unit testing. Project Saturn management noticed the highly divergent data, and decided the code metrics were meaningless for managing projects. No one understood that differing processes caused the data disparity, rather than quality of the tool.

Solutions:

A) If you have process documentation, determine how the tool will best fit within your established development and quality processes. If the tool is complex, you might include a separate work instruction for using the tool and modify the main process to reference the new work instruction. If the tool is straightforward, you could simply modify the current process documentation to include the new tasks associated with the tool. Each function should examine their processes and coordinate changes.
B) If you do not have process documentation, at a minimum, draw a high-level flowchart of your development process. Mark where the tool will be used, and who will be using it. Then describe steps for using the tool. Even without other written process documentation, rudimentary documentation on using the tool increases successful tool adoption. Additionally, after several tools have been introduced and documented, your organization might see the benefit of documenting the remainder of your development process.
C) Make the tool documentation easily accessible. Consider providing a quick reference card or online process help, so users can easily find information about using the tool in the context of your process.

5. Inadequate Training

Joe acquired an estimation tool for the Minerva division. He trained all project managers at once so they would use the tool consistently. Joe tracked when the tool was used, and found project managers were not using the tool consistently. The problem became more pronounced over time. The data in the estimation database became unusable because there was too much variance.

Solutions:

Copyright 2001-2002, Karen S. King. All Rights Reserved

A) Train users "just in time". If they are trained before they need to use the tool, they might not see the relevance of the training (it's not what I'm doing now), and they will not effectively learn. Even if they do learn, they will probably forget their training by the time they need to use it.
B) Provide hands-on training. People need to try new ideas before they can accept them. Create an instance of the tool where users can safely experiment without corrupting live data. For example, when implementing a defect database, have a test database where users can enter defect data and route defects among users without affecting project metrics.
C) Develop training for new employees. It is not uncommon to give new employees documentation and tell them to figure out the tool themselves. Slightly better is when a current user demonstrates the tool to the new employee. Often only less experienced users have time to demonstrate the tools, and the training is inconsistent.
D) Have tool experts available on call to help users with any problems. Users will be more likely to use the tool if they know there are resources available to help.

6. No Follow-up on Tool Deployment

Project Arachne purchased a GUI test tool. The quality assurance engineers wrote tests and developed automated test scripts. About a year later, one of the quality assurance engineers was out during a verification cycle. Their manager assumed the test cases were run automatically, and asked to see the test report. There was no test report because tests were run manually over the past year. When questioned, the QA staff said the tests were run manually because it took more time to analyze the results from the automated scripts than running the tests manually. No one had questioned the effectiveness of the tool, and the tool became shelfware without management's knowledge.

Solutions:

A) Schedule periodic times to talk with users about how the tool is working. Collect information on positives, negatives, and opportunities for improvement. Implement suggestions that will improve the system.
B) Develop metrics to monitor tool usage. For a configuration management tool, you might want to monitor number of check-ins, average size of check-ins, number of code branches, etc. for significant changes. If the metrics vary from past history (positively or negatively), talk with the users to understand why. Do not use the metrics to make judgements about people.
C) Set expectations on the results you expect in the tool adoption. In the short term, the tool will probably not improve productivity. You will not see results until the organization is able to use the tool effectively, which will take multiple experiences. Many companies have abandoned tools before the organization has mastered the new tool, and the result is extreme frustration for everyone.

7. No Resources for Adopting Tool

Bob was working on a critical project on a tight schedule. He was concerned about keeping the project on track, and convinced Hilandares to acquire a project tracking tool. Bob asked his team to use the tool to track their progress on the project. At the same time,

Copyright 2001-2002, Karen S. King. All Rights Reserved

there was extreme pressure on the team to complete the project as soon as possible. The team was overwhelmed, and figured they would enter status information when they had time available. As a result, Bob was not able to use the tool effectively because status information was not up to date. Bob soon became frustrated, and stopped using the tool.

Solutions:

A) Rather than using "volunteer time", work with a sponsor to get a time commitment, preferably from an improvement resource, as well as the people on the tool adoption team. If possible, try to get dedicated resources. If that isn't possible (project is small or organization is small), at a minimum, get project relief for team members. Set realistic expectations about the amount of time required to work on the tool adoption project. In Bob's situation, his highest priority was getting the product out on time. Although he had strong incentives to adopt the project tracking tool, it was a "spare time" activity. If Bob had help from an outside improvement resource, there would have been more effort available to deal with the user resource issues.
B) Before acquiring the tool, ensure all users will have time to learn the new tool. If users are already overloaded, look for ways to offload some of their work to ensure they can use the tool.

8. No Plan for Maintaining Tool

Susie worked with customer service to acquire a customer complaint system. Through detailed interviews with customers, Susie found the most critical aspect was usability, and Hilanderas' customers wanted a tool with pulldowns to easily select answers. Susie and her team implemented the system, and the customers were very pleased. Shortly afterwards, the number of complaints logged in the system decreased drastically. Hilanderas management believed they were being more responsive to their customers' complaints, which resulted in happier customers. Instead, the salespeople reported the customers were frustrated because they could not report their actual complaints because the selections on the pulldowns were too limited. Although the product offerings had changed, the product names in the pulldowns had not been updated. No one had been assigned to maintain the system after roll out, and what had been designed as a user-friendly system quickly became unusable.

Solutions:

A) Develop a maintenance plan at the beginning of the project. Be sure to provide enough resources to answer user questions, as well as make necessary changes to the system. The maintenance plan should be updated as the team learns more about the final solution.
B) Include extra maintenance time throughout roll out and into production. The new system is almost never perfect, and users will need help in successfully transitioning to the new system. Also, users always have suggestions on ways to improve the system. It is important to be responsive to new ideas to help users accept the new system.

Copyright 2001-2002, Karen S. King. All Rights Reserved

Putting it All Together

Now you have ideas about improving the way you adopt tools, but how do you put it all together? It's helpful to have a collection of best practices, but the next step is to determine how the practices can be incorporated into your process. Do you currently have a process for adopting tools? If so, do you follow it? This section includes a high-level process for tool acquisition you can use as an outline to define your own specific process.

Typical Tool Adoption Process

Figure A shows a basic tool adoption process that you can tailor for your organization. Customize this process to include your specific steps. When developing the process, include guidelines for determining ways to tailor the process for different tools. For example, you might not want to follow all steps in the process when adopting a simple tool that all users accept. When determining processes, also consider the consequences of failure. If critical business processes are dependent on successfully adopting the tool, it is more important to follow the complete process.

Planning – Prepare for the project by developing an improvement plan covering all aspects of the project. The plan should be documented, agreed to by all major stakeholders, and updated when there are changes to the project. (1A,1B,1C,1D,2A,2B,2D,6C,7A,7B,8A,8B)

Requirements – Collect requirements from all stakeholders. As a group, reach consensus on the priorities. Document how each requirement will be verified. Maintain all requirements under configuration control to keep track of requirements changes. Document an owner for each requirement. The owner will be the expert on the requirement. If there are any questions or issues related to the requirement, there will be a single focus. (2C,3B)

Selection – Choose the supplier that best meets your requirements. Note that the best supplier might not meet all requirements. If not, revisit your requirements to reevaluate the importance of missing requirements. (3A,3C,3D)

Copyright 2001-2002, Karen S. King. All Rights Reserved

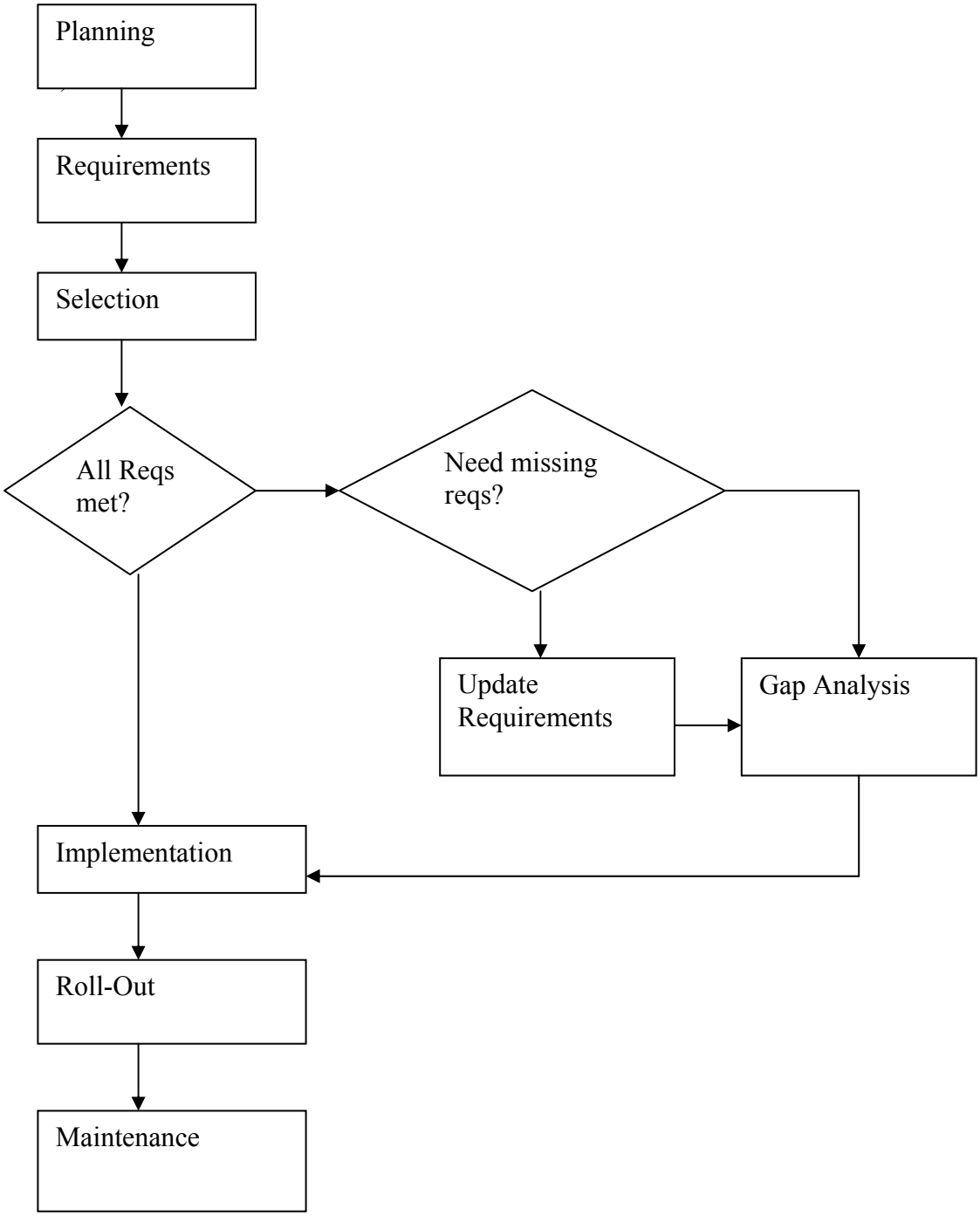


Figure A: Basic Tool Adoption Process

Copyright 2001-2002, Karen S. King. All Rights Reserved

Gap Analysis – If no product meets all requirements, consider either developing your own customizations, or negotiate with the tool vendor to have customizations developed. If you need customizations, treat the development as a miniature development project, including project planning, requirements, project tracking, etc. Note that it might not always be cost effective to continue the project if the gap is too great. At this point, evaluate whether the project is still cost effective. Update the plan to reflect changes from the gap analysis.

Implementation – Develop the new process, training, documentation, etc. If there is any product development required, it would be done during implementation. Track progress against the plan throughout the implementation phase. (4A,4B,4C,5A,5B,5C,6B)

Rollout – When the implementation is complete, the new tool will be rolled out based on the rollout plan. Rollout might include test runs with a pilot group. The rate of rollout should be dependent on the number of people who will be using the new tool, number of resources available for rollout, as well as the complexity of the tool. (5D,6A)

Maintenance – Once the tool is rolled-out, the improvement project proceeds to a maintenance phase. Like development projects, improvement projects are almost never perfect. Users will provide feedback on areas to change, as well as ideas for improvement. It is critical to have resources available to monitor the success of the improvement project and make any necessary changes. (6A,6B)

Conclusion

There are many reasons tools become shelfware, many of which you can mitigate by implementing best practices and good project planning. Try spending more effort on each tool adoption to ensure your organization will get maximum benefit from the tool. In addition to saving money on wasted tools, you will help your organization achieve its goals by using tools effectively.

Copyright 2001-2002, Karen S. King. All Rights Reserved

QW2002 Paper 8P2

Danny R. Faught
(Tejas Software Consulting)

The Making of an Open Source Stress Test Tool

Key Points

- Introduce the audience to an open source stress test tool that they can use.
- Explain the benefits of a general-purpose reusable stress test tool.
- Describe some of the implementation details involved in building a heavy-duty test driver.

Presentation Abstract

Describes a reusable stress test tool that I designed while at Convex Computer Corporation, and is now available under an open source license.

About the Author

Danny R. Faught is an independent software quality consultant, calling his practice Tejas Software Consulting. He has been in the industry for ten years. He has a BS in Computer Science from the University of North Texas. He is a senior member of the American Society for Quality, and is also a member of the Project Management Institute and the Fort Worth Chamber of Commerce. Danny is the maintainer of testingfaqs.org and is cofounder of the swtest-discuss mailing list. He serves on the Practicality Gauntlet for STQE magazine. He speaks and writes frequently on software quality topics.

The Making of an Open Source Stress Test Tool

Danny R. Faught
Tejas Software Consulting
Quality Week 2002

Objectives

- Introduce you to an open source stress test tool, the stress_driver, which you can use and modify in your own environment.
- Explain the benefits and limitations of a general-purpose reusable stress test tool.
- Describe some of the implementation details involved in building a heavy-duty test driver using a scripting language.

© 2002 Danny R. Faught

History

- Perl prototype
- C++ “production” version
- Then back to Perl because of problems with the C++ version
  - Ran on SPP-UX, HP-UX 9 & 10
- Dormant for several years
- Now open-sourced, and ported to Windows/Cygwin and Linux

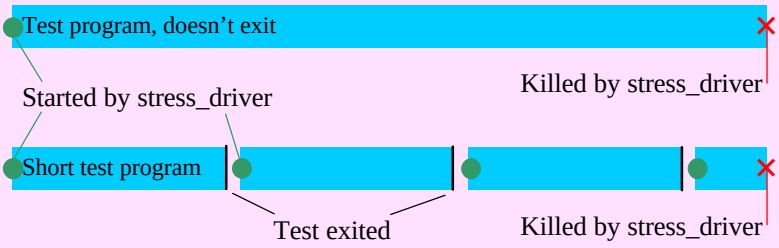
© 2002 Danny R. Faught

Stress_driver command line

stress_driver [-log *logfile*]
[-life *time*] [-iterate *n*]
[-tmout_max *time*] [-tmout_min *time*]
[-max *n*] [-min *n*] [-user *username*]...
[-user_array *basename*]... [-seed *n*]
[-sig *signame*] [-fail_max *n*] [–iterate *n*]
[-config *filename*]
[-- (*test program args*)] *test_program*

© 2002 Danny R. Faught

Two test program styles



© 2002 Danny R. Faught

When to stop?

- By default, stress_driver never stops. To specify an end do one or more of these –
  - Use -iterate
  - Use -life
  - Use -fail_max
  - Set the stress_time environment variable

© 2002 Danny R. Faught

How to run tests?

- -max and –min specify how many
- -tmout_max and –tmout_min specify how long to run
- -user and -user_array specify which account to run under
- Specify test program arguments after "--" on the command line or in the config file

© 2002 Danny R. Faught

Randomize test arguments

- Example:
  `[string1 string2 string3] [0-4]`
- Each time it starts a test, stress_driver can choose randomly between:
  - a list of strings, or
  - a integer range

© 2002 Danny R. Faught

Stress_factor scaling

- Stress_factor 1-10 chosen by user
- Config file says what to do with stress_factor:

```
--
# Factor 6-10
-s
```

- Run with something like:

```
stress_driver -config testprogconfig
  /usr/bin/testprog
```

Longer stress_factor example

```
# stress_driver
  configuration for
  the misc/forker05
  test
-fail_max 100
# Factor 1
-max 1
# Factor 2
-max 2
# Factor 3
-max 4
# Factor 4
-max 6
# Factor 5
-max 8
# Factor 6
-max 10
# Factor 7
-max 12
# Factor 8
-max 15
# Factor 9
-max 20
# Factor 10
-max 30
```

Processor scaling

- Use xCPU to scale on the number of processors:

```
#Factor 1
-min 1xCPU -max 2xCPU
...
#Factor 10
-min 10xCPU -max 30xCPU
```

*Not working at the moment

One more example: shell_stress

```
-tmout_min 1
-tmout_max 30
-user test -user cshtst -
  user shtst -user nettst
#Factor 1
-min 1xCPU -max 2xCPU
#Factor 2
-min 2xCPU -max 4xCPU
#Factor 3
-min 3xCPU -max 6xCPU
#Factor 4
-min 4xCPU -max 8xCPU
#Factor 5
-min 5xCPU -max 10xCPU
#Factor 6
-min 6xCPU -max 12xCPU
#Factor 7
-min 7xCPU -max 15xCPU
#Factor 8
-min 8xCPU -max 20xCPU
#Factor 9
-min 9xCPU -max 25xCPU
#Factor 10
-min 10xCPU -max 30xCPU
#Factor 1-10
--
-seed [1-4294967295]
-shell [/usr/bin/sh
  /usr/bin/csh
  /usr/bin/ksh:sh]
-o
```

Reproducible test runs?

- Ideally, using the same pseudo-random seed should produce the same test run and find the same bugs
- In reality, nondeterministic responses from the OS causes skew
- Even if stress_driver did everything exactly the same, we still would have hard-to-reproduce bugs

© 2002 Danny R. Faught

Porting to Cygwin & Linux

- Test-First Maintenance
  - used Test::More and Test::Harness
- Event module
- Perl 4'isms
  - h2ph files & syscall vs. POSIX
  - local vs. my
  - NGetOpts vs. GetOptions
    - config file hack

© 2002 Danny R. Faught

Limitations

- Designed for OS testing on the SUT
- Requires Cygwin on Windows, command-line oriented
- Generality leads to inefficiencies
  - Each test is a new process
  - Wrappers to add additional functionality also require starting extra processes

© 2002 Danny R. Faught

Ideas for enhancement

- Increase test coverage.
- Port and test on other systems.
- Fix the cpu scaling feature.
- Add a graphical interface using Tk.
- Validate the math used in the adaptive scheduler.
- Provide hooks that would allow dynamically modifying the stress_driver and test program arguments.
- Improve process management.
- Implement a slow start feature.
- Allow math expressions in arguments.

© 2002 Danny R. Faught

Getting the code

- You're all invited to use and contribute to the project
- The code is at: http://www.tejasconsulting.com/stress_driver/
- For Windows, get Cygwin (including the Cygwin Perl) at http://cygwin.com/
- Install Event and Test::More from http://cpan.org/. May need to upgrade Test::Harness.

Acknowledgements

- Eric Schnoebelen, for getting the stress_driver code released under an open source license
- Bob Clancy and Mark Wiley, for reviewing the paper
- The Convex System Software Test Team for their contribution to stress_driver
- Art credit – Alex, age 8, and Kelly, age 3

References

- "Test-First Maintenance: A diary", Danny R. Faught, Dallas/Fort Worth Unix Users Group Newsletter, June 2002. http://tejasconsulting.com/DFWUUG/test-first-maintenance.html (included with the paper)
- "Event-Driven Scripting", Danny R. Faught, presented at the July 12, 2001 meeting of the Dallas/Fort Worth Unix Users Group. Slides at: http://tejasconsulting.com/papers/event-driven/Event-Driven.htm.
- "Perl Scripting: A Test Automation Task Master", Danny R. Faught, Software Test Automation Conference tutorial, Fall 2002. Includes a walkthrough of some of the stress_driver code.

Thanks for listening!

Danny Faught
faught@tejasconsulting.com
http://tejasconsulting.com/
+1 817 294 3998

To stay in touch, sign up for my newsletter mailing list at http://tejasconsulting.com/#news.

The Making of an Open Source Stress Test Tool

Danny R. Faught
http://www.tejasconsulting.com
faught@tejasconsulting.com

The objectives of the paper are:

1. Introduce you to an open source stress test tool, the stress_driver, which you can use and modify in your own environment.
2. Explain the benefits and limitations of a general-purpose reusable stress test tool.
3. Describe some of the implementation details involved in building a heavy-duty test driver using a scripting language.

History

The saga of the stress_driver tool starts in 1993, when I wrote a prototype of a general-purpose stress test tool using the Perl scripting language. I then handed it over to another test developer to reimplement in a compiled language because I felt that a Perl script would not have sufficient performance to be able to stress the supercomputers I was testing.

A year later, we were trying to track down some mysterious problems in the tool, and I declared the C++ version to be unmaintainable. It didn't help that the programmer had left the company and we didn't have many C++ experts on staff. So I dusted off the Perl version, and that's the code base that survives to this day.

Some very recent news that makes the story much more interesting is that Hewlett-Packard, the current owner of the stress_driver tool after acquiring Convex, has granted an open source license for a large body of test tools and automated test cases, including the stress_driver and a suite of stress tests that use it. So I (no longer an HP employee) have been able to resume the development of the tool.

The code is now available for download, but it's buried within 21 megabytes of other data, nobody knows that it's there, and it would only work on specially configured versions of Perl running on SPP-UX or HP-UX 9, which few people now have access to. I have ported the stress_driver to Windows and Linux using the standard Perl distribution so that it's useful for a much broader audience.

The stress_driver tool is currently about 700 lines of Perl code, plus a manual page. The stress_driver runs a given test program, possibly scheduling random numbers of parallel invocations and randomly choosing parameters based on the user's

© 2002, Danny R. Faught, Tejas Software Consulting

specifications. It can scale the load based on the number of CPUs in the system and a user-specified 1-10 scale, and it can run the tests under different user IDs. It can vary the run time of each test process within a given range, and it can vary the number of parallel invocations of the test within a given range using an adaptive scheduling algorithm. The tool was used by a suite of operating system stress tests, testing the filesystem, memory management, process and thread management, and it was also a key part of a large-scale system reliability test.

What stress_driver does

The stress_driver is a generic stress test tool; you must provide a test program for stress_driver to run. You can use stress_driver in a variety of ways. There are basically two different ways it interacts with the test program. If the test program is designed to run for an indefinite period of time, then stress_driver will run the program once for each time slot that it sets up for the test, and it will kill the test at the end of the time slot. The degenerative case is when you only want one copy of the test to run at a time. Stress_driver doesn't add much value in that case, except to stop the test when the time period that you specify is done.

You can tell stress_driver how many tests to run at a time, and stress_driver will start that many copies of the test. You might randomize the parameters that each test receives, and you might scale both the test's parameters and the number of test programs according to a 1-10 scale provided at runtime. Stress_driver also used to be able to scale up automatically based on the number of processors on the system, but I've disabled this feature until I get access to another multiple-processor system.

The second type of interaction with the test program is the case where the test executes some defined transaction and then exits. In this case, stress_driver will usually need to schedule more than one iteration of the test during each time slot. Perhaps the degenerative case of just running one test at a time is somewhat more useful here, because stress_driver will continue re-running the test until the specified time period or number of iterations is complete.

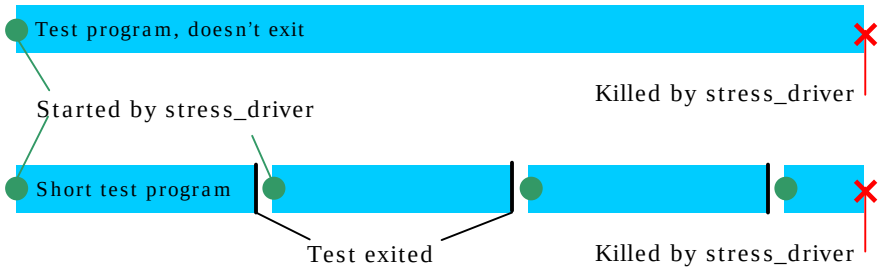


Figure 1. One stress_driver time slot, for a test with no built-in end point, and for a short-running test.

© 2002, Danny R. Faught, Tejas Software Consulting

Here's an overview of stress_driver's options. For a more complete reference, see the manual page in the stress_driver source distribution. A Unix man-style synopsis is:

stress_driver [–log *logfile*] [–life *time*] [–iterate *n*] [–tmout_max *time*] [–tmout_min *time*] [–max *n*] [–min *n*] [–user *username*]... [–user_array *basename*]... [–seed *n*] [–sig *signame*] [–fail_max *n*] [–iterate *n*] [–config *filename*] [–– (*test program args*)] *test_program*

Stress_driver keeps a detailed log, named "stress_driver.log.<pid>" where <pid> is stress_driver's process id. The –**log** option specifies a name for the log file. On Unix-like systems, you can specify /dev/tty as the log file to see the log in your terminal window without cluttering the disk with a log file.

By default, stress_driver will never exit. There are three options and an environment variable that affect stress_driver's immortality. You can use the -life option to define the lifespan of the test run in minutes. Or with this and all the other time options, you can append the letter "s" to the time and it will be interpreted as seconds, which is useful when you're testing the tool. You can also use the stress_time environment variable, which works the same as -life if you don't use the -life option.

Another way to specify the end of a test run is using -**iterate**, which gives a maximum number of iterations of the test that stress_driver will run before exiting. And finally, there's the -**fail_max** option, which gives stress_driver a bit of common sense, so it will exit after encountering the specified number of errors in the test program (either a non-zero exit code or dying from a signal) and any internal errors. You may use all three of these options, and stress_driver will use the first one that applies.

There are also a few options that relate to the lifetime of the test programs. By default, the test programs are not interrupted except at the very end of the stress_driver run. The -tmout_min option specifies that individual invocations of the test program will not be allowed to run longer than the specified time. If you also use -tmout_max, then stress_driver will randomly choose a time between the min and the max timeout each time it starts the test program.

Stress_driver's default action when it decides it needs to stop a test program is to send a SIGINT signal. You can use the **-sig** option to specify a different signal. The signal is specified using its symbolic name, without the "SIG" prefix (like "TERM", "HUP", etc.). The test program may catch the timeout signal if it needs to do any cleanup. It needs to be able to clean up and exit within 30 seconds, or else it will receive a KILL signal.

If the test program starts any child processes, it is responsible for cleaning them up. The test program should not report an error just because it received the timeout signal—this is a normal occurrence. It's okay if the test program simply dies from the timeout signal, though. Stress_driver doesn't log an error in this case.

© 2002, Danny R. Faught, Tejas Software Consulting

If you don't specify the **-max** option, stress_driver will start only one invocation of the test program at a time. The **-max** gives the maximum number of test programs to run in parallel. This is really where the tool starts to become useful! If you use -max and not -min, then stress_driver will try to keep the maximum specified number of test programs active all the time. If you use both -max and -min, then stress_driver will first start at the maximum, and then let the load vary randomly between the max and min. It uses an adaptive algorithm to try to keep the average number of active test programs close to the average of max and min. Note that the load may never fall down exactly to the minimum specified.

There is no built-in maximum for the number of test programs that stress_driver can run. In practice, the maximum will be determined by the demands of the test program, and the level of resources that are available on the test system (including memory, the size of the process table, and the processing horsepower). Stress_driver doesn't have any mechanism for distributing the load across more than one test system, though it is conceivable that an intermediary program between stress_driver and the test program could facilitate this with no change to the stress_driver design or to the test program.

The **-seed** option specifies a seed for the pseudo-random number generator. If you don't specify the seed, you'll get different random choices every time you run stress_driver. You can attempt to reproduce the results of a previous run by looking at the seed that is stored in the log file (even if you didn't use the -seed option before) and then feeding that seed into a later run. But later in this paper I'll explain why this isn't very useful.

You can tell stress_driver how to manage user accounts. By default, the test program will run under the same user id as the user who runs stress_driver. If you run stress_driver with administrator privileges, you can use the **-user** option to specify one or more accounts to use instead. If you give more than one account, stress_driver will randomly choose one of them each time it starts the test program. It won't guarantee that the accounts won't be reused for another invocation of the test program at the same time, though.

For more sophisticated account handling, you can use **-user_array**, which specifies the root name of a list of accounts that you've created (like "user1", "user2", "user3", etc.). Stress_driver assumes that the accounts are named using the root name you specify, followed by a number counting up to the maximum number of concurrent test programs allowed. You can use -user_array more than once to give multiple root names (I've done this before to have three different banks of users, with each bank set to a different login shell). Each bank must have enough accounts to handle the maximum load.

Rather than put all of the arguments on the command line, you can create a config file and tell stress_driver where it is using the **-config** option. You can put any command line options and test program options in the config file except for the test program,

© 2002, Danny R. Faught, Tejas Software Consulting

described below. Arguments on the command line will override any arguments that are specified in the config file.

After you have specified all of the stress_driver options, you can specify options to send to the test program. First use "--" on the command line to designate the end of the options for stress_driver. Then you may include any arguments that you want stress_driver to pass down to the test program every time the test program starts.

You may randomize the test program arguments either by providing a list of strings, or by specifying an integer range. Here are two randomized test program arguments:

```
[string1 string2 string3] [0-4]
```

Stress_driver will pass two arguments to the test program based on this specification - first, either "string1", "string2", or "string3", and then an integer in the range 0-4 inclusive. Note that in some shells you need to quote the square brackets when using this notation from the command line, though in practice, I generally use a config file when I use randomized parameters.

The final argument on the stress_driver command line is the absolute pathname for the test program. The test program must always be specified as the last argument on the stress_driver command line, not in a config file.

Stress_factor scaling

There are two different ways to scale a stress_driver run, based on a user-specified stress_factor environment variable, and based on the number of processors in the system.

The stress_factor environment variable is an integer from 1 to 10. (The fact that it's an environment variable and not a command line argument is based on the historical design of the test infrastructure at Convex that ran on top of the stress_driver.) The default stress_factor is 1—this is intended to be a minimal load for the software under test. A stress_factor of 10 is the maximum load that the software can withstand without encountering spurious errors related to resource shortages. For example, you wouldn't want to exhaust the memory on the system unless you're testing memory management. Numbers between 1 and 10 can be gradations in between.

To scale the test based on stress_factor, the test engineer must use a config file, because of the line-oriented syntax of the "factor" lines. Sections of the config file are partitioned using "factor" lines that look like "#Factor n[-m]". The "n" represents a number from 1 to 10. The optional "-m" turns it into a range, like 1-3. All lines after the factor line and before the next factor line will be used if the current stress_factor setting is within the specified range. Thus, the stress_factor scaling involves no magic, just a mechanical way to select options from the config file as they were set up by the test engineer. The engineer is responsible for configuring the test at each stress level, either by scaling the arguments to stress_driver, the arguments to the test program, or both.

© 2002, Danny R. Faught, Tejas Software Consulting

Note that you don't have to set up 10 distinct stress levels. In fact, the different stress_factor settings don't even have to have any relationship to each other, but they are set to scale up from 1 to 10 by convention.

As a simple example, consider a case where the test program only takes one argument, "-s", and whatever it does, the argument makes the test more stressful. We will only have two distinct stress levels, so we decide that stress_factor 1-5 will be the low stress level, and 6-10 will be the high stress level.

```
--
# Factor 6-10
-s
```

The "--" tells stress_driver that you're going to list test program options, just like on the command line. Then we have a factor line, telling stress_driver only to use the following lines if the stress_factor is in the range 6-10. We didn't give any test program arguments for stress_factor 1-5, so the test program won't get any arguments when we set stress_factor somewhere from 1 to 5. If the test program is "/usr/bin/testprog", and the config file is named "testprogconfig" in the current directory, we could call stress_driver like this:

```
stress_driver -config testprogconfig /usr/bin/testprog
```

Here's a more complex example from the Convex test suites:

```
# stress_driver configuration for the misc/forker05 test
-fail_max 100
# Factor 1
-max 1
# Factor 2
-max 2
# Factor 3
-max 4
# Factor 4
-max 6
# Factor 5
-max 8
# Factor 6
-max 10
# Factor 7
-max 12
# Factor 8
-max 15
# Factor 9
-max 20
# Factor 10
-max 30
```


© 2002, Danny R. Faught, Tejas Software Consulting

The -fail_max argument comes before any of the Factor lines, so it applies to all stress levels. Then we define a different stress level for each of the 10 stress_factor settings, by passing in a different -max argument to the stress_driver at each level. Note that the -max settings do not scale linearly. Stress_factor 10 is 30 times as stressful as stress_factor 1. This format gives us the freedom to scale however we want to.

This 1-10 scaling scheme makes more sense when we look at a suite of tests. The Convex operating system stress test suite was designed to run under the CITE functional test harness. Sometimes we would run a stress test by itself, in which case CITE didn't provide much value. But sometimes we wanted to do a regression test where we would run all of the stress tests for a brief period of time. So we could set the stress_time environment variable to, say, 30 minutes, and we could set stress_factor to, say, 3. Some of the tests recognize both stress_time and stress_factor, some by using stress_driver, and some using other mechanisms. Others may use one or the other. Tests that don't use stress_time are designed to do just one task and then exit. Anyway, with these settings, we'll get uniform coverage across all of the stress tests, at a fairly low stress level, for a fairly short period of time. So we can have global control across all the tests by setting these two environment variables.

Processor-based scaling

There's another type of scaling that we might want to do on a multi-processor system. A test that is stressful on a single-processor system might not be stressful at all on a system with eight processors. So stress_driver had the ability to scale the test based on the number of processors on the system. Note that this feature is currently not functioning in the version of stress_driver that I'm distributing, because it worked only on systems supported by the "getsysinfo" utility that was part of the Convex test suites. But the infrastructure for doing the scaling is still in the code, and all that is needed is a mechanism to count the number of processors on the system in order to get it working. The code currently assumes that there is only one processor on the system. The mechanism is worth discussing nonetheless.

To use processor scaling, you append the text "xCPU" to an integer argument. Here are two sections from the config file for the shell_stress test that show two types of scaling at work:

```
#Factor 1
-min 1xCPU -max 2xCPU
...
#Factor 10
-min 10xCPU -max 30xCPU
```

On a single processor system, at stress_factor 1, the number of test programs will vary from 1 to 2. If there are 4 processors, at stress_factor 1, the number of test programs running will range from 4 to 8. And at stress_factor 10, with 4 processors, the number of test programs will range from 40 to 120 (10 times 4 to 30 times 4).

© 2002, Danny R. Faught, Tejas Software Consulting

You can also use floating point numbers when you use xCPU. The fractional part will be truncated after multiplying, so the result will always be an integer. For example, you might want to use finer control with the scaling like so:

```
-min 1xCPU -max 2.5xCPU
```

So with 1 processor, the range is still 1 to 2, but with 4 processors, the range is 4 to 10.

You may combine xCPU with randomized integer ranges for test program arguments, and you may use fractional numbers here as well.

```
-foo [1-4]xCPU -bar [1-1.5]xCPU
```

The scaling is applied before the randomization, so you get the full range of possibilities. So with 4 processors, the above example is scaled to:

```
-foo [4-16] -bar [4-6]
```

And then the randomization is done within the multiplied ranges.

Further examples

Here is the first part of the config file for the thread01 test. It illustrates the ways you can get creative with the Factor lines. The arguments to stress_driver have two different stress levels. But the arguments to the test program have ten different levels (the first two are shown here).

```
# Factor 1-5
-min 2 -max 8
# Factor 6-10
-min 4 -max 12
--
# Factor 1
200
# Factor 2
400
...
```

The shell_stress test tries to accurately simulate an interactive user load on an operating system. This is probably the most elaborate use of stress_driver in the Convex tests. The shell_stress tool itself is a fairly complex tool, but it's designed to only simulate one user, so it integrates quite well under stress_driver. For this test, we call stress_driver like so (this is one long line):

```
stress_driver -fail_max 100 -log shell0.log
              -config shell_stress.cf $testbin/shell_stress
```


© 2002, Danny R. Faught, Tejas Software Consulting

The stress_driver arguments are split across the command line and the config file for no good reason that I can recall. Here is the full config file:

```
-tmout_min 1
-tmout_max 30
-user test -user cshtst -user shtst -user nettst

#Factor 1
-min 1xCPU -max 2xCPU
#Factor 2
-min 2xCPU -max 4xCPU
#Factor 3
-min 3xCPU -max 6xCPU
#Factor 4
-min 4xCPU -max 8xCPU
#Factor 5
-min 5xCPU -max 10xCPU
#Factor 6
-min 6xCPU -max 12xCPU
#Factor 7
-min 7xCPU -max 15xCPU
#Factor 8
-min 8xCPU -max 20xCPU
#Factor 9
-min 9xCPU -max 25xCPU
#Factor 10
-min 10xCPU -max 30xCPU

#Factor 1-10
--
-seed [1-4294967295]
-shell [/usr/bin/sh /usr/bin/csh /usr/bin/ksh:sh]
-o
```

There are some stress_driver options at the top that apply to all stress levels. Note that I took advantage of the free-form format of the file to try to make it more readable. I specify four different user accounts to choose from. These were standard accounts that were always set up on systems that were configured to run any of the operating system tests.

The -min and -max arguments to stress_driver are scaled based on the stress_factor and the number of processors, as described earlier. Then at the bottom of the file, I specify the test program arguments that don't scale on stress_factor. I likely forgot the "#Factor 1-10" line when I first wrote the config file, and was surprised to find that my test program only got its options at stress_factor 10, since the "#Factor 10" line is still in effect until the next Factor line.

For the test program arguments, I set the pseudo-random seed for shell_stress. This is based on a random choice across a wide range, and that random decision in turn is based on stress_driver's seed. This was an attempt to make the test run reproducible, so that all random decisions at all levels are tied to stress_driver's seed. Note that I

© 2002, Danny R. Faught, Tejas Software Consulting

didn't use stress_driver's -seed option here - we just let stress_driver randomize the seed. It's easy to confuse the two different drivers here. For our big reliability test, which had a somewhat different setup, we did hard-code a seed for stress_driver, using a large prime number.

In practice, I found that the results for two different shell_stress runs with stress_driver using the same seed weren't necessarily the same. Keep in mind that the types of bugs that shell_stress found often depended on exact timings that occurred by random chance more than intentional test design. Even if we ignore that factor, just comparing the logs from two stress_driver runs with the same seed show that stress_driver wasn't making the same random decisions in both cases.

Why did the seed not do what I wanted it to? I haven't studied the reason in depth, but here's a theory. Complex computer systems are not completely deterministic. When we have hundreds of processes running, there is no guarantee that they will exit in the same order each time. Perhaps the disk is fragmented in a different way and its response time is different, or perhaps you ran a command on the system that was the equivalent of a butterfly flapping its wings and changing the weather on the other side of the globe. In any case, as soon as a stress_driver action is done in a different order than the previous run, then the next number in the pseudo-random sequence may be applied for a different purpose than for the last run. Then the place where that number was used last time gets a later number in the sequence instead. That's all it takes for the test run to skew wildly. Further study would be needed to figure out how to prevent this, and whether identical behavior from stress_driver is likely to have much effect on reproducing failures in the first place.

The "-shell" argument is a use of the string type of randomization, telling shell_stress which shell to use. You may have noticed that the names of some of the user accounts also suggest a type of shell—these are the login shells for the accounts. Neither stress_driver nor shell_stress (in this particular test) does a full login, so the shell is chosen independently of what the login shell for the account is. The “:sh” notation tells shell_stress to use Bourne shell-style syntax when setting the shell prompt and checking that status of the commands.

Porting stress_driver

Now we can fast forward to 2002. Eric Schnoebelen, another ex-Convex employee, was doing contracting work for the Hewlett-Packard division that had acquired Convex. HP was no longer actively using the test suites that it had acquired with Convex. Eric convinced HP to release the tests and their associated tools under an open source license. Eric volunteered his time to audit the tests to remove the functional tests that Convex has licensed from Perennial, and the tests are now available, along with the CITE test harness that many people had requested a copy of during it heyday.

I decided to pull out one particular part of this valuable but obscure resource, and help others take advantage of it. So I ported stress_driver to the Cygwin environment on Windows (a library that facilitates porting Unix utilities to Windows, plus the Unix-

© 2002, Danny R. Faught, Tejas Software Consulting

style utilities that use it), and Linux. I did most of the work on Windows because I had a Windows system handy. When I tried the tool on Linux for the first time, it required no changes in order to run properly. The tests, however, required some significant porting work because of the way I had designed them.

I adopted one tenet of extreme programming, and did a sort of test-first development, or in this case, test-first maintenance. I published a brief write-up about this effort, in the article "Test-First Maintenance", which is included at the end of this paper.

One of the first things to go from the stress_driver code was the references to several "h2ph" files. These files are produced by a script that tries to convert C header files into Perl. I referenced some of these files for the advanced signal handling that is required to support the event-driven aspects of stress_driver. These perl headers were very fragile, and I had to work around a few bugs in them. Another big hack in the code was a use of the "syscall" function to invoke a system call directly from perl, also for the purpose of advanced signal handling.

I wanted to rip out my home-grown event-handling code and use something like the Event module instead. The Event module is part of the Comprehensive Perl Archive Network (CPAN), though it doesn't install with Perl by default. I was apprehensive about removing my event handling code. I had put a lot of work into making it robust, and it was a core part of the code, though there was still an occasional mysterious failure. I decided to put off the port to the Event module for a while.

Since I wanted to make the script portable, I decided to port the signal handling code to use POSIX signals. The POSIX module was not available when I first wrote stress_driver using Perl 4. Using the POSIX module would not only make the code more portable, but it would also get rid of the dependence on the most egregious hacks in the code—the h2ph files and the use of the syscall function. I had the port partially done, and at the same time I was writing automated tests to verify the stress_driver code. I found that one of my tests was failing intermittently. It looked like I had a race condition in my code. At this point, I decided it was time to bite the bullet and make the big changeover to the Event module rather than trying to fix the existing code.

The changeover wasn't as traumatic as I had feared. I ended up removing about 90 lines of code that were replaced by functionality in the Event module. I still need to do more testing to verify that stress_driver is working as well it used to, though.

There are some other Perl 4'isms that I've been working on removing. To parse the command line arguments, I used the NGetOpts function from the newgetopt.pl library, which was the latest and greatest method for argument handling in Perl 4. I used NGetOpts to get stress_driver's command line arguments, and I also crafted a hack to use NGetOpts to parse the config file. I have ported the code to use the GetOptions function in the Getopt::Long module instead. So some nasty perl 4 hacks with "package" are replaced with some "write-only" code that deals with the hash that now

© 2002, Danny R. Faught, Tejas Software Consulting

stores the arguments. A bit of a hack is still required to convince GetOptions to process the config file, but it's not nearly as crufty.

Another Perl 4 relic was the fact that I localized my variables using local(), which uses dynamic scoping. For Perl 5 programs, programmers are strongly encouraged to use my() instead of local(). The my() syntax specifies lexical scoping, which is a safer and much more familiar mechanism (even if you don't know what lexical scoping is). But for the config file hack mentioned above, I found that I still had to use "local(@ARGV)" because GetOptions references @ARGV as a global variable. When I naively tried "my(@ARGV)", the value wasn't available to GetOptions because of the lexical scope.

Limitations

While stress_driver was written to be general-purpose, it's not likely to be appropriate for everyone. It was designed for operating system testing, and it runs directly on the system under test. So there are no special features for starting the application under test.

Stress_driver is Unix-centric, and it doesn't have a graphical user interface. Though it runs under Windows with a lot of help from the Cygwin environment. someone who isn't familiar with Unix or Cygwin may have trouble using the tool.

You can only specify one test program to stress_driver. If you wanted to use more than one test program, you could write a wrapper program that called your test programs using whatever criteria you wanted. In fact, you could consider the shell_stress test to be an example of this. Shell_stress runs many different programs from its user profile database. The downside is that all these layers of control steal performance from the system (and shell_stress itself is two layers - a perl script to parse the database and an expect script to execute the commands). While running the shell_stress test, I found that the system spent a sizable fraction of its resources running all the driver scripts. This takes resources away from the tests themselves. If you write a test driver specifically for a particular type of test, you have more opportunity to optimize the driver. This is the tradeoff we make for a general solution.

The performance issue might could be mitigated if stress_driver could execute tests in a distributed fashion, so that one machine executes the driver and other machines execute the tests. It is possible, of course, to add a layer underneath stress_driver that distributes the tests, which would bring with it all the caveats of the previous paragraph. For one incarnation of shell_stress, I did add such a layer. I only used it to test networking in a loopback, i.e., doing telnet, rlogin, and ftp back to the same machine, but it did serve as a proof of concept for doing distributed testing.

Another factor to consider is that each invocation of the test program requires starting a new process. This wouldn't be ideal for situations where the test cases are very lightweight. For example, if the test runs in a tenth of a second, then the time required to start a new process, clean up after it, and log each step along the way would dwarf the time spent actually running the test code.

© 2002, Danny R. Faught, Tejas Software Consulting

Ideas for enhancement

There are always more features to implement that resources available to implement them. Here are a few ideas for stress_driver that may also give you ideas for improving other tools that you work with. Whether any of these get implemented will depend on the interest from the user community and on how many people volunteer to help with the development.

- *Increase test coverage.* The current test suite is very sparse and there are likely many more bugs to root out.
- *Port and test on other systems.* It's likely to port easily to any system that supports Perl and the Event module (probably only Unix-like environments).
- *Fix the cpu scaling feature.* For each supported operating system, it just needs to have a mechanism to count the number of processors on the system. Also, perhaps add a command-line option to specify the number of processors, which could be used before the automatic processor count is ready, and could also be used to spoof the number of processors for purposes of experimentation.
- *Add a graphical interface using Tk.* This would make stress_driver easier to set up and monitor.
- *Validate the math used in the adaptive scheduler.* I suspect that it doesn't quite work the way it's supposed to, in managing the average number of active test processes.
- *Provide hooks that would allow dynamically modifying the stress_driver and test program arguments.* Users could use their own adaptive algorithms.
- *Improve process management.* Several possibilities here, such as: using process groups to make cleanup more robust, modify priorities so stress_driver gets more cpu cycles when under heavy load, and test stress_driver's operation when the process table or memory is full.
- *Implement a slow start feature.* Rather than always blasting the system with the maximum number of tests all at once, it might be useful to be able to start up more slowly in order to mimic more of a real-world scenario.
- *Allow math expressions in arguments.* The current scaling mechanisms are fairly flexible, but we could get even more flexibility by allowing arbitrary expressions and further generalizing the scaling.

Call for participation

Consider yourself invited! Perhaps you want to be a user of the tool, or you just want to borrow the code for other purposes. You could hone your test automation skills by contributing to the stress_driver test suite, or you could exercise your perl programming skills by working on the stress_driver code itself. Both the original code from Hewlett-Packard and my enhanced version of the stress_driver is released under a modified Apache Project license.

© 2002, Danny R. Faught, Tejas Software Consulting

To obtain the enhanced version of the stress_driver and its test suite, go to http://www.tejasconsulting.com/stress_driver/.

To run stress_driver on Windows, you will need to install Cygwin and the Cygwin build of Perl. See http://www.cygwin.com/. On Linux and Unix systems, you will need a recent version of Perl. I used Perl 5.6.1 on both Windows and Linux.

You will need the optional Perl modules: Event, Test::More, and probably an upgrade of Test::Harness. The easiest way to install these if you have a live Internet connection is to run "perl –MCPAN –e shell" and type "install Event" and "install Test::Harness". Note that on Windows 2000, one of the Test::Harness self-tests will fail, so you have to do a forced install.

For the original stress_driver as of the time of its 2002 release from HP (the script was actually last modified in 1996), including the suite of stress tests that used it, see ftp://ftp.cirr.com/pub/cite/test-suites-19961217.tar.gz. The script is located in bin/stress_driver, and the test suites are under os/stress. This version of the script is also included in my stress_driver distribution, named "stress_driver_orig".

The test suites are designed to run using the CITE test harness, which can be found at ftp://ftp.cirr.com/pub/cite/cite-4.4.tar.gz. Stress_driver itself is not dependent on CITE.

Note that the originally released testware in test-suites-19961217.tar.gz and cite-4.4.tar.gz run only on a limited set of now outmoded platforms. It's difficult even to determine which platforms they did run on when development ceased. So don't expect to be able to make use of what you find there without porting it to your platform.

Acknowledgments

Big thanks go to Eric Schnoebelen for taking the initiative and applying the elbow grease to get the stress_driver code and megabytes of other interesting test assets released to the public, and to Hewlett-Packard for agreeing to make it available. Thanks also to the System Software Test Team at Convex Computer Corporation for their role in the development of stress_driver, the infrastructure it worked within, and all the suites that used it.

I'd like to acknowledge Bob Clancy for reviewing early drafts of this paper and for being to first to volunteer to participate in the further development and testing of stress_driver.

References

"Test-First Maintenance: A diary", Danny R. Faught, Dallas/Fort Worth Unix Users Group Newsletter, June 2002. (included below)

© 2002, Danny R. Faught, Tejas Software Consulting

Event-Driven Scripting, Danny R. Faught, presented at the July 12, 2001 meeting of the Dallas/Fort Worth Unix Users Group. Slides at: http://tejasconsulting.com/papers/event-driven/Event-Driven.htm.

“Perl Scripting: A Test Automation Task Master”, Danny R. Faught, Software Test Automation Conference tutorial, Fall 2002. Includes a walkthrough of some of the stress_driver code.

© 2002, Danny R. Faught, Tejas Software Consulting

Appendix

Test-First Maintenance: A Diary

Published in the Dallas/Fort Worth Unix Users Group Newsletter, June 2002

Hurrah! A stress test tool that I wrote while employed at Convex Computer Corporation has been released with an open source license. It's called "stress_driver" and it's sitting hidden in a 20 meg tar file where no one will likely find it, and anyone who does find it won't know what to do with it. It runs on an operating system version that few people use. It was written using Perl 4, and though it's been ported to Perl 5, it still uses a Perl 4 style, including requiring some header files that are conjured via black magic. But I found that it was a very useful tool, and I bet that it could easily be ported to other operating systems.

I've been talking to Extreme Programming (XP) and other agile development advocates about test-first development. So why not test first maintenance? The idea with test-first development is that when you develop a new feature, you first write a test, you run the test to verify that it fails, you develop the feature, then run the test and see if it passes.

Here I have an 816-line perl script that doesn't run on any system I have access to. There are no tests. I'm going to dive into the deep end and try test-first maintenance for legacy code, while porting stress_driver to the Cygwin environment on Windows NT 4.0. I'm keeping a diary along the way. Here are some highlights and extra commentary.

Oh, by the way, I'm not familiar with Perl's test harness modules, though I know that several exist. Having run the test suite for Perl itself and some of its modules, I choose the same basic "Test" module that they use, and I decide to use Test::Harness in a script that will kick off all of the tests.

2002-05-08
2:08pm
The simplest test I can write is one that uses no command line arguments. It turns out that this is a negative test - the expected result is an error message, because at least one argument is required. I don't think agile developers write a lot of negative tests. Oh well. I write the "badopt" test. It passes, but I didn't verify the text of the error message. It turns out that the stress_driver is croaking because I haven't starting porting it to Cygwin yet. So I add another check based on the error message, and that fails.

Seems like I have a lot of work to do to get this first test to pass. Hmmm, the XP tenets say I should keep things simple. So I simply comment out the parts that don't work and are preventing the program from getting as far as the code that checks the command line arguments. I have my first passing test!

© 2002, Danny R. Faught, Tejas Software Consulting

I decide to add some subtests to "badopt." As a tester, I find negative tests more fun than positive tests. :-) One new test passes a test program path to stress_driver that doesn't exist. (Talking about testing a test tool can be confusing - when using stress_driver, you give it the path of a test program that it will run) Cool, that passes.

2:24pm
I want to add another test that specifies a program that isn't executable. Now is when I start wishing for a more complete test environment. I need to create a file and make sure the execute bits are off. Normally, I expect the test harness to give me a working directory where I can create any files that are needed. I hack my test harness script so it creates a working directory and put the path in an environment variable. That subtest passes. In retrospect, I wonder why I didn't just create a non-executable file ahead of time in the test suite directory. Maybe because it it's too easy for file permissions to be botched when installing a test suite.

2:45pm
Okay, I'll force myself to write some positive tests. I create the second test, named "simple." I'll tell the stress_driver run "sleep 100000" and then interrupt it shortly after it starts. There is a -life option that tells stress_driver how long to run. Unfortunately, the lowest it can go is one minute, which is unacceptable for a test case that should be able to do its job in a few seconds. I modify stress_driver so that the -life option can understand seconds as well as minutes.

Testers often have to ask developers to add testability features to their programs. It's such an easier sell when I'm both the tester and the developer. I recall when I originally developed the code, I modified it so that minutes were interpreted as seconds while I was testing, but since I didn't write any reusable tests, I didn't bother to support both.

2002-05-11
9:33am
A big change that I've been planning to make is to rip out my home-grown event loop and use the Event module instead. I now have 12 subtests in three files, 15 seconds runtime. All usually pass, but one intermittently fails in the event code. I decide it's time to do the big changeover rather than trying to fix the old code.

2002-05-13
2:41pm
All tests are now passing after the event code changeover (and the code is about 90 lines leaner now). But I'm suspicious - that was too easy. I examine the logs created from running the "simple" test, and I see that stress_driver never actually started any test programs. My tests need to do a lot more verification. I realize that I'm using a unit testing framework to do high-level functional testing. Verification would be much easier and more thorough if I were doing true unit testing and had more access to the program state.

© 2002, Danny R. Faught, Tejas Software Consulting

2:49pm
Oops! I learn that when commenting out some of my event code libraries, I also commented the code that initially starts the child processes, which is still needed in the new event design. Fixed. I'm glad I tend to comment out code and test the program before actually deleting the code.

10:05pm
Fixed several other problems, and the post-Event module code now passes all 12 tests.

2002-05-17
5:19pm
I'm getting tired of setting -life to one second for my positive tests. It's not elegant, and it's still not as optimized as it could be. So I give stress_driver a new -iterate option that specifies the maximum number of times to iterate the test program. For many of my stress_driver tests, I'll specify just one iteration, and many will complete in less than a second. I wonder why I never thought to create that feature before. Chalk up another one for testability.

Well, that's where I'll leave you for now. Along the way, I found bugs in my original stress_driver design (including a minor Y2K bug) as well as the new code I added. I found bugs in the Event module and perl itself, including a reproducible crash in the perl interpreter. I found myself wishing for a more full-featured test environment, so I plan to investigate the other Test modules that are available.

If you're a Perl hacker who's interested in participating in the test-first maintenance project and in using an alpha version of a general-purpose stress test tool, let me know. There's plenty more testing to be done.

© 2002, Danny R. Faught, Tejas Software Consulting

QW2002 Paper 9P1

Lauri MacKinnon
(PhaseForward Incorporated)

Testing Efficiency: Taking Advantage of Test Overlap

Key Points

- Discuss the current software development environment and how it effects test teams
- Describe methods for combining test activities to complete them in shorter timeframes
- Ms. List tips and techniques for working more efficiently as a test team

Presentation Abstract

With today's fast-paced software development cycles, there is seldom enough time for complete testing. Shortened test cycles, a smaller staff, and an increasing need for product quality mean that test groups have to do more testing in less time. This presentation identifies some ways that test groups can work more effectively in shortened timeframes. It describes common testing activities and discusses ways to combine them to get more test coverage. The presentation also covers ways your test team can work more efficiently and more effectively. Attendees will discover:
·Methods for combining test activities to complete them in shorter timeframes
·Tips and techniques for working more efficiently as a team
·Resources for more information on effective testing practices

About the Author

Lauri is currently a Principal QA Engineer at Phase Forward Incorporated. She is a QA professional with 13+ years of experience in QA methodologies, documentation, testing, QA implementations, and team leadership across all phases of product development. Her background in programming lends to her experience in the more technical aspects of testing.

Lauri's main role at PhaseForward is to implement the testing of several webbased and client-server software applications created specifically for use by the pharmaceutical industry. Lauri recently worked at Vanteon where her main task was to drive the research, training, and implementation of new technologies, testing methodologies, and automation tools across multiple projects for clients in a variety of industries, including financial, educational, tool vendor, ecommerce, graphics, and shrink-wrapped markets.

Lauri has presented at several conferences on topics including configuration testing, performance testing, usability testing, and test management, and has written technical white papers that have been printed in industry newsletters. She is a Segue-certified eConfidence Performance Consultant and has been involved with UL ISO9000, ISO9001 certifications, and SEI quality assessment processes, standards, and inspections. Lauri is a participant in local chapters of ASQ, IEEE, SPIN, and NESQAF. Lauri has a Bachelor's degree in Computer Science and a Master's degree in Computer Science from Boston College.

International Quality Week Software Research

15th International Internet & Software Quality Week 2002

Testing Efficiency:

Taking Advantage of Test Overlap

Lauri MacKinnon

PHASEFORWARD™

San Francisco, California, USA
3 - 6 September, 2002

Agenda

- Describe the current testing environment
- Define common test types
- Discuss ways to combine testing for greater efficiency
- List some team efficiency tips
- List resources

The Current Test Environment: Industry Effects

- Increased market pressure
- Reorganizations
- .com failures
- Budget cutbacks leave little money for
  - Employees
  - Contractors and consultants
  - Equipment and tools
  - Incentives and reimbursements

The Current Test Environment: The Software Development Industry

- Shorter time-to-market
- Rapid product development
- Feature sets must be maintained
- Quality of product becomes critical
- Usability becomes more important and affects a wider product audience

The Current Test Environment: The Effect on QA Groups

- Less money for testing tools and equipment
- Smaller test staff
- Less time for testing
  - Shortened test cycles
  - Fewer test cycles

Something has to give…

Test Techniques: Cutting Back on Testing Tasks

More Risky to Skip	*Less Risky to Skip*
Testing new and changed code	Testing legacy code
Testing vital code	Testing little-used code
Testing prominent and/or advertised code	Testing older configurations
Testing complex code	Repetitive tests that are easy to automate
Critical test types	Unit testing that can be done by development

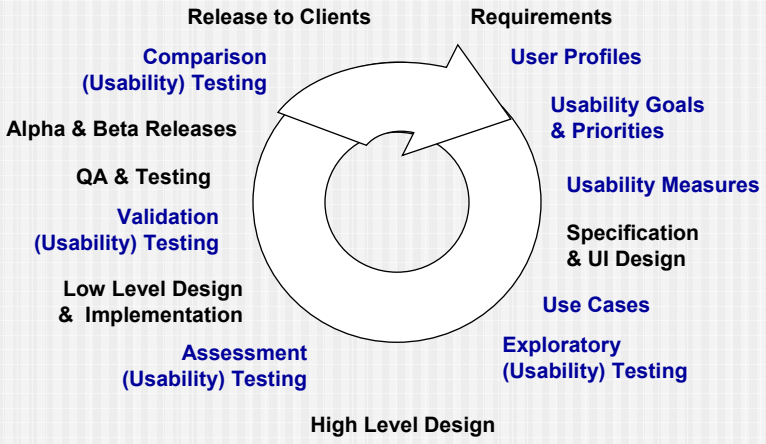
Common Test Types & Activities

- Product Familiarization
- Test documentation
- Acceptance testing
- Feature testing
- System testing
- Configuration testing
- Compatibility testing

Common Test Types & Activities

- Performance testing
- Unit testing
- Error / Recovery testing
- Load testing
- Stress testing
- Ad-hoc testing
- Scenario-based testing
- Usability testing

Usability Testing During the Product Lifecycle



Test Techniques: Combining User Profiles & Requirements Gathering

- Target User:
  - User role identification
  - Demographics (age, education, etc.)
  - Learning style
  - Skills background
  - Domain knowledge (subject matter)
  - Computer skills (overall experience and daily use)
  - Special restrictions or considerations

Test Techniques: Combining User Profiles & Requirements Gathering

- Product Usage:
  - Goal or purpose
  - Frequency of use
  - Task analysis
  - Work flow analysis
  - Surfing vs. searching
  - End-user environment

Test Techniques: Combining Usability Evaluations & Product Familiarization

- Usability Evaluations:
  - Heuristic / Expert Evaluation
  - Live Tests (Focus Group, Talk Aloud, Paper Prototypes, Video/Audio)
  - Questionnaires and Survey
  - Consistency Evaluation (Style Guides)
  - Standards Evaluation
  - Feature Evaluation

Test Techniques: Combining Usability Walkthroughs & Test Documentation

- Use Case or Task Scenario Review (Cognitive Walkthrough)
- Multi-user Evaluation (Pluralistic Walkthrough)
- Test Documentation
  - Test Plans
  - Test Cases
  - Traceability Matrix
  - Use Cases

Test Techniques: Re-use of Use Cases

- Usability Testing
- X-treme Programming Testing
- Scenario Testing
- Acceptance Testing
- Requirements Validation
- Design Validation
- Documentation Templates

Test Techniques: Combining Acceptance Testing & Configuration Testing

- Stick to supported platforms
- Automate minimal acceptance test suite
- Execute simultaneously on platforms
  - Automated
  - Manual
- Alternate execution across configurations

Test Techniques: Combining Scenario Testing & System Testing

Write Scenario test cases that:

- Walk through typical user tasks (start-to-finish)
- Exercise System test areas:
  - Recovery testing
  - Security testing
  - Stress testing
  - Performance testing

Test Techniques: Combining Load Testing & Performance Testing

- Use a configurable automated load testing tool
- Baseline with 1 user; doubles as performance test
- Increase the load in increments
- Record and compare results
- Save results as a benchmark
- Compare to future product releases

Test Techniques: Combining Usability Testing & Beta Programs

Beta program participants can re-run:

Usability tests

- Familiarization tasks
- Use cases
- End-user tasks

Other test types

- Requirements validation
- Acceptance tests
- Scenario tests
- Ad-hoc tests
- Configuration tests

End-users may also find issues that testers and developers may not notice

Test Techniques: High Performance Work Teams

- Multi-task
- Rotate tasks & staff
- Track product area coverage
- Cross-train through concurrent activities
- Overlap training and testing
- Leverage testers' strengths and talents

Test Techniques: High Performance Work Teams

- Alternate full feature & directed test passes
- Have frequent communication
- Track progress
- Use post-mortem results
- Hold bug triage reviews
- Improve their estimating and scheduling
- Use simpler methods when necessary
- Delegate common IT infrastructure tasks
- Reward staff

Test Techniques: Using Resources Outside of the QA Team

Inside the company

- Development Group
- Customer Support Group
- Documentation & Training Groups
- Sales & Marketing Groups
- Managers & Executives
- IT Group

Outside the company

- Beta Sites
- Customers
- Contractors
- Consultants
- Outsourcing companies

Summary

- Described the current QA environment
- Went over some of the more common test types and activities
- Discussed ways to combine test activities to complete them in shorter timeframes
- Identified some tips and techniques for working more efficiently as a team

Resources – Books & Articles

- Buckingham, Marcus and Coffman, Curt. 1999. First, Break All the Rules. Simon & Schuster. ISBN: 0684852861.
- Connor, Daryl R. 1992. Managing at the Speed of Change: How Resilient Managers Succeed and Prosper Where Others Fail. Random House. ISBN: 0679406840.
- Drucker, Peter. 1995. Managing in a Time of Great Change. Dutton/Plume. ISBN: 0525940537.
- Pressman, Robert S. 1992. Software Engineering, A Practioner's Approach. ISBN: 0-07-050814-3.

Resources – Books & Articles

- Jakob Nielsen, Designing Web Usability
- Jakob Nielsen, Usability Inspection Methods
- Donald Norman, The Design of Everyday Things
- Stephen Krug, Don't make me think!
- Ben Schneiderman, Designing the User Interface: Strategies for Effective Human- Computer Interaction
- Russell R. Hurlbut, A Survey of Approaches for Describing and Formalizing Use Cases, Document: XPT-TR-97-03

Resources – Web Sites

- http://www.useit.com
- http://www.jnd.org
- http://www.microsoft.com/usability/
- http://webword.com
- http://usableweb.com
- http://usability.gov
- http://www.asktog.com
- http://www.usablesites.com

QW2002 Paper 9P2

Mr. Cem Kaner
(Florida Institute of Technology)

Measuring the effectiveness of software testers

Key Points

- Measures of individual performance are often dysfunctional
- In many companies, we have to measure individual performance
- Some combinations of measures might do more good than harm

Presentation Abstract

The Software Test Managers Roundtable (STMR) meets twice a year to discuss issues of interest to experienced test managers. The May 2002 meeting discussed measurement of the effectiveness of software testers. This talk reports ideas from that meeting.

We will look at several examples of dysfunctional approaches to employee measurement, why they are dysfunctional, and how you might explain to your management that they are best left unused. We will also look at examples of multi-dimensional approaches that different test managers have found useful, asking why they were useful and how you might apply the ideas underlying them to your job. Finally, we'll note that much employee performance "measurement" is qualitative and interactive. Some managers operate by discussion and demonstration, much more than by the numbers. We'll discuss some of the interactions that some people have found effective.

About the Author

Cem Kaner is Professor of Computer Sciences at the Florida Institute of Technology. Prior to joining Florida Tech, Kaner worked in Silicon Valley for 17 years, doing and managing programming, user interface design, testing, and user documentation. He is the senior author (with Jack Falk and Hung Quoc Nguyen) of TESTING COMPUTER SOFTWARE (2nd Edition) and (with David Pels) of BAD SOFTWARE: WHAT TO DO WHEN SOFTWARE FAILS.

Through his consulting firm, KANER.COM, he teaches courses on black box software testing and consults to software publishers on software testing, documentation, and development management. Kaner is also the co-founder and co-host of the Los Altos Workshop on Software Testing, the Software Test Managers' RoundTable, the Workshop on Heuristic & Exploratory Techniques, and the Florida Workshops on Model-Based Testing.

Kaner is also attorney whose practice is focused on the law of software quality. He is active (as an advocate for customers, authors, and small development shops) in several legislative drafting efforts involving software licensing, software quality regulation, and electronic commerce. Kaner holds a B.A. in Arts & Sciences (Math, Philosophy), a Ph.D. in Experimental Psychology (Human Perception & Performance: Psychophysics), and a J.D. (law degree). He is Certified in Quality Engineering by the American Society for Quality.

QW2002 Paper 2V1

Mr. Mitch Krause
(TestQuest, Inc.)

Carrier Compliance Testing of Mobile Handsets

Presentation Abstract

The presentation will provide an insight into an automated testing environment employed at a wireless carrier which allows the same test environment and the same automated test scripts to be executed against different models of handsets from different handset manufacturers with little maintenance or setup from the carrier.

About the Author

Mitch Krause is a Managing Consultant in the Mobile and Wireless business division of TestQuest Inc, a leading provider of operating system independent testing solutions. He has over 15 years of software test experience including commercial, military and class III medical device software development environments.

Wireless Carrier Compliance Testing of Mobile Handsets

testQuest

Presentation Outline

- ***Problem Statement and Challenges***
- Test Scenarios
- Testing Options
- Test Solution

testQuest

Challenges for Mobile & Wireless Products

testQuest

Mobile Market Objectives

1. Maximize ARPU
2. Minimize subscriber churn
3. Increase market penetration of services and devices
4. Deploy revenue generating services
5. Increase adoption of advanced handsets

testQuest

Primary Influencers

1. Postpaid & high-end subscriber demand for reliable, high-bandwidth data services for enterprise applications
2. Prepaid & low-end subscriber need for dependable communication appliances
3. Subscriber perception of on-air Quality of Service
4. Differentiated device capabilities and value-add service

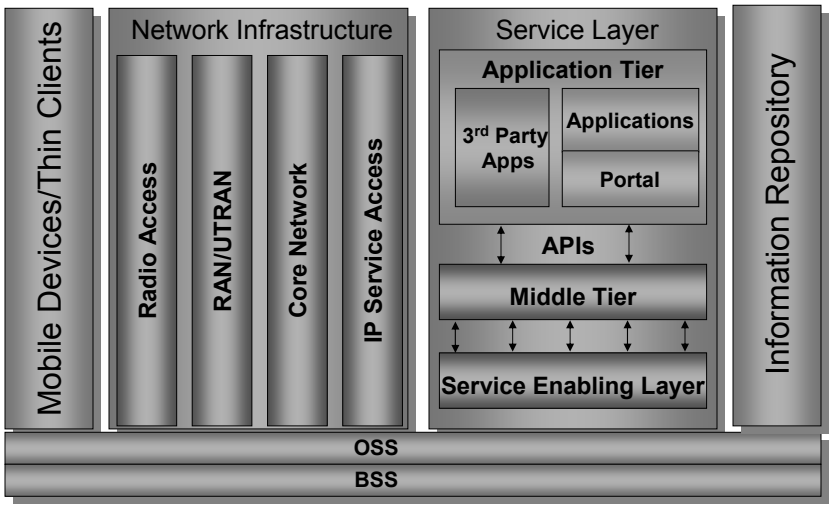
testQuest

Carrier Testing Problem

1. Typical vendor has 15 models & 8-10 revs/model
2. Typical service provider certifies about 35 handsets a year from each of about 12 vendors
3. Certification takes 90-120 days
4. Involves 8-10 revs of handset software and firmware
5. Tests are repeated when problems are found & corrected
6. Certification is generally manual
7. Handset complexity is increasing with convergence, 3G, and data services

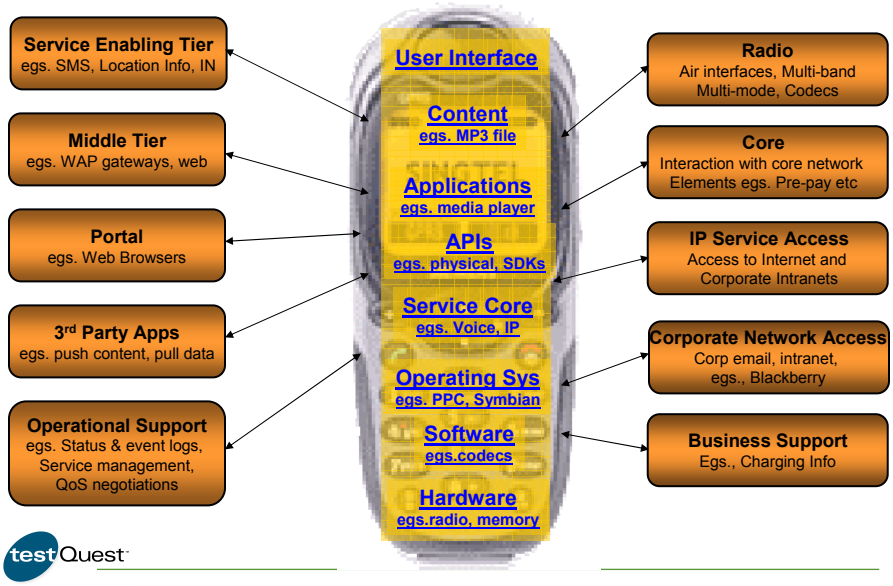
testQuest

End-to-End Network Architecture



testQuest

Mobile Device – Subscriber's View



testQuest

Presentation Outline

- Problem Statement and Challenges
- ***Test Scenarios***
- Testing Options
- Test Solution

testQuest

Carrier Test Scenarios

1. Stage 1: RF Parametric
   - In conjunction with BSEs and T&M systems
2. Stage 2: Mobile station – Base Station Interop
   - System determination; mobile originate; SMS handling
3. Stage 3: Drive test on live infrastructure
   - Multi-band performance; call performance
4. Handset functionality
5. Value-add services

testQuest

Application Examples

- Functional and Regression Testing
- User Interface (UI) testing
- Application testing – browser, SMS, phonebook, etc.
- End-to-end test:
  - PC/handset synchronization
  - Server & handset (client/server applications)
  - Base station emulator control
  - Bluetooth and wi-fi
- Field testing
- Interoperability test (IOT)

testQuest

Presentation Outline

- Problem Statement and Challenges
- Test Scenarios
- ***Testing Options***
- Test Solution

testQuest

Testing - Options

1. Grow the manual test organization
   - Does not shorten the test cycle
   - Does not scale to technical requirements
   - Expensive
2. Automate handset certification
   - Multiple handsets can be tested in parallel
   - Potential to significantly shorten the test cycle
   - Shrink time-to-market
   - Can handle complex test scenarios
   - Tests can be moved upstream to handset vendors thus reducing burden on service providers

testQuest

Presentation Outline

- Problem Statement and Challenges
- Test Scenarios
- Testing Options
- ***Test Solution***

testQuest

Solution Goals

1. Increase quality and maturity level of handsets released to service providers for test
2. Reduce vendor & service provider test cycles by 50%
3. Deploy completely carrier-specific turnkey test solutions

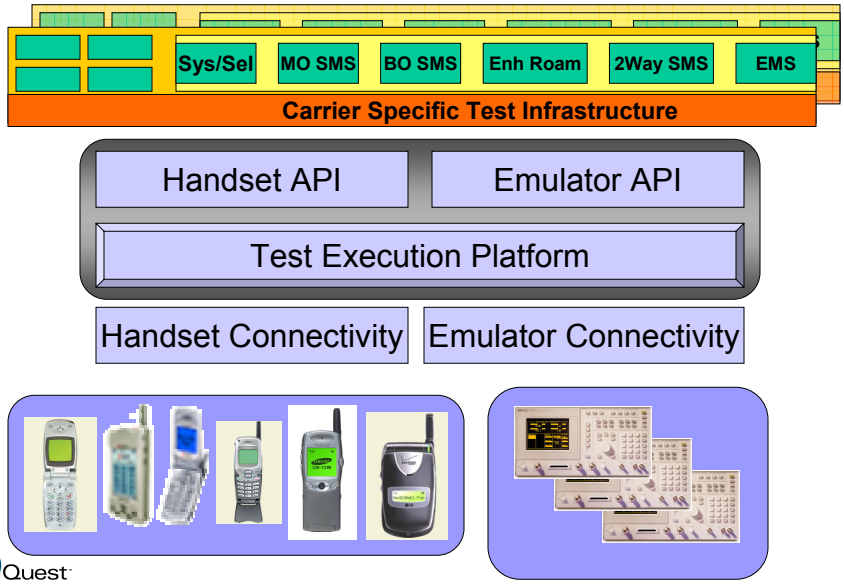
testQuest

Solution Elements

1. TestQuest Mobile and Wireless Test System
   - Baseline test systems and software
2. Carrier Specific Test Infrastructure
   - Test framework for a specific carrier including standard reports, databases, logs etc
3. Carrier Specific Test Cases
   - Implementation of a specific carrier's test cases
4. "Stay-Connected" Handset Connectivity
   - Package of hardware and software connectivity to handsets
5. Network Emulator Connectivity
   - Library for controlling network emulators

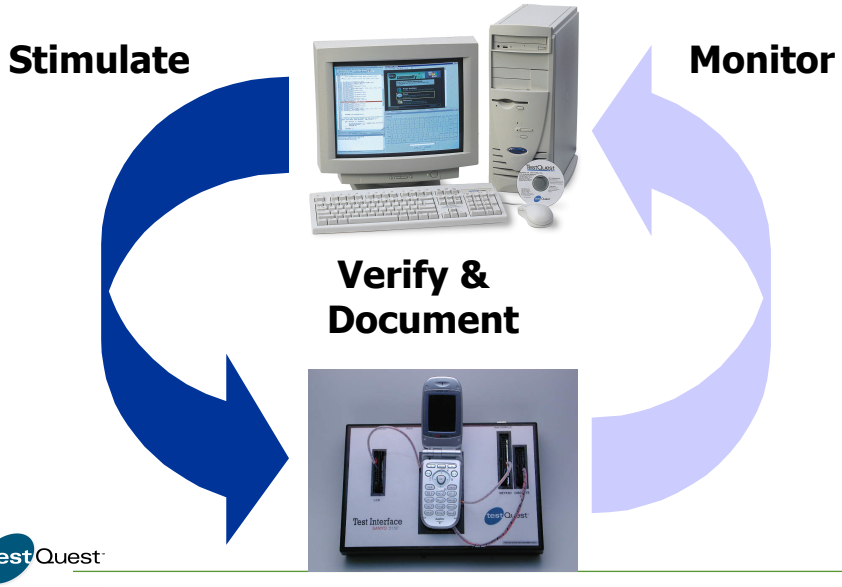
testQuest

TestQuest Carrier Test Solution



testQuest

TestQuest Pro System Approach



testQuest

TestQuest Pro – Product Overview

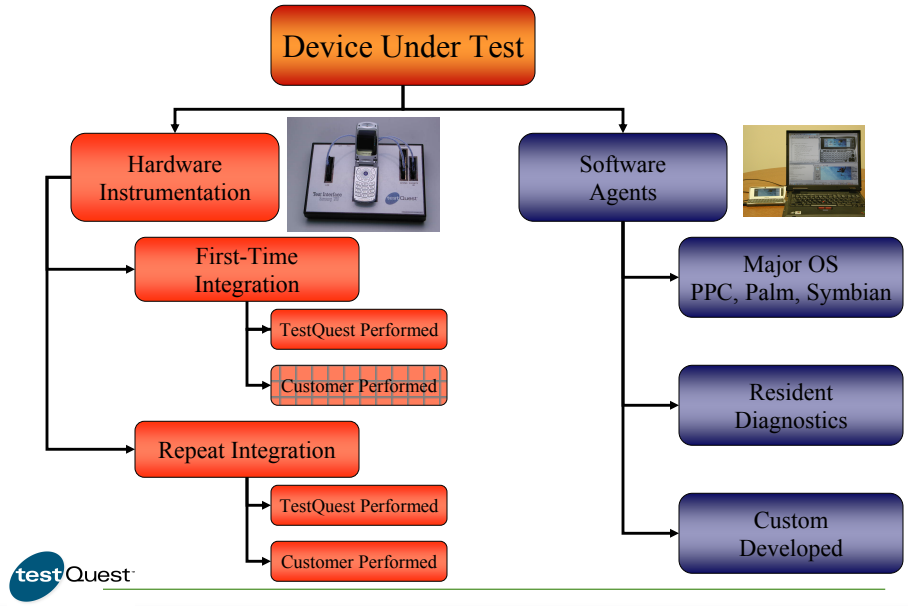
- Test Automation Tool for mobile and wireless devices applications, and services
- High productivity script development environment
- Industry standard scripting language - C/C++
- Open platform for integration of optional modules

TestQuest Pro – Product Modules

Three main modules in the product which together enable high-fidelity functional test automation

1. Connectivity hardware/software
   - Drive the inputs and capture the outputs of the SUT
2. A powerful Integrated Development Environment (IDE) & Software libraries
   - Generate, edit, and debug automation test scripts
3. Test Management
   - Manage and execute scripted automation

Target Connectivity Methods



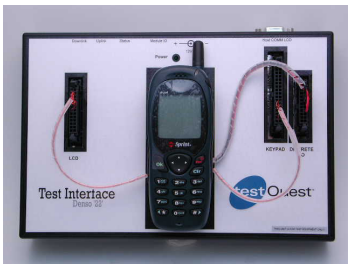
testQuest

Handset Connectivity Options

1. Hardware instrumented connectivity
   - Zero software footprint; maximum coverage; complementary with software tools
2. Software agents on major commercial OSs
   - Symbian, PPC, Palm
3. Agents for proprietary OSs
   - iTRON, in-house
4. Connection to Baseband chipset DM ports
   - CDMA; W-CDMA;

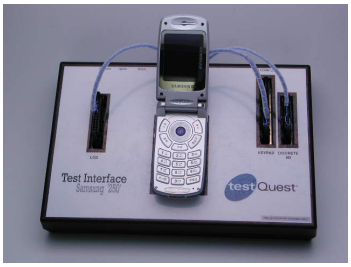
testQuest

Handsets – Hardware Connectivity



Denso 2200

Sanyo 5150



Samsung 250

testQuest

Software Connectivity

testQuest

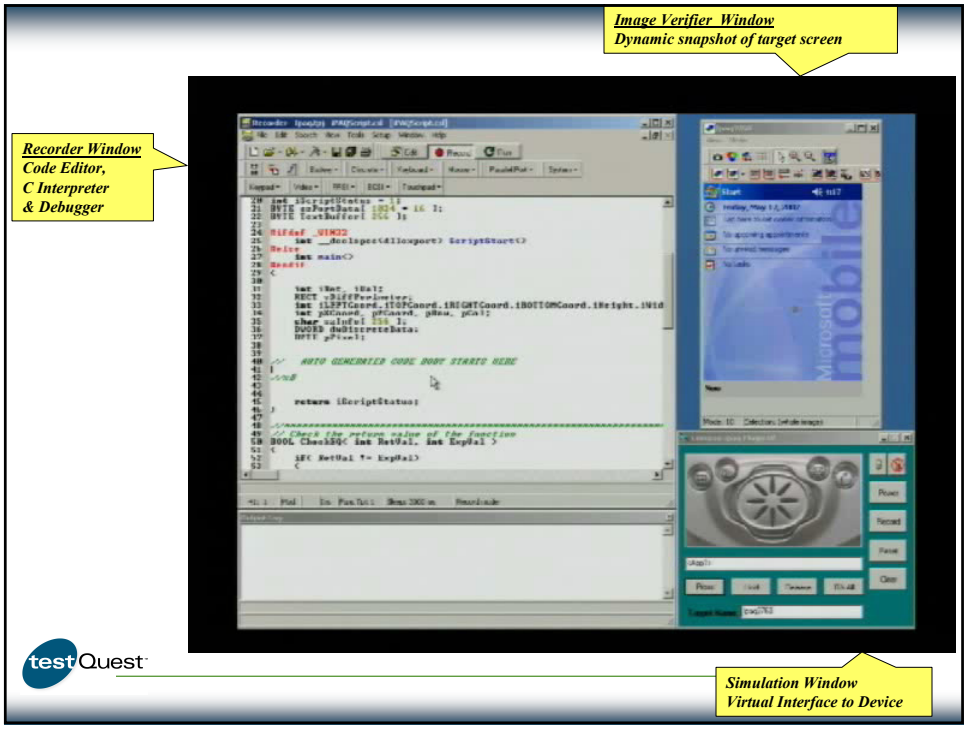
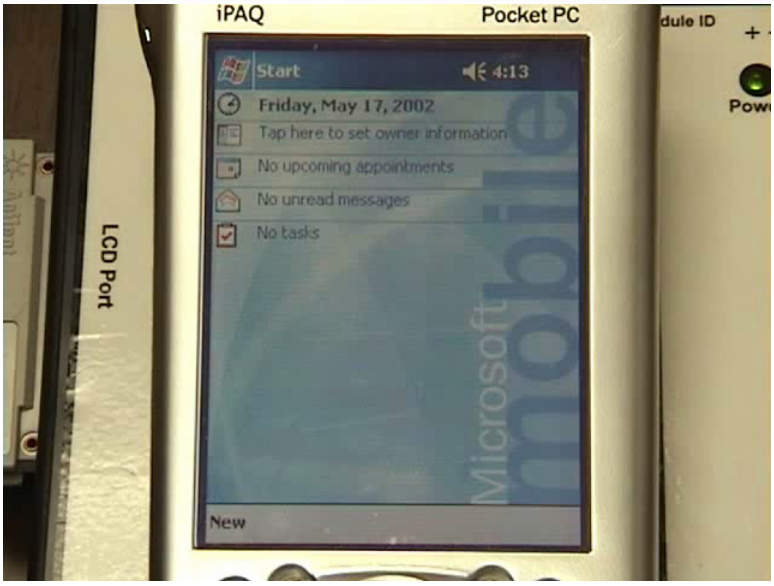
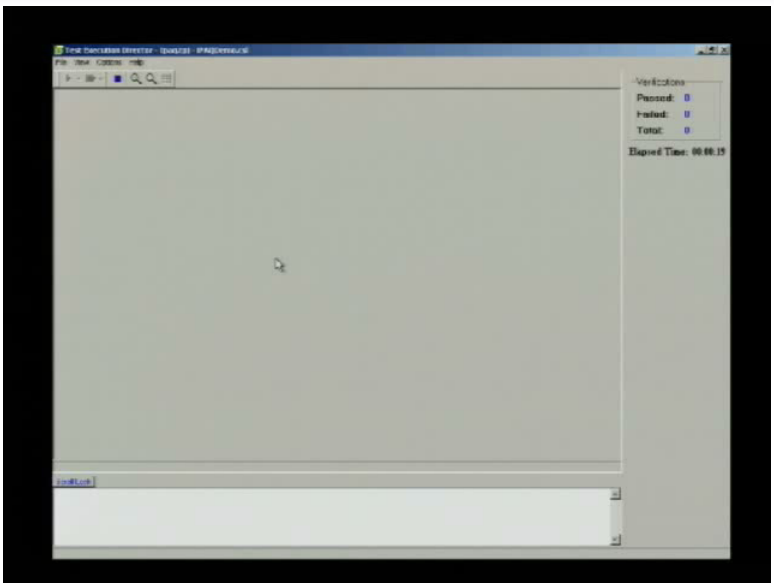


Image Verifier Window
Dynamic snapshot of target screen
Recorder Window
Code Editor,
C Interpreter
& Debugger
Simulation Window
Virtual Interface to Device
testQuest


iPAQ
Pocket PC
Start
4:13
Friday, May 17, 2002
Tap here to set owner information
No upcoming appointments
No unread messages
No tasks
New
LCD Port
testQuest

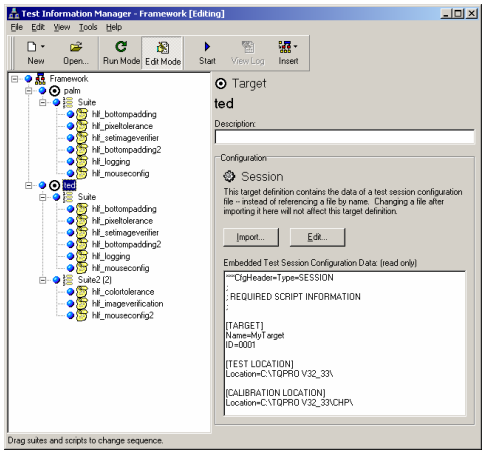


testQuest

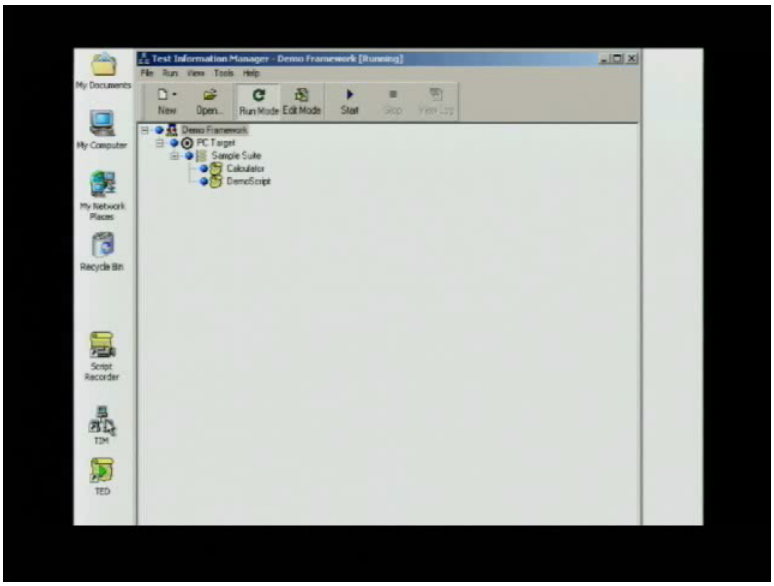
Test Runtime & Management

Test Information Manager (TIM)

- Enables organization of tests into test frameworks
- Allows structured execution of hundreds of tests
- Enables unattended testing
- Provides an organized approach to extensive product testing



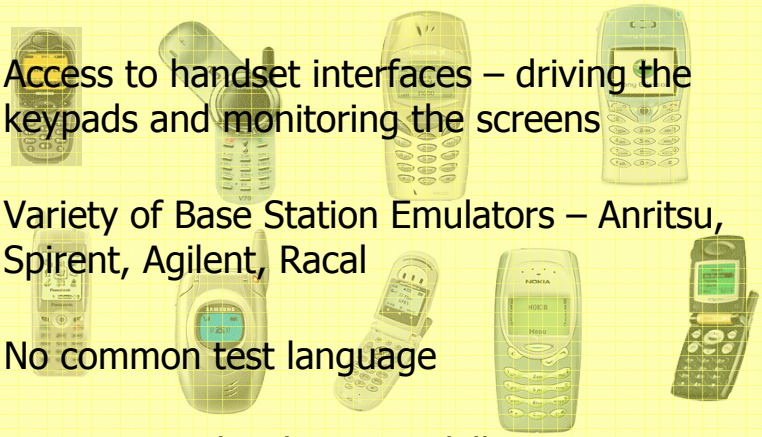
testQuest



testQuest

Automation – Rollout Challenges

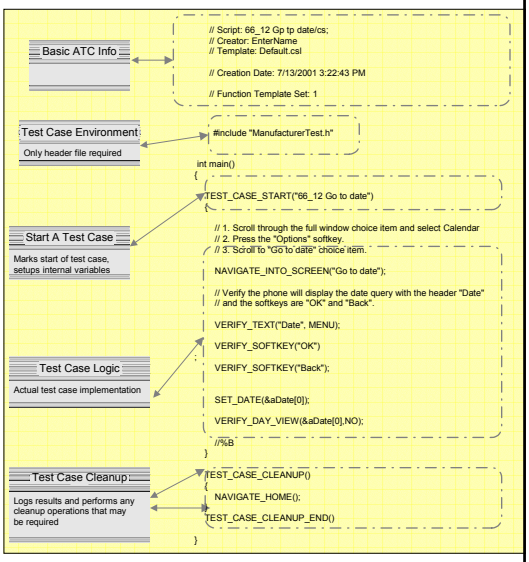
1. UI and Navigation differences across handsets
2. Access to handset interfaces – driving the keypads and monitoring the screens
3. Variety of Base Station Emulators – Anritsu, Spirent, Agilent, Racal
4. No common test language
5. Automation development skills



testQuest

Mobile Test Architecture

1. Approximately 150 handset and BSE neutral commands
2. Test automation is expressed in problem-specific terms vs. handset-specific terms
3. Insulates test implementation from handset and BSE specifics
4. Greatly reduces test development and maintenance requirements



testQuest

Summary

Vendors and carriers needed a common broadly accepted and "endorsed" turnkey test solution

- TestQuest understood the shared problem that carriers and handset manufacturers have
- TestQuest has the solution that has been widely adopted and endorsed by handset vendors and carriers
- TestQuest has expertise with handsets, carriers, and test to ensure the success of our customers

testQuest

QW2002 Paper 2V2

Mr. Jessee Ring
(Software Quality First)

Agile Software Process Improvement

Presentation Abstract

Software process improvement (SPI) is an effective means by which software development and maintenance organizations can move toward higher performance. But SPI can easily become a bureaucratic, slow, ineffective endeavor. This does not have to be the case. This presentation will describe an approach to software process improvement that is streamlined but effective. It can be used by large and small organizations to develop “lean and mean” SPI projects.

About the Author

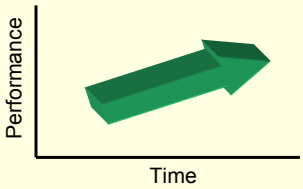
Jessee Ring has twenty-five years experience in systems and software development. He has fulfilled roles of software developer, tester, integrator, systems analyst, project manager, department manager, software quality manager, corporate executive, and independent consultant. He has experience in both large and small organizations and diverse environments. He has a Bachelor’s and Masters degree in electrical engineering, and achieved numerous scholastic honors. He is currently working as an independent consultant in the areas of software quality, testing, process improvement, metrics, and project management.

Agile Software Process Improvement

Jessee Ring
Software Quality First
40119 San Carlos Place
Fremont, CA 94539
510-915-2353
Fax: 510-573-7464
Email: sqa1st@attbi.com
Web: www.sqa1st.com

Certification vs. Improvement

- Use the process model as a guide to making improvements.
- Using the model as a criteria for "certification" often leads to irrational behavior.
- All improvement projects should be linked to business



© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

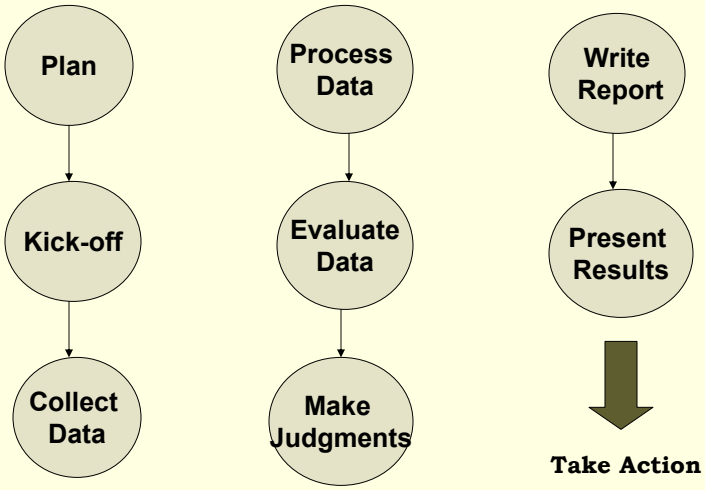
Agile SPI - Appraisal

- **Done by one person**
- **Must have extensive experience in multiple areas:**
  - Systems and software development
  - Testing
  - SQA
  - SPI
  - As an individual contributor & manager

© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Appraisal Process

Reference: "CMM Appraisal Framework" – SEI publication dated Feb., 1995



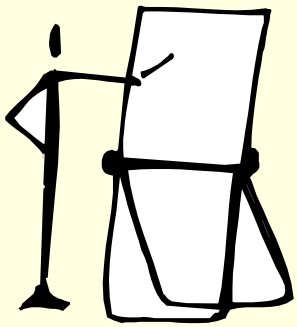
© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Time Requirements For Appraisal

- One Calendar Week
- 40 – 50 hours by the appraiser
- Kick-off Meeting: 1 hour
- Participants: 2 – 3 hours during the week
- Results presentation: 1 Hour

© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Appraisal Data Collection



- Questionnaire
  Based upon the process model being used.
- Small group discussions
  Groups can be by job function or cross-functional.
- Document review
  Selected documents.

© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Appraisal Questionnaire Data Analysis

Name	RM-1	RM-2	RM-3	RM-4	RM-5	RM-6	RM-all
Person 1	1	1	1	1	1	1	
Person 2	1	1	1	1	1	1	
Person 3	1	1	1	1	2	2	
Person 4	1	1	1	1	2	2	
Person 5	1	1	2	1	2	2	
Person 6	1	1	2	2	2	2	
Person 7	2	1	2	1	3	4	
Number of Yes (1)	6	7	4	6	2	2	
% Yes	85.7	100.0	57.1	85.7	28.6	28.6	64.3
Number of No (2)	1	0	3	1	4	4	
% of No	14.3	0.0	42.9	14.3	57.1	57.1	31.0
Number of N.A. (3)	0	0	0	0	1	0	
% of N.A.	0.0	0.0	0.0	0.0	14.3	0.0	2.4
Number of Don't Know (4)	0	0	0	0	0	1	
% of Don't Know	0.0	0.0	0.0	0.0	0.0	14.3	2.4

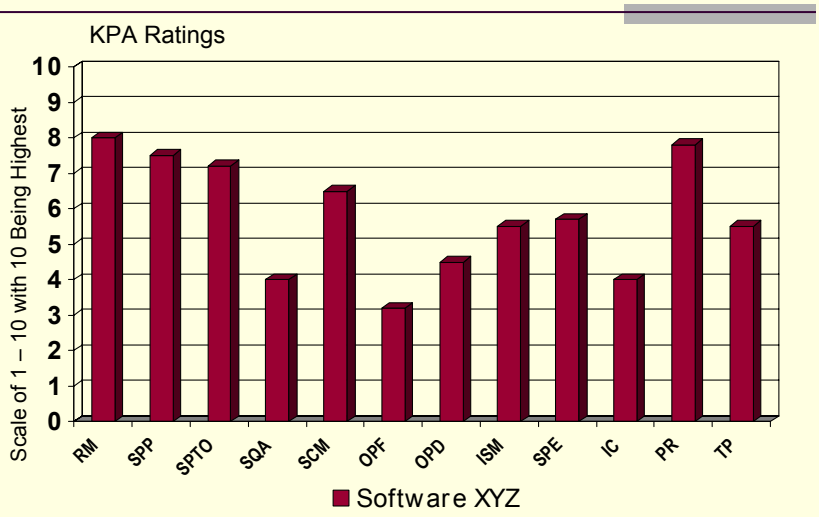
© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Process Model Based Ratings

	Rating	Average Per KPA
Requirements Management		
Goal 1: System requirements allocated to software are controlled to establish a baseline for software engineering and management use.	10	10.0
Goal 2: Software plans, products and activities are kept consistent with the system requirements allocated to software.	10	
Software Project Planning		
Goal 1: Software estimates are documented for use in planning and tracking the software project.	3	5.0
Goal 2: Software project activities and commitments are planned and documented.	5	
Goal 3: Affected groups and individuals agree to their commitments related to the software project.	7	

© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Process Model Based Ratings – Graphical Presentation



© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Appraisal Results

- Summary of Results
  - Overall Strengths
  - Major Issues
  - *Consequences*
  - *Recommendations*
- Detailed Results
  - Per KPA
  - Strengths
  - Issues
  - Recommendations

© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Appraisal Results

- "Consequences" of the issues that were found are described.
- Makes the issues of more immediate interest.
- Answers the question: "So what?".
- Justifies the allocation of resources.

© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Appraisal Results

"Recommendations" are a critical element of the Agile SPI approach. By including them in the Appraisal Report, people become focused on Improvement action. It also speeds up the whole process since the organization doesn't have to wait to find out what to do.

© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Improvement Action

- **Don't do anything just because the process model tells you to do it.**
- **All improvement actions should be driven by a belief in the business benefit to be received.**
- **Be results oriented. Identify some metric that will show the effect of improvement actions.**

© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Metrics Plan

Two types of metrics: product & process

Product Metrics

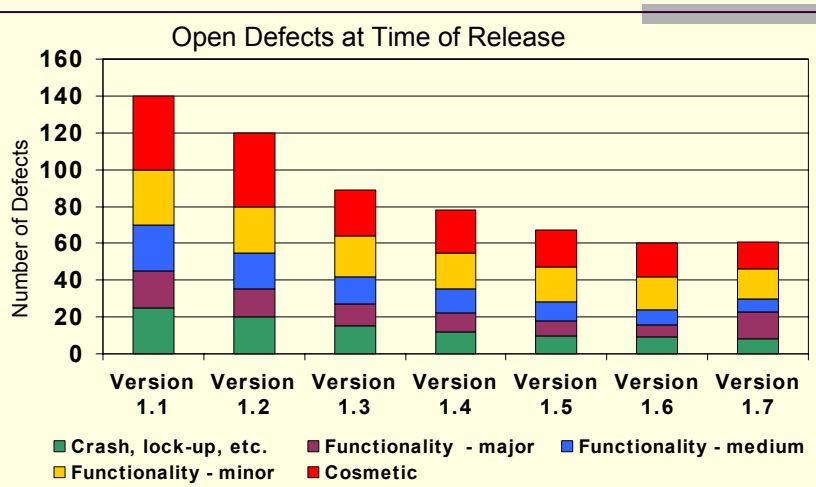
- Number of open defects by severity
- Defect density
- Defect arrival rate
- Total size of the software
- Number of lines of code added, deleted, modified
- Number of new features and/or fixes contained in a new version
- Reliability growth
- Complexity of the software

Process Metrics

- Test effectiveness (number of "escaped" bugs)
- Percent of bugs found by test and inspections.
- Amount of time to find a bug by test and by inspections.
- Cycle time of regression testing.
- Test coverage
- Inspection coverage
- Process maturity ratings over time

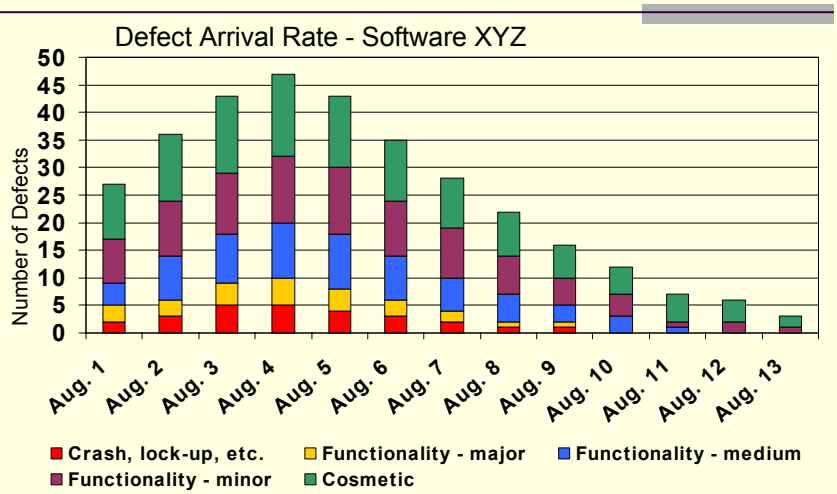
© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Sample Metrics



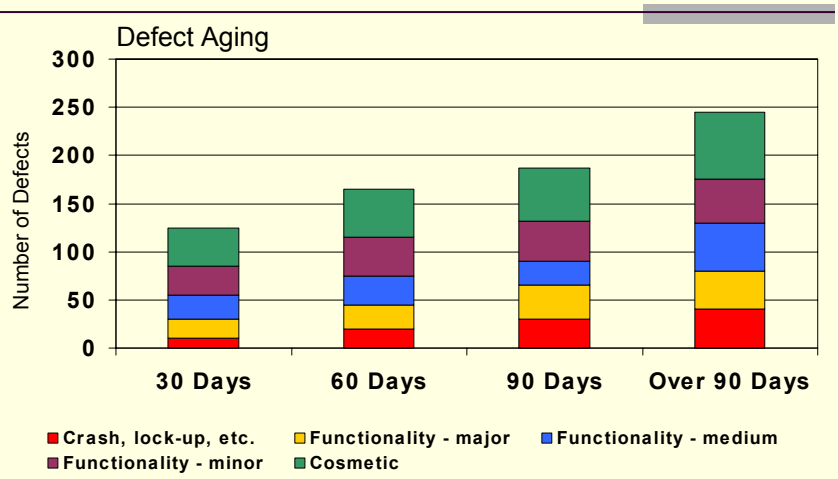
© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Sample Metrics



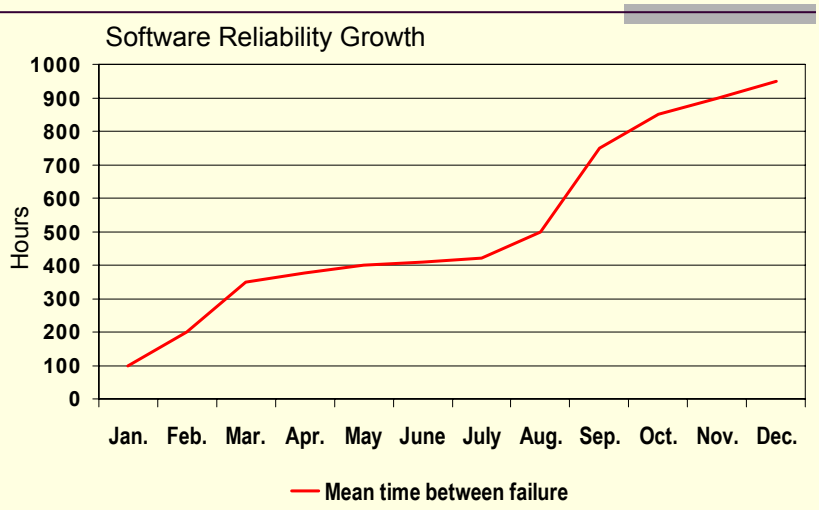
© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Sample Metrics



© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI – Sample Metrics



© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

The Need For Software Process Improvement

Mars Climate Orbiter

Lost because of failure to convert from English to metric units in the navigational (orbiting) program.

Mars Polar Lander

Lost because of a "missing line of code" that caused the braking engines to shut down prematurely.

Result

- $360 million out-of -pocket cost
- Indefinitely postpone all future launches to Mars
- Address shortcomings at JPL in areas ranging from systems analysis and testing to staffing and communications.

© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

Agile SPI - Summary

- Use the process model as a guide; not a mandate.
- Link all process improvements to business results.
- Use process maturity ratings to guide improvement actions, not as a final goal. Don't publish them.
- Process appraisal done by one very experienced person.
- In the process appraisal report, identify consequences of the issues that were found, and recommendations.
- Use metrics to manage the improvement actions.
- KISS

© 2002 Software Quality First sqa1st@attbi.com www.sqa1st.com

QW2002 Paper 3V1

Mr. Joe Ponczak
(McCabe & Associates)

Change Impact Testing

Presentation Abstract

As a result of increased competition, businesses are currently pushing their products out to the market in record time. Some e-commerce businesses even are on a weekly release cycle. Add to that the current economic conditions coupled with the demand for higher quality software, testers are now facing unprecedented challenges.

Many test groups have developed hundreds if not thousands of automated and manual test scenarios. As developers constantly change code, and as the amount of time given to testers decreases, finding out how to test the changed code becomes more and more problematic. It is neither practical nor efficient to run all the scenarios on such a frequent basis due to a few code changes. Approaches that address the impact of code changes and identify which scenarios to execute will save time and increase quality.

This presentation will discuss an effective unit/integration testing strategy for rapid release schedules. The focus will be on

- Identifying changes
- Assessing the change impact
- Identifying which scenarios to execute

Time permitting, client success stories will be discussed as well.

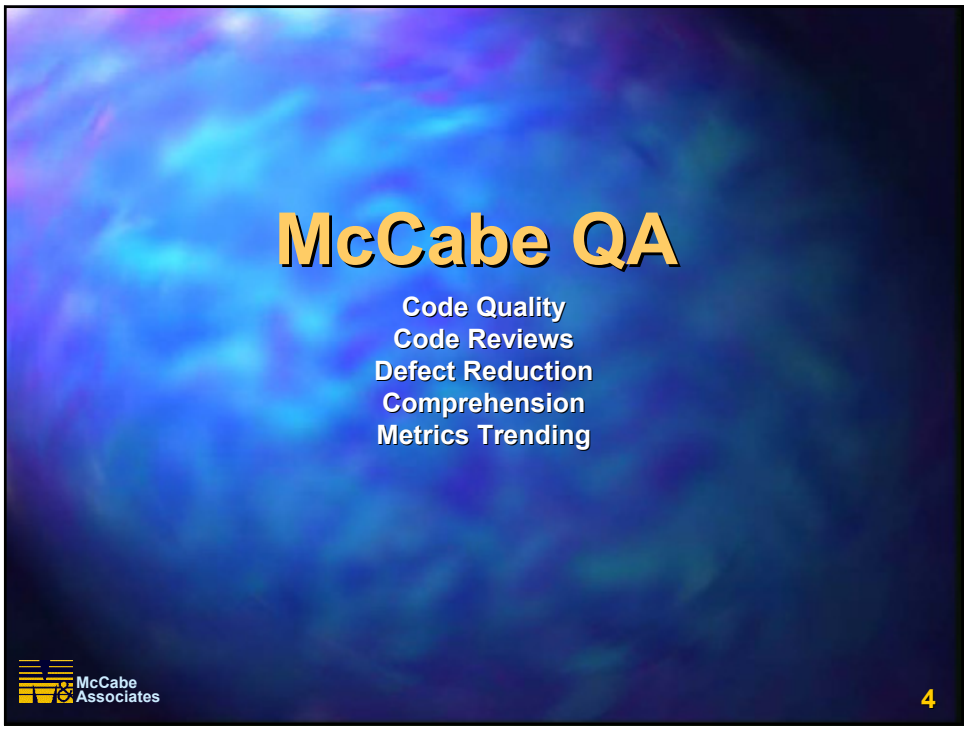
About the Author

Joe Ponczak is a Regional Account Manager for McCabe & Associates, a leading provider of QA and CM solutions. He has 10 years of QA and consulting experience in a wide range of programming languages, applications, and environments.

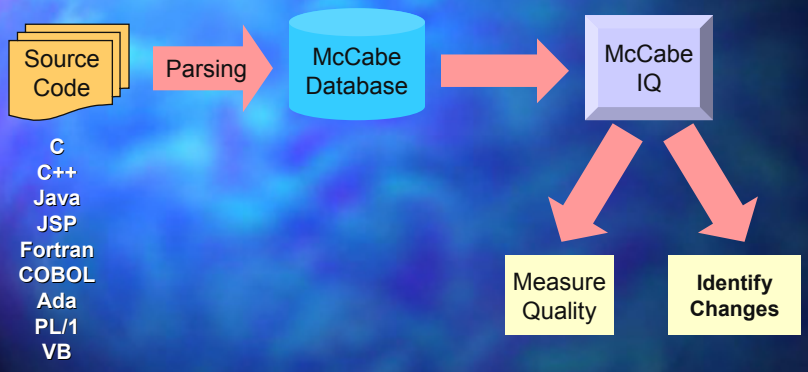
McCabe in the Process

- Development
  - Code Reviews
  - Developer Testing
  - Quality Measures
- Maintenance
  - Comprehension
  - Change Management
  - Regression Testing
- Quality Assurance
  - Measurement
  - Automated Reports
  - Metrics Trending
- Functional/System Test
  - System-Wide Coverage
  - Focus Testing
  - Prioritise Effort
  - Improve Efficiency

McCabe
Code Quality
Developer Testing
Code Reviews
Testing Time
Defect Reduction
Testedness
Comprehension
Metrics Trending
Coverage of Changed Code
Tracking Changes
INTEGRATED QUALITY
McCabe Associates
3


McCabe QA
Code Quality
Code Reviews
Defect Reduction
Comprehension
Metrics Trending
McCabe Associates
4

What is McCabe IQ?



C
C++
Java
JSP
Fortran
COBOL
Ada
PL/1
VB

Static Analysis Process

McCabe & Associates

5

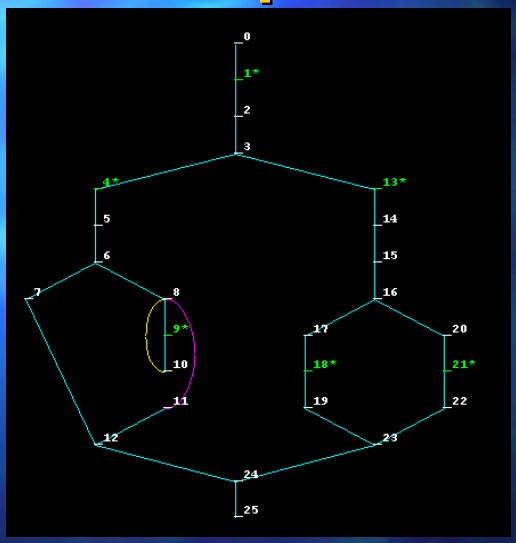
Measuring Quality
flowgraphs

```
    function_test(y)
0   {
        x=3;
1       if ( y < 4 )
2           x=sin(y);
        else
3           x=cos(y);
4       x=x*x;
5   }
```

McCabe & Associates

6

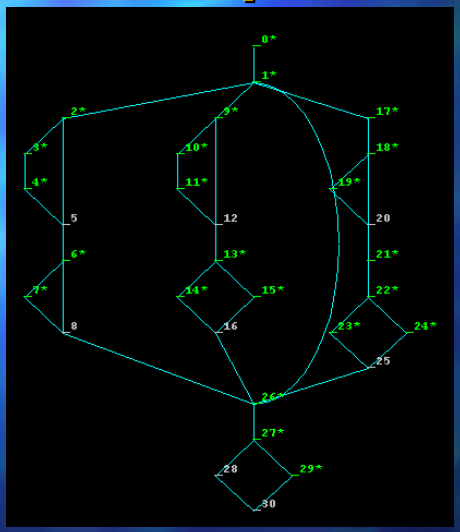
Example 1



McCabe Associates

7

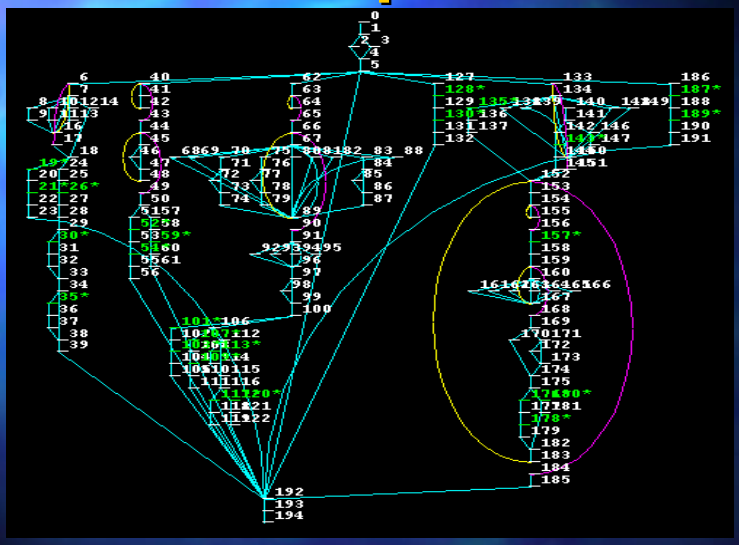
Example 2



McCabe Associates

8

Example 3

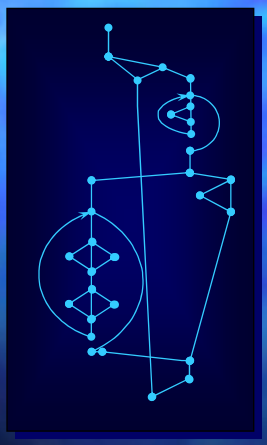


McCabe Associates

9

Software Measurement

complexity



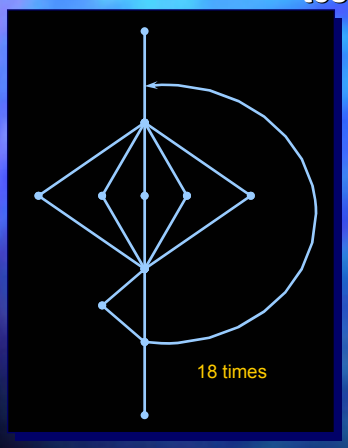
- Flowgraphs Visualize Logic
- Useful for:
  - Comprehension
  - Test Derivation
- How Can Tests be Derived Using Flowgraphs?

McCabe Associates

10

Software Measurement

testing effort



How many tests
are needed to exercise
every line of code
at least once?

Code Coverage
Testing Technique

McCabe & Associates

11

Software Measurement

code coverage

Example 'A'

Example 'B'

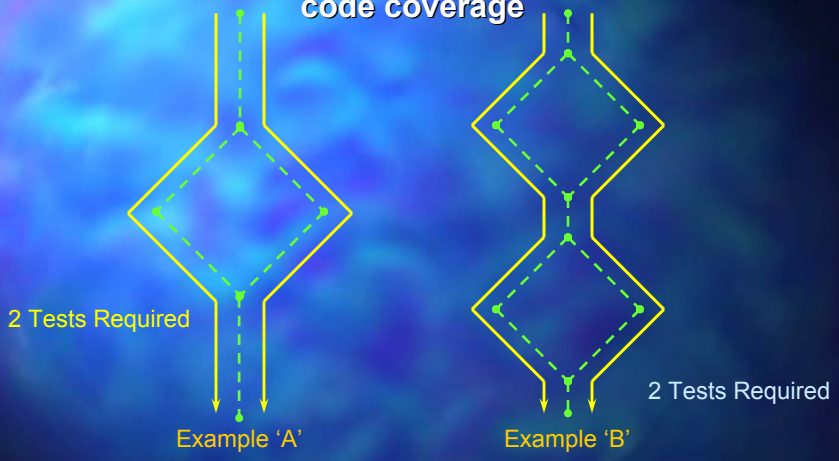
Which function is more complex?

McCabe & Associates

12

Software Measurement

code coverage

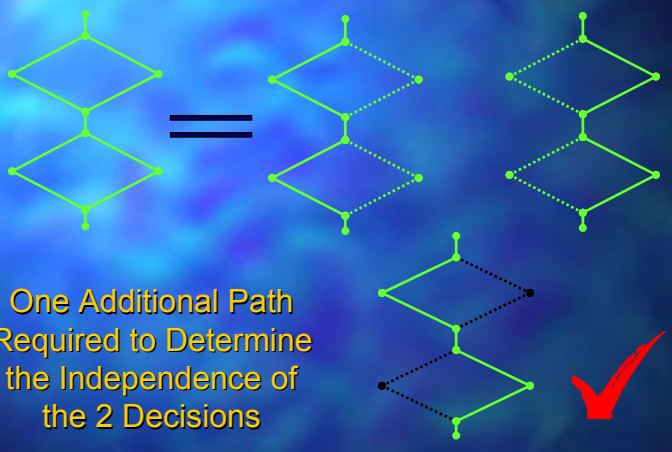


Code Coverage is <u>not</u> proportional to complexity

McCabe & Associates

13

Software Measurement



One Additional Path Required to Determine the Independence of the 2 Decisions

McCabe Complexity

Number of linearly independent paths

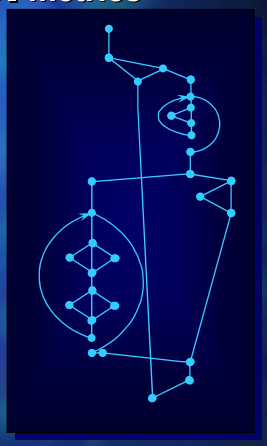
McCabe & Associates

14

Software Measurement

McCabe unit level metrics

- Complexity
  - Simple / Scalable
  - Language Independent
  - Easy to Measure
  - Indication of Testing Effort
- Essential Complexity
  - Structure
  - Maintainability
  - Re-engineering effort



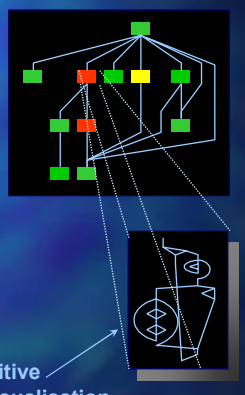
McCabe & Associates

15

Software Comprehension

Visualize software

- System Level Views
  - Complete System Architecture
  - Overlay of Quality Metrics
  - Interaction of Modules
  - Identify impact of changes
- Unit Level Views
  - Flowgraphs & Code Listings
  - Detailed Metrics Analysis
  - Summary Reports

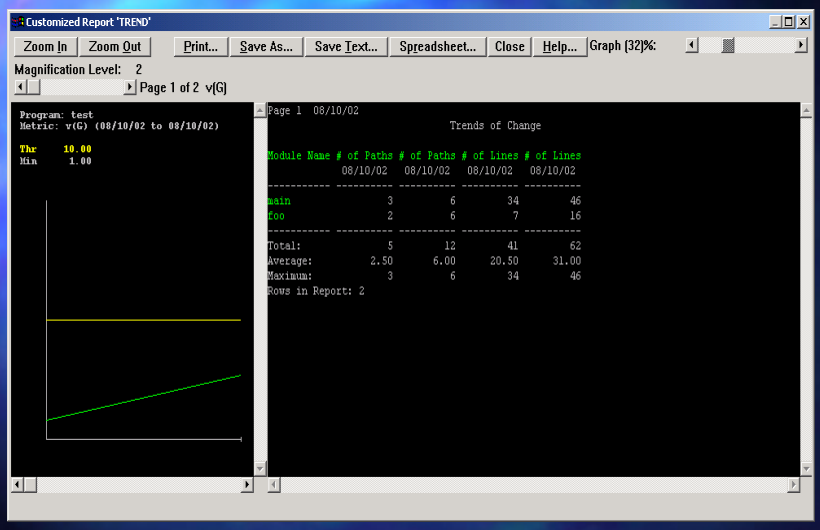


Context Sensitive
Measurement and Visualisation

McCabe & Associates

16

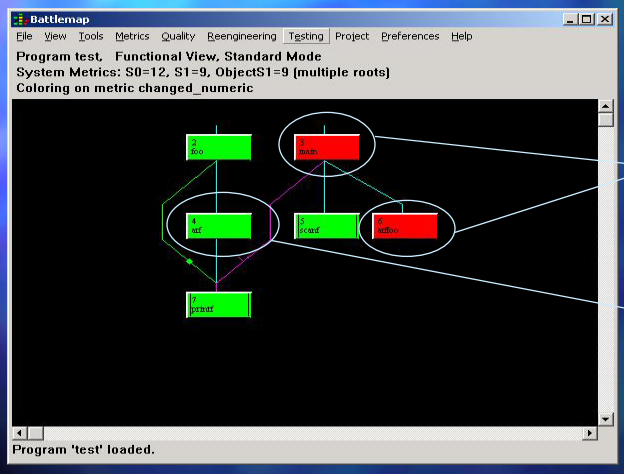
Metric Impact of Change



McCabe Associates

17

Impact of Change on Structure



Changed Code from last version

Unchanged Code from last version

McCabe Associates

18

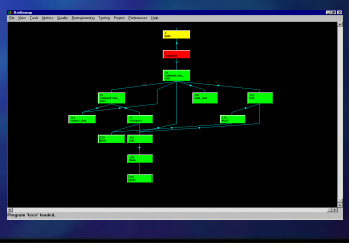
Platforms / Languages

Languages:

- COBOL
  - Many Dialects
- C
- C++
- Java/JSP
- Fortran
- Ada
- Visual Basic
- PL/1
- Model 204

Platforms:

- Windows
  - NT/2000/XP
  - 95/98
- Unix
  - Solaris
  - AIX
  - HPUX



McCabe Associates

19

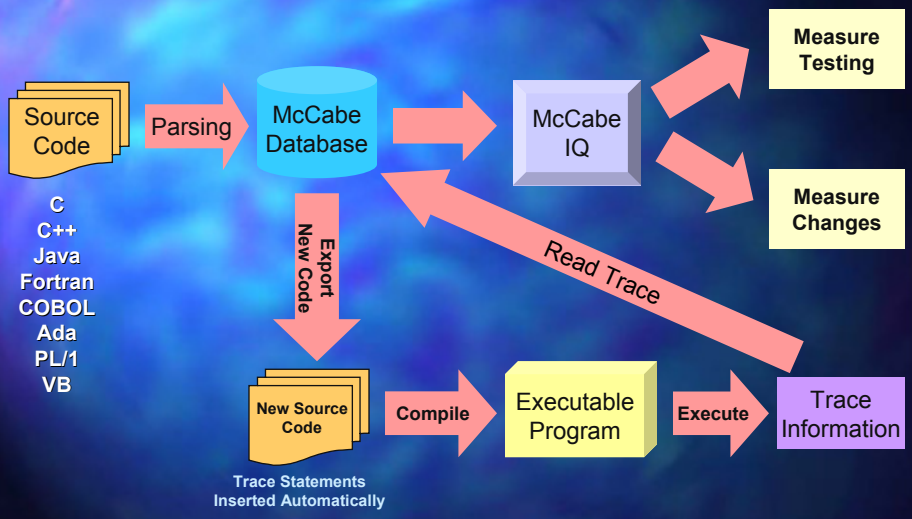
McCabe Test

Developer Testing
Testing Time
Testedness
Coverage of Changed Code

McCabe Associates

20

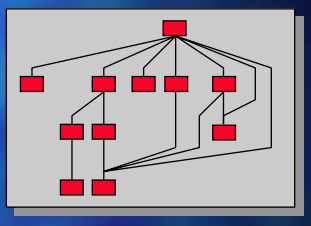
What is McCabe IQ?



Dynamic Analysis Process

Coverage Mode

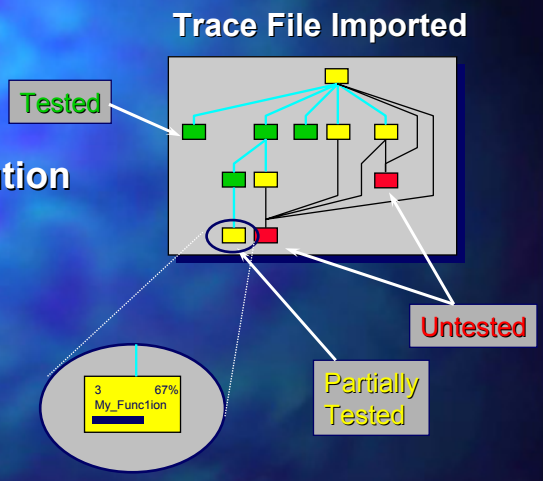
- Static Battlemap
- Color Scheme Represents Coverage



No Trace File Imported

Coverage Results

- Colors Show "Testedness"
- Lines Show Execution Between Modules

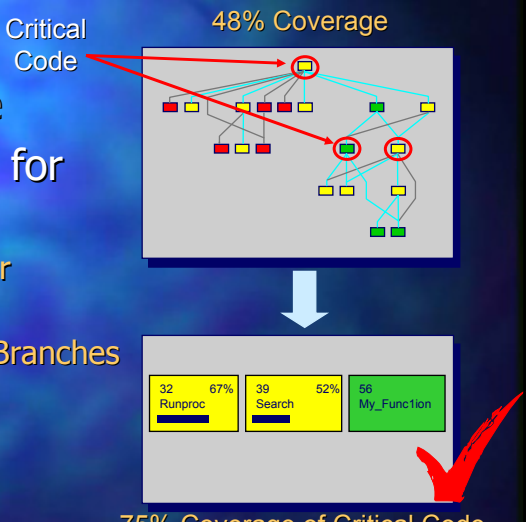


McCabe & Associates

23

McCabe Test

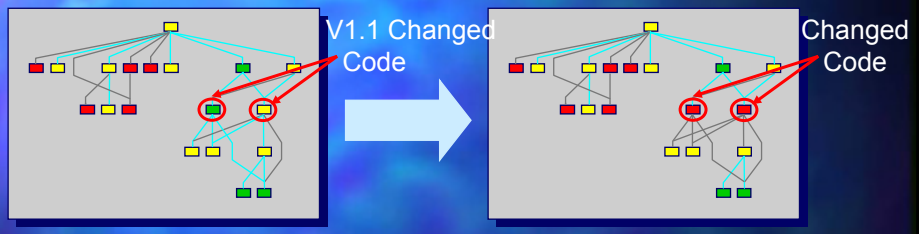
- Import Coverage
- Assess Coverage for "Critical" Code
  - Coverage Report for "Critical" Group
  - Examine Untested Branches



McCabe & Associates

24

McCabe Change



Version 1.0 - Coverage Results

Version 1.1 - Previous Coverage Results Imported Into New Analysis

McCabe Associates

25

McCabe Test

- Verify Test Suites
  - Confirm Test Effectiveness
- Derive Additional Tests
  - New Functional Combinations
- Prioritise Remaining Test Efforts
  - Focus on Critical Code Elements
- Remove Duplication
  - Remove "Low Value" Test Sets

McCabe Associates

26

Demonstration

McCabe IQ
McCabe Test

27

QW2002 Paper 3V2

Dr. Ira Baxter
(Semantic Designs, Inc.)

DMS: Software Quality Enhancement via Automated Software Analysis, Modification and Generation

Presentation Abstract

Most software engineering quality tools are analysis-centric point-solutions: they provide one kind of very specific specific analysis. If you can't find what your company needs off the shelf, however, building a custom tool is almost impossibly expensive. What is really needed is an available, agile cost-effect technology for not only doing custom analyses, but coupling those analyses to mechanically reliable change processes to effect improvements suggested by the analyses.

This talk will describe DMS, a customizable analysis and modification engine that can automatically analyze and change large scale software systems. We will show how the undelying technology works and can be used to read and transforms programs written in arbitrary langauges. We will provide a number of interesting production application examples, including removal of preprocessor directives, XML parser generation, duplicate code detection, and automated translation of one programming language to another.

About the Author

Dr. Baxter has been building systems software for over 30 years. He is presently the Chief Technology Officer of Semantic Designs, a software-tools building company. He is also active in academic conferences, and is presently the Program CoChair for the International Conference on Software Maintence 2002 (Montreal).

DMS®

Software Quality Enhancement via Automated Software Analysis, Modification and Generation

Ira D. Baxter

Semantic Designs, Inc.

www.semdesigns.com

Quality Week 2002

Automation In Software Engineering

- Manual coding/analysis expensive
  - Typical: $100K/year/man/4KSLOC → $25/SLOC
  - Presently not much automated help; very expensive to build
- Automation possible
  - For problem *domains* with well-defined *semantics*
    - Computer languages, specification languages, …
  - For well-defined tasks
    - Analysis: error detection, test support, documentation extraction, reverse engineering, …
    - Modification: structure improvement, error handler insertion, API change, code porting, …
    - Code generation: from specs, diagnostics, test cases,….
  - Using *Program Transformation* technology
    - = Generalized compiler componentry
    - Researched by community over past 25 years

DMS® Software Reengineering Toolkit

- Customized, *automated*
  analysis, modification, porting or generation
  - Enables wide variety of source-based SE tasks to be automated
  - For sources for large scale software systems
    - Scalable to millions of source lines, tens of thousands of files
    - Parallel processing foundations to support scale
  - Handles many *and mixed* languages simultaneously
    - C, C++, Java, Ada, Fortran, SQL, XML, assembler, …
  - Generalized compiler technology conveniently *integrated*
    - Parsing, Analyzing, *Transforming*, Prettyprinting
    - Enables practical customization for desired automation task
    - Predefined support for standard computer languages
    - Huge infrastructure cost amortized over many tasks/customers
- Semantic Designs Supporting Services and Tools
  - Consulting & Training to customer on DMS usage
  - Implementation of DMS customization
  - Selected SE tasks prepackaged: formatting, test coverage, …

DMS Impact on Quality and Process

- Quality
  - (Re)use of tested specification techniques
    - avoids ad hoc descriptions
  - Reuse of *abstract* generative components (transforms)
    - *Not code reuse, but rather implementation knowledge reuse*
  - Specifications and implementation steps inspectable by others
  - Reuse of tested synthesis/modification methods
  - *Mechanical, reliable construction of product*
  - Easier recovery from errors: correct mistake, re-execute task
- Process
  - Reliable components --> avoid rework after changes
  - Mechanically repeatable implementation steps
  - Focus on knowledge acquisition rather than repeated coding events

Overview

- ***DMS***® Software Reengineering Toolkit
  - Defining notations ("domains") for specs and legacy systems
  - Parsing and prettyprint
  - Transformation mechanics
- Applications for Software Quality Improvement
  - C++ preprocessor conditional removal
  - Automatic Code Generation (XML Parsers)
  - Clone Detection/Removal
  - Porting application software to new languages
- Purpose: *Educate audience about new generation of tools*

DMS Domain Parts

- Syntax
  - External Form (what you can say: string or graphical)
  - Internal Form (How DMS stores it)
  - *Parser* (how to convert external form to internal form)
  - *PrettyPrinter* (how to display the Internal Form)
- Semantics (what the Internal Form *means*)
  - *Optimizations* (how to optimize in the domain)
  - Refinements (how to transform IF to another IF)
  - *Analyzers* (how to analyze in the domain)
  - Attachments (procedures to enhance DMS efficiency)

DMS Domain for Java
Parser + Pretty Printer

```
nested_class_declaration = nested_class_modifiers class_header class_body ;
  <<PrettyPrinter>>:  { V(H(nested_class_modifiers,class_header),class_body); }

class_header = 'class' IDENTIFIER ;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER); }
class_header = 'class' IDENTIFIER 'implements' name_list ;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER,'implements',name_list); }
class_header = 'class' IDENTIFIER 'extends' name;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER,'extends',name); }
class_header = 'class' IDENTIFIER 'extends' name 'implements' name_list ;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER,'extends',name,'implements',name_list);

class_body = '{' class_body_declarations '}' ;
  <<PrettyPrinter>>:  { V(H('{',STRING(" "),class_body_declarations),'}'); }

nested_class_modifiers = nested_class_modifiers nested_class_modifier ;
  <<PrettyPrinter>>:  { H(CH(nested_class_modifiers[1]),nested_class_modifier); }
```

… + 300 more rules…(COBOL is 3500!)

Parsing to Abstract Syntax Trees

A Program Representation analyzable by Computers

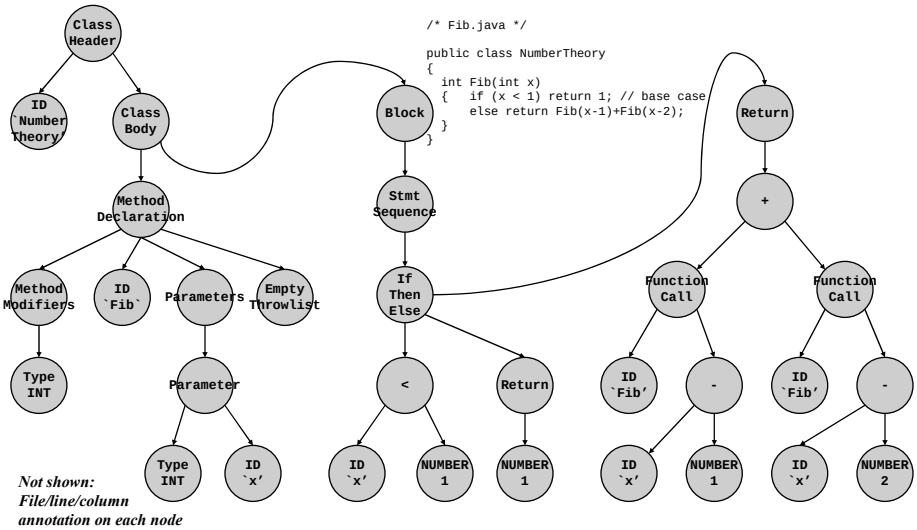
- Use DMS grammar domain to define language syntax
- DMS generates lexer/parser automatically
- Parser reads source file(s)
  - Captures comments
  - Carries out lexical conversions (e.g, FP text -> IEEE binary fp)
  - Builds Abstract Syntax Tree
  - Records Position of every node (file, line, col)
- Present capability for the following domains
  - *Specification:* Spectrum, BNF, Rose Models
  - *Technology:* XML, IDL, SQL
  - *Implementation:* C/C++, COBOL, Java, Ada, VB6, Fortran, Verilog

A Simple Java Program

```
001  /* Fib.java */
002
003  public class NumberTheory
004  {
005    int Fib(int x)
006    {   if (x < 1) return 1; // base case
007        else return Fib(x-1)+Fib(x-2);
008    }
009  }
010
```


Abstract Syntax Tree (AST) for Fib Class

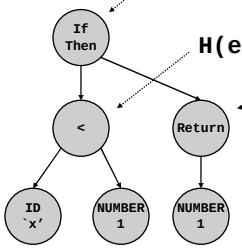
... free of lexical properties ('text shape') of program ...



PrettyPrinting: “AntiParsing”

- Conversion of AST back to text file
- Handles indentation, comments, literal formats...
- Uses DMS Box language to compose PP fragments

```
V(H('if','(',condition,')'),
  I(then_stmt));
```




```
H(expression1,'<',expression2);
```

```
H('return',expression,';');
```

Prettyprinted result:

```
if (x<1)
  return 1;
```


Winner: Obfuscated “C” Contest

The Maintenance Programmer’s Nightmare

```
#include                    <sys/time.h>
#include                    <math.h>
#include                    <X11/Xlib.h>
#include                    <X11/keysym.h>
                            double L,o,P
                            ,_=dt,T,Z,D=1,d,
                            s[999],E,h=8,I,
                            J,K,M[999],m,n,O
                            ,i[999],j=.038-3;i=
                            153,(r=rand)(),u,v,w,S=
                            74.5,t,x=72,y=97;main(){
                            a_B,A=32,c,*f,*W;
                            int N,q,C,y,p,U;
                            Window z;char f[52]
                            ,*d="d_ P@wH'E+B-TD; for(o=(I=D*w+E
                            sin(j); a=B*T-D+E*W; XClearWindow(e,z); t=T+E+ D*B*W; J+=d-_*D-_*FE; P=W*E*B-T*D; for(o=(I=D*w+E
                            *T*B,E=d/K*B+v+B/K*F*D)_; P<y; )t=p[s]+I; E=c-p[W]; D=n[p]-L; K=D*m+B*T+*E; I=(p [n]+W[ p]+p[s
                            ]=_*0)K+fabs(W+T*I-I*E+*D*F)<1&&(D=t,h=2-_*E)>K)N=1e4; else{ d=W/K*4E2+262; c=_*2+W*42/K
                            *D+I*fabs(W+T*I-I*E+*D*F)<1&&(D=t,h=2-_*E)>K)N=1e4; else{ d=W/K*4E2+262; c=_*2+W*42/K
                            XDrawString(o,z,K,20,380,f,17); D=1/(1+15*I-I*E+B*W+_*2*Z-_*D*F); for(;;XPending(e);u *=S)=1){
                            XEvent z; XNextEvent(e,&z);
                            ++*((N=XLookupKeysym
                            (&z.xkey,0)-ITT?
                            N-LT? UP-N?&E:&
                            J:& u: &h); --*(
                            DN-N? N-DT ?N==
                            RT?&u: & W:&h:&J
                            ); J=m!=15*F/I;
                            c+=(I=W/ l,l?H
                            :1+*W+X*F-I-H
                            E=-1+X*4.9/l;t
                            =T*m/32-I*T/24
                            )/S; K=F*M+(
                            h*.164/l-(T+
                            E*5*T+E)/382
                            )/S-X*d-B*A;
                            a=2.63 /l*d;
                            X+=( d*l-T/S
                            *(.19*E+a
                            ),64+D*1e3
                            )*.64+D*1e3
                            Z)*.4*_ V+=
                            K*.2-_ W+=q-
                            sprintf(f,
                            "%5d %3d"
                            "%7d",p =l
                            /1.7,(C=9E3+
                            0*57.3)%8559,(int)i); d+=T*(.45-14/l*
                            X-a*130-J* .14)*_/1252+F*v; P=(T*(47
                            *I-m* 52+E*94 *D-t* 38+u*.21*E) /1e2+W*
                            179*v)/7312; select(p=0,0,0,46); v-=(
                            W*F-T*(.63*m+T* .086+m*E*19-D*25- .1T*u
                            )/10*e2)_; D=cos(o); E=sin(o); } }
```

Pretty Printing to un-obfuscate

```
#include <math.h>
#include <sys/time.h>
#include <X11/Xlib.h>
#include <X11/keysym.h>
double L, O, P, _, t = dt, T, Z, D = 1, d, s[999], E, h = 8, I, J, K, w[999],
       M, m, O, n[999], j = 3.3e-2, i = 1e3, r, t, u, v, W, S = 7.45e1,
       l = 221, X = 7.26, a, B, A = 3.22e1, c, F, H;
int N, q, C, y, p, U;
Window z;
char f[52];
GC k;
main()
{
  Display * e = XOpenDisplay(0);
  z = RootWindow(e, 0);
  for (XSetForeground(e, k = XCreateGC(e, z, 0, 0), BlackPixel(e, 0));
  scanf("%lf%lf%lf", y + n, w + y, y + s) + 1; y++);
  XSelectInput(e, z = XCreateSimpleWindow(e, z, 0, 0, 400, 400,
                0, 0, WhitePixel(e, 0)), KeyPressMask);
  for (XMapWindow(e, z);; T = sin(O))
    {
      struct timeval G = { 0, dt * 1e6 };
      K = cos(j);
      N = 1e4;
      M += H * _;
      Z = D * K;
      F += _ * P;
      r = E * K;
      W = cos(O);
      m = K * W;
      H = K * T;
      O += D * _ * F / K + d / K * E * _;
      B = sin(j);
      a = B * T * D - E * W;
      XClearWindow(e, z);
      t = T * E + D * B * W;
      j += d * _ * D - _ * F * E;
      P = W * E * B - T * D;
      for (o += (I = D * W + E * T * B, E * d / K * B + v + B / K * F * D) * _; p < y;)
        {
          T = p[s] + i;
          E = c - p[w];
          D = n[p] - L;
          K = D * m - B * T - H * E;
          if (p[n] + w[p] + p[s] == 0 | K < fabs(W = T * r - I * E + D * P) | fabs(D = t * D + Z * T - a * E) > K)
            N = 1e4;
          else
            {
              q = W / K * 4e2 + 2e2;
              C = 2e2 + 4e2 / K * D;
              N - 1e4 && XDrawLine(e, z, k, N, U, q, C);
              N = q;
              U = C;
            }
          ++p;
        }
      L += _ * (X * t + P * M + m * l);
      T = X * X + l * l + M * M;
      XDrawString(e, z, k, 20, 380, f, 17);
      D = v / l * 15;
      i += (B * l - M * r - X * Z) * _;
      for (; XPending(e); u *= CS != N)
        {
          XEvent z;
          XNextEvent(e, & z);
          ++ * ((N = XLookupKeysym(& z.xkey, 0)) - IT ? N - LT ? UP - N ? & E : & J : & u : & h);
          -- * (DN - N ? N - DT ? N == RT ? & u : & W : & h : & J);
        }
      m = 15 * F / l;
      c += (I = M / l, l * H + I * M + a * X) * _;
      H = A * r + v * X - F * l + (E = 1e-1 + X * 4.9 / l, t = T * m / 32 - I * T / 24) / S;
      K = F * M + (h * 1e4 / l - (T + E * 5 * T * E) / 3e2) / S - X * d - B * A;
      a = 2.63 / l * d;
      X += (d * l - T / S * (1.9e-1 * E + a * 6.4e-1 + J / 1e3) - M * v + A * Z) * _;
      l += K * _;
      W = d;
      sprintf(f, "%5d  %3d"
                 "%7d", p = l / 1.7, (C = 9e3 + O * 5.73e1) % 0550, (int) i);
      d += T * (4.5e-1 - 14 / l * X - a * 130 - J * 1.4e-1) * _ / 1.25e4 + F * _ * v;
      P = (T * (47 * I - m * 52 + E * 94 * D - t * 3.8e-1 + u * 2.1e-1 * E) / 1e2 + W * 179 * v) /
2312;
      select(p = 0, 0, 0, 0, & G);
      v -= (W * F - T * (6.3e-1 * m - I * 8.6e-2 + m * E * 19 - D * 25 - 1.1e-1 * u) / 1.07e4) * _;
      D = cos(o);
      E = sin(o);
    }
}
```


Optimization transform for DMS Rewrite Rule Language

```
default base domain Java;
```

Domain Name

```
rule merge-ifs(\condition1,
               \condition2,
               \then-statements)
    "if (\condition1)
        if (\condition2)
           { \then-statements
           }
    "
    rewrites to
    "if (\condition1 && \condition2)
        { \then-statements }   ";
```

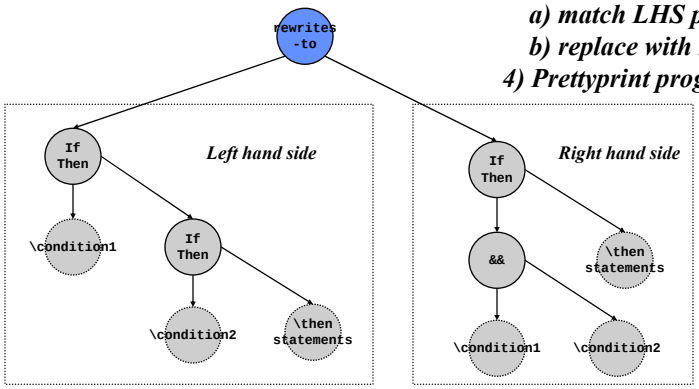
Domain Syntax

We'll see this idea again later.

DMS transforms work on ASTs, not text

Not fooled by any lexical properties of text!

To modify programs:
1) define transforms
2) Parse program
3) Apply transforms
a) match LHS pattern
b) replace with RHS substitution
4) Prettyprint program



Overview

- ***DMS***® Software Reengineering Toolkit
  - Defining notations ("domains") for specs and legacy systems
  - Parsing and prettyprint
  - Transformation mechanics
- Applications for Software Quality Improvement
  - C++ preprocessor conditional removal
  - Automatic Code Generation (XML Parsers)
  - Clone Detection/Removal
  - Porting application software to new languages
- Purpose: *Educate audience about new generation of tools*

Useless Conditional Elimination

- Problem: Too many configuration #IFs
  - Application on many platforms: WNT, SUN, VAX, ...
  - #IFs still in code
  - Too many to remove by hand, confusing to manage
  - Does delivered system work with all combinations?
- Solution: Use DMS to remove designated #IF
  - Engineer names dead configuration variables (VAX)
  - DMS use transforms to remove #IFs

C++ simplifying transforms

```
rule simplify_and_false(e:expression): expression->expression
   = "\e & 0" -> "0".

rule simplify_and_true(e:expression): expression -> expression
   = "\e & 1" -> "\e".

rule simplify_or_true(e:expression): expression->expression
   = "\e ! 1" -> "1".

rule simplify_or_false(e:expression): expression -> expression
   = "\e ! 0" -> "\e".

rule pp_if_true_simplify(b:block):statement->statement
    = "#if 1
       \b
       #endif" -> "{\b}".

rule pp_if_false_simplify(b:block):statement->statement
    = "#if 0
       \b
       #endif" -> ";".


rule pp_if_then_else_false_simplify(b1:block,b2:block):statement->statement
    = "#if 0
       \b1
       #else
       \b2
       #endif" -> "{\b2}".
```

C++ sample code… simplified

Add Rule for dead configuration variables

```
rule VAX: expression -> expression = "VAX" -> "0".
```

Before

```
#IF VAX!UNIX
    syslog(logfile->file_descriptor,"display output");
#ENDIF
   ...
#IF VAX
    sysclose(logfile->file_descriptor);
#ELSE
    fclose(logfile->file_descriptor);
#ENDIF
```

After

```
#IF UNIX
    syslog(logfile->file_descriptor,"display output");
#ENDIF
   ...
    fclose(logfile->file_descriptor);
```


XML Parser Generation

- XML enables Electronic Data Interchange
  - Neutral form for moving information between systems
- Problem: Need System1 to XML to System2 translators
  - For arbitrary data
  - Must be fast to support high-volume EDI
- Typical solution: Use Standard XML -> DOM reader
  - Incomplete: Doesn't solve "to XML" problem
  - Slow: Parse arbitrary XML, validate against DTD schema
  - Slow: Procedural interface interpreting DTD for data access
  - Clumsy: DOM calls clutters application code
- Idea: Generate DTD-specific XML parser/generators
  - Produce application-target-language code specific to DTD
  - Free validation; XML data in direct-access native data structures

  ⇒ Faster parsing/processing, Easier application coding

Simple XML DTD for OrderForm

```
<?xml version='1.0' ?>
<!DOCTYPE orderform [

<!ELEMENT orderform (name,company,address,items) >
<!ELEMENT name ( firstname, lastname )>
<!ELEMENT firstname ( #PCDATA )>
<!ELEMENT lastname ( #PCDATA )>
<!ELEMENT company ( #PCDATA )>
<!ELEMENT address ( street, city, country )>
<!ELEMENT street ( #PCDATA )>
<!ELEMENT city( #PCDATA )>
<!ELEMENT country ( zipcode | nation )>
<!ELEMENT zipcode ( #PCDATA )>
<!ELEMENT nation ( #PCDATA )>
<!ELEMENT items (item)+ >
<!ELEMENT item ( partnumber, quantity, unitprice)>
<!ELEMENT partnumber ( #PCDATA )>
<!ELEMENT quantity ( #PCDATA )>
<!ELEMENT unitprice  ( #PCDATA )>
]>
```

`items` ***is "list of*** `item`***"***

Sample orderform

```
<orderform>
  <name>Wiley Coyote</name>
  <company>Dinner, Inc.</company>
  <address><street>1 Mesa Highway</street>
           <city>Southwest Park</city>
           <country><zipcode>98765</zipcode></country>
  </address>
  <items>
     <item><partnumber>RocketSkates</partnumber>
           <quantity>2</quantity><price>29.95</price></item>
     <item><partnumber>BirdSeed</partnumber>
           <quantity>2000</quantity><price>.01</price></item>
  <items>
</orderform>
```


Java Code Generation Plan

- For each DTD element:
  - Produce an "element" class to hold data for that element
    - PCDATA for leaves
    - Class references for non-leaves
  - Produce an element-specific parsing procedure
  - Produce an element-specific unparsing produre
- Handle sequences as *array* of element class references
- Handle choices as class reference + "which alternative" integer (1..n)

```
<!ELEMENT name ( field )+ >
```

⇨

```
name Parse()
    {  while …  sequence[i]=field.Parse(); }
class XML_Element_name implements Union
  { public Union sequence[] = new Union[]; }
```

Sample Code Generation Transforms

```
pattern CLASS_SEQUENCE(name:NAME,sequence:cp_sequence):class_body_declarations@Java =
   "\JavaClassName\(\name\) Parse() throws XML.NotValidForDTD
     { if (!XML.QueryOpeningTag(\JavaXMLOpenTagString\(\name\))) return null;
       \JavaClassName\(\name\) result=new \JavaClassName\(\name\)();
       { \SEQUENCE_FETCH\(\sequence\,\SEQUENCE_LENGTH\(\sequence\)\,sequence1\) }
       XML.RequireClosingTag(\JavaXMLCloseTagString\(\name\));
     return result; }
```

Generates Parser method

```
     class \JavaClassName\(\name\) extends XML_IO implements Union
        { \XML_PARSER_DECLARATIONS_FOR_CP_SEQUENCE\(\sequence\,1\,\>XML\:NAME
                    \%contentspec sequence1 \<\:NAME\) // produces nested subclasses
          public Union sequence[]= new Union[\SEQUENCE_LENGTH\(\sequence\)];
          public Generate()
            { XML.WriteOpeningTag(\JavaXMLOpenTagString\(\name\));
              for (i=1;i<=\SEQUENCE_LENGTH\(\sequence\);i++)
                  sequence[i].Generate();
              XML.WriteClosingTag(\JavaXMLOpenTagString\(\name\));
            }}
    ".

rule refine_ELEMENT_sequence_to_class(name:NAME,sequence:cp_sequence)
                :elementdecl -> class_body_declarations =
  "\%markupdecl <!ELEMENT \name ( \sequence ) >" -> CLASS_SEQUENCE(name,sequence).
```

Generates class

About 100 rules ~~ 2000 lines for all transforms

Generated Java code for `items`

```
class XML_Element_items extends XML_IO implements Union
  {  public Union sequence[] = new Union[1]; } }

XML_Element_items Parse()
   throws XML.NotValidForDTD {
   if ( !XML.QueryOpeningTag("items"))
     return null;
   XML_Element_items result = new XML_Element_items();
   {   if ((sequence[1] = XML_Element_item.Parse()) == null)
       throw XML.NotValidForDTD;  }
   XML.RequireClosingTag("items");
   return result;  }
```

```
class XML_Element_item extends XML_IO implements Union
{ public Union sequence[] = new Union[3]; }

XML_Element_item Parse()
  throws XML.NotValidForDTD {
  if ( !XML.QueryOpeningTag("item"))
    return null;
  XML_Element_item result = new XML_Element_item();
  {  if ((sequence[1] = XML_Element_partnumber.Parse()) == null)
      throw XML.NotValidForDTD;
    if ((sequence[2] = XML_Element_quantity.Parse()) == null)
      throw XML.NotValidForDTD;
    if ((sequence[3] = XML_Element_unitprice.Parse()) == null)
      throw XML.NotValidForDTD;
  }
  XML.RequireClosingTag("item");
  return result;
}
```

Generated Java code for item content: partnumber, quantity, unitprice

```
class XML_Element_partnumber extends XML_IO implements Union
  { public String PCDATA; }  }

XML_Element_partnumber Parse()
    throws XML.NotValidForDTD {
    if (XML.QueryOpeningTag("partnumber"))
      return null;
    XML_Element_partnumber result = new XML_Element_partnumber();
    result.PCDATA = XML.AcceptNonEmptyPCDATA();
    XML.RequireClosingTag("partnumber");
    return result;  }
```

```
class XML_Element_quantity extends XML_IO implements Union
  { public String PCDATA; } }

XML_Element_quantity Parse()
    throws XML.NotValidForDTD {
    if (XML.QueryOpeningTag("quantity"))
      return null;
    XML_Element_quantity result = new XML_Element_quantity();
    result.PCDATA = XML.AcceptNonEmptyPCDATA();
    XML.RequireClosingTag("quantity");
    return result; }
```

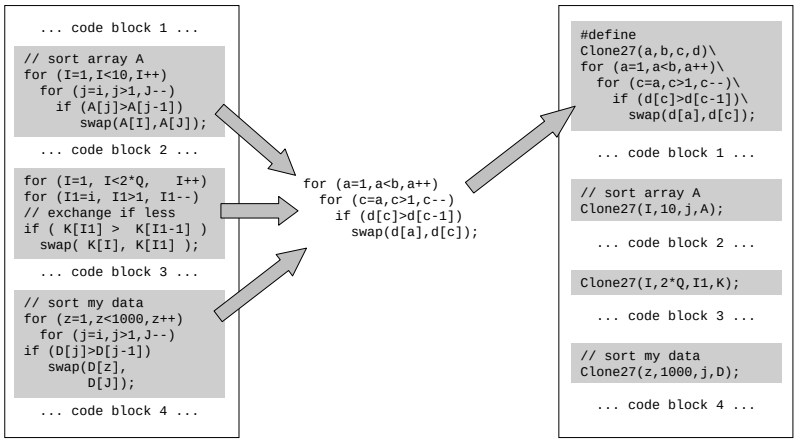
```
class XML_Element_unitprice extends XML_IO implements Union
  { public String PCDATA;  }

XML_Element_unitprice Parse()
    throws XML.NotValidForDTD {
    if (XML.QueryOpeningTag("unitprice"))
      return null;
    XML_Element_unitprice result = new XML_Element_unitprice();
    result.PCDATA = XML.AcceptNonEmptyPCDATA();
    XML.RequireClosingTag("unitprice");
    return result;  }
```


Application Code to Print Orderform

```
Orderform orderform =new Orderform.Parse();  // exception thrown if invalid w.r.t. DTD
{ Print("Customer: ");

  Print(orderform.name.firstname.PCDATA);Print(" ");

  Print(orderform.name.lastname.PCDATA); }

{ Print("Company: "); Print(orderform.company.PCDATA); PrintNewline(); }

{ Print("Address: "); Print(orderform.address.street.PCDATA); PrintNewLine();
                     Print(orderform.address.city.PCDATA); PrintNewline();
                     if (orderform.addreess.region.which=1)
                        Print(orderform.address.region.zipcode.PCDATA);
                     else  Print(orderform.address.region.country.PCDATA);
                     PrintNewline(); }

Print ("ITEMS"    Product     Quantity       Cost     Extension");
float Total=0;
for (item=1;item<length(orderform.items.item);1)
   {  PrintNumber(item); PrintTab();
      Print(order.items.sequence[item].partnumber.PCDATA); PrintTab();
      Print(order.items.sequence[item].quantity.PCDATA); PrintTab();
      Print(order.items.sequence[item].price.PCDATA); PrintTab();
      float extension=Value(order.items[item].quantity.PCDATA)
                     *Value(order.items[item].price.PCDATA);
      PrintNumber(extension);PrintNewline();
      Total+=extension;
   }

Print "Invoice total";PrintNumber(Total);PrintNewline();
```

Note direct access to XML data

DMS Scale Application: Clone Removal



Original System with code clones
1 M SLOC

Skeleton of detected clones

DeCloned System with automatic names
10-20% reduction

COBOL Clone Detection/Removal

- Find Clones by matching every program fragment (AST) to *every* other
  - *Expensive!*
- California Community Colleges application
  - Course Inventory Construction for each campus
- Parameters:
  - 77,000 SLOC ANSI COBOL 85 --> 774,645 AST nodes
    - 128 Second Parse time, 40 minutes for clone detection
    - 500 Mb RAM, 6 CPUs
  - Number of exact clone pairs = 78, near miss pairs = 95
  - *Largest clone: 5 copies, 1017 lines each, *1* parameter!*
  - Number of cloned lines = 30727 --> can remove 15363
- *SLOC reduction by removing clones = 19.7%*

Sample COBOL Clones

```
Similarity = .99178082191781
from 35179 to 35204 file = example.cbl

  9700-OUTPUT-REPORT-TOTALS.
      MOVE CURRENT-COLLEGE-ID
          TO REPORT-CODE1  REPORT-CODE2  REPORT-CODE3.
      SET EDIT-ERROR-LITERAL-INDEX TO 1.
      SET DISTRICT-COUNT-ROW-INDEX TO 1.
      PERFORM 9710-OUTPUT-TOTALS1
          UNTIL EDIT-ERROR-LITERAL-INDEX > 30.
      PERFORM 9720-OUTPUT-TOTALS2.
      PERFORM 9730-OUTPUT-TOTALS3.


  9710-OUTPUT-TOTALS1.
      MOVE REPORT-SUM1 TO TOTALS-ID1.
      MOVE ELEMENT-NUMBER (EDIT-ERROR-LITERAL-INDEX)
              TO DED-NUMBER.
      MOVE DISTRICT-COUNT (DISTRICT-COUNT-ROW-INDEX  1)
              TO EXCEPT-COUNT.
      MOVE DISTRICT-COUNT (DISTRICT-COUNT-ROW-INDEX  2)
              TO UNKNOWN-COUNT.
      MOVE DISTRICT-COUNT (DISTRICT-COUNT-ROW-INDEX  3)
              TO REASON-COUNT.
      MOVE DISTRICT-COUNT (DISTRICT-COUNT-ROW-INDEX  4)
              TO GRP3-COUNT.
      WRITE REPORT-TOTALS-RECORD1.
      SET EDIT-ERROR-LITERAL-INDEX UP BY 1.
      SET DISTRICT-COUNT-ROW-INDEX UP BY 1.
```

#1

#2

Report by College-ID

```
#2  ==>  30
    ==>  17
---------
#1
==>     CURRENT-COLLEGE-ID
==>     HOLD-DISTRICT-ID
```

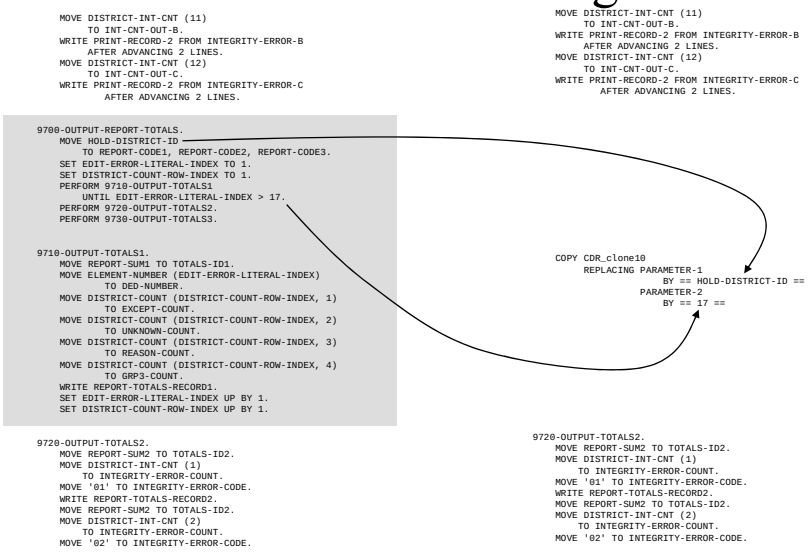
```
-----------------------------------
from 15368 to 15393 file = example.cbl

  9700-OUTPUT-REPORT-TOTALS.
      MOVE HOLD-DISTRICT-ID
          TO REPORT-CODE1  REPORT-CODE2  REPORT-CODE3.
      SET EDIT-ERROR-LITERAL-INDEX TO 1.
      SET DISTRICT-COUNT-ROW-INDEX TO 1.
      PERFORM 9710-OUTPUT-TOTALS1
          UNTIL EDIT-ERROR-LITERAL-INDEX > 17.
      PERFORM 9720-OUTPUT-TOTALS2.
      PERFORM 9730-OUTPUT-TOTALS3.


  9710-OUTPUT-TOTALS1.
      MOVE REPORT-SUM1 TO TOTALS-ID1.
      MOVE ELEMENT-NUMBER (EDIT-ERROR-LITERAL-INDEX)
              TO DED-NUMBER.
      MOVE DISTRICT-COUNT (DISTRICT-COUNT-ROW-INDEX  1)
              TO EXCEPT-COUNT.
      MOVE DISTRICT-COUNT (DISTRICT-COUNT-ROW-INDEX  2)
              TO UNKNOWN-COUNT.
      MOVE DISTRICT-COUNT (DISTRICT-COUNT-ROW-INDEX  3)
              TO REASON-COUNT.
      MOVE DISTRICT-COUNT (DISTRICT-COUNT-ROW-INDEX  4)
              TO GRP3-COUNT.
      WRITE REPORT-TOTALS-RECORD1.
      SET EDIT-ERROR-LITERAL-INDEX UP BY 1.
      SET DISTRICT-COUNT-ROW-INDEX UP BY 1.
```

Report by District-ID

The generated COPYLIB

```
9700-OUTPUT-REPORT-TOTALS .
    MOVE PARAMETER-1
        TO REPORT-CODE1  REPORT-CODE2  REPORT-CODE3 .
    SET EDIT-ERROR-LITERAL-INDEX TO 1 .
    SET DISTRICT-COUNT-ROW-INDEX TO 1 .
    PERFORM 9710-OUTPUT-TOTALS1
        UNTIL EDIT-ERROR-LITERAL-INDEX > PARAMETER-2    .
    PERFORM 9720-OUTPUT-TOTALS2 .
    PERFORM 9730-OUTPUT-TOTALS3 .
9710-OUTPUT-TOTALS1 .
    MOVE REPORT-SUM1 TO TOTALS-ID1 .
    MOVE ELEMENT-NUMBER ( EDIT-ERROR-LITERAL-INDEX )
            TO DED-NUMBER .
    MOVE DISTRICT-COUNT ( DISTRICT-COUNT-ROW-INDEX 1 )
            TO EXCEPT-COUNT .
    MOVE DISTRICT-COUNT ( DISTRICT-COUNT-ROW-INDEX  2 )
            TO UNKNOWN-COUNT .
    MOVE DISTRICT-COUNT ( DISTRICT-COUNT-ROW-INDEX 3 )
            TO REASON-COUNT .
    MOVE DISTRICT-COUNT ( DISTRICT-COUNT-ROW-INDEX 4 )
            TO GRP3-COUNT .
    WRITE REPORT-TOTALS-RECORD1 .
    SET EDIT-ERROR-LITERAL-INDEX UP BY 1 .
    SET DISTRICT-COUNT-ROW-INDEX UP BY 1 .
```

Source file change



Clone Detection/Removal Statistics

Application	*Language*	*SLOC*	*Removable SLOC (est.)*	*Removed SLOC*
Process Control	C	427921	54346 12.7%	N.A. (pre removal capability for C 6/98)
PARLANSE Compiler	C	42413	4133 9.7%	2568 6.0%
CCC Registration	COBOL85	77631	15500(39967) 20.0 (51.5)%	15034 19.9%
Roccade Accounting	IBM COBOL	567798	130494 23%	N.A. (not yet performed)
SLC Insurance	PROGRESS	356413	58783 16.5%	N.A. (not yet performed)
SWING Toolkit	Java	241460	26355 10.9%	N.A. (not yet performed)

Typical Porting Scenarios

- JOVIAL73 on MIL1750 → C on PowerPC
  - Military Avionics + Weapons management
- COBOL74 + IDMS → COBOL85 + SQL
  - UNISYS 1100 retirement; must move data, too!
- K&R C + custom RTOS → ANSI C + VXworks
  - Microprocessor modernization
- Clipper + green screen → Delphi + GUI
  - Legacy 3GL data processing language
- MODCOMP ASM → C
  - Defense Radar modernization; 12 computer languages!
- Verilog → VHDL
  - Reuse of Chip Design in new context

How DMS handles Porting

- Accepts definitions of source, target and *design* languages
  - Syntax, Semantics, Optimization Transforms and Analysis rules
- Accepts specifications of (porting) transformations
  - Written in terms of the language syntax, conditioned by analyses
  - Source-language idioms often map directly to Target-language idioms
  - Transforms for Complex idioms/OS/Library calls abstracted to design languages, then refined to target languages
- Parses entire source system (thousands of files!)
- Apply Porting transformations, then Optimizing transforms
- PrettyPrints the results in compilable target language form
- Test Result using Application Regression Test
- Revise transforms and repeat till done

A few DMS porting transforms

Jovial to C

```
default source domain Jovial;
default target domain C;

private rule refine_data_reference_dereference_NAME
                (n1:identifier@C,n2:identifier@C)
                   :data_reference->expression
  = "\n1\:NAME @ \n2\:NAME" -> "\n2->\n1".

private rule refine_for_loop_letter_2
               (lc:identifier@C,f1:expression@C,
                f2:expression@C,s:statement@C)
                     :statement->statement
  = "FOR \lc\:loop_control :
       \f1\:formula BY \f2\:formula; \s\:statement"
    ->
       "{ int \lc = (\f1);
         for(;;\lc += (\f2)) { \s } }"
        if is_letter_identifier(lc).
```

Domain Name

Pattern Variables

Source Domain Syntax

Target Domain Syntax

Porting Transforms in Action

Jovial to C

JOVIAL Source:

```
FOR i: j*3 BY 2 ;
    x@mydata = x@mydata+I;
```

Translated C Result:

```
{ int i = j*3;
  for (;;i+=2)
     { mydata->x = mydata->x + i}
```

Typically lots of small transforms for full translation
~1500 rules to translate full Jovial

A More Complex Example

Jovial to C

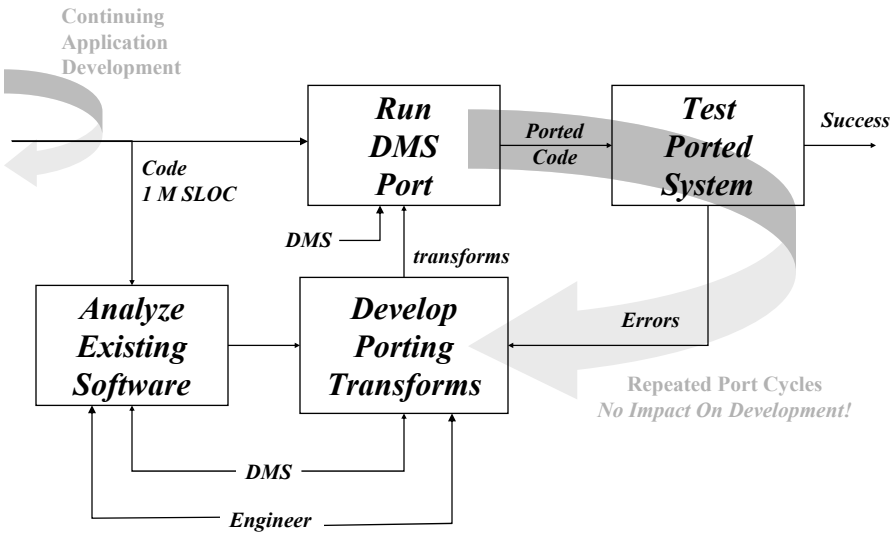
```
START
  TABLE TFP'D'TWRDET (1:109,12:37);
    BEGIN
      % Main status boolean %
      ITEM TFP'G'TWRDET STATUS (V(YES),V(NO));
    END
    TYPE TFP'D'TWRDET'TABLE TABLE (7:23) W 3;
      BEGIN
        ITEM TFP'ITM S 3 POS(0,3); "cube axis"
      END

  %begin proc%
  PROC PROC'A(c1) S;
    BEGIN
      ITEM match'count U 6;
        %an item%
      ITEM c1 C 5; "parameter value"
      ITEM c2 C 7;
      IF c1 <= c2 AND c2 > c1;
         match'count = UBOUND(TFP'D'TWRDET,0) +
                       UBOUND(TFP'D'TWRDET'TABLE,0);
      "result off by 1 so adjust"
      match'count = match'count+1;
      BEGIN
        match'count=match'count/2;
        PROC'A = match'count; % return answer %
      END "cleanup and exit";
    END "end proc"

TERM
```

packed tables with bit offsets, typedefs, functions, string operations, comments

```
#include "jovial.h"
static struct
  { /* Main status boolean */
    enum { V(yes$OF$tfp_g_twrdet$OF$tfp_d_twrdet),
           V(no$OF$tfp_g_twrdet$OF$tfp_d_twrdet) }
                        tfp_g_twrdet _size_as_word;
          } tfp_d_twrdet[109][26];
typedef union
  { W(3);
    struct
      { POS(0, 3) S(3) tfp_itm:4 _align_to_bit; /* cube axis */
      };
   } tfp_d_twrdet_table[17];

static S proc_a(C(5) c1);
/* begin proc */
static S proc_a(C(5) c1)
{ __typeof__(proc_a(c1)) RESULT(proc_a);
  _main:
    { U(6) match_count;
      C(7) c2;
      if (CHARACTER_COMPARE(BYTE_CONVERT(C(7), c1, 7), c2) <= 0
          && CHARACTER_COMPARE(c2, BYTE_CONVERT(C(7), c1, 7)) > 0)
        match_count = UBOUND(tfp_d_twrdet, 2, 0) + 16;
      /* result off by 1 so adjust */
      match_count = (S(6))match_count + 1;
      { match_count = (S(6))match_count / 2;
        RESULT(proc_a) = (S(6))match_count; /* return answer */
      } /* cleanup and exit */
      ;
    }
  _return:
    return RESULT(proc_a);
} /* end proc */
```

Equivalent C (used with hand-coded macro library)

Porting Process



Porting by DMS is practical

- Enabled by generalized compiler technology
  - Requires:
    - Specification of source, target and design languages
    - Specification of *inspectable* transforms
  - Automates:
    - Source file parsing and prettyprinting of results
    - Application of sets of transforms
  - Scalable, fast, repeatable
    - Transform thousands of files/millions of lines in one day
    - Iterative development/testing of porting transforms/result
- Organizational benefits
  - Development team not disturbed by porting team
  - Application functionality preserved → happy users!
  - Far more cost effective than hand translation

DMS: Conclusion

- Useful to automate analysis/modification of programs
  - Many possible custom reengineering possibilities
  - A key technology for software quality improvement
- Need generalized compiler-like infrastructure
  - Definable parsers, prettyprinters, transforms
  - Must *scale* to application systems with MSLOCs
- DMS provides these capabilities
  - on path to design capture/analysis/modification
  - growing infrastructure and standard language modules

www.semdesigns.com

Where can *DMS* be applied?

- *Program modification*
  - *Application evolution*
    - Functionality change
    - Performance change
    - Technology change
  - Massive changes
    - Porting: new language, target APIs, ...
    - Restructuring: *Clone removal*, Y2K fix, ...
    - Optimization: Dead code, parallelize, ...
  - Customize reusable component in new context
- Program Analysis
  - Metrics
    - SLOC, conditional, complexity
  - Organization style checking
  - Programming information extraction
    - Clones, Slices, Call Graphs, Side effects
  - Domain information extraction
    - Business rules
    - Idiom recognition
  - Semantic Faults
    - erroneous/dead/useless code
- Domain-specific *program generation*
  - Partial Differential Equation solvers
  - Factory control synthesis
  - Entity-Relationship compilers
    - DB conversion generators
  - Protocol compilers
  - Automated Test Generation
- Legacy code reverse engineering
  - design recovery to domain abstractions
    - aid code understanding
    - enable application evolution
  - *Incremental* design capture
  - reusable component extraction
  - component extraction for domains
  - Legacy mergers
    - Unify data schemas
    - Modify applications
    - Convert existing data
  - Business rule extraction
    - Make explicit, easy to read/change

QW2002 Paper 6V1

Mr. Hans Buwalda
(Logigear Corporation)

Test Automation, Build to Last Action Based Testing Methods and Tools at Work!

Presentation Abstract

Action Based Testing (ABT) is becoming a more popular method to architect testing and test automation. The method has been proven for a wide range of applications including GUI, embedded software, Web, and client-server systems; and industries such as bioscience, logistic, transportation, telecommunication, finance and consumer product. In this session, Hans Buwalda offers a summary of ABT practices through the demonstration of the ABT Toolset™.

This test-development oriented toolset will elegantly puts you in control of the development, management and automation of even the most complex test sets. Take this opportunity to see how the method and supporting tools together will deliver the hidden weapon that has been missing from your testing arsenal.

About the Author

Hans Buwalda, ABT Chief Architect, leads LogiGear's Action Based Testing™ (ABT) research and development, including ABT Toolset™ operations, and oversees the practice of the ABT methodology. Prior to joining LogiGear, Hans served as Project Director at CMG in The Netherlands where he was the original architect behind what has now become TestFrame™ - an integrated method for planning, managing, and deploying software testing and test automation. Hans is an internationally recognized expert specializing in test automation, test development, and test management. He speaks and presents workshops at international conferences on testing concepts such as Action Based Testing, The Three Holy Grails of Test Development, Soap Opera Testing, and Testing in the Cold. Hans authored (along with Dennis Janssen and Iris Pinkster) Integrated Test Design and Automation: Using the TestFrame Method (Addison Wesley 2001). He holds a Master of Science degree in Computer Science from Free University, Amsterdam.

QW2002 Paper W1

Eric Siegel
(Keynote Systems)

Introduction to Performance on the Internet and Web

Key Points

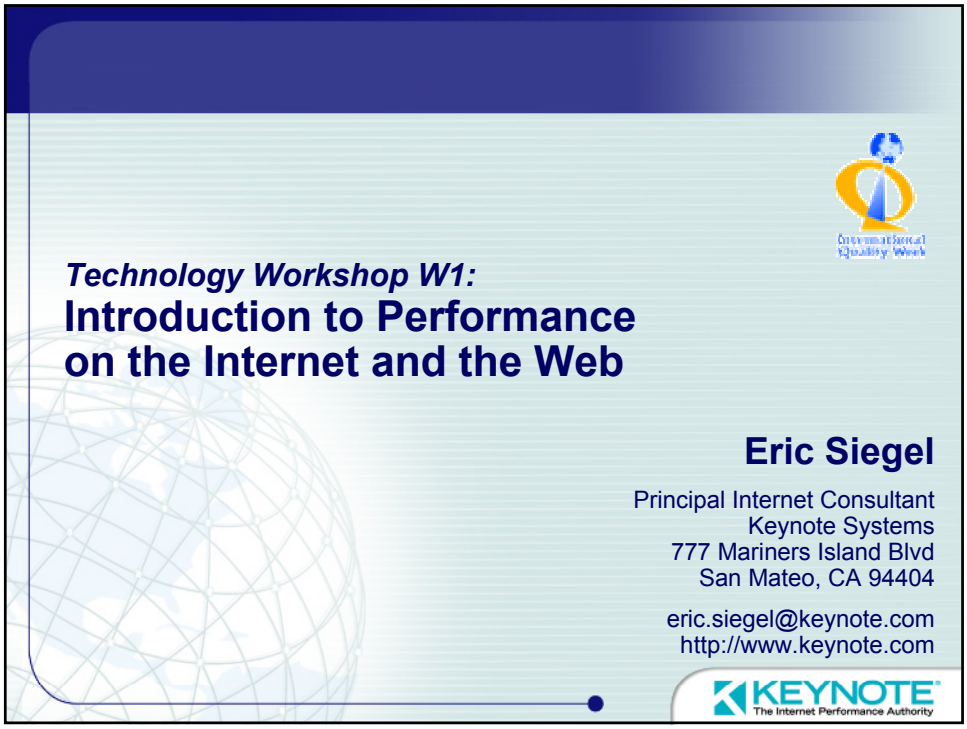
- The main factors contributing to performance and availability on the Web and Internet
- Covers relevant Web and Internet protocols (TCP, IP, HTTP, SSL, DNS)
- Covers peering, routing ; server-room connectivity; caching; CDNs

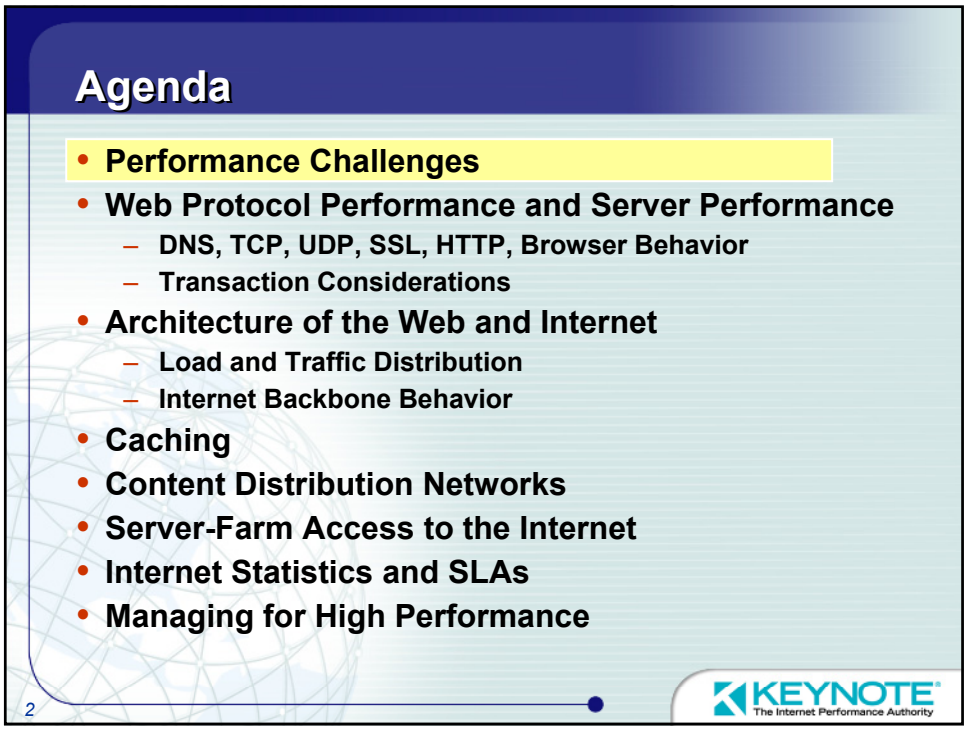
Presentation Abstract

This tutorial provides a thorough grounding in Web and Internet technologies that affect performance and availability. It explains the fundamentals of Internet performance engineering, which is not just a matter of bandwidth! We'll cover both the protocols and the Internet and Web architectures, but only those pieces that are relevant to performance. It concentrates on the differences between the protocols used on legacy networks (e.g., SNA) and those of the Internet (TCP/IP, DNS, HTTP), along with the differences between a private network based on owned links (leased lines, frame relay, etc.) and a network based on the Internet (Internet routing protocols, connection points between networks owned by different organizations, internal caching, load distribution technologies, etc.) If you're new to the problem of assuring quality over the Internet, come to this tutorial to gain a solid foundation in the relevant technologies.

About the Author

Eric Siegel, Principal Internet Consultant at Keynote Systems, has been a member of the Internet community since 1978. He is the author of "Designing Quality of Service Solutions for the Enterprise" (John Wiley & Sons) and is an instructor and panelist in Internet performance and QoS at major industry conferences such as Networld+Interop, CA World, Service Networks, WWW Conferences, Quality Week, and CMG. Before joining Keynote, Mr. Siegel was a Senior Network Analyst at NetReference, Inc., where he specialized in network architectural design for Fortune 100 companies, and he was a Senior Network Architect with Tandem Computers, where he was the technical leader and coordinator for all of Tandem's data communications specialists worldwide. Mr. Siegel also worked for Network Strategies, Inc. and for the MITRE Corporation, where he specialized in computer network design and performance evaluation. Mr. Siegel received his B.S. and M.E.E. degrees in Electrical Engineering from Cornell University, where he was elected to the Electrical Engineering honor society.





Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Performance Is Important!

- **"Twenty-eight percent of shoppers who have suffered failed performance attempts said they stopped shopping at the Web site where they had problems, and six percent said they stopped buying at that particular company's off-line store."** (Boston Consulting Group, quoted in Infoworld / Computerworld 3/00)
- **"[One site] found the bailout rate fell immediately from 30% to 6-8% just because of one tiny second of load time!"** (Zona Research 4/99)
- **"Surfers say that slow-loading Web sites are the biggest cause of irritation ... Seven percent of respondents say they hit their equipment ... 2% say they've become so upset they've hit the person who sits next to them."** (Market & Opinion Research International 2/02)
- **"Perhaps as much as $4.35 billion in e-commerce sales in the U.S. may be lost each year due to unacceptable download speeds and resulting user bailout behaviors."** (Zona Research 4/99)

KEYNOTE The Internet Performance Authority

What Is "Good" Performance?

- **Commonly-cited "Eight-Second Rule"**
- **But a better measure is competitors' performance**
  - **What is your end-user's frame of reference?**
    - Competitors
    - Commonly-accessed consumer sites (Yahoo!, etc.)
    - (how will you measure these sites?)
  - **What does your end-user care about?**
    - Impulse buying from a "fat" home page
    - Quickly browsing up and down your catalog's tree structure
    - Placing orders quickly, without failures or delays
    - Does a "fat" page generate more revenue? You must watch competitors closely: "Why are they doing that?"
  - **Benchmarks: www.keynote.com/company/html/services.html**
- **The location of your end-user affects expectations.**
  - **At home, on a 28.8k modem**
  - **At work, on an uncongested T3?**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

A Definition of Performance

- **Web e-commerce performance measures the user's experience interacting with your Web site, *not your in-house experience or the experience inside your Web hosting center.***
  - **Download time**
    - The dissatisfaction and abandonment points are different for different types of pages and different classes of users
  - **Transaction Time**
    - banking, stock trading, purchasing
  - **Availability**
  - **Errors**
    - Failed connection attempts
    - Missing or incorrect pages
    - Missing page components
    - Broken links
    - Transaction failure

KEYNOTE The Internet Performance Authority

Performance Challenges

- **24x7 availability and geographic distribution; expectation of universal access**
- **A shared network resource**
- **No control over customers' environment**
- **Multiple servers and caches, which may be geographically distributed, participate in a single user interaction**
- **Poor support for session structures; difficult or impossible to detect transaction abandonment**
- **Potentially massive peak volumes ("flash" loads)**

The Web and Internet are very different from legacy client:server and terminal:mainframe environments!

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Agenda

- Performance Challenges
- Web Protocol Performance and Server Performance
  - DNS, TCP, UDP, SSL, HTTP, Browser Behavior
  - Transaction Considerations
- Architecture of the Web and Internet
  - Load and Traffic Distribution
  - Internet Backbone Behavior
- Caching
- Content Distribution Networks
- Server-Farm Access to the Internet
- Internet Statistics and SLAs
- Managing for High Performance

KEYNOTE The Internet Performance Authority

Web Page Technology Overview

- The Domain Name System (DNS), a worldwide hierarchy of directories, translates fang.dog.com into 10.9.23.22.
- TCP/IP carries the data between your browser and 10.9.23.22; it detects errors and corrects them by retransmitting.
- The data consists of HTTP, HTML, and the page's information.
- HTTP (Hypertext Transfer Protocol) carries the Hypertext Markup Language (HTML) and provides the basic Web page commands:
  - GET
  - Query String (e.g., GET fang.dog.com/filename?fur=matted)
  - POST
- HTML describes the page:
  - Formatting
  - Content, and the servers/files from which that content can be downloaded (e.g., pix.fang.dog.com/gifs/picture1.gif)
  - Links

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Domain Name System (DNS) Record

- A domain's owner creates the Authoritative Record, which provides:
  - host name → IP address translation
  - IP address → host name translation ("reverse lookup")
  - email server information, etc.
  - It can provide alternative or multiple translations
- The Authoritative Record is stored on a DNS Server controlled by the domain's owner

KEYNOTE The Internet Performance Authority

DNS Search

- The DNS Resolver in the end-user's system asks the end-user's local DNS server to translate the host name (fang.dog.com) into an IP address (10.9.22.23)
- If the end-user's DNS server doesn't have the requested information, it searches the DNS tree to find the authoritative record or a cached copy
  - It may find cached information for a less-specific domain (e.g., dog.com) that points to the authoritative server
  - It may have to go to the root (e.g., .com) to find a pointer to the authoritative server
  - DNS servers cache DNS information for a period of time (Time To Live, TTL) controlled by an entry in the authoritative record

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

DNS Issues

- DNS performance and resilience
  - DNS failures occur during 0.4% of Keynote's Web page retrieval attempts
  - Long TTL uses the DNS tree's caches, increasing performance and resilience but decreasing your control
  - Short TTL gives you greater control, but there will be a delay when cache expires and a new fetch of the authoritative record is necessary
  - Authoritative server (or backup) must always be available
- Some alternative DNS systems provide greater control and performance than the standard system
  - But they're not free

KEYNOTE The Internet Performance Authority

TCP Fundamentals

- TCP creates a virtual circuit
  - Received data, if it arrives, will arrive without errors, in the correct sequence.
  - Setup is to the receiving host's TCP stack only, not all the way into the receiving application.
  - TCP header's port number identifies an application
  - When first connected, TCP selects a packet size.
    - Typical packet size = 1500 bytes
- TCP typically sends an acknowledgement:
  - When it can piggyback on data going in the other direction
  - If a time delay (typically 200+ ms.) has elapsed
- Packet loss is *implied* by repeated receipt of the same acknowledgement or by timeouts
  - TCP stack performs a complex calculation of network round-trip time (RTT)

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

TCP Window

- The TCP Window equals the number of bytes that the receiver is currently willing to accept.
  - Windows/98 default receive window = 8192 bytes; Windows/NT = 8760 bytes
- Long Fat Networks (e.g., satellite) need large windows
  - Maximum window size can be increased by window scale option (RFC 1323)

KEYNOTE The Internet Performance Authority

TCP Flow Control

- Slow Start ramps up traffic slowly, instead of abruptly trying to fill the receiver's window.
  - Used at start of connection
  - Used after a long pause in transmission or a lost end packet
- Slow Start begins by transmitting *only one packet* and waiting for the ACK.
  - *When the ACK is received*, it transmits *two* packets.
  - When the ACKs for those two are received, up to *four* packets can be transmitted (assuming the window has capacity), etc. — an exponential increase.
  - The standards now allow Slow Start to begin with 2 packets.
- TCP can't send packets faster than ACKs arrive back at the sender
  - ACKs are delayed by slow links
  - "Self-Clocking Behavior"

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

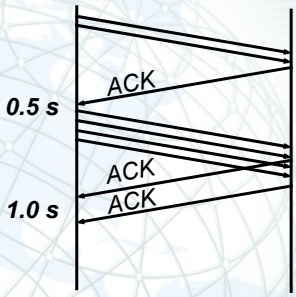
TCP Congestion and Packet Loss

- **TCP *infers* congestion from packet loss**
  - **If a packet in the middle of the received stream was lost:**
    - transmission rate is cut in half, then increases linearly, depending on Round Trip Time (RTT): large RTT, slow increase
  - **If the packet at the end of the received stream was lost:**
    - Transmission starts all over again with Slow Start
    - When it recovers to half of the original transmission rate, it uses the midstream-loss algorithm to increase the rate linearly
    - Timeouts cause much more degradation than midstream losses

KEYNOTE The Internet Performance Authority

TCP Performance Example

- **Bandwidth is not the same as latency**
  - **"Ping" isn't a good measure of download time!**
  - **Example: with 250 ms. one-way latency on a 1 GB pipe. . . *throughput is 16 kb/s with standard Windows/98 defaults!***
    - Windows/98 default receive window = 8192 bytes; /NT = 8760
  - ***And a 9 kb file takes one second to transmit!***
    - Slow Start, with frame size = 1500 bytes



Typical one-way, direct-link latencies:

3 ms. through a local digital telephone exchange
12 ms. through a local analog telephone exchange
20 ms. New York to San Francisco
100 ms. New York to Melbourne, Australia
260 ms. through a geosynchronous satellite

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

TCP Performance Notes

- Use large frame size, window sizes, SACK (RFC 2018)
  - Window size is limiting factor for broadband
  - The defaults are often small
- Set Slow Start to start with two packets, not one
  - TCP Slow Start is limiting factor for small objects
  - Startup of a TCP flow and recovery from errors depends on round-trip time.
- If you control both ends of the link, and you're using a Long Fat Network, use large windows and SACK.
- TCP works best if the error rate is low

KEYNOTE The Internet Performance Authority

User Datagram Protocol (UDP)

- UDP's very simple header carries port numbers
- Each UDP packet stands alone
  - No sequence numbering
  - No guarantee that the packet arrives at the destination
    - No error detection
    - No automatic retransmission to correct errors or lost packets
  - No flow control or windows to handle congestion
    - Network can't signal, even indirectly, to UDP flow's source
- Used by:
  - Audio and video real-time transmission
    - There's no time to retransmit bad packets anyway!
  - Broadcast and Multicast
    - Who would generate the acknowledgements?

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

UDP Performance

- UDP applications are responsible for managing their own performance
  - They rarely respond as quickly as TCP to congestion
  - Some don't respond to congestion at all
- Uncontrolled UDP applications may interfere with important TCP traffic
- Router configurations can use queues to throttle UDP flows; excess traffic should be discarded at the network entrance.

KEYNOTE The Internet Performance Authority

Secure Sockets Layer

- SSL (HTTPS:) is used for secure connections
  - It fits between HTTP (or FTP, etc) and TCP
- There are two major SSL phases:
  - Connection Handshake Phase
    - This phase can be time-consuming
    - Each session has a Session ID
    - The setup information is cached and can be reused
  - Record Transfer Phase
    - Encryption overhead is not large
    - Encrypted data is not compressible and is not cacheable

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

HTTP Behavior

- HTTP is a simple protocol, consisting of requests:
  - GET, GET with Query String
  - POST
- And responses:
  - 200 (OK)
  - 300 (Redirect)
  - 400 (Client Error)
  - 500 (Server Error)
- Redirection may be done by HTTP (e.g., 302), by "Refresh," by JavaScript, etc.

KEYNOTE The Internet Performance Authority

Browser Behavior

- The browser may use a separate TCP connection for each GET
  - Server closes connection after file is delivered
- Or it may use persistent connections:
  - All GETs in a persistent connection must go to the same host
  - Each file needs a length header or "chunking" to indicate file end
- The browser may support a number of connections in parallel, but they all share the same bandwidth
  - Parallel connections speeds up slow-starts
  - Each browser version/OS combination has a different maximum number of parallel connections, reconfigurable by user

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Cookies

- **Cookies are short `name=value` pairs stored on the browser by a server**
  - **The cookie is returned in the browser's HTTP Request in every subsequent request to that domain**
    - This may be restricted to requests for specific directories
  - **Cookie expiration time is set by the server**

Cookie set by server:

```
HTTP/1.1 200 OK
...
Set-Cookie: ASPSESSIONIDGQQGGKWQ=FGMHJFMDK; path=/fang
(file)
```

Cookie returned by browser:

```
POST /fang/dog.htm HTTP/1.1
Cookie: ASPSESSIONIDGQQGGKWQ=FGMHJFMDK
(form data)
```


KEYNOTE The Internet Performance Authority

HTTP Example

Request (GET with Query String):

```
GET /dog/fangface/bonnie.htm?fur=matted HTTP/1.1
Host: web202.furbearers.com
Accept: */*
User-Agent: Mozilla/4.0 (compatible;MSIE 4.01;WindowsNT; )
```

Reply (including content length, cookie, and cache control):

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Sat, 06 May 2000 18:06:02 GMT
Content-Length: 483
Content-Type: text/html
Set-Cookie: ASPSESSIONIDGQQGGKWQ=FGMHJFMDKEDJ; path=/
Cache-control: private

(file)
```


KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Dynamic Page Generation

- Page is built "on the fly" instead of being a pre-built, unchanging, fixed-length file
  - Usually can't be cached
  - Because length changes, needs special handling to insert length headers for persistent connections
- Typical dynamic page generators:
  - Server-Side Include (SSI, .shtml)
  - Common Gateway Interface (cgi-bin)
  - Java Server Pages, Active Server Pages (.jsp, .asp)
  - Others (Cold Fusion, etc.)

KEYNOTE The Internet Performance Authority

Web Transactions – 1

- ***Scaling transactions is much more difficult than scaling simple Web page delivery!***
  - This has massive implications for load balancing and geographic distribution schemes
- Need to maintain transaction context between Web pages, associating a user with a transaction
  - Use remote IP address?
    - Different users of one ISP can have same address
    - Users can switch IP addresses in mid-transaction
  - Use a cookie?
    - Set cookie when user first appears, or after login
    - Cookie remembers transaction ID, etc.
    - User can set browser to refuse cookies
    - Load balancing devices can't handle encrypted cookies unless they can decrypt SSL

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Web Transactions – 2

- **Embed user, state information within each page and link?**
  - Requires dynamic page generation
  - Page is generated with each link modified to contain user and state information:

```
<A HREF="dog.htm?userID=123abc">
```

    or with custom-built POST form data:

```
<FORM NAME=FANG METHOD=POST ACTION=dog.htm>
<INPUT TYPE=HIDDEN NAME="userID" VALUE="123abc">
...
</FORM>
```

  - Load balancing devices can't read a URL or page data unless they can decrypt SSL
- **Need to recover resources from abandoned transactions**
  - **Web transactions usually do not tell server when browser has abandoned transaction**
  - **Timeout is a reasonable technique**

KEYNOTE The Internet Performance Authority

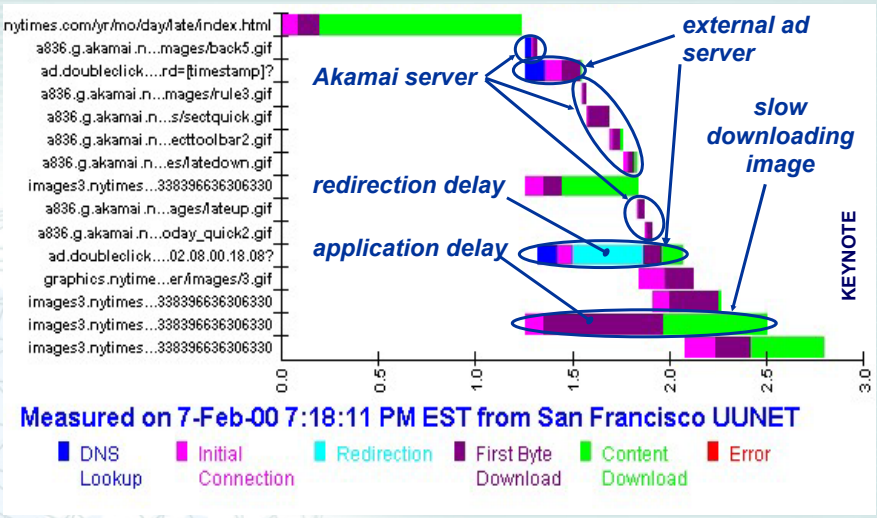
Summary of Web Performance Factors

- **The Web page seen by the browser is often generated from a number of different sources:**
  - **Ad servers**
  - **Geographically-distributed content servers**
  - **Caches**
- **Download performance is affected by:**
  - **Protocol behavior**
  - **Number of concurrent download connections; persistence**
  - **Geographic location of the browser**
  - **Congestion and latency between servers and browser**
  - **Performance of load-distribution and load-distribution schemes**
  - **Performance of the servers and their back-end databases**
- **For example...**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Web Page Download Time Components



This page includes "Akamaized" content distribution and DoubleClick ad servers

KEYNOTE The Internet Performance Authority

Web Protocol and Server Performance

- **Now that we've finished our discussion of the Web's technical underpinnings . . .**
- **It's time for a review and list of "best practices" that combines:**
  - **Our performance notes from those discussions**
  - **Our experiences from dealing with performance issues at thousands of major Web sites . . .**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Web Protocol and Server Performance – 1

- Decrease the size of page elements
  - Shrink the color palette and the number of pixels/inch
  - Use www.gifwizard.com, Pagemaker, Macromedia Fireworks...
- Minimize the number of elements on each Web page
  - Have a reason that you're using each element
  - Combine small elements into larger ones
  - Reuse elements within and among pages when possible, and give them identical names.
  - Allow an element to be obtained and cached in the browser before you use it again.
- Minimize the total size of the page
  - Regardless of the number of concurrent download threads, or the degree of pipelining, all data shares the same bandwidth
  - Use style sheets
  - For slow-speed links, compressing the HTML may be useful

KEYNOTE The Internet Performance Authority

Web Protocol and Server Performance – 2

- Design the page to allow rendering to start quickly
  - Minimize use of long or deeply-nested tables and Frames
  - Avoid items that are difficult or slow to render
  - Specify HEIGHT and WIDTH of embedded images
  - Flush partially-built buffers
  - Put major information and links at the top of the page
- Avoid changing hostnames in the page
  - Avoid using different hostnames for the same server
- Avoid redirection
  - Redirection takes time
    - HTTP (30x) redirection is the fastest
    - HTML redirection (HTTP-EQUIV="refresh"...) or other script or applet redirection requires that the page be parsed first
  - Avoid accidental redirection
    - If you use a directory name as a URL, you should end it with " / "

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Web Protocol and Server Performance – 3

- Enable Persistent Connections (in HTTP 1.1)
  - Allows reuse of a TCP connection to GET more than one file from a host, but it may tie up server ports
  - Use "Content-Length:" headers, chunked encoding, etc.
  - Look at a trace to be sure that persistence is really working!
- Applets are slow and may need to download libraries
- Avoid animated .gifs
- Secure pages are slow
  - SSL overhead
    - Try to have all TCP threads directed to the same server or server process, so that they can share a Session ID
    - Consider using a hardware-accelerated SSL device
  - Encrypted data is usually uncompressible and uncacheable
    - A string of spaces will be uncompressible after encryption

KEYNOTE The Internet Performance Authority

Agenda

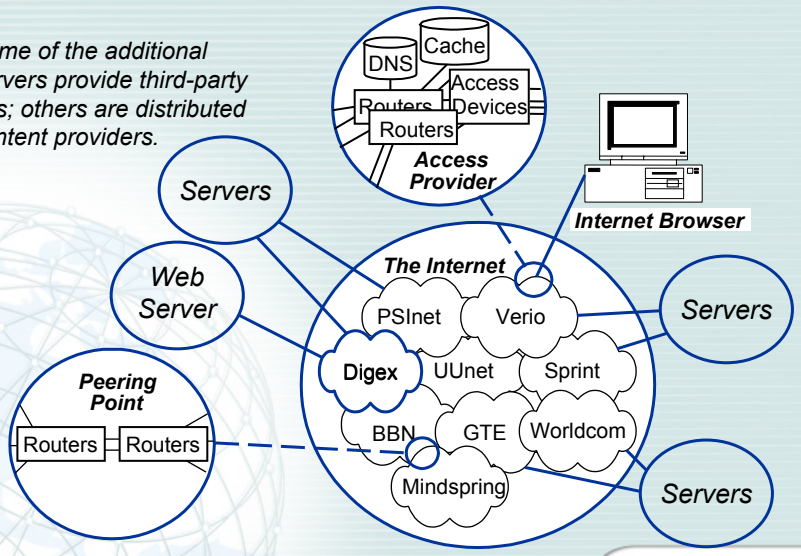
- Performance Challenges
- Web Protocol Performance and Server Performance
  - DNS, TCP, UDP, SSL, HTTP, Browser Behavior
  - Transaction Considerations
- Architecture of the Web and Internet
  - Load and Traffic Distribution
  - Internet Backbone Behavior
- Caching
- Content Distribution Networks
- Server-Farm Access to the Internet
- Internet Statistics and SLAs
- Managing for High Performance

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

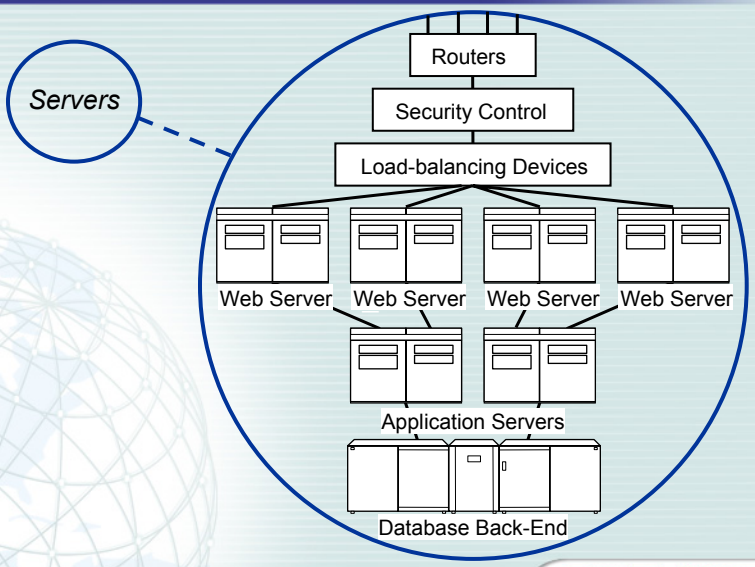
Internet Architecture Overview

Some of the additional servers provide third-party ads; others are distributed content providers.



KEYNOTE The Internet Performance Authority

Server Farm Architecture



KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Load-distribution Requirements

- Load-distribution distributes the page-serving and transaction load across all servers in a server farm
  - To increase system capacity and scalability
  - To improve system availability
  - While using mid-scale conventional computer servers
- Challenges:
  - Context-sensitive page serving ("stickiness")
  - Handling SSL and Session ID
  - Fine-grain scalability
  - Encouraging locality of reference
  - Building a resilient, manageable system
  - Security, Denial of Service defense
  - Special handling for particular clients

KEYNOTE The Internet Performance Authority

Basic Load Distribution Technologies

- DNS-based
  - DNS Authoritative Server sends different IP addresses in response to resolution request
  - Address will be cached near requester if DNS TTL > zero
  - Traffic will be sent to a server even if it has failed
- Network Address Translation (NAT)-based
  - Load-distribution device appears as a "virtual IP address/port" on the Internet. That address is then mapped to a back-end server whose address/port is concealed.
  - Load-distribution device provides some security and QoS

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Web Content Switches

- "Layer 4/7" or "Web Content Switches"
  - Examine incoming request's details
  - "Delayed Binding" terminates browser's connection in load balancer, creates connection to desired Web server and moves packets between the two connections.
  - Send image requests, searches, etc. to dedicated servers
- Some balancers perform SSL processing
  - Performs the SSL key-handling and decryption in the load distributor, usually in hardware
  - Allows load distributor to examine headers and cookies

KEYNOTE The Internet Performance Authority

Health/Performance Check

- Load distribution system should look at health and performance of servers, to increase reliability and end-user performance:
  - Monitor for server failure, incorrect pages, failed transaction
  - Monitor server statistics (CPU busy, queues, etc.) for performance
  - Monitor the number of in-process connections
  - Monitor round-trip query time to servers

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Association Persistence

- Do you need persistence?
  - Build context-independent pages
  - Put all context into the dynamic page or the cookie
- Share context data among servers
  - Shared data; single-system-image server
  - But you'll still need a transaction ID in the GET or POST
- Load-distribution device can provide "sticky" context-based forwarding based on:
  - Remote IP address
  - Cookie
  - Query String or POST Parameter
    - Load-distributor reads Transaction ID in GET query string

KEYNOTE The Internet Performance Authority

Load Handling

- If all traffic passes through load distributor, it:
  - Creates two separate connections (to browser; to server), or
  - Rewrites headers between browser and server
- Sophisticated schemes may bypass load distribution device after first routing decision
- Flash load handling:
  - Sends overflow to overflow servers with special content
  - Automatically replicates frequently-requested files onto overflow servers

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Geographic Traffic Distribution

- What is the requesting browser's address?
  - Approximate it by using the DNS request
    - Traffic Distributor acts as Authoritative DNS Server, examines source of DNS request
    - DNS request comes from the browser's local DNS server, *not from the browser itself*
    - The local DNS server may be far away from the browser!
  - Wait for TCP Connection
    - Must redirect with HTTP or "Refresh" or JavaScript, etc.
- Which server is best?
  - Classify by requester's IP address
  - Discover topological distance
  - "Ping" the browser address or local DNS server address from the various Web servers to evaluate round trip time
    - May resemble a Denial of Service attack!
  - Evaluate load on server or responsiveness of server
- Decision may need to be persistent

KEYNOTE The Internet Performance Authority

Agenda

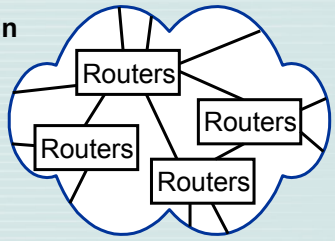
- Performance Challenges
- Web Protocol Performance and Server Performance
  - DNS, TCP, UDP, SSL, HTTP, Browser Behavior
  - Transaction Considerations
- Architecture of the Web and Internet
  - Load and Traffic Distribution
  - Internet Backbone Behavior
- Caching
- Content Distribution Networks
- Server-Farm Access to the Internet
- Internet Statistics and SLAs
- Managing for High Performance

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Internet Routing Within An AS

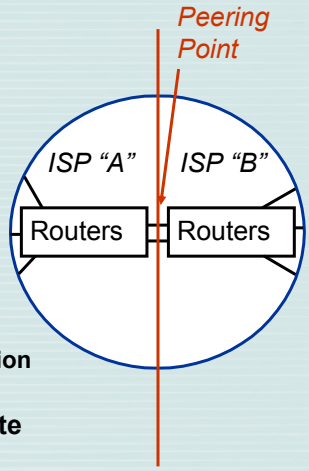
- An AS (Autonomous System) is a domain on the Internet
- Routers read every packet's header and select an outgoing path for the next hop
  - Each hop adds delay
  - Routing information is imperfect
- Packets can be lost
  - Hopelessly looping packets are discarded. (Each packet has a "Time To Live" that's decremented at each hop; an error message is sent and the packet is discarded when TTL = 0.)
  - Noise in the communications link can corrupt packets, causing them to be discarded by the hardware
  - Temporary overflow of router buffers causes packet loss



KEYNOTE The Internet Performance Authority

Internet Routing Between ISPs (Peering)

- Internet Service Providers enter into legal contracts to carry each other's traffic.
  - Traffic transfer between ISPs occurs at peering points
  - Some peering points are public; e.g., MAE-EAST (and MAE-WEST !)
  - Other peering points are privately arranged, with the ISPs sharing the cost
  - Peering philosophies differ among ISPs
- Congestion may occur at peering points, especially public ones.
  - The primary inter-ISP routing protocol, BGP-4, usually does *not* look at congestion
- The end-to-end route in one direction is usually different from the end-to-end route in the other direction!
  - Depends on legal and financial arrangements between ISPs, etc.



KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

ISP Peering Performance

- *Peering Costs Money*
  - The ISP can't use the router port to connect to a customer
  - The ISP must pay some of the cost of the connection link
  - The ISP may need to pay the other ISP if the amount of traffic in the two directions isn't roughly equivalent
  - The other ISP may not be willing to peer
- Therefore, *an ISP's business model is usually to congest the peering points!*
  - If customers don't measure and complain, the ISP has no incentive to improve peering.
  - Some ISPs concentrate on their internal backbone performance and don't spend much money on peering
- You *must* evaluate your ISP's peering to your customers' ISPs

KEYNOTE The Internet Performance Authority

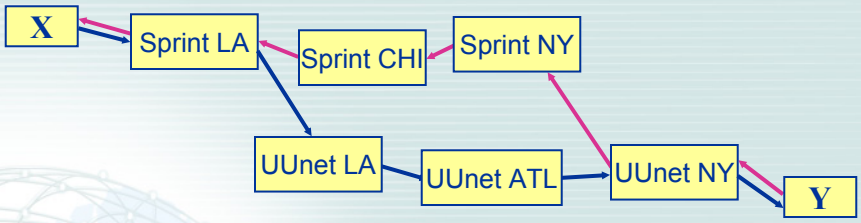
BGP-4

- BGP-4 handles routing among ASes and ISPs
  - The interior details of an AS are hidden
  - BGP-4 can contain rules to enforce peering agreements
    - Route selection can depend on the local AS's policies as well as the policies and decisions made by all other ASes in the path!
    - ISPs manipulate routes depending on financial arrangements and business decisions!
    - "Inter-AS routing has often been described as an art rather than a science, and a black art at that." — Geoff Huston, *ISP Survival Guide*
- *BGP-4 does not usually consider congestion*
  - It balances load by looking at destination, not congestion
  - It does not directly consider latency in route selection
- BGP-4 withdraws a route only if it is *completely* dead
  - Network operators will sometimes kill a congested route to force BGP-4 to re-route around it.

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Asymmetric Internet Routing



- In "hot potato routing," each ISP tries to hand off the packet as quickly as possible to the next ISP, to avoid the expense of carrying it.
  - The route is *asymmetric*.
  - For example, the ACKs for a file transfer may travel over a very different (longer?) route than the file data itself — but delayed ACKs will delay the transfer!
- Some ISPs act differently, to maintain control over the packet.

KEYNOTE The Internet Performance Authority

Traceroute

TTL value

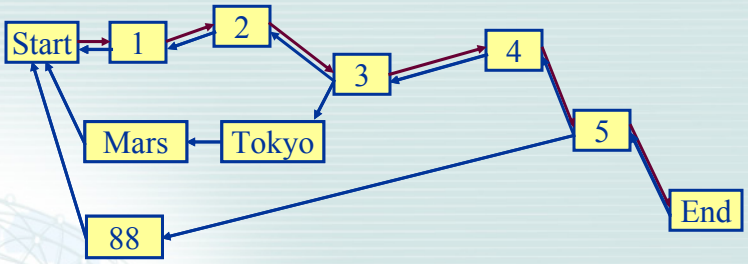
```
1 <10 ms  <10 ms   10 ms  63.67.132.1
2 <10 ms  <10 ms  <10 ms  Ser7.GW3.PAO1.ALTER.NET [157.130.230.53]
3 <10 ms  <10 ms   10 ms  5.ATM2-0.X2.PAO1.ALTER.NET [152.63.53.94]
4 <10 ms   10 ms  <10 ms  2.at-1.XR2.SAC1.ALTER.NET [152.63.49.206]
5  10 ms   10 ms  <10 ms  1.ATM.BR2.SAC1.ALTER.NET [152.63.52.85]
6  10 ms   10 ms   10 ms  208.50.172.25
7  80 ms   70 ms   80 ms  4-622M.cr2.LGA2.gblx.net [208.48.234.106]
8  70 ms   70 ms   80 ms  1-622M.hr3.LGA2.gblx.net [208.48.234.121]
9  70 ms   70 ms   80 ms  208.48.26.200
```

- Traceroute uses ICMP Echo or UDP packets
- Traceroute shows ***separate round-trip*** times to each node.
  - A packet with a short "Time To Live" (TTL) is sent; an ICMP "TTL time limit exceeded" or "port unreachable" error returns. That is the round-trip time.
- Each measurement (9 x 3 of them in this example) is *separate*.
- ***Return path for the ICMP TTL error packet is probably different than the outgoing path!***

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Traceroute Example (*Surprise!*)

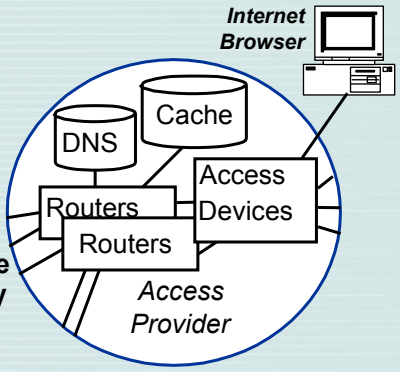


- Return paths can be stranger than this example
- This certainly affects the round-trip timings!
  - *Round-trip timings to intermediate nodes can be very misleading.*
  - (What happened to that route through Mars? *It isn't used by the real end-to-end route*; we shouldn't care about it!)
- The only accurate way to see the return path is to initiate a traceroute *from* the End *to* the Start
  - (And that still won't tell you about the irrelevant trip through Mars!)
  - *One-way traceroutes can be very misleading!*

KEYNOTE The Internet Performance Authority

Internet Access Providers

- End-users connect to an Access Device maintained by an Internet Access Provider or by their corporate IS department
  - Dial-in, DSL, cable from home
  - LAN link at the office
- Access Device connects to routers and then to the Internet
- End-users convert host name (fang.dog.com) into an IP address (10.9.23.22) by using Domain Name System (DNS) distributed directory
  - A worldwide hierarchy of directories
  - Controlled by authoritative record created by host name's owner
- Cache or Distributed Content system may also be locally available



KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Agenda

- Performance Challenges
- Web Protocol Performance and Server Performance
  - DNS, TCP, UDP, SSL, HTTP, Browser Behavior
  - Transaction Considerations
- Architecture of the Web and Internet
  - Load and Traffic Distribution
  - Internet Backbone Behavior
- Caching
- Content Distribution Networks
- Server-Farm Access to the Internet
- Internet Statistics and SLAs
- Managing for High Performance

KEYNOTE The Internet Performance Authority

Caching

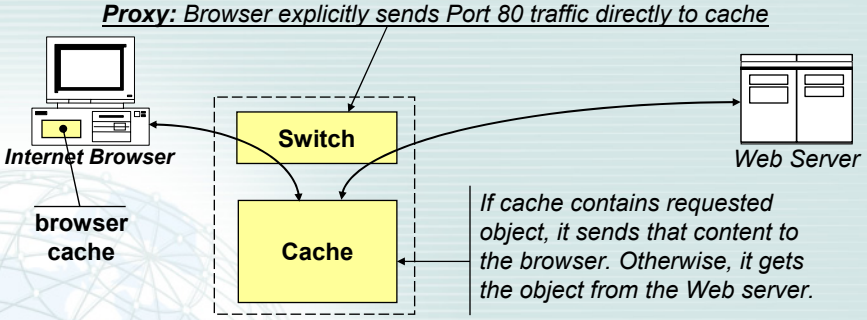
- Caching systems are usually located between server and browser
  - They temporarily store commonly-requested, unchanging objects (images, streaming media files, etc.)
  - Caching usually improves performance
  - There may be a "cache tax" — a slight delay for all objects
  - Caches may be used to assist traffic distribution by mirroring some server content at a remote location
  - Some caches may be pre-loaded to handle flash loads
- We'll look at:
  - Three types of caches (browser, client-side, server-side)
  - Cache control
  - Content distribution
  - Performance notes

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Basic Web Cache Operation

Transparent: *Port 80 traffic goes through cache regardless of original destination*
Proxy: *Browser explicitly sends Port 80 traffic directly to cache*



- Traffic is diverted by external or internal switch or router
- Cache may handle http, ftp, JavaScript files, streaming media, etc.

Browser Caching

- Every browser has a built-in cache
  - Varies in size; can be set by user
- User can set browser's basic caching rules
  - Whether or not to cache secure files
  - When to check for object changes
- User can force generation of "`pragma: no-cache`" in HTTP request header, which (theoretically) bypasses network caches and goes directly to server.
  - This is not done just by pressing the "Reload" button, which checks for object expiration and reloads only if necessary

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Client-Side Caching

- Client-side caches ("forward proxy caches") are located at the Internet Access Provider
  - Decreases the amount of traffic that the provider must pull through adjacent ISPs and usually increases the speed.
  - May be configured in the browser ("proxy" cache)
  - May be invisible to the browser ("transparent" cache)
- Extensive use by enterprises, access providers, AOL
  - Enterprises use the cache to provide firewall, filtering, and caching functions
  - Access providers use cache to decrease uplink costs and increase performance
    - Cache can use more efficient TCP parameters (window size, MSS, SACK, etc.) than the typical browser
    - Especially useful in locations that are far away from servers

KEYNOTE The Internet Performance Authority

Server-Side Caching

- Server-side cache ("reverse proxy cache," "content accelerator") is placed in front of the server farm
  - Decreases load on the servers; increases speed, availability
- Can cache more types of content, because it's owned by the enterprise and is easily controlled
  - May be able to cache the results of queries, .asp, etc.
  - Can cache long ftp files, applets, streaming media
  - Server, through an API, may be able to force flushing of specific cached content (e.g., invalidated query results)
    - "Event-based" freshness, instead of "time-based"
  - System administrator can force flushing of any cached content
- Can be placed at a remote location for content replication

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Caching Headers – 1

- Object's server tries to control network caching by using HTTP reply headers

```
HTTP/1.1 200 OK
Date: Sat, 06 May 2000 18:06:07 GMT
Content-Type: text/html
Set-Cookie: IsClient=0; path=/MyKeynote
Cache-control: private ← Cache in browser only, not in network
```

- Network caches do not look at HTML caching instructions
  - HTML's META HTTP-EQUIV tags are *not* seen or used by network caches, although they are used by browser caches
  - Some Web servers automatically build HTTP headers from the META HTTP-EQUIV tags when they transmit objects
- Non-HTTP objects (e.g., FTP files) don't have headers
  - Expiration times are set by cache administrator

KEYNOTE The Internet Performance Authority

Caching Headers – 2

- The Expires header is the basic way to set expiration
  - `Expires: Mon, 30 Oct 2000 14:19:41 GMT`
  - A zero or -1 means "expired"
  - HTTP/1.1 also has `Cache-Control: max-age = <seconds>`
- If there's no Expires header, cache may use a Last-Modified header to infer an expiration time
  - Header indicates the time when the object was last modified (e.g., its file's modification date)
  - Cache heuristics then derive an expiration time. Example:
    - Delete after object becomes 20% older than it was when fetched
- Most caches will not cache an object that doesn't have either a `Last-Modified` or an `Expires` header
  - A surprising number of servers don't generate these headers by default!

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Caching Headers – 3

- **Cache sends Get If Modified Since request to server**
  - `GET...<cr> IF-MODIFIED-SINCE : <timestamp>`
- **Caching is controlled by direct header commands:**
  - `Pragma: no-cache`
    - This is normally placed in an HTTP *request*; it asks the cache to fetch the object from the server instead of from cache
    - This is not officially a response option in HTTP; therefore, it is usually ignored when it is in a *reply*.
  - `Cache-Control:` **headers are for HTTP/1.1 only**
    - `Cache-Control: Public` (cache anywhere)
    - `Cache-Control: Private` (cache in browser only)
    - `Cache-Control: No-Cache` or `No-Store` (don't cache)
      - Internet Explorer won't cache this for use with "back" button

KEYNOTE The Internet Performance Authority

Cache Restrictions

- **Dynamic objects are not usually cached**
  - **Caches won't store cgi, .asp, .jsp, GET query-string, POST**
    - Server-side caches may store some dynamic objects and provide an operator interface and/or an API to control behavior
  - **HTTP 302 Redirect without Expires header isn't cached**
  - **Server Side Include (SSI) pages usually don't have cache headers, so they usually aren't cached**
- **Cookies shouldn't be cached**
  - **Cache is supposed to remove the Set-Cookie header string**
  - **It's safest to mark an object containing a cookie as "private"**
- **Caches usually will not handle negotiated content**
  - **For example, a server may look at USER-AGENT field to decide which object to send**
  - `Vary:` **header may be used, but many caches don't handle it**
- **Secure objects are usually not cached**
  - **Objects with authorization headers may be cached**
  - **SSL-encrypted objects are not cached**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Content Distribution

- Cache and traffic-distribution vendors have created tools that can be used to manage the distribution of content *to remote caches that you control*
  - Tight control over expiration
  - Can handle versioning/rollback
  - Can distribute only changed content
  - Additional content can be distributed as needed by load
  - Regional content can be distributed to regional caches
  - Content can be pre-positioned before major events
- Distribution can be over the Internet or by alternate paths (satellite, dedicated link)

KEYNOTE The Internet Performance Authority

Performance Notes: Caching – 1

- Set the HTTP headers to encourage caching
  - Use the "Last Modified" and "Expires" headers. Set the expire date far into the future; if you need to kill the element, rename it.
  - Some caches ignore the HTTP "Pragma: No-Cache" tag, especially if it is in the reply instead of in the request.
  - Non-browser caches generally ignore HTML caching instructions
- Reuse content and use identical URLs each time
- CGI, .ASP, .JSP pages are usually not cached
- You may force cache headers into Server Side Include (SSI) pages to allow caching
- Query-string results and POSTs are usually not cached
- Avoid URLs with embedded session numbers that make them unique and uncacheable
- Redirected responses are usually not cached
  - Don't redirect by accident
- Cookies have varying interactions with caches
- Secure objects are usually not cached

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Performance Notes: Caching – 2

- If possible, avoid content negotiation and parsing of User-Agent
- Very long files and applets are sometimes not cached
- If you need to count hits, use a small one-pixel uncacheable .gif
- Use server-side caching
  - Some dynamic pages can be cached on server-side caches
- If streaming media is cached, it may be possible for it to serve different bandwidth feeds from one cached file
- Optimize your cache's TCP, HTTP stack
  - MSS, Receive Window, SACK, Slow Start, HTTP/1.1 persistence
  - The browser may have a non-optimal stack, but you can use an optimized stack to fetch data from the server!
- Performance varies with cache location
  - A cache that's close to the server will handle more traffic than a cache that's farther away, as cache misses are handled faster
- Public caches may disobey your caching instructions
  - And you shouldn't assume they'll use advanced caching techniques, store streaming media, etc.

KEYNOTE The Internet Performance Authority

Agenda

- Performance Challenges
- Web Protocol Performance and Server Performance
  - DNS, TCP, UDP, SSL, HTTP, Browser Behavior
  - Transaction Considerations
- Architecture of the Web and Internet
  - Load and Traffic Distribution
  - Internet Backbone Behavior
- Caching
- Content Distribution Networks
- Server-Farm Access to the Internet
- Internet Statistics and SLAs
- Managing for High Performance

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Content Distribution Networks

- Content Distribution Networks (CDNs) have constructed worldwide systems of caching, geographic traffic distribution, and content distribution management devices.
  - Usually in partnership with local ISPs
  - CDNs specialize in static content (.jpg, .gif, etc.); they may also be able to handle authentication, streaming media and some dynamically-generated Web pages.
  - In some cases, they use distribution systems (leased lines, satellite, etc.) that completely bypass the Internet's core and may preload cache content.

KEYNOTE The Internet Performance Authority

CDN Technology

- CDN typically hosts only static content
  - Not the base HTML
  - Not the dynamic content (stock price, ads, etc.)
- End-user must go to a hosting facility to obtain base HTML and dynamic content
  - Remember the example at the beginning of this workshop?
- The base HTML and/or the content provider's hosting setup probably will need some modification
  - End result is that browser asks CDN's Authoritative DNS record for directions to the CDN-served content, and the CDN uses very sophisticated traffic-distribution technology when it gives those directions
- True transactions must be handled by hosting facility

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

CDN Advantages – 1

- **Performance**
  - **Akamai study with Keynote measurements shows median download time of major sites was 7 times faster on a CDN**
  - **Speedup during peak business hours was much larger**
    - This is because the CDN insulates the end-user from most congestion at peering points and at the server farm
    - CDN greatly decreases end-user performance variation
  - **Error rate was also much lower using a CDN: 30% or more**
  - **Additional content can be distributed as needed by load**
  - **Content can be pre-positioned before major events**
- **Offloads hosting Web center**
- **Streaming media files can be pre-positioned at CDN**
  - **CDN can transcode (derive) lower-bandwidth streams locally**

KEYNOTE The Internet Performance Authority

CDN Advantages – 2

- **Management tools**
  - **The advantages of running content distribution with caches, but you don't have to pay for it all yourself!**
    - Tight control over object freshness
    - Can handle content synchronization / versioning / rollback
    - Can distribute only changed content
    - Can distribute large files, applets, streaming media and know that the cache will store them instead of throwing them out
  - **Better statistics than by using "one-pixel gif" objects**
  - **Some traffic distribution devices can distribute to CDNs**
- **Better localization**
  - **CDN usually excellent at finding geographic location**
  - **Regional content can be distributed to regional caches**
- **May save money**
  - **Billing typically based on actual bandwidth used**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Evaluating CDNs

- How does the CDN choose the "closest" server?
  - If it's dependent on DNS, how does it ensure that most of the DNS requests come from DNS servers close to the browsers?
- What is the latency between the CDN's "closest" servers and your customers? What is the capacity?
  - Some CDNs have thousands of small cooperating caches; some have a few dozen large data centers; others have a mix
- How does the CDN ensure that the data is available at the server and can be delivered quickly?
- How can you control your data?
- Does the CDN provide useful metrics?
  - To count user page views and gather near-real-time statistics
  - To evaluate CDN performance
    - Be sure to evaluate CDN performance *vs.* the optimized, cache-friendly performance of your non-CDN Web page

KEYNOTE The Internet Performance Authority

Agenda

- Performance Challenges
- Web Protocol Performance and Server Performance
  - DNS, TCP, UDP, SSL, HTTP, Browser Behavior
  - Transaction Considerations
- Architecture of the Web and Internet
  - Load and Traffic Distribution
  - Internet Backbone Behavior
- Caching
- Content Distribution Networks
- Server-Farm Access to the Internet
- Internet Statistics and SLAs
- Managing for High Performance

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Access to the Internet

- **Each server farm must choose:**
  - **The type of access they'll have to the Internet**
  - **The ISPs that will be used to provide access**
- **And those connections must then be tuned for performance and managed**

KEYNOTE The Internet Performance Authority

Access Types – 1

- **Single-homed**
  - **One connection to one ISP is used for all Internet Access**
    - Load may be shared across multiple parallel access links
  - **BGP-4 isn't necessary**
- **Multiconnected, single-homed**
  - **One ISP, but multiple connections to that ISP**
    - Each connection may have multiple parallel access links that share the load
    - ISP picks "closest" connection to enter into server farm
    - Server farm shares outgoing load by manipulating outgoing default router for different address ranges
      - *This is not based on load*
    - Connections provide mutual backup for each other
  - **Routes may be asymmetric, entering on one connection and exiting on another**
  - **BGP-4 isn't necessary**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Access Types – 2

- **Multihomed**
  - **Multiple ISPs and multiple connections**
    - Each connection may have multiple parallel access links that share the load
    - Choice of entry and exit connections may be complex
      - May pick “closest” connection to destination address
      - May use BGP-4 filtering and other rules
      - *This is not based on load*
    - Connections provide mutual backup for each other
  - **Routes may be asymmetric, entering on one connection and exiting on another**
  - **BGP-4 is required**
  - **Large hosting facilities should be multihomed**

KEYNOTE The Internet Performance Authority

Choosing ISPs

- **Measure your ISP’s peering to your customers’ ISPs**
  - **Performance depends on the number of peering points, their location relative to your position and your customers’, the congestion at and surrounding the peering points, the configuration of BGP-4, and more.**
    - ***It’s easy to tell a story; you’d better measure!***
  - **Performance will be changing rapidly over the next years as the industry changes**
  - **Performance should be measured in terms of latency and page download time; latency and error rate are insufficient for evaluating Web and streaming media performance.**
- **Satisfaction with an ISP depends on much more than technical performance and cost. For example:**
  - **Technical help with your routing configurations and tuning**
  - **Influence with other ISPs at their peering points**
  - **Responsiveness to problems**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Working With ISPs – 1

- **A few hints about working with ISPs to improve performance:**
  - **Your only real leverage is through ISPs that you pay**
  - **Remember that Traceroute shows the path in only one direction, the intermediate timings can be irrelevant, and Internet routing is asymmetric. There's a chance that you're blaming the wrong ISP.**
  - **The ISPs can look at their internal router statistics, and get a definitive answer, much more quickly than you can run a series of Traceroutes to get much less useful information.**
    - No ISP will ever let *you* see their router statistics.
  - **There's inter-ISP politics and money involved**
- **Therefore:**
  - **Look for a way to do *triage,* not diagnosis**
    - "All the Keynote Agents on UUnet are reporting problems"
  - **Work through your own ISP**

KEYNOTE The Internet Performance Authority

Working With ISPs – 2

- ***An accurately-reported squeaky wheel gets the grease***
  - **Fixing problems takes thinking, and peering costs money**
    - And there's a shortage of thinkers and money nowadays (sigh)
  - **If you don't complain, it won't get fixed**
    - But if you repeatedly complain about problems that don't exist, or using traceroutes that you don't understand . . .
    - And if you have six different people complain to the ISP about the same problem without coordinating with each other . . .
    - You'll build a reputation that isn't helpful when you need help!
- **Try to build a relationship with your support person**
  - **Coordinate the contacts at your end**
  - **Avoid finger-pointing, but present convincing triage data when you claim there's a problem**
- **Educate your own support team**

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

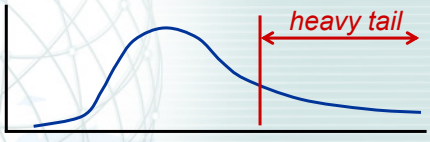
Agenda

- Performance Challenges
- Web Protocol Performance and Server Performance
  - DNS, TCP, UDP, SSL, HTTP, Browser Behavior
  - Transaction Considerations
- Architecture of the Web and Internet
  - Load and Traffic Distribution
  - Internet Backbone Behavior
- Caching
- Content Distribution Networks
- Server-Farm Access to the Internet
- Internet Statistics and SLAs
- Managing for High Performance

KEYNOTE The Internet Performance Authority

Statistical Behavior of Internet Traffic

- Internet traffic is self-similar and heavy-tailed
  - Result of mixing long files, small files, ACKs, compressed video, human think time, etc.
  - (Connection arrival rate is still normally distributed)
- Self-similar traffic is very bursty
  - No natural length to the bursts
  - Bursts are not quickly smoothed by larger time scales
  - Capacity problems can appear at 50% utilization!
- Heavy-tailed (right tail) traffic can have an arbitrarily large amount of very large values



KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Performance Statistics for Heavy-Tailed Traffic

- **With heavy-tailed traffic:**
  - **Arithmetic Mean may be misleading.**
    - Individual long measurements can distort the mean, especially if the typical measurement is very short (e.g., DNS, TCP Connect)
    - This is more troublesome for SLAs and long-term trending than for problem diagnosis, which involves detailed examination of datapoints
    - If you have 10,000 measurements of 0.02 seconds, plus TEN measurements of 20 seconds, the arithmetic mean is 0.04 seconds — a *doubling* of the mean because of only *ten* measurements!
    - Geometric Mean or Median are better, more stable indicators of typical performance as seen by users
  - **Standard Deviation is *very* misleading in measuring heavy tail data**
    - Can be massively distorted by a single large measurement!
    - Geometric Deviation or 85th percentile are better, more stable indicators of variance

KEYNOTE The Internet Performance Authority

Computation of Performance Statistics

- **Arithmetic Mean is easy and useful for "quick looks"**
  - **Arithmetic means can be aggregated (e.g., DNS + TCP Connect time) by simple addition; other statistics cannot.**
    - (85th Percentile of DNS) + (85th Percentile of TCP Connect) is NOT the 85th Percentile of (DNS + TCP Connect)
  - **"Trimming above *n* seconds" is a computationally-simple method for improving the stability of the arithmetic mean, but the number of trimmed values must also be considered.**
- **Other measures require more manipulation**
  - **Percentiles give a reasonable, but coarse, estimate**
    - Median is the 50th percentile
  - **Geometric Mean is the best measure**
    - It's the nth root of the product of the *n* measurements.
- **Geometric Deviation is a good measure of variance**
  - **"Geometric Deviation" is a *factor* = 10^(std deviation{log(x_i)})**
- ***See Keynote website "Resources/White Papers" for statistics articles***

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Agenda

- Performance Challenges
- Web Protocol Performance and Server Performance
  - DNS, TCP, UDP, SSL, HTTP, Browser Behavior
  - Transaction Considerations
- Architecture of the Web and Internet
  - Load and Traffic Distribution
  - Internet Backbone Behavior
- Caching
- Content Distribution Networks
- Server-Farm Access to the Internet
- Internet Statistics and SLAs
- Managing for High Performance

KEYNOTE The Internet Performance Authority

Why Measure?

- Without measures, we can't ensure that we have good performance in the unstable Internet technical and business environment.
  - The Internet backbones are rapidly changing
  - Massive new loads are appearing daily
  - Internet routing tables are becoming unwieldy
  - Some ISPs are having financial problems
- Measurement Goals:
  - Trending and Service Level Agreements
  - Executive Status Reporting
  - Quick problem detection and credible triage
  - Diagnosis
  - Tuning
  - Load testing and test validation

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Measurement of Individual Elements

- Measurement of individual network and server elements is:
  - Great for system operators and for triage
  - Necessary for tracking load *vs.* element utilization
- Typical element measures:
  - Device status: CPU, memory, link utilization; queue sizes; queue overflows
  - Identity of heavy users; traffic on hardware ports
  - Bandwidth usage
  - Application statistics (page hits, user counts, abandoned shopping carts, etc.)
  - Some passive measurement tools can examine frame or packet headers to track response times, etc.
- SNMP Polling is the basis for most element measurements

KEYNOTE The Internet Performance Authority

Measurement of the End-User Experience

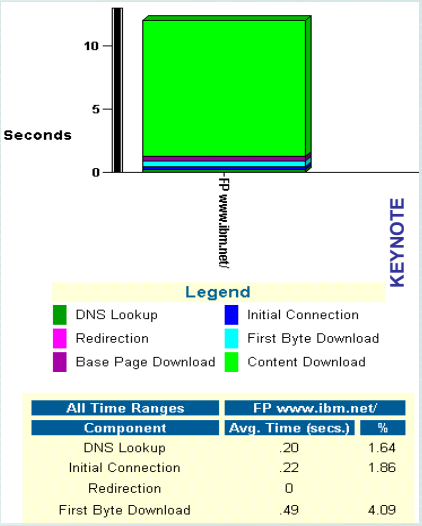
- Network-level pings, etc. are useful for problem diagnosis, but are not a true measurement of end-user experience.
  - Reaches only outskirts of Web hosting system, not the application
  - Does *not* indicate the health of application
  - Usually is not directly correlated with end-user's Web page experience
- Automated measurement agents run scripts to download Web pages and run transactions.
  - Includes non-network (e.g., server) time
  - May include detailed component measurements that are useful for triage and trending
  - Finds errors as seen by users

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Typical Active End-User Measurement

- Details of download can be timed and displayed.
- Download details can be trended over time.
- Includes:
  - DNS lookup
  - TCP connection complete
    - round-trip network time
  - First packet ("first byte")
  - Redirections complete
  - Base page complete
  - Content (images, etc.) complete



All Time Ranges	FP www.ibm.net/	
Component	Avg. Time (secs.)	%
DNS Lookup	.20	1.64
Initial Connection	.22	1.86
Redirection	0	
First Byte Download	.49	4.09

KEYNOTE The Internet Performance Authority

Load Testing an Enterprise System *vs.* Load Testing a Web System

- For example, a call center vs. a Web site:
  - Call center operators do not abandon ("click away") if response time is poor; they remain on-line and will eventually continue through the transaction sequence.
  - Call center protocols usually give a clear indication of session termination; Web protocols usually do not.
    - Abandoned Web sessions use resources until they're timed out.
    - Web site's "concurrent session" statistics may be misleading.
  - Call center load are much more uniform than Web loads, and they cannot exceed the number of operators.
    - Excess incoming traffic is buffered by busy signals, etc.
  - Call center operator skills are more uniform.
  - Call center access speeds are more uniform.
  - Web networks contain elements that affect performance of the enterprise site and must be evaluated.
  - Web client ("browser") is usually more complex than legacy terminal or client

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

Web Load Testing Must Include:

- **Variable loading**, for realism and to test flash loads
- **Satisfaction measures**, based on response time
- **Abandonment behavior**, for realism and to evaluate resource recovery mechanisms
- **Distributed loading**, to test the Internet connections, peering, caching, and content distribution networks
- **Multiple geographic profiles**, to represent the wide variations in Web access technology and location
- **Multiple user connectivity profiles**, because latency and bandwidth affect the load seen by the servers
- **Multiple user profiles**, to model the wide variation in user skills present in the Web community
- **Script variability**, of both data and paths, to avoid creating unrealistic hot spots that are handled by caching
- **True browser emulation**, of varied browsers, including browser caching, cookies, Javascript, SSL, etc.

KEYNOTE The Internet Performance Authority

Keynote Systems (www.keynote.com), "*The Internet Performance Authority®,*" **is the world's leading supplier of Internet performance measurement, diagnostic, load testing, and consulting services to companies with e-commerce Web sites.** Keynote® captures over 20 million performance measurements daily, using Keynote's global infrastructure of over 1500 measurement computers connected to the major Internet backbones from over 100 statistically selected Internet access locations in 50+ metropolitan areas worldwide. Internet performance and availability data are collected at Keynote's sophisticated operations center and are instantly available to customers through any Web browser, by real-time XML transfer, or by FTP. **Keynote currently measures individual Web pages as well as transactions, streaming media, and wireless. Keynote also supplies highly-accurate, distributed Web load testing services.**

Eric Siegel is Principal Internet Consultant with Keynote Systems, Inc. and is the author of *Designing Quality of Service Solutions for the Enterprise* (John Wiley & Sons, 1999). Before joining Keynote Systems, Mr. Siegel was a Senior Network Analyst at NetReference, Inc., which specializes in network architectural design and strategic planning, and he was a Senior Network Architect with Tandem Computers, where he was the technical leader and coordinator for all of Tandem's data communications specialists worldwide. Mr. Siegel also worked for Network Strategies, Inc. and for the MITRE Corporation, where he specialized in computer network design and performance evaluation. Mr. Siegel received both his B.S. and M.E.E. degrees in Electrical Engineering from Cornell University, and he has been a member of the Internet community since 1978.

KEYNOTE The Internet Performance Authority

Copyright © 2002 by Keynote Systems and Eric Siegel. All Rights Reserved.

QW2002 Paper W2

Mr. Hung Nguyen & Mr. Hans Buwalda
(LogiGear Corporation)

Jump Starting Your Test Automation

Key Points

- Implement effective and maintainable tests using Action Based Testing™ (ABT) approach
- Win and keep the necessary management commitments
- Deal with common technical challenges such as testing Web applications, legacy systems, load testing, and complex multi-platform environments

Presentation Abstract

Whether you're looking for a new testing approach through automation or re-examining existing test automation strategies, your objectives are the same: Maximize test coverage, save time, and keep testing under control. A delicate combination of technical problems and management issues, test automation is a challenging task. Hung Nguyen and Hans Buwalda discuss how to start up or improve test automation to meet your goals. They address topics such as how to set up the right team, options for an effective and stable architecture, and how to deal with managerial issues you're likely to encounter.

About the Author

Hung Q. Nguyen is founder, president, and CEO of LogiGear® Corporation. He's held leadership roles in business management, product development, business development, engineering, quality assurance, testing, and information technology. Hung is an international speaker and contributor to industry publications. He authors and teaches software testing curriculums for LogiGear University and the University of California. He is the original architect of TRACKGEAR™, a Web-based defect management system, and the author of Testing Applications on the Web (Wiley). He also wrote (with Kaner and Falk) the best-selling book Testing Computer Software (Wiley), which is also published in Japanese. He holds a B.Sc. in Quality Assurance from Cogswell Polytechnical College, is an ASQ-Certified Quality Engineer, and a member of the Advisory Council for the Department of Applied Computing and Information Systems at UC Berkeley Extension.

Hans Buwalda, ABT Chief Architect, leads LogiGear's Action Based Testing™ (ABT) research and development, including ABT Toolset™ operations, and oversees the practice of the ABT methodology. Prior to joining LogiGear, Hans served as Project Director at CMG in The Netherlands where he was the original architect behind what has now become TestFrame™ - an integrated method for planning, managing, and deploying software testing and test automation. Hans is an internationally recognized expert specializing in test automation, test development, and test management. He speaks and presents workshops at international conferences on testing concepts such as Action Based Testing, The Three Holy Grails of Test Development, Soap Opera Testing, and Testing in the Cold. Hans authored (along with Dennis Janssen and Iris Pinkster) Integrated Test Design and Automation: Using the TestFrame Method (Addison Wesley 2001). He holds a Master of Science degree in Computer Science from Free University, Amsterdam.

Applications Workshop W2

Jump Starting Your Test Automation

Hung Nguyen
Hans Buwalda
LogiGear® Corporation

© 2002 LogiGear Corporation. All Rights Reserved.

Agenda for Today

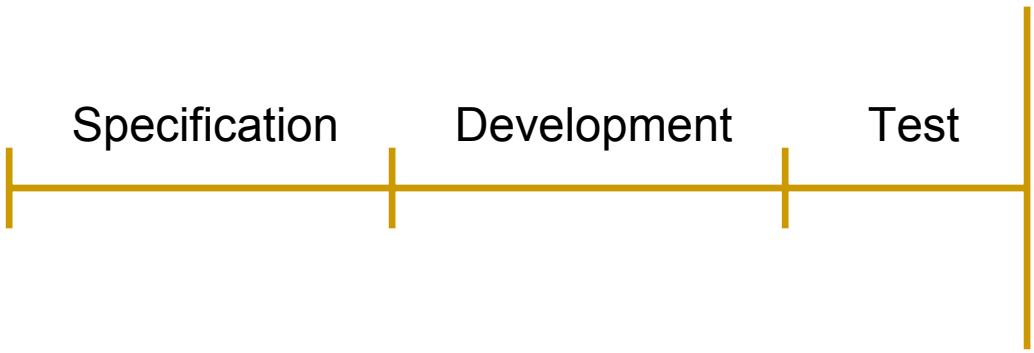
- Overview of test automation strategies
- Introduction to Action Based Testing as an approach
- Jump starting your test automation with an integrated approach
  - Test design strategies and methods
  - Automating tests ("Navigation Engineering")
  - Test management

Objectives

- Focus on sharing testing automation strategies through
  - Ideas
  - Concepts
  - Experiences
- To a lesser extent:
  - In-depth treatments
  - Full coverage of possible issues
- Illustrating an integrated approach to test automation through "Action Based Testing"
  - Incorporates most of the key ideas and concepts
  - Home match for your teacher

© 2002 LogiGear Corporation. All Rights Reserved.

Testing Under Pressure



© 2002 LogiGear Corporation. All Rights Reserved.

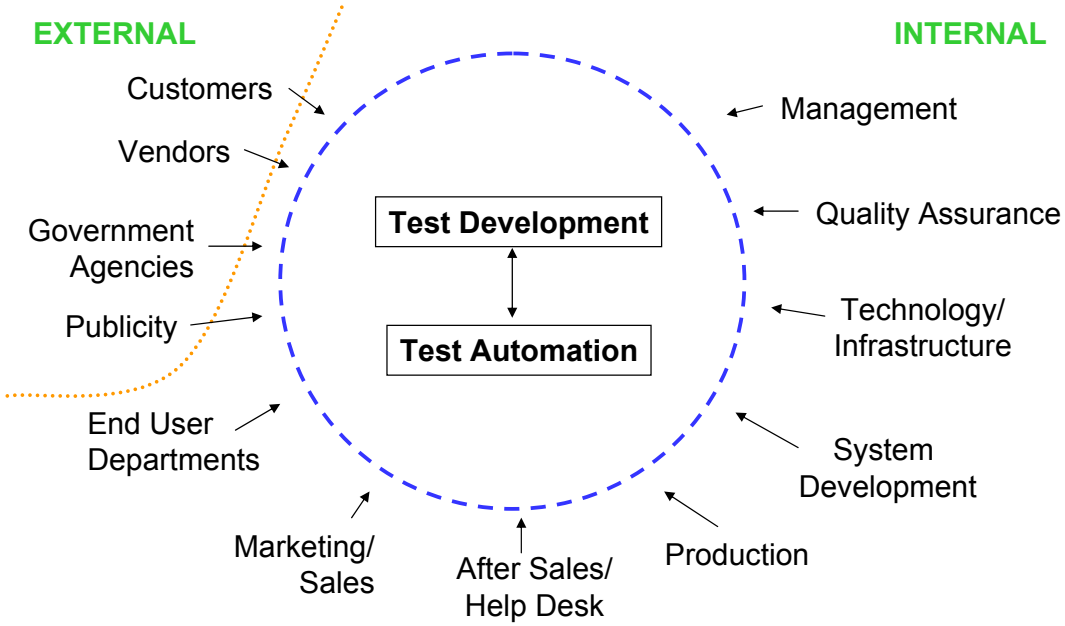
Testing Under Pressure

Develop tests in time:

- Test design
  - Description of tests (Actions and checks)
- Ways to execute tests
  - Manually or automatically

© 2002 LogiGear Corporation. All Rights Reserved.

Strategic Position of Test Processes



© 2002 LogiGear Corporation. All Rights Reserved.

Key Issues in Testing

- Effective test design
- Comprehensive automation architecture
  - Consider manageability and maintainability.
- Methodical management of the tests
  - Tests and test scripts are products. They need to be managed.
- Competent management of the test process
  - Managers want to know what is going on. Give them what they want.
- Thorough documentation
- Clear communication through useful reporting
  - Communicating progress status and test results.
- Firm distinction between testing and quality assurance
  - Solicit involvement of stakeholders, users, auditors and decision makers.

© 2002 LogiGear Corporation. All Rights Reserved.

What are the Typical Problems?

- Poorly designed test cases won't catch bugs
- Lack of test automation architecture makes automation expensive and difficult to manage and maintain
- Disorganized test processes
  - Test designers and test automators require/have different skill sets
  - The stakeholders don't know what is going on
- Testing effort is underestimated or deferred
  - Testing is difficult and expensive
  - Management might be in denial
  - It is unattractive to spend money on testing
  - Cost shifting
- Unclear about the three separate test automation problems that need to be addressed: Test design, automating tests, and test management.
- No clear direction.
- Focus on tools and technology
  - The engineers are bottlenecks

© 2002 LogiGear Corporation. All Rights Reserved.

Recommendations

- Integrate testing and test automation
  - Don't automate manual testing
- Put the test engineers in control of the test logic independent to test execution strategies
- Pay attention to good test design and test automation architecture
- Actively solicit and manage the involvement of stakeholders
- Structurally implement testing and test automation strategy
  - Test automation is not a hobby

© 2002 LogiGear Corporation. All Rights Reserved.

Test Design Considerations

- Understand the difference between passive and active test
- Understand the difference among black-box, white-box and gray-box testing
- Understanding S.I.C.O
  - Storing (data), Input, Computing, and Output
- Start with component-based and file-based inventory lists
- Identify various types and points of interface
- Identify various actions used by each interface
- Identify various data used by each interface

© 2002 LogiGear Corporation. All Rights Reserved.

Why Do We Automate Tests?

- It is more interesting and fun
- Let the machine do the work
- Save time and money
- Reduce the misuse of valuable test engineering resources
- Formulate a structured way of working
- Shorten time-to-market
- Improve test coverage

© 2002 LogiGear Corporation. All Rights Reserved.

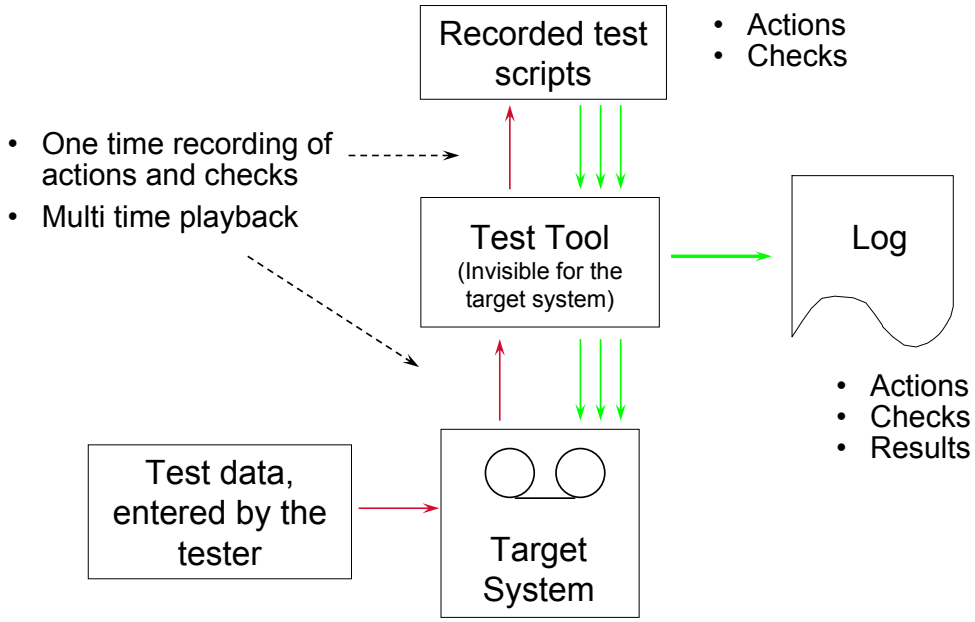
Some Approaches to Test Automation

- Dumb Monkeys
- Record/Playback
- Scripted Approach
- Data-Driven Testing
- Action Based Testing

"Software Test Automation" by Mark Fewster and Dorothy Graham, Addison Wesley, 1999

© 2002 LogiGear Corporation. All Rights Reserved.

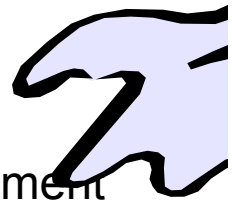
Record/Playback



© 2002 LogiGear Corporation. All Rights Reserved.

Beware of the [illegible]/P Pitfalls

- Sensitivity to ch[illegible]s leads to high mai[illegible]ance cost



- [illegible] difficult to access
- [illegible]he working system or environment
- Only suitable for GUI systems
- Upside
  - A useful way to learn about the tool and automation

© 2002 LogiGear Corporation. All Rights Reserved.

Scripted Approach

- Automation is regarded as a programming task
- Improved reusability
  - Parameterize hard-coded values
  - Separate data from code by moving variables to include files
  - Create utility functions to be shared
- Produce and maintain like any other software
- Train test specialists to run the scripts

© 2002 LogiGear Corporation. All Rights Reserved.

Scripted Approach

- A much better approach
- Reusability of scripts for common tasks
- Improved maintainability
- Issues
  - Tests are programmed, and are relatively hard to access
  - Affected by a working system or environment
  - Test implementation must be done by engineers
  - 3000 tests mean 3000 automated scripts
  - Manageability is still an issue

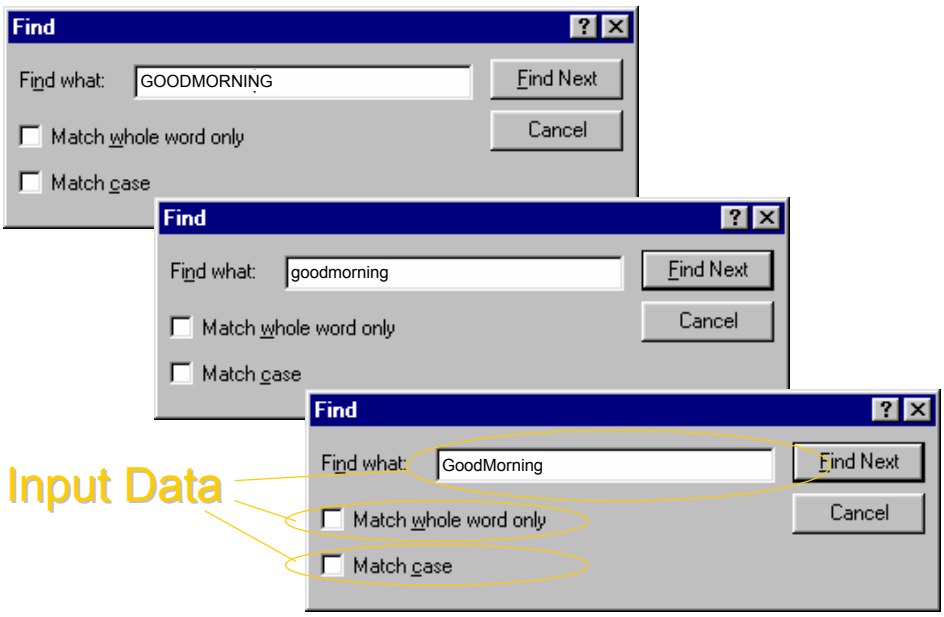
© 2002 LogiGear Corporation. All Rights Reserved.

Data-Driven Approach

- Take advantage of tester's familiarity with test case design and creation using tables and matrices
- Accommodate localization projects
- Recognize the importance of patterns in test cases
- Enable testers to catalog test cases with Excel spreadsheets
- Enable testers to specify expected results in spreadsheets

© 2002 LogiGear Corporation. All Rights Reserved.

Data-Driven Example



© 2002 LogiGear Corporation. All Rights Reserved.

Data-Driven Example

- File with test data:

nr	case	whole	pattern	matches
1	off	off	GOODMORNING	4
2	off	on	GOODMORNING	3
3	on	off	GOODMORNING	1
4	on	on	GOODMORNING	1
5	off	off	goodmorning	4
6	off	on	goodmorning	3
7	on	off	goodmorning	1
8	on	on	goodmorning	1
9	off	off	GoodMorning	4
10	off	on	GoodMorning	3
11	on	off	GoodMorning	2
12	on	on	GoodMorning	1

Text file used in the test:

```
goodmorning
GOODMORNING
GoodMorning
GoodMorningVietnam
```

for each line in the file do
  open the find dialog
  read a line from the file
  use the values to fill dialog
  press find button
  check amount of matches
  close the dialog

© 2002 LogiGear Corporation. All Rights Reserved.

Scripted/Data-Driven Approach

- Separation of action and data
- Test data can go in separate file or spreadsheet
- Improved reusability and maintainability
- Issues
  - Actions (containing business logic) remained in automated scripts
  - Sensitivity to changes
  - Manageability is still a burden

© 2002 LogiGear Corporation. All Rights Reserved.

Action Based Testing Approach

Test Cluster
(Spreadsheet, Table, ...)

Test Development

A	B	C	D
. . .			
transfer sum	Houston	Klein	210
check balance	Klein	210*	
transfer sum	Savy	Klein	150
check balance	Klein	360*	
. . .			

*expected values

Navigation Engineering

. . .	
transfer sum	
check balance	
. . .	

Navigation Scheme

© 2002 LogiGear Corporation. All Rights Reserved.

Example of an ABT Test Cluster

cluster	**EXAMPLE OF A TEST CLUSTER**			
version	1.0			
author	Hans Buwalda			
section	**1. Entering clients and balances**			
	last name	first name	account nr	balance
enter client	Green	John	458473948	1500
enter client	Wood	Anna	422087596	2100
section	**2. Money Transfers**			
	from	to	sum	
transfer	458473948	422087596	500	
transfer	422087596	785793025	1201	
section	**3. Checking names and numbers**			
	account nr	last name	first name	
check name	458473948	Green	John	
check name	422087596	Wood	Anna	
	account nr	sum		
check balance	458473948	1000		
check balance	422087596	1399		

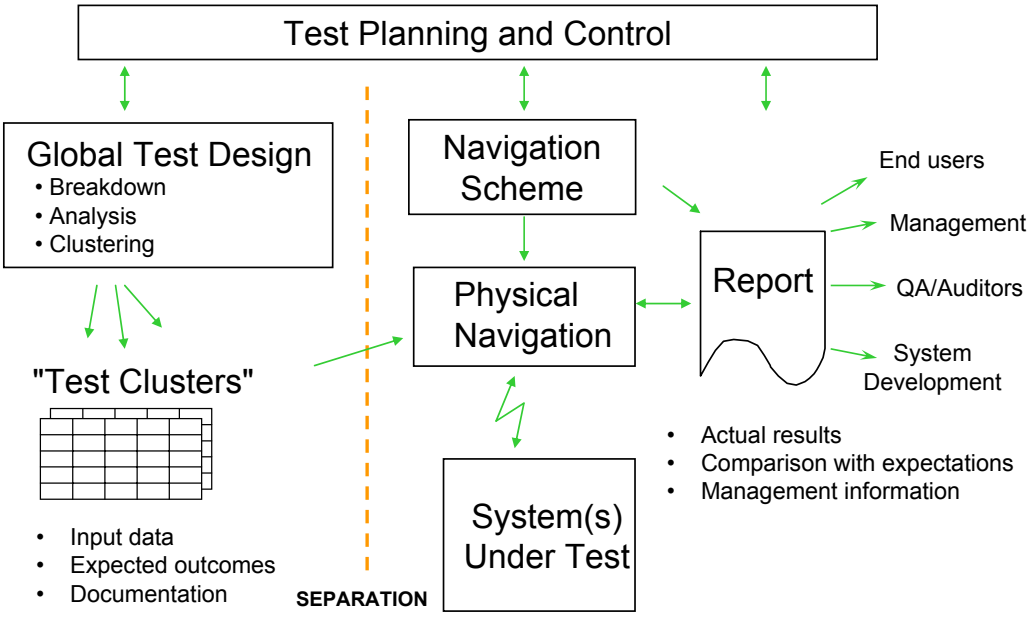
HEADER

INPUT

EXPECTED OUTCOMES

© 2002 LogiGear Corporation. All Rights Reserved.

Example of a Test Product Life Cycle



© 2002 LogiGear Corporation. All Rights Reserved.

Example of ABT Test Reporting

TEST RUN SUMMARY

test cluster:	Minibank business test
test sheet:	demonstration scenarios
test version:	1.2
test author:	Hans Buwalda, all rights reserved
test date:	April, 2001
run date and time:	Tue Nov 06 12:49:06 2001

SCENARIO MB01 -- Entering customers using manual numbering
In this scenario the account numbers are entered manually.

Enter Customers - Entering 5 customers in the Relation Entry screen

14, enter customer: Johnson Jean 500103381 1500

15, enter customer: Juet Christian 423137538 2100

16, enter customer: Savy Anne 848656467 1700

⋮

© 2002 LogiGear Corporation. All Rights Reserved.

Example of ABT Test Reporting

:

Check Balances - Check if the balances are conform the entry and the transfers

36, check balance: 500103381 1000
check: balance
expected: 1000
recorded: 1000
result passed

37, check balance: 423137538 555
check: argument 2
expected: 555
recorded: 2600
result failed

RESULTS TEST EXECUTION

overall statistics for this test run:
number of checks: 21
number of passes: 20
number of failures: 1

fail(s) were found in these line(s):
37

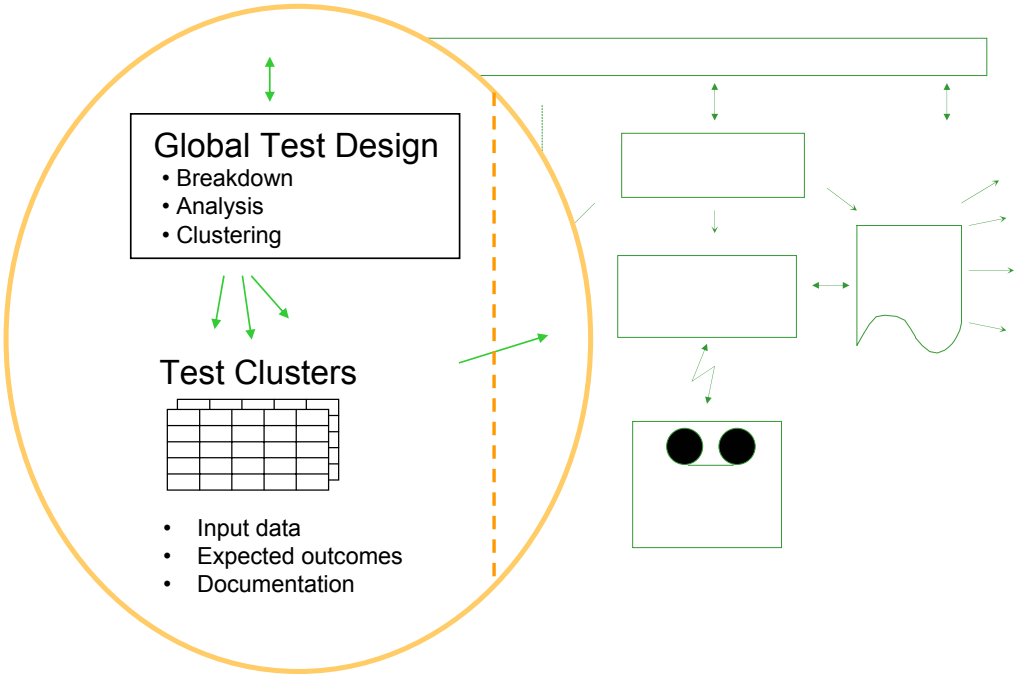
© 2002 LogiGear Corporation. All Rights Reserved.

Designing a Tool Independent Architecture

- For example, ABT sits on top of many test automation tools including, but not limited to:
  - WinRunner
  - QA Run
  - SILKTest
  - Visual Test
  - Rational Robot
  - ATF
  - eValid
  - C/C++, Visual Basic, Java, etc.
  - Various scripting languages

© 2002 LogiGear Corporation. All Rights Reserved.

Test Design Strategy



© 2002 LogiGear Corporation. All Rights Reserved.

Test Design Strategy

- About 70-80% of the work
- Separated from the automation
- The bulk of the work is done at the cluster level
  - Easier to organize and manage
- Separation between "test objectives" and test cases or test scenarios
- Global Test Design is can be organized based on either logical abstraction or physical abstraction

© 2002 LogiGear Corporation. All Rights Reserved.

Test Design Strategy

1. Global Test Design
   - Key product: Cluster identification

Project Level

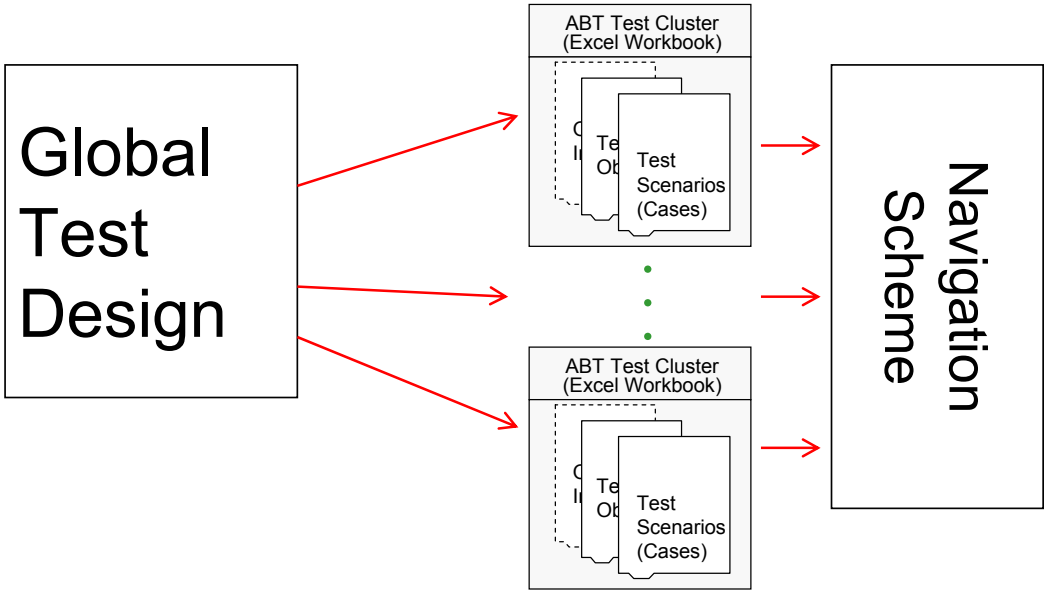
2. Test Analysis
   - Key product: Test objectives

3. Test Development
   - Key product: Scenarios/Cases with actions

Cluster Level

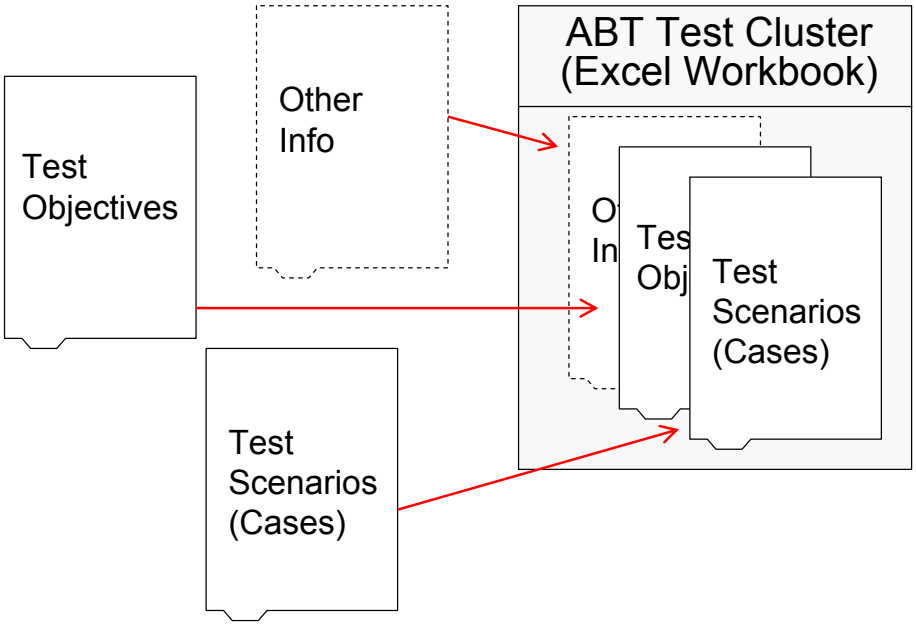
© 2002 LogiGear Corporation. All Rights Reserved.

Test Product Management Strategy



© 2002 LogiGear Corporation. All Rights Reserved.

Test Cluster Build-Up Strategy



© 2002 LogiGear Corporation. All Rights Reserved.

Test Case and Test Objective Referencing

...
TO-3.51 The exit date must be after the entry date
...

test objective	TO-3.51		
	name	entry date	exit date
enter employment	Bill Goodfellow	2002-10-02	2002-10-01
check error message	The exit date must be after the entry date.		

© 2002 LogiGear Corporation. All Rights Reserved.

Clustering Recommendations

- Logical to all concerned
- Independent from other clusters
- Well differentiated and clear in scope
- Fitting the priorities and planning of the project
- Balanced in size and amount

© 2002 LogiGear Corporation. All Rights Reserved.

Examples of Clustering Criteria

STRAIGHTFORWARD

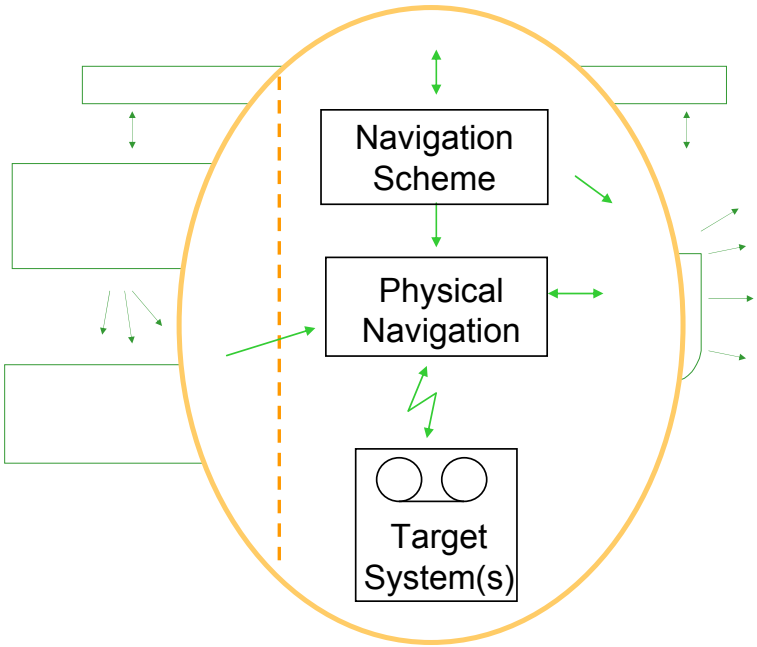
- Architecture of the system under test
- Functionality and other requirements
- Quality attributes
- Level of detail

ADDITIONAL

- Planning and control
- Level of risks involved
- Complexity of the test
- Technical aspects of test execution
- Stake holders
- Code hand-offs (Brian Marick)

© 2002 LogiGear Corporation. All Rights Reserved.

Navigation Engineering Strategy



© 2002 LogiGear Corporation. All Rights Reserved.

Role of Test Execution Tools

- The leading test tools are high quality
- Embedded in suites
- Strong in GUI and Web testing
- Less dominant in technical testing

© 2002 LogiGear Corporation. All Rights Reserved.

Common Components of a Playback Tool

- Recorder of actions
- Check points
- Playback engine
- Test results bookkeeping
- Built-in script language
- GUI handling

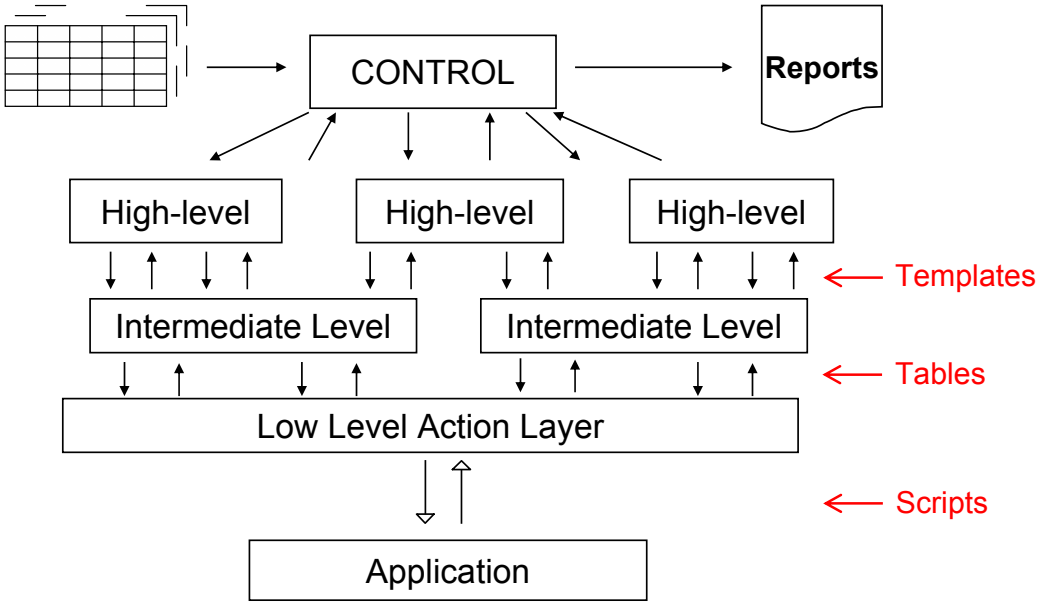
© 2002 LogiGear Corporation. All Rights Reserved.

Scripting Language

- Regular programming language
  - control flow (like if-then-else, loops, etc.)
  - constants, variables and data structures
  - numeric and string expressions
  - functions, subroutines
- Extra: specific testing functions
- Interpreted
  - Easier to debug
  - Slower than a compiled language (but that is usually not a problem)
- Extra facilities to help the non-technical user
  - However, automation remains a programming task

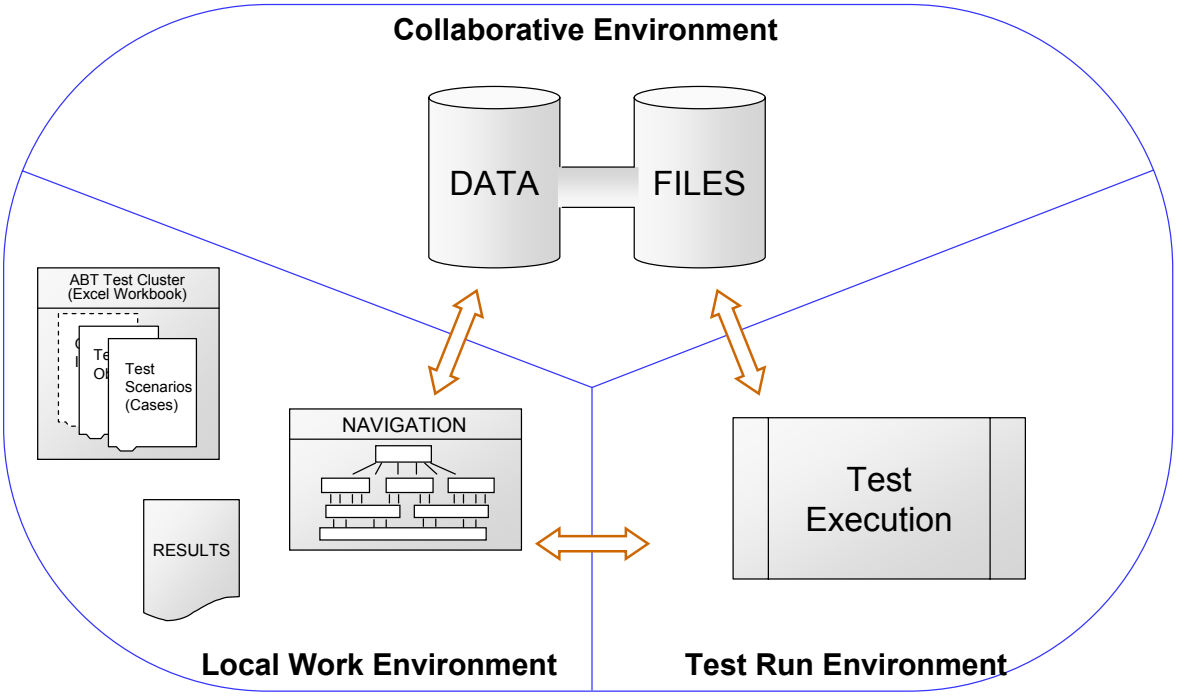
© 2002 LogiGear Corporation. All Rights Reserved.

Multi Level Navigation Strategy



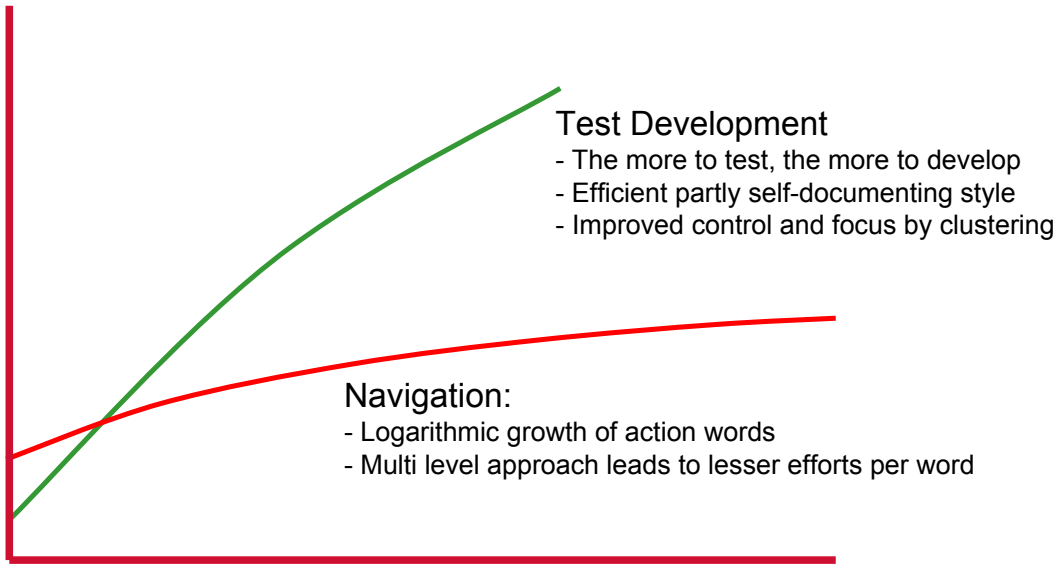
© 2002 LogiGear Corporation. All Rights Reserved.

Test Product Architecture



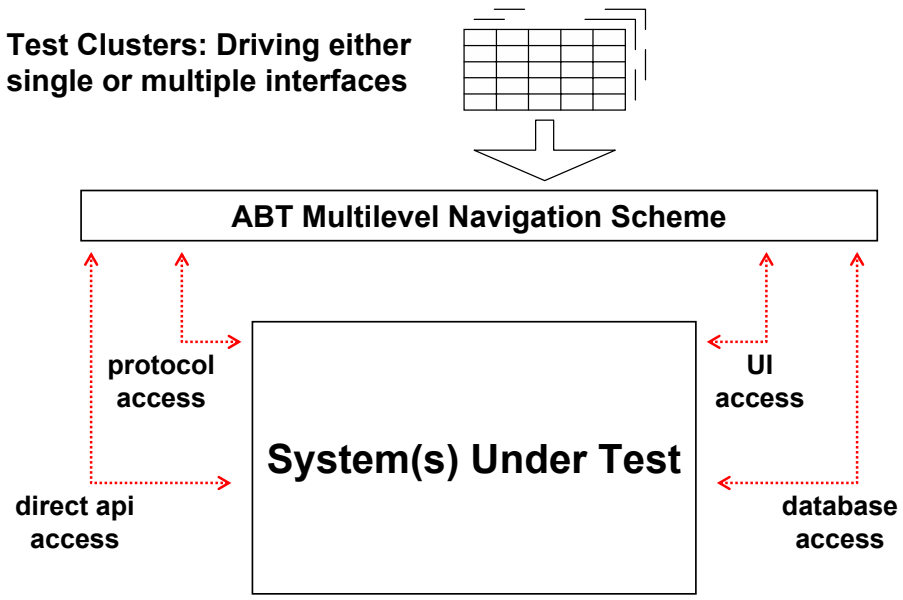
© 2002 LogiGear Corporation. All Rights Reserved.

Typical Division of Labor



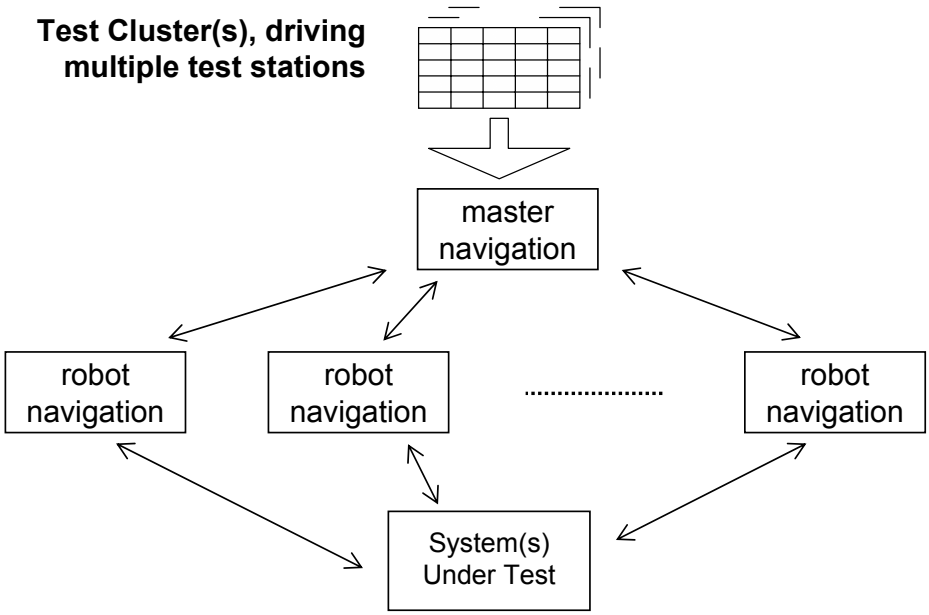
© 2002 LogiGear Corporation. All Rights Reserved.

Multiple System Access Strategy



© 2002 LogiGear Corporation. All Rights Reserved.

Multiple Test Station Distributing Strategy



© 2002 LogiGear Corporation. All Rights Reserved.

Web Testing

- Shooting at a moving target
- Performance is number one issue
- Selection and planning can help
- Use clustering to manage

© 2002 LogiGear Corporation. All Rights Reserved.

There is a Lot of Web Testing

- "Classic" functionality
- Multimedia
- Links
- Compatibility
  - (Combinatorial explosion, see www.argreenhouse.com)
- Client and server scripts
- Databases, middleware, etc.
- Load and performance
- Reliability

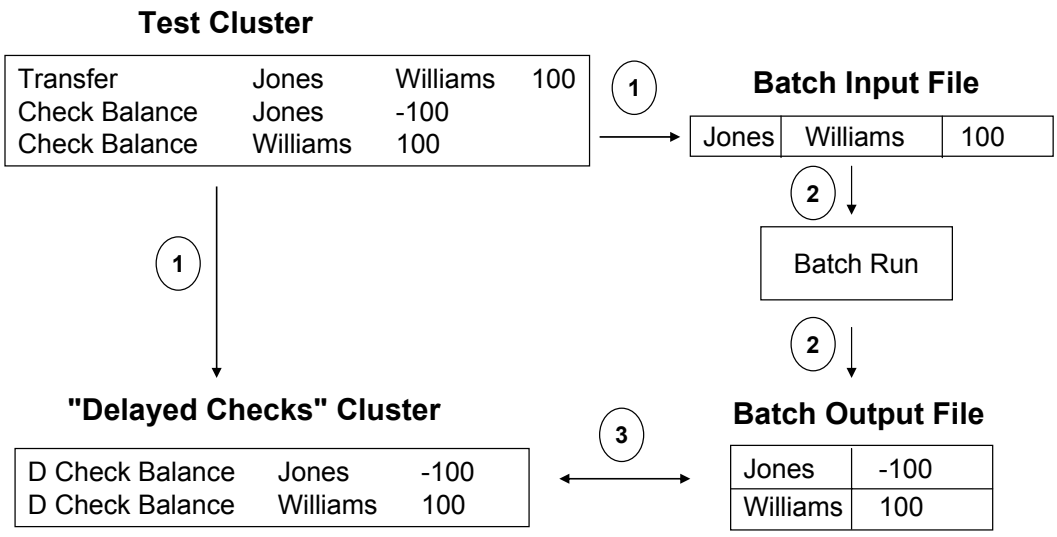
© 2002 LogiGear Corporation. All Rights Reserved.

Web Testing

- Order in the chaos
- Test specification is not particularly hard
- Navigation can be hard, depending on the tooling

© 2002 LogiGear Corporation. All Rights Reserved.

Legacy Batch System Testing



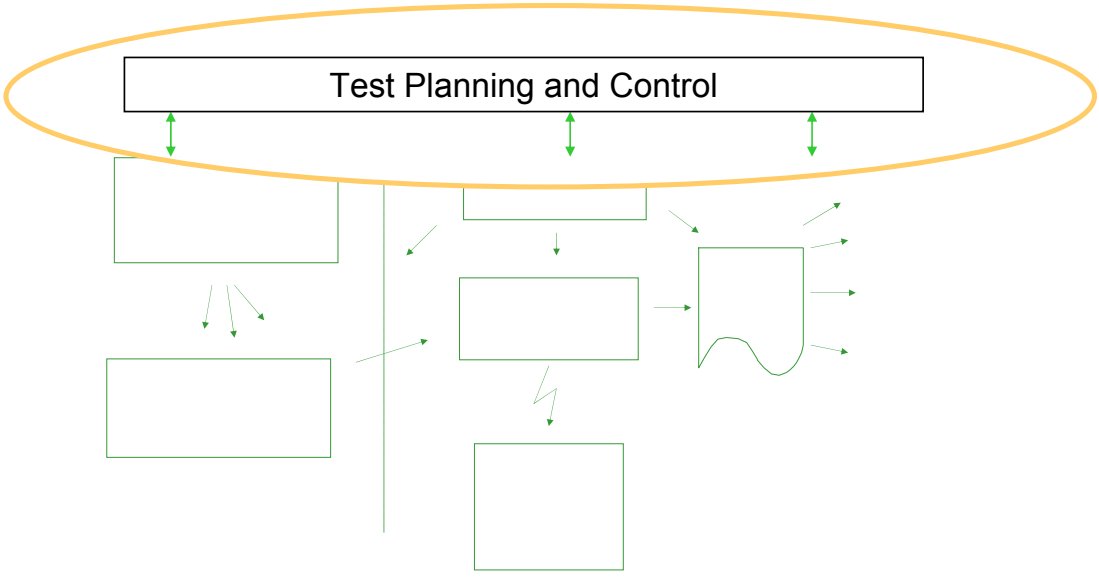
© 2002 LogiGear Corporation. All Rights Reserved.

Performance Testing

- The topic is complex
- To formulate tests in actions is usually surprisingly straightforward
- Navigation can be straightforward as well, depending on your tooling and environment

© 2002 LogiGear Corporation. All Rights Reserved.

Test Management Strategy



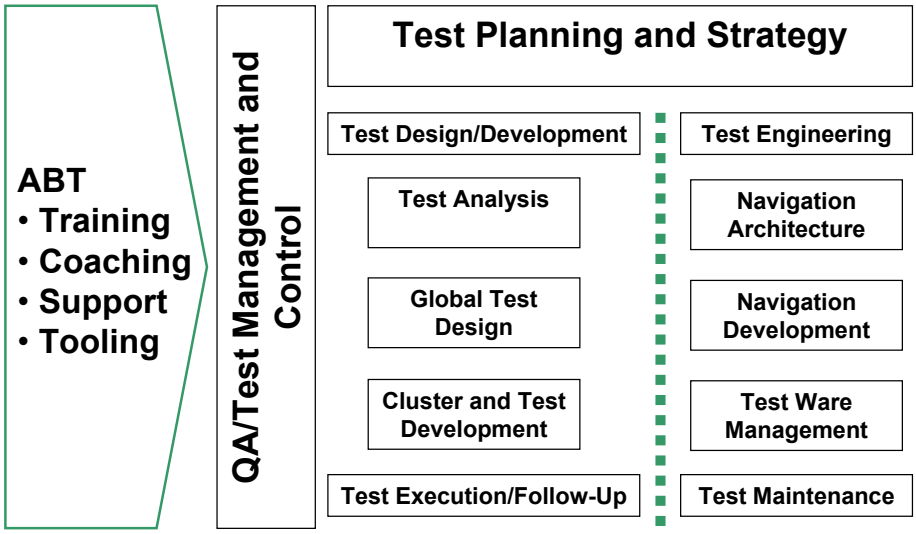
© 2002 LogiGear Corporation. All Rights Reserved.

Key Strategic Issues

- Shorter Time to Market
- Improved Quality to Market
- Better Control
- Commitment

© 2002 LogiGear Corporation. All Rights Reserved.

ABT Project Model



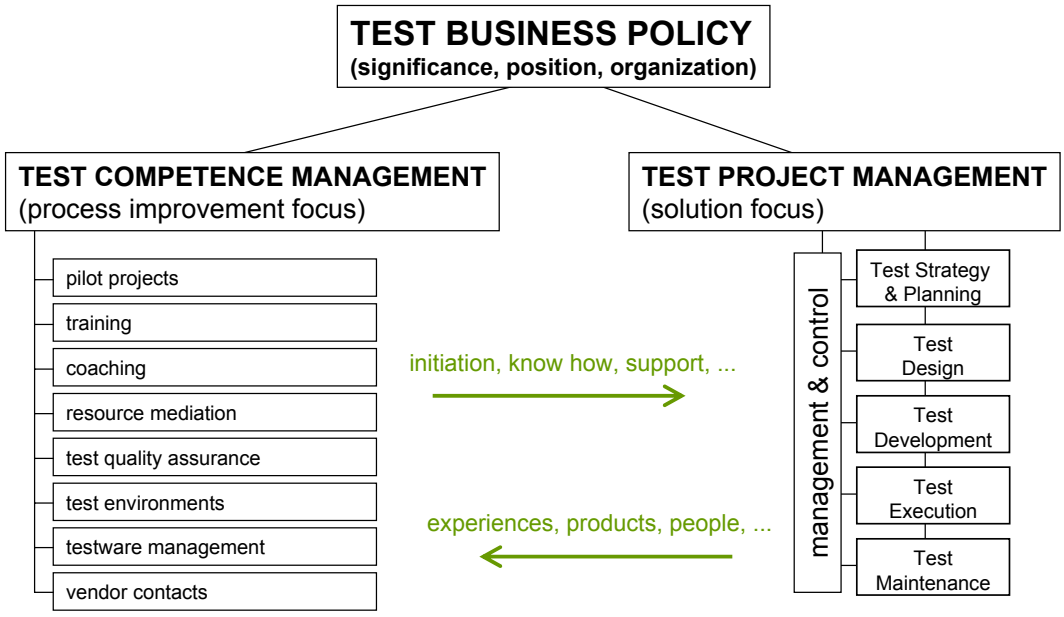
© 2002 LogiGear Corporation. All Rights Reserved.

Assembling the Team

- Within a project
  - Test consultant: setup and coaching
  - Test management: managing the test process
  - Test development: production of the test clusters
    - test leads
    - test analysts
    - end users, business specialists
  - Navigation Engineering: production of the navigation scheme for automatic execution
    - senior navigator / test architect
    - one or more navigators
- General, at the organization level
  - Support on the method
  - Keeping navigation schemes
  - Keeping test clusters

© 2002 LogiGear Corporation. All Rights Reserved.

ABT Test Governance Model



© 2002 LogiGear Corporation. All Rights Reserved.

Getting Commitment for Test Automation

- Testing is often not popular
- Nobody wants an extra problem (No time, next year, ...)
- Offer solutions, not additional problems
- Tell managers that a good tested system creates a positive image (Not only negative reasons for testing)
- Present/show what you're doing (Glass box)
- Try to get clarity about “Test Policy”
- Keep in mind that managers want things to be under control (Give information about progress and results)
- Use outsiders, use books and/or published articles to make your case
- Try to find some bugs

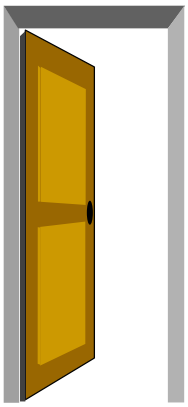
© 2002 LogiGear Corporation. All Rights Reserved.

Getting Commitment for Test Automation

- "No time, no money, ..."
  - Back to the problem
  - You should not become the problem owner!
- "It is so expensive/It is so difficult"
  - Testing is expensive and difficult
  - Test automation is difficult
- "The others should do the testing"
  - Figure this out
  - You can't deal with this yourself
- General vagueness
  - Hidden problems and conflicts

© 2002 LogiGear Corporation. All Rights Reserved.

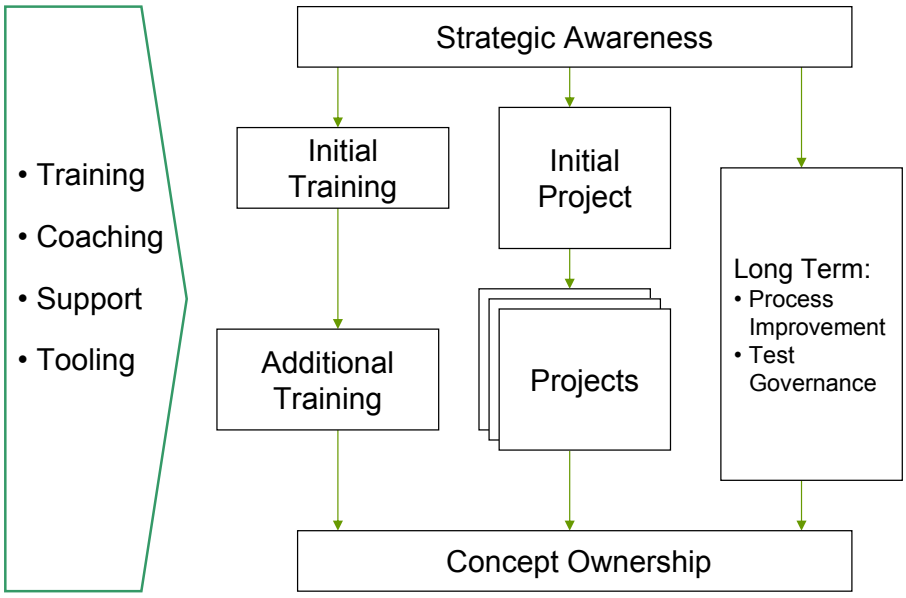
Jump Starting Automated Testing



- Treat it as an organizational change (Although it is not necessarily a reorganization)
- Use pilots, training and coaching
- No standard recipe. It depends on
  - Skills available
  - Experience with testing and test automation
  - Time (and budget) constraints
- Coaching and support is essential
  (There are many lessons to learn)

© 2002 LogiGear Corporation. All Rights Reserved.

ABT Jump Starting Model



© 2002 LogiGear Corporation. All Rights Reserved.

LogiGear and Action Based Testing

- The leading provider of quality engineering strategy and testing services
- The leading provider of quality engineering and test automation knowledge-transferring
- The leading Action Based Testing practitioner
- ABT partnership model offering flexibility in offerings:
  - Training
  - Consultancy
  - Implementation
  - Tooling
  - Outsourcing

© 2002 LogiGear Corporation. All Rights Reserved.

Some References

- Buwalda, Hans, *Testing with Action Words, Abandoning Record and Playback*, Eurostar 1996
- Buwalda, Hans, *Testing with Action Words*, STAR 1998
- Buwalda, Hans, and Kasdorp, Maartje, *Getting Automated Testing Under Control*, STQE Magazine, November Issue 1999
- Buwalda, Hans, *Soap Opera Testing*, STAR East 2000
- Buwalda, Hans, Janssen, Dennis, and Pinkster, Iris, *Integrated Test Design & Automation Using The TestFrame Method*, Addison Wesley, 2001
- Buwalda, Hans, *The Three Holy Grails of Test Development*, Quality Week 2001
- Graham, Dorothy, and Fewster, Mark, *Software Test Automation*, Addison Wesley, 1999
- Kaner, Cem, Bach, James, and Pettichord, Bret, *Lessons Learned in Software Testing*, 2001 (expected)
- Kaner, Cem, Nguyen, Hung Quoc, and Falk, Jack, *Testing Computer Software, 2nd Edition*, John Wiley & Sons, 1999
- Kit, Edward, *Software Testing in the Real World*, Addison Wesley Longman, 1996
- Marick, Brian, *New Models for Test Development*, www.testing.com/writings/new-models.pdf, 1999
- Nguyen, Hung, *Testing Applications on the Web*, Wiley, John & Sons, Incorporated, 2000
- Nyman, Noel, *Using Monkey Test Tools*, STQE magazine, Issue January 2000
- Splaine, Steven, and Jaskiel, Stefan, *The Web Testing Handbook*, STQE Publishing, 2001

© 2002 LogiGear Corporation. All Rights Reserved.

About *Logi*Gear® Corporation

LogiGear Corporation is the first Silicon Valley-based software testing company to offer a full range of solutions to advance individual and organizational excellence in software testing. LogiGear offerings include in-depth technical and management expertise in software quality engineering, comprehensive advanced test engineering such as Action Based Testing™, a structured approach to testing and testing automation, and outsource testing solutions, skill-based training curriculum for software testing professionals through LogiGear University, and world-class testing support products including TRACKGEAR, a Web-based defect management solution.

LogiGear® Expert Testing... Real World Solutions.

www.LogiGear.com

© 2002 LogiGear Corporation. All Rights Reserved.

QW2002 Paper W3

Dr. Subraya BM
(Infosys Technologies Ltd)

Web Performance Testing & Issues

Key Points

- How to prepare for the Performance Testing? : Planning phase
- How to conduct Performance Testing? : Test Execution phase
- How to analyze the results and identify the bottlenecks? : Post-test phase

Presentation Abstract

Performance problems related to Web applications are known to all organizations. Many companies have developed or integrated their critical applications using Web Technologies. As these applications become more complex, testing of such applications is critical, particularly from the standpoint of performance. Performance depends, among other things, upon the load on the system at peak time under varying conditions. To test such complex applications requires proper planning, which includes business benchmarking and profiling, setting up environment and testing process. Performance testing is normally conducted in a simulated environment with the help of performance testing tools. Such testing process will have many practical issues to address. This tutorial provides an in-depth study of how to plan and conduct performance testing in an automated environment. Also, the tutorial provides how to analyse the performance results to find the bottlenecks and provide remedies through a case study.

About the Author

Dr. Subraya B.M. currently working as Senior Principal Consultant to Education and Research Department of Infosys Technologies Limited, Bangalore, India. Before he comes to Infosys, he was working as a Professor and head of Computer Centre, at S.J. College of Engineering, an affiliated college of University of Mysore, a reputed engineering college in southern part of India, since more than 20 years.

Dr. Subraya holds PhD from Indian Institute of Technology, Delhi from Computer Science & Engineering Department, in the area of Testing and Verification. He has guided many projects at graduate and under graduate level. His area of interest includes Software Engineering, Operating System and Distributed databases.

He is currently responsible for managing the Web Performance Testing Centre of the Organization. He has offered tutorials on Web Performance Testing in many pre-conference Tutorials including QAI, SEPG, and PSQT international Conferences. He has published many papers in international conferences in the area of Performance Testing of Web applications.

Performance Testing & Issues: Web Applications

Dr. Subraya BM
subraya@infy.com
Infosys Performance Testing Centre(InPTC)
ITL Infosys, Bangalore
www.infy.com

INTRODUCTION

Please introduce Yourself
What you Expect from this Tutorial!

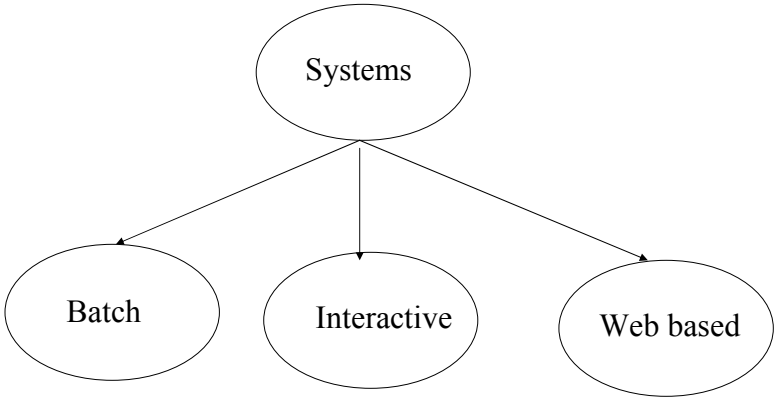
Session Plan

- **Performance in General**
  (Introduction, definitions, objectives, need for Performance testing)
- **Issues in Performance Testing**
  (Quality characteristics, Architecture, Performance Peculiarities)
- **Preparation for Performance Testing**
  (Definition Phase, Design and Build Phase)
- **Performance Test Execution**
  (Setting parameters, execution, analyse results)

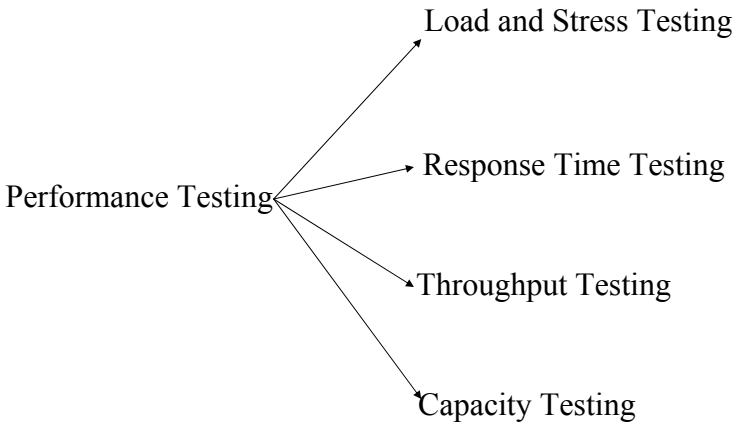
What is Performance Testing?

- Performance means the capacity to function(well)
  (as in The Cassell – concise Dictionary)
- Performance means
  - response time
  - throughput
  - resource utilization
- Performance Testing is not Features Testing

What kind of Systems?



Performance Testing: Different Views



How do You Conduct PERFORMANCE TESTING ?

- Similar to Black box testing
- Looks like an end user to the system under test
- Performed by recording or scripting the actions

Myths on Performance Testing

• Performance problems can usually be fixed by simply plugging in a more powerful Processor, …	**Bulldozer Approach**
• If features work correctly, users do not mind a little slow	**Qualitative Management**
• No elaborate plans are required; it is intuitively obvious how to measure the system's performance	**Simplistic Approach**
• Needs few hours to check performance before the deployment	**Unstructured Approach**
• Do not required expensive tools; few users can play the role of the tools	**Testing after Thought**
• Anyone can measure and analyze the performance; does not require any specialized skills	**Layman Approach**

Need for Performance Testing

Survey Speaks itself

Performance Testing Practices	Organization in which Operational Performance	
	Was Acceptable	Was not Acceptable
Reviewed or simulated (Performance during RA, and design phases)	21%	0%
Test early in development	35%	38%
Test late in development	38%	26%
Did post-deployment testing	0%	8%
Did not do performance or load testing at all	6%	60%

Ref: Computer world 1999(survey results of 117 organizations)

PERFORMANCE TEST OBJECTIVES

- Support of maximum number of concurrent users(for proposed configurations for acceptable performance)
- Assess performance of current configurations
- Identify location of bottlenecks within the application architecture
- Impact of a software or hardware change
- Address Scalability issues

Bottom Line: How do you achieve?

Performance Testing : Definition

- **Measured as a weighted mix of three characteristics of a system:**
  1. **Throughput**
  2. **Response**
  3. **Availability**

- **Carried through:**
  - **Load Testing : Measurement of performance under heavy load: peak or worst-case conditions**

  - **Stress Testing : Deliberate stress on a system beyond specified limits**

Variations Of Performance Testing

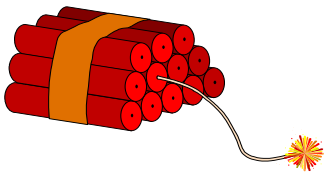
Test for

- Load
- Endurance
- Stress
- Spike

Test-LESS: An approach to ensure complete Performance Testing (LESS is More!)

Poor Performing Systems Will Cost You...

- Resources
  - Time
  - Money
  - Staff
- Profits
- Customers



Benefits of Performance Testing

- Improved quality from a user's perspective
- Reduced cost of change
- Reduced system costs
- Increased profits
- Early identification of major application defects and architectural issues

Cost Per Defect		
1	10	100
Design	Test	Production

Discussions

1. **What are the important objectives of Performance Testing?**
2. **Why the load testing cannot be conducted manually?**
3. **Specify at least two reasons why Performance testing is mandatory ?**

Discussion on Case Study

Study the banking case study and identify the performance requirements, if any.

Issues to be Addressed

- Client/Server vs. Web Applications
- Setting the Quality
- Understanding the Architecture
- Testing Types and Issues
- Addressing Peculiarities

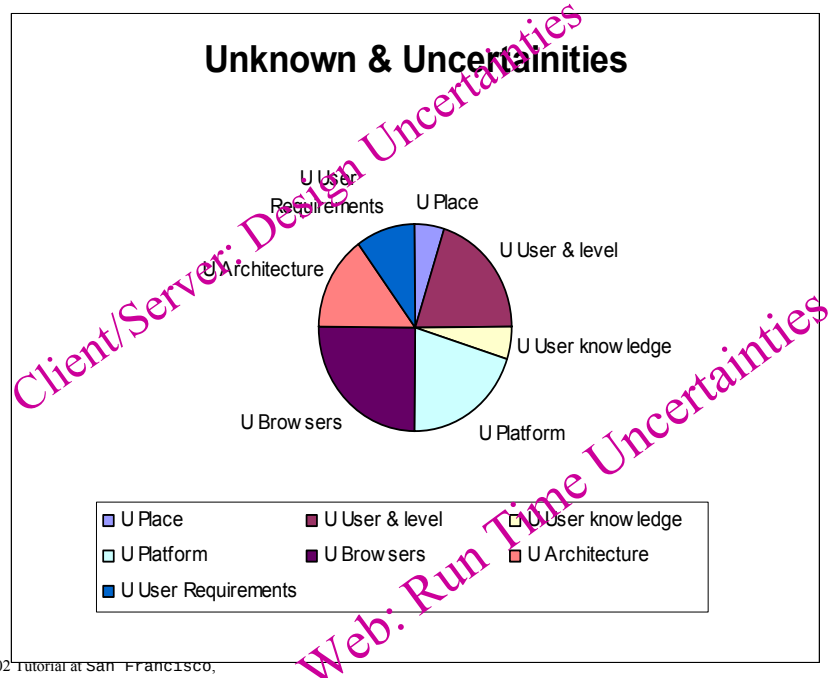
Client/Server vs. Web Applications

CLIENT/SERVER vs. WEB APPLICATION TESTING

- Web is relatively a new area compare to C/S
- There is a difference in testing Client/ Server and Web Applications
- Automated testing of Websites is an opportunity and a Challenge

C/S Application vs.Web Site ?

- **How do you differentiate Web Application & typical Client/Server Application?**
- **Who are all the clients for an Web Site compare to C/S?**
- **Can you suggest HW/SW for a Web client?**



Client/Server: Design Uncertainties

Web: Run Time Uncertainties

Client/Server vs. Web Applications

What do you Expect?

We Expect:

- Client friendly
- Most Reliable
- Performance Conscious

Bottom Line: How do you achieve?

Setting the Quality

Quality ?

Bottom Line: How do you set Quality aspect?

Quality Characteristics

Setting the Quality

Types	Client/Server	WEB
Functionality	C	C
Reliability	C	C
Efficiency	C	C
Usability	C	C
Portability	M - C	L – M
Maintainability	M	M
Extensibility	M	C
Scalability	L - M	C

Legend: Critical(C) , Moderate(M), Low(L)

Impact Of Quality

Setting the Quality

- Poor Quality
  - broken pages and faulty images, CGI-Bin error messages, user experience, etc(WEB)
  - Impact on Client leads to cost/effort(C/S)

HOW to ENSURE QUALITY AUTOMATICALLY?

Setting the Quality

- Design Test suites
- Plan Test sessions
- Conduct set of repeatable tests automatically
- Analyse Test Results

WHAT TO TEST?

Setting the Quality

- C/S Applications: Client + Server
- Web Applications: Network + Server

Bottom Line: Be clear, which side testing produce effective results?

Discussions

1. **What are the essential differences between Web Testing and Client/Server Testing?**

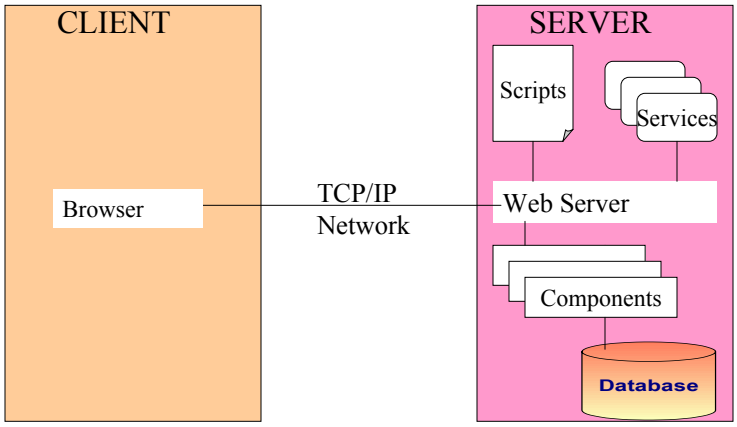
2. **What is the most relevant quality characteristics that you have found in your project?**

Understanding the Architecture

Web Technology

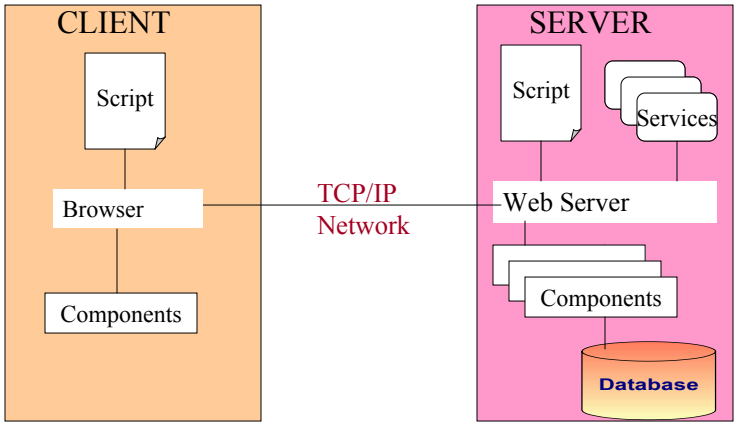
A Web-based Thin client

Understanding the Architecture

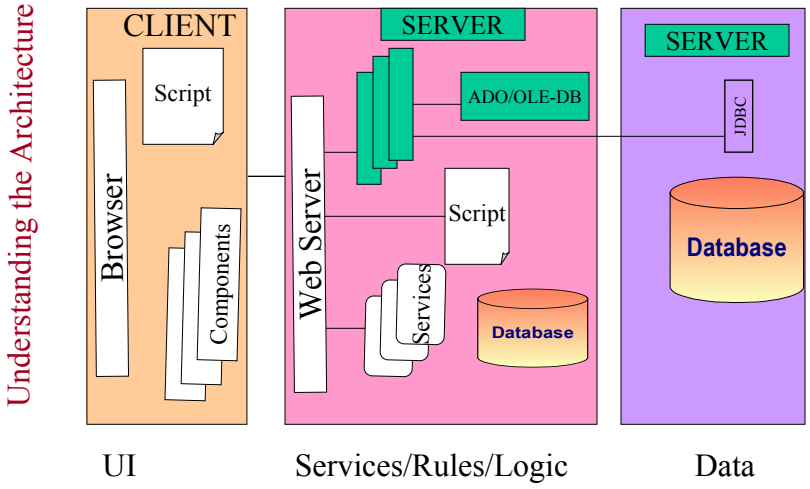


A Web-based Thick client

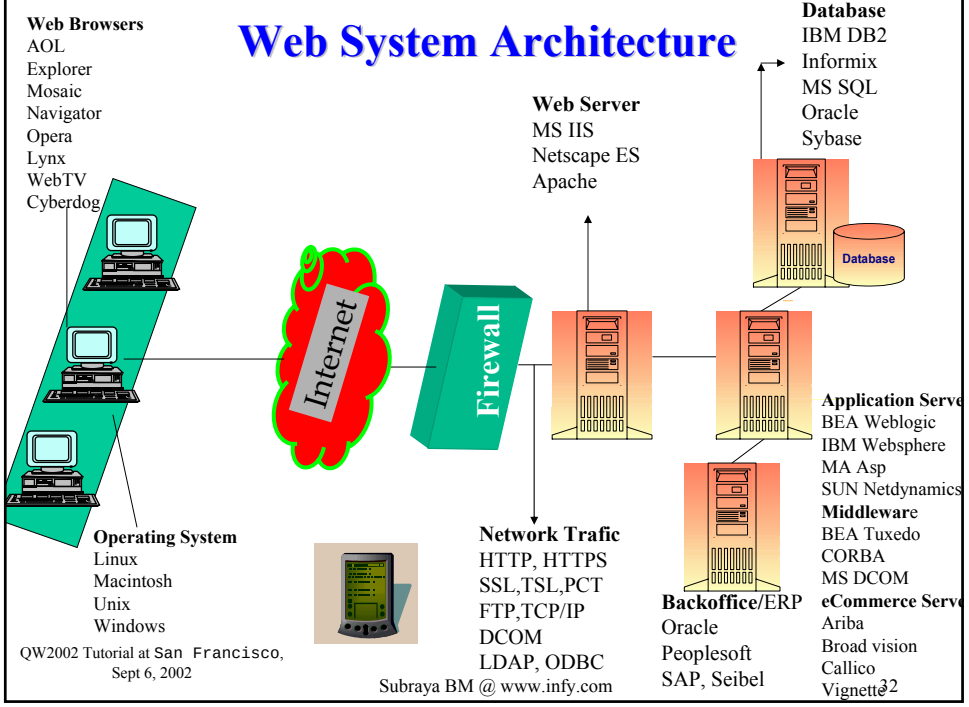
Understanding the Architecture



Three-tiered Web-based System

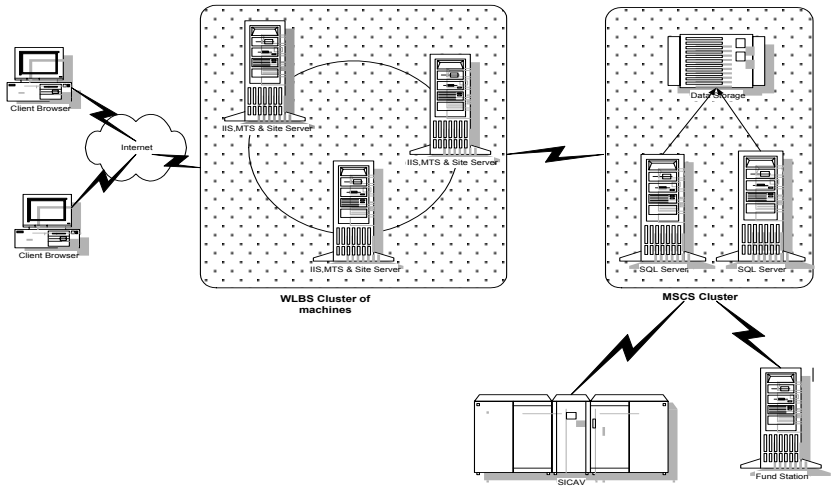


Web System Architecture



Typical E-Architecture

Understanding the Architecture



Types of Testing

Testing Types and Issues

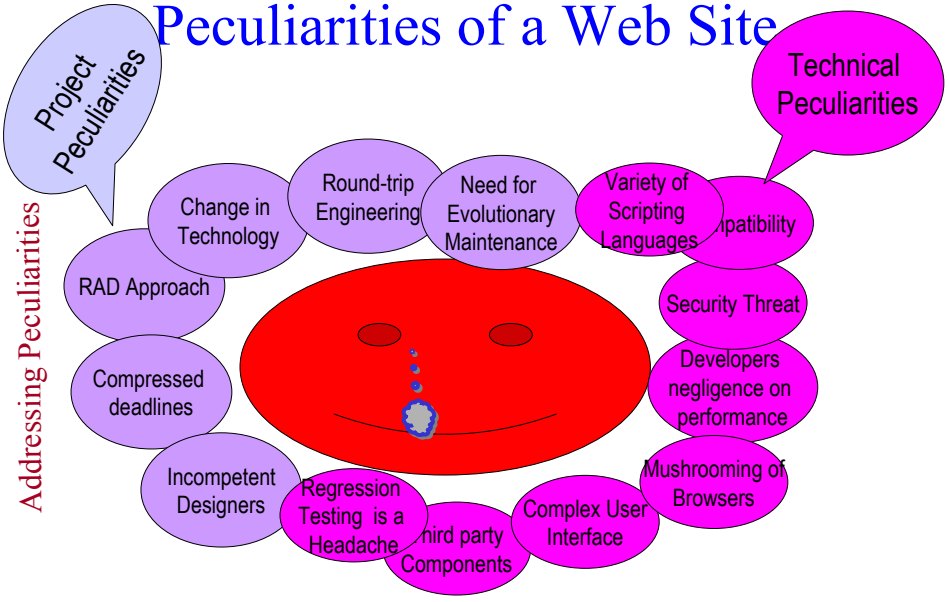
- Business Related functional Testing
  - Unit and integration Testing
- Client Related functional Testing

Client Related functional Testing

Testing Types and Issues

Testing Types	C/S	Web
Code related Validation	Yes	Yes
Navigation	Yes	Yes
Compatibility	No	Yes
Web Page	No	Yes
Frames	No	Yes
Forms	Yes	Yes
Presentation	Yes	Yes
Usability	Yes	Yes
Reliability & Recovery	Yes	Yes
Security	No	Yes
Animation	No	Yes
Performance	Yes	Yes

Peculiarities of a Web Site



Bottom line: How do you test such a complex environment?

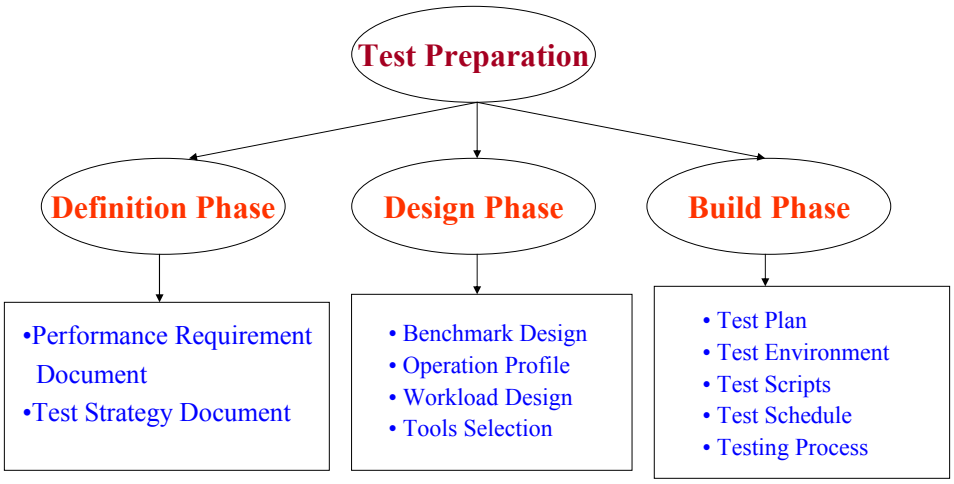
Possible Strategy

- **Understand the Performance concept of Web Applications**
- **Define Performance goals unambiguously**
- **Develop Workloads based on Business Benchmarking**
- **Prepare proper Test Plan and Schedule**
- **Use suitable Environment for Testing**
- **Analyze Results & use good feedback mechanism for improvement**
- **Ensure the Reliability**

Discussions

1. **Does the Technology influences the performance of the system?**
2. **Should I understand the architecture before working on performance Testing?**
3. **Did you face any compatibility issues with systems from multi-vendors?**
4. **Do you conduct performance testing with out a test plan?**
5. **Whether Performance testing can be done prior to the functional testing?**

Preparation for Performance Testing

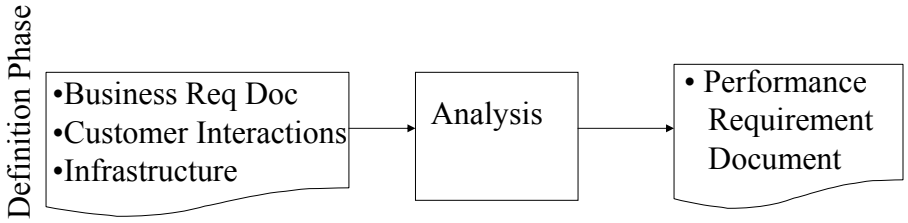


Performance Requirements Document

Definition Phase

- **Business Related**
  - Understand both C/S and Web applications business plans
    - Business Background
    - Business Operations
    - Business Objectives for the System
    - Logistics of Operations
    - Business Growth Projections
    - Infrastructure
- **Proposed System Related**
  - Functions of the System
  - Interfacing System, If any
- **Service level Agreements**
- **Management Conditions and Constraints**

How to Prepare Performance Requirement Document?



Exercise I

Study the requirements of the case study given to you. From the case study, list the following :

- **What are the performance objectives of the proposed system?**
- **How do you organize the performance requirements? List them properly.**

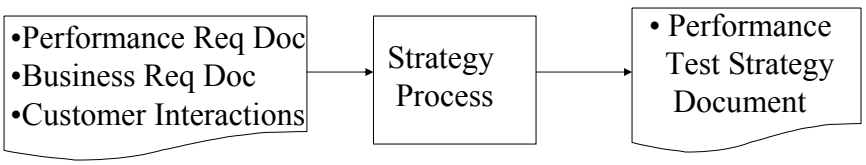
Build Strategy Document

Definition Phase

- **Performance Test Related**
  - Identifying Performance Test Objectives
  - Defining Performance pass/fail criteria
  - Identifying measurement criteria
  - Operation Profile
  - Analyzing Test approaches and selecting best one
  - Responsibilities
  - Automated Testing facilities

How to Build Strategy document

Definition Phase



Exercise II

Study the strategy document given to you. Answer the following:

1. **Identify the performance objectives listed in the document**
2. **List the pass/fail criteria which, you think, is most critical**
3. **What is the scope of the performance testing?**
4. **List the key transactions for effective performance testing**

Benchmarks: What and Why

Design Phase

- What is a benchmark?
- Domain specific
  - No single metric possible
  - The more general the benchmark, the less useful it is for anything in particular.
  - A benchmark is a distillation of the essential attributes of a workload
- Desirable attributes
  - Relevant ➔ meaningful within the target domain
  - Understandable
  - Good metric(s) ➔ linear, orthogonal, monotonic
  - Scaleable ➔ applicable to a broad spectrum of hardware/architecture
  - Coverage ➔ does not oversimplify the typical environment
  - Acceptance ➔ Vendors and Users embrace it

Benefits and Liabilities

Design Phase

- Good benchmarks
  - Define the playing field
  - Accelerate progress
    - Engineers do a great job once objective is *measurable* and *repeatable*
  - Set the performance agenda
    - Measure release-to-release progress
    - Set goals
    - Something Users/managers can understand (!)

Benchmark Design Issues

Design Phase

- A representative workload of a real life Scenario used for the Performance Testing
- More than one benchmark is required for better Performance Results
- Consider mix of demands on the system and the frequency of occurrence of the various demands
- Complexity depends on the Size of the Benchmark
- Select transactions with more concurrency and interacting with others
- Consider Background Noise during Benchmark design

Benchmark Standards (Web)

Design Phase

- **TPC-W (Web) is a transactional web benchmark.**
- **TPC-W models a controlled Internet Commerce environment that simulates the activities of a business oriented web server.**
- **The application portrayed by the benchmark is a Retail Store on the Internet with a customer browse-and-order scenario.**
- **TPC-W measures how fast an E-commerce system completes various E-commerce-type transactions**

TPC-W Characteristics

Design Phase

- The simultaneous execution of multiple transaction types that span a breadth of complexity.
- On-line transaction execution modes.
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationship.
- Multiple on-line browser sessions.
- Secure browser interaction for confidential data.
- On-line secure payment authorization to an external server.
- Consistent web object update.
- Transaction integrity (ACID properties).
- Contention on data access and update.
- 24x7 operations requirement.

TPC-W Work Profile

Design Phase

- There are three workloads in the benchmark, representing different customer environments.
  - Primarily shopping (WIPS). Representing typical browsing, searching and ordering activities of on-line shopping.
  - Browsing (WIPSB). Representing browsing activities with dynamic web page generation and searching activities.
  - Web-based Ordering (WIPSO). Representing intranet and business to business secure web activities.

Number of Benchmarks

Design Phase

- Separate benchmark is needed for each significant and distinct work load
- A typical project employs five benchmarks:
  1. Routine Business workload
  2. Routine Peak demand
  3. Specialized requirement
  4. Top managements specialized demands
  5. Projected growth based demand

Discussions

1. **Discuss different possible benchmarks for the given Banking System?**

2. **Design a TPC-W equivalent benchmark for the ABC banking System**

Operation Profile

Design Phase

- Lists the major demands on a system or likely demands of the system actually occurring in live operations
- Provides a usage patterns of the expected system
- Requires Business Knowledge to prepare the operation profile

Characteristics of Operation Profile

Design Phase

- Target Users, their demographics and number of users
- How the target users use the proposed system
- Target time of usage
- Target Locations
- Rates of utilization of set of Activities/Scenarios

Operational Scenarios

Design Phase

- Reflects a performance business goals
- Attributes
  - Identification of Key Transactions
  - Identification of Transaction Mixes
  - Selection of Data and Size
    - Small data volume
    - Large data volume
- Determine number of Operational Scenarios

What is a Workload ?

Design Phase

- An instrument required to simulate the real world environment
- Represents the user groups and activity types
- Needs a benchmark of typical business to formulate a workload
- There may be several workloads for a Project

Characteristics of a Workload

Design Phase

- Selection of type of transactions
  - Business Domain Specific
  - Resource hungry
  - Concurrency
  - Interacting
  - Infrastructure related
- Grouping of Transactions vs. Users group
- Weightage to each transactions
- Sequencing of transactions within a user group

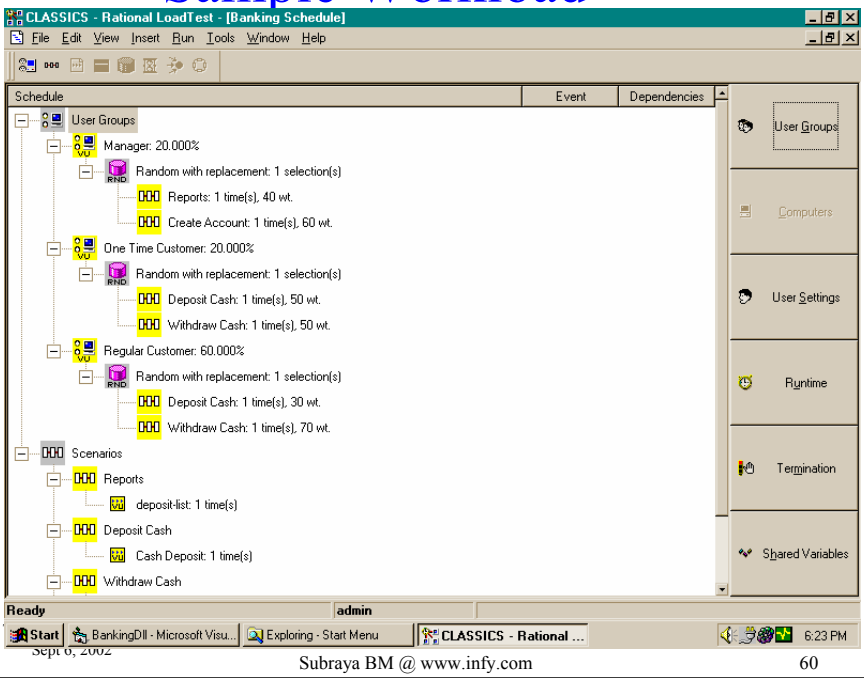
Workload Planning

Design Phase

- Identify User Groups
- Identify Representative subset of users.
- Develop Activities/Scripts (Test cases).
- Consider whether any batch jobs need to be started manually in the background.
- How long should run last?
- Identify the different times of day.

Sample Workload

Design Phase



Selection of Test Automation Tool

Design Phase

- Depends on goals and Performance Measurements
- Cost of the tool
- Ability to use virtual databases users
- Support of database connectivity methods (ODBC, RDO, DBLib, etc,)
- Types of protocols support

Discussions

1. **What are the parameters that determines the accuracy of the performance testing results?**
2. **Design a simple workload for the given ABC banking system**

Performance Test Plan

Build Phase

- Plan Identification
- Description of the system being tested
- Performance goals of the System
- Performance Test Objectives & Success Criteria
- Test Constraints
- Performance Measurements
- Operational Profile(s)
- Test Environments
- Test Execution Process
- Test Schedule

Plan data

Build Phase

- How much data?
- Which data?
- How do I generate it?
- How do I reset it (- and do we need space to store copy)?
- Do I have enough disk space on the driver machine(s)?

Plan Team

Build Phase

- Project manager
- Real users for recording
- Scripter/Scheduler
  - Should be technical with the ability to change a program
- DBA should be available
  - Need passwords
  - Restore/backup
- Sys Administrator
  - Install products
  - Restore/backup
- Network Administrator

Plan Test Environment

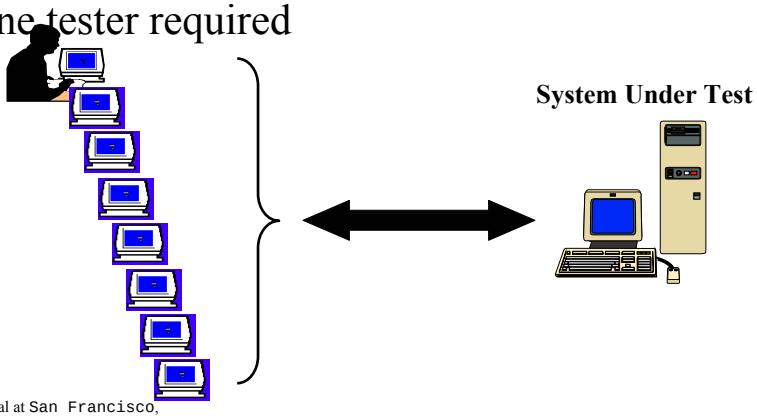
Build Phase

- Replicate the real world operating environment as much as possible
- Check whether multiple environment is possible
- Choose appropriate configuration for the performance testing
- Avoid shared environment as for as possible

Hardware-Based Testing

- Many physical computers required
  - One per test
- One tester required

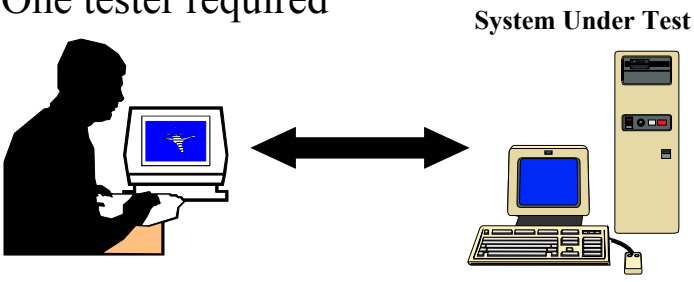
Build Phase



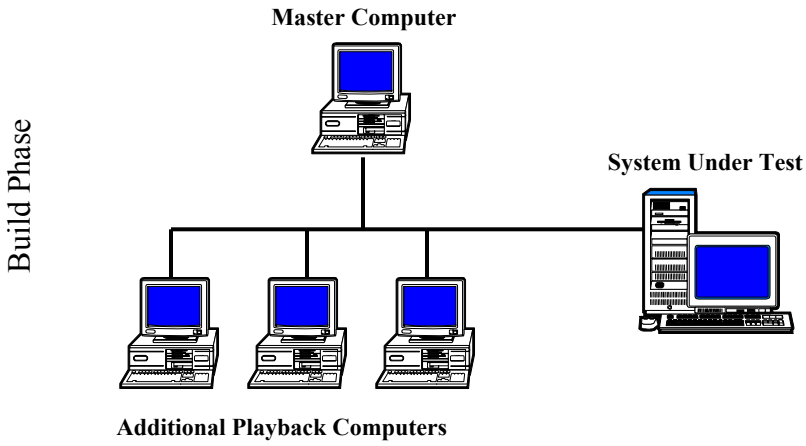
Software-Based Testing

- At least one physical computer emulating many users
- One tester required

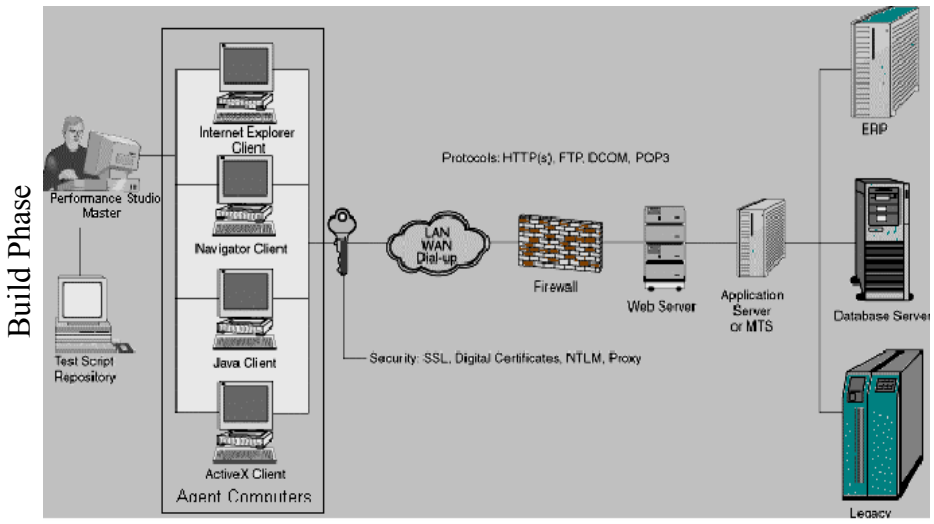
Build Phase



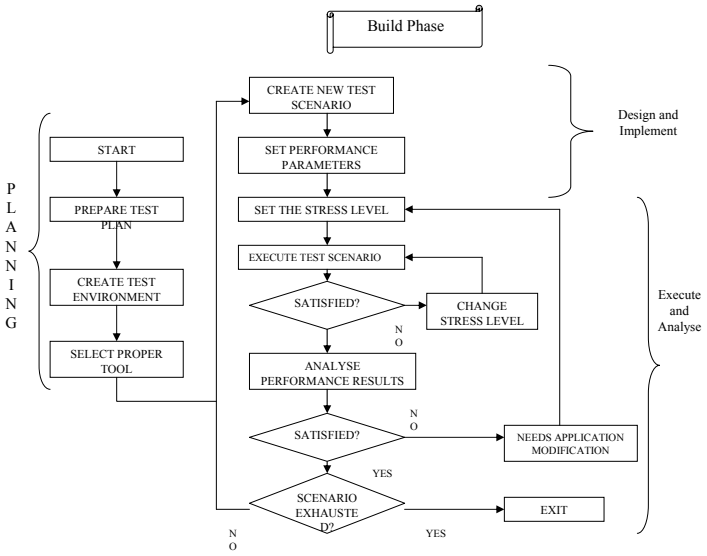
Combined Hardware and Software-Based Testing



Typical Testing Environment(WEB)



Testing process



Test Execution Phase

- Entry/ Exit Criteria
- Elaboration Testing
- Issues in Test Schedule
- Issues in Test Execution
- Analyze results
- Address feedback mechanism

Entry Criteria/ Pre-requisite

- System under test must be functionally error free
- System development must be completed

Exit Criteria

- Once completed the number of runs planned
- In each run, once a specified duration reaches
- When it reaches the required number of virtual users
- When the performance measurement matrix is satisfied

Issues in Test Schedule

- Scripts must be ready before the schedule
- In C/S, clients must be initialized before the schedule
- Sequence the schedule with different configurations
- Ensure the Performance Test teams availability

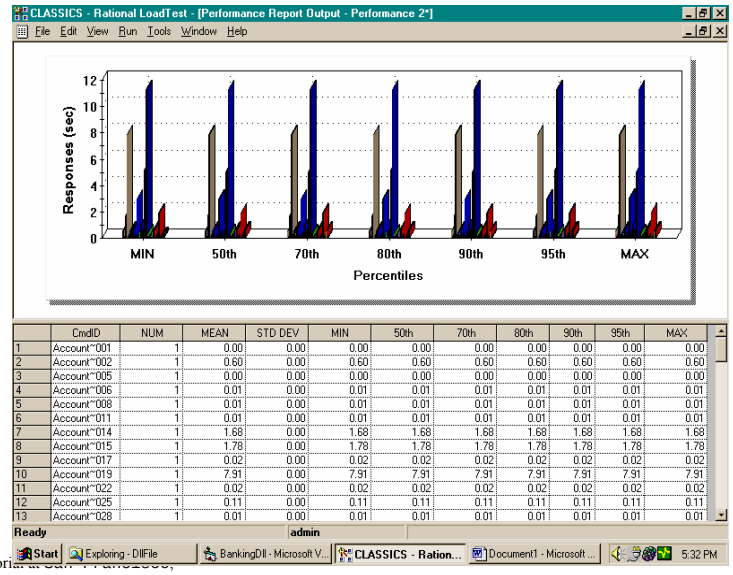
Issues during Test Execution

- **Run Performance testing more than once with different levels of background activity(noise) to get the true picture**
- **Before start testing, check for following Test entry criteria**
  - **All major business functions and features are present and working**
  - **All major defects have been fixed and re-tested**
  - **Test environment is ready and test it properly**
  - **Test workloads and related tools are ready**
  - **Proper configuration management and change control are in place**
  - **Test Schedule be ready**

Issues during Test Execution(contd)

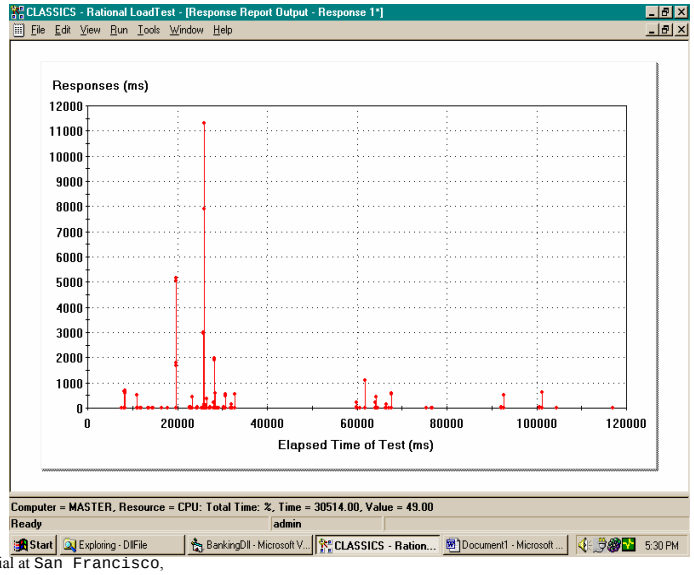
- Conduct Exploratory Test
- During Execution
  - Monitor performance counters
  - Monitor test duration
  - Avoid external interference
  - Monitor the test execution by probing client
- Log the results with timings

Sample Results
(generated from Rational Test Manager)



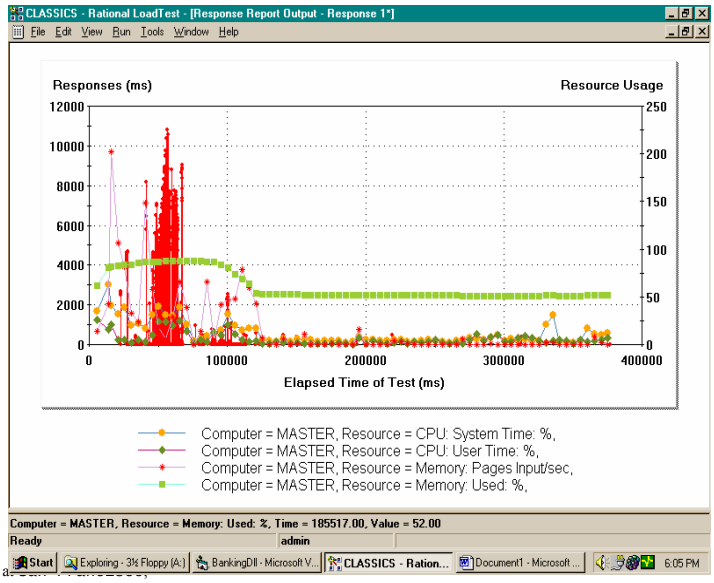
Sample Results

(generated from Rational Test Manager)



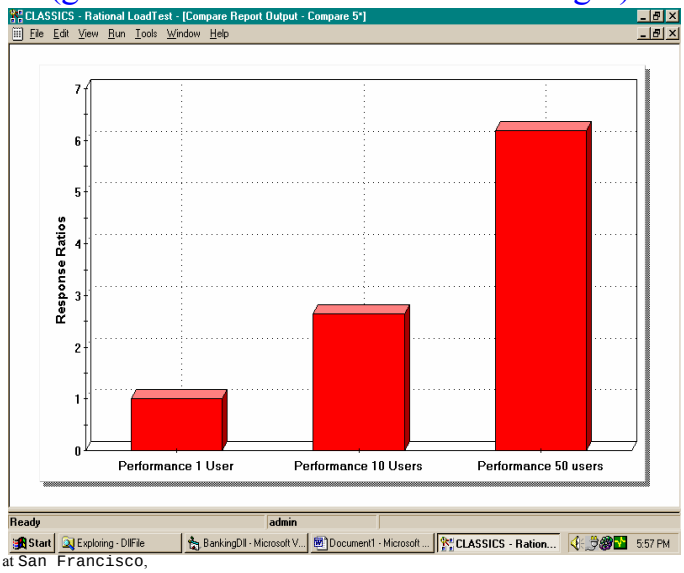
Sample Results

(generated from Rational Test Manager)



Sample Results

(generated from Rational Test Manager)



Analysis of Results and Reporting

- Who will be using the results
  - Business managers, Marketers, Sr. management, etc,.
- Critical success factors for each category
- Representation of results to different stakeholders
  - Graphs, Tables, abstracts, bottlenecks isolation, ….
- Follow guidelines for analysis

General Guidelines for Analysis

- Identify Resource utilization and bottlenecks with any resource
  - In C/S, clients resource utilization is also important
- Identify script level problems
  - script level response time, resource usage, …
- Isolate excess queue lengths at the server, if any.
- Address code level optimization at the end

Load Simulation

- Single Browser for both C/S and Web
  - Mostly in C/S, single browser is targeted
- Multiple Independent Browsers
- Multiple Coordinated Browsers

Suggestion For Load Testing

- Identify the system components
- Describe each configuration
- Create a chart of use cases
- Develop load-testing objectives for acceptance criteria
- Allow time for script development
- Perform proof-of-concept to see if the tool works with the application

Performance Issues

- Poor understanding of User Perceptions
- Complexity in Developing Usage Profiles
- Difficulty in setting up the environment
- Expensive testing tools: ROI not well understood
- Performance Testing is a late activity in the life cycle
- Difficulty in selecting performance parameters

CONCLUSION

- **Functional Testing is a must before addressing performance issues**
- **Plan for performance testing from the beginning**
- **Prepare a strategy and a method for performance Testing**
- **Select suitable automation tools for Web Performance Testing**
- **Select proper environment for Web applications**
- **Give more importance to benchmarking and workload**
- **Analysis of results needs experienced skill set**

Reference

- Capacity Planning and Performance Modeling by Daniel a. Menasce etal
- The Web Testing Handbook by Steven Splaine and Stefan p. Jaskiel
- www.softwareqatest.com
- www.tpc.org
- www.w3c.com
- www.sof.com

Thank you

subraya@infy.com

Please give/send your valuable feedbacks

ABC BANK

Brief Business Requirements Document

Name:	QW2002 Tutorial
Application:	Performance Testing Automation
Author:	Dr. Subraya BM(subraya@infy.com)
Date:	6-Sep-2002

A Simplified Banking System

ABC Bank is a global banking organization that provides standard banking services to its customers spread across the globe. The vision of the bank is **anywhere banking any time**. The head office is located in Chicago with several hundred centres situated outside USA. The aim of this proposed banking system is to create a paperless Bank. Both the bank employees as well as account holders will use the proposed system through the web-based interface. However, majority of the bank employees prefer to use the system in a Client/Server(C/S) environment. The bank has several branches in different cities; each branch is identified by an n digit code.

Types Of Accounts:

1. The Bank Provides Current, Cash, Savings and Loan Accounts to its Customers
2. The Bank will maintain Office Accounts internally for Double Book Entry system of Accounting
3. Inter Bank and Foreign Bank accounts will be maintained by the bank to facilitate clearing of cheques of different banks, countries and currency.

Opening an A/C:

1. Account Number should be generated by system in the ascending order.
2. The system should record the name and address of the A/C holder. These are mandatory.
3. The minimum amount to open an A/C is $500/-.
4. Each Account Number Starts with '100' and is followed by the n digit branch code
5. Each user gets a user name and password that he can use to login from web

Operating an A/C:

Deposits:

1. A deposit should be for a minimum amount of $100/-.
2. Any Amount Greater than equal to $100 is valid provided it is with in the normal range.
3. Whenever a deposit is made, the final balance should be made known to the A/C holder.
4. The account may be operated using a web interface
5. Fund Transfers, DD Request, Password Changes, Account Statements, Request for Cheque Books etc may be obtained using web interface

Withdrawals:

1. Minimum amount of withdrawal is $100/-.
2. A/C balance should never be less than $500/-.
3. Whenever a withdrawal is made, the final balance should be made known to the A/C holder.
4. Withdrawals may be made from any of the branches or through ATM outlets as designated by the bank
5. Using ATMS should be possible irrespective of a centralised or distributed database and the ATMs must be immune to link failures and should be able to operate in both online and offline modes

Loans:

1. Bank will provide loans to its account holders charging a higher rate of interest than that given by the bank to saving and current account customers.
2. Loans that are not paid as per the schedule will attract a fine.

Closing an A/C:

1. The Record will be deleted permanently on closing.
2. All Accounts that have not been operated for 10 years will be considered as dormant accounts

Other details:

1. The Account Numbers should be generated beginning with 100
2. Account number should be generated in sequence.
3. It should not be possible to modify the Account Number at any stage.
4. An enquiry option should be provided, so those users can see the Account Number, name and balance sorted either by Account Number or by A/C holder's name.

The banking system is divided into three modules:

1. **Accounts Maintenance:**

   This module is used to perform the operations on accounts such as opening an account, closing of existing account, Navigation through accounts etc. Some of the characteristics of the module are:

- The module is displayed through a form.
- When the form loaded it displays the first Account Number and it's details.
- The 4 navigational buttons used to move across accounts.
  They are…
- *MoveFirst* : Navigates to the first record
- *MovePrevious* : Navigates to the previous record
- *MoveNext*: Navigates to the next record
- *MoveLast*: Navigates to the last record.
- *Insert* Button allows the user to insert a new account in the *Accounts* table. First it will clear all the text fields on the screen and except the *Account Number* text box, here new Account Number is generated automatically by the system. The user has to fill up all the remaining fields legibly and then presses *commit* to insert the record.

Validations:

- The initial deposit should be a number greater than $500/-, otherwise it will display an error message "Invalid amount…enter number greater than $500".
- The date should be a valid date, other wise it will display an error message "Invalid Date".
- *Delete* button allows the user to delete the current record of which details are existing on the screen. Upon selection of *Delete* the system display a message box "Are you sure you want to delete?" If the user selects 'Yes' it will delete the record else it will undone the operation.
- *Close* closes the child window that is accounts operations screen.

2. Transactions Maintenance:

This module provides facilities to user to perform different transactions on accounts such as deposits, withdrawals etc.

Validations:

- After the withdrawal transaction if current balance less than $500/- then system will prompt the message "Balance less than $500/-".

- If the user enter any non numeric data or number less than 100 for amount then the system will report an error message “Enter number greater than 100”.
- If the date given is not a valid date then there will be an error message says that “Invalid Date”.
- If the user didn’t select any transaction and then click *Commit* the error message comes which prompt that ‘Select type of Transaction (deposit/Withdraw)’

3. Account Enquiry Details:

This module helps to enquire the list A/C details (A/C number, A/C holder’s name, address and balance) sorted either by *Account Number* or by A/C holder's name.

The user can select either sorting by *Account Number* or *name.*

- If he chooses sorting by Account Number, he has to select *Account Number* radio button. He may choose a starting Account Number of 1001. In that case, details of all A/Cs whose Account Number has the digits ‘1001’ will be displayed in the ascending order of account number in a separate page. The user can navigate back to the enquiry screen through a link.
- Similarly, he may choose a listing of details sorted by name and then *Name* radio button has to be selected. If the starting name field is populated with "Ra", system displays all the records sorted by name in the ascending order, whose name has the characters "Ra". If the “starting name” field is not populated, a listing of all account holders should be displayed in the order of increasing names.

4. Transaction Enquiry Details:

This module facilitates the user to generate the reports and statements of a particular account in the specified range of dates.

- By default the system displays the following screen with all existing Account Numbers under combo box, so that the user has to select one *Account Number* and also the *type of transaction* (withdraw or deposit) and the range of dates between which the transactions has committed. After that when (s) he selects ‘*Retrieve*’ it will display all the transactions on specified account of type either withdraw or deposit or both based on selection criteria of check boxes.

Validations:

- The user has to select either *Deposits* or *Withdrawals* or both. If (s) he selects neither then the error message “Select type of Transaction (Deposit/Withdraw)” will be displayed.

- The user has to enter the *start date* in such a way that it should be a valid date and also greater than accounts open date other wise the error message “Invalid date” will appear. The same message will appear if the user selects the end date, which is greater than current date or less than accounts open date.

ABC Bank

Performance based Requirement Document

Name:	QW2002 Tutorial
Application:	Performance Test Automation
Author:	Dr. Subraya BM(subraya@infy.com)
Date:	06-Sep-2002

Performance based Requirements Document
ABC Bank

ABC Bank is a global banking organization that provides standard banking services to its customers spread across the globe. The vision of the bank is **anywhere banking any time**. The head office is located in Chicago with several hundred centers situated outside USA. The proposed system supports both Client/Server and Web platform. A detailed specification on business functions is available in a separate document (see business Requirement documentation). In this document, we capture the details that are required for performance testing in addition to the existing business requirements document.

Following Sections provides brief details on performance-based requirements necessary for addressing the performance issues of the proposed system.

A. Business Related:

1. Business Background

To create a unique banking system through automation of all its services like customer service, managerial operations and inter-banking transactions in a more competitive prices.

To support this business goal, ABC Bank is in the process of building a comprehensive new information system.

2. Business Operations

ABC Bank is organized into four main groups: a) Senior Management, b) Customer Division, c) Middle Management d) The Information System Groups. The functions of each group is given elsewhere and not highlighted in this document.

3. Business Objectives of the System

The overall business goals of the system are to a) grow business at the rate of 20% every year, b) improve profitability, and c) increase customer satisfaction.
The specific business objectives for the new proposed system are as follows:

- Fulfill all requirements listed in the requirements doc
- Improve the productivity in the customer service by 30%
- Maintain the highest level of security for all accounts
- Ensure the system as performance conscious for the customer

4. Logistics of Operations

The business activities of the bank are distributed across the globe since the Sr. management believes in **anywhere banking** concept. However, the major business activities is concentrated in and around of USA with the head quarter located in Chicago. Most of the customer oriented business is carried out through Internet which is in turn depends on ISP providers and quality of connection to the system at the customer place. Set of in-house operations is carried out in an C/S environment.

5. User base

User base is split into two categories:

- **Internal users**
  Most of the administrative functions are performed by internal users (employees of the bank) in an C/S environment. Current internal users base is around 2000, which will grow at the rate of 20% every year.
- **Customers**
  Customers are both Account holders and prospect customers who want to start business with the bank in due course. All customers are Web clients who will be using the system via Internet. Current Web clients base is around 30,000 and the growth projection is given elsewhere in this document.

6. Business Growth Projections

The system must be upgradeable to support growth rates of 20% per year for the next 5 years. The growth depends on providing an Internet based banking and percentage of changing over from traditional banking to Internet banking. The traditional banking is carried out in C/S environment. The growth rate depends on customer satisfaction, which is of paramount importance to the business. One of the customer satisfactions is the quick response time for any query. The changing over from traditional banking to the Internet banking is proposed as:

	Percentage of banking	
Time frame	**By traditional banking**	**By Internet banking**
Current Year	90 %	10 %
2 years ahead	40 %	60 %
4 years ahead	70 %	30 %
5 years ahead	10 %	90 %

7. Infrastructure

The infrastructure proposed/available for ABC Bank is outlined below:

Network Environment:

The system will operate in a client/server network as well as supported by Web. There will a cluster of LAN situated at different places. Some of the clusters may have their own web server. All clusters are linked to the head quarters by a wide area network (WAN) situated in India. The client server network will include approximately 2000 users, with one client workstation per user. A complete topology of the Network environment is available with the IS department of the Bank.

The Backend Servers:

The system will have a centralized database servers situated at the head office. There will a set of multiple database servers, which will be used for data base mirroring. In other

words, all updates will be made concurrently to both copies of the databases, and these updates will be periodically monitored by the database management system (DBMS).

A complete back-up copy of the database will be taken once every night and the process will take approximately 30 minutes. During this time, the user response time may be slow. No guarantee will be given to the user about the amount of slowness, i.e. worst-case performance of the system. However, the managers would like to know the complete details, which is within the scope of the performance objective.

The Web Servers:
There will a host of Web servers connected to routers, which in turn connected to the Internet through multiple T1 lines. Each T1 line has a sufficient capacity to accommodate required sessions using 56 kbps modems. Firewalls are provided on the Web servers, which may slow down the overall performance.

Though the Web servers will have its own databases, it will not contain all the data needed to answer queries or process the requests, as they need to access the data base servers.

Software:
Standard software like Windows 2000, Oracle, Apache software are used.

B. Proposed System Related:
Based on the business process and requirements, the proposed system is viewed from two angles: i. Business functionality and ii. Performance of the system. The business functions are mapped to a set of use cases, which will be used to model the system. While modeling the system, performance issues also to be considered. Following Sections provide information on both functionality and performance issues.

8. Proposed Functions of the System
The major functions of the banking system are:

- Creation and maintenance of accounts
- Providing fast queries
- Routine banking transactions
- Faster Transactions thru ATMs
- Overall maintenance of the system

The proposed system must comply with the W3C standards. The detailed functions are available in the business requirements document.

9. Interfacing System, If any,
ABC Banking is planning to connect with other global banking system for inter-banking transactions.

C. Performance Test Related:

In order to ensure the performance of the system, it is necessary to conduct the performance testing in an isolated environment with automated tools. To use the automated tools we need to define the objectives and responsibilities before starting the performance testing. Following Sections provides brief information on requirements for conducting performance testing.

10. Responsibilities

The main responsibilities include testing the performance of the new system for response time, throughput, scalability and ability to handle peak loads. Immediate requirement is to prepare high-level strategy to describe the above task.(See high level Strategy document)

11. Performance Test Objectives

- Working out a satisfactory response time of the system under a realistic load. Response time consistency has to be worked out in consultation with the IS team (say 25 %). This means, if the average response time for a transaction is expected to be 2 seconds, then 90% or more of the measured response times should fall within the range of 1.5 and 2.5 seconds.
- Reliability of the system (checking memory leaks etc,.)
- Capability of handling of peak load
- Understanding of the systems capability to handle load beyond its planned capacity.
- Scalability

12. Operation Profile

Main features of the system are high lighted in the business requirements document. The frequency of utilization for each group is listed below:

User Group	**Features/Business Functions**	**Frequency of Utilization**		**Priority**
		Normal Hours	Peak hours	
Customer	Web site hits by external users			
Service	Main page only	2000	8000	high
Group	Banking features query	1500	5000	high
	Request for new A/Cs	100	300	avg
	Deposit amount	500	1500	high
	Withdraw amount	900	1800	high
Managers	Creation of new A/Cs	50	250	avg
Group	Deletion of new A/Cc	10	20	low
	Routine report generation	5	08	low
	Monitoring of A/Cs	150	600	avg
Senior	Ad-hoc query	0	100	high
Management	Daily on-line status report	15	150	avg
Group	Weekly status summary	20	200	low
	Monthly/Quarterly reports	10	50	low

Note:

- Frequency of utilization is expressed in terms of transactions per hour.
- Normal load is obtained during the routine business hours of transaction observed over a week.
- A peak hour of transaction represents the expected load during the worst-case hour in a typical week.
- The operation profile do not include overhead and background transactions, if any.

13. Testing Methodology

It is proposed to have two different teams: a) Functional testing team, which interacts with the development team and conforms the correctness of business functions of the system, b) Performance testing team, which plans and conducts the performance testing in cooperation with the Functional testing team. Each team will use their own test plan and strategy.

14. Automated Testing facilities

The team will use appropriate testing tools for functional and performance testing. They will use Rational Robot for functional and Test Manager for performance testing.

15. Service level Agreements (SLA)

A separate SLA has been developed for each operational groups and the agreement has been signed by the manager of the bank and the IS department.

16. Miscellaneous:

The Sr. Management would like to know the impact on performance if the web site generates dynamic contents.

17. Management Conditions and Constraints

No budget constraints to achieve the goal. No deadline on the schedule. However, once the strategy document is prepared and presented, cost and deadline will be determined.

ABC Bank

High level Strategy document for Performance Testing

Name:	QW2002
Application:	Performance Test Automation
Author:	Dr. Subraya BM(subraya@infy.com)
Date:	6-Sep-2002

High level Strategy document for Performance Testing
ABC Bank

This document provides high-level strategies for the smooth conduction of performance testing of the proposed banking system before the deployment. The exhaustive list will not be presented to the senior management, but it has been prepared as a necessary document before taking their approval to proceed with the project.

1. Identifying the Performance Testing Objectives

The primary goal of this system is to provide adequate information to the senior managers about the likelihood that the system will perform adequately in actual operation. This includes information about the capability of the system to handle, a) typical daily business workloads, b) daily, weekly, and monthly periods of peak demand, c) specialized needs such as the need to generate volumes of inter banking transactions, documents transfer, etc, d) adhoc demands like bursting loads during an odd day, e) likely future growth over the next 5 years.

Main questions that typically arises are:

- Is it the response time is acceptable when the system is operating under a realistic load? (Acceptable response time will be elaborated in the SLA)
- Is the system capable of handling peak load?
- Does the system operate correctly when accessed simultaneously by multiple users? (Check problems such as business functions, data base optimization, resource contention and transaction priorities)
- Whether the system is Scalable?
- Does the system provide the consistency in performance and reliability?
- Is the entire system tuned optimally?
- Does the system degrade gracefully or fail abruptly when the load is pushed beyond its limit?

2. Defining Performance Pass/Fail Criteria

To determine the success of the performance testing activity, it is of paramount importance to determine pass/fail criteria for each of the key performance goals. We need to define what constitutes passing the goal and what constitutes falling short of achieving it. The pass/fail criteria should be unambiguous so that it is measurable in absolute terms. Otherwise the clients may challenge the pass/fail criteria later.

3. Identifying the Measurement Criteria

Once the performance objectives and pass/fail criteria for each of them is successfully established, then we can start determining how each will be measured. Performance must be measured in absolute terms so that it can be easily compared with the pass/fail criteria. In order to establish proper measurements, it may be necessary to fine-tune or re-examine the goal as well as pass/fail criteria. An example might be instead of providing overall response time for an event, it may be required to determine the split response times among various hardware's devices that associated with the execution of the event.

4. Stakeholders

Expectations on performance among many groups with in the organization is different and difficult satisfy all. For example, senior management may expect the results at abstract level whereas middle management wants more microscopic level details. Therefore identifying the stakeholders is an important activity. Let us assume for the ABC banking as:

- Senior managers of ABC Bank are the major stakeholders.
- Customer Service Group needs to have sufficient confidence that the system will be able to support the workload of their department. Is it possible to satisfy the productivity goal (if any) as desired by the Sr management?
- The IS group of the Bank needs information about the levels of compliance with the SLAs which are likely to be achievable in live system operation.
- The clients expect the system must be performance conscious and robust one.

5. Project Scope

The scope of performance testing will include load, stress, spike and endurance testing in C/S and Web environment. The performance testing team must also analyse the results and suggest solutions to bottlenecks (if any) in fulfilling the SLA.

Note: Functional testing and related problems will not be part of the scope. It is assumed that the system is functionally correct before handing over to the performance test team. (Activities, which are outside the scope of the performance testing, must be elaborated in-detail).

6. Identifying the Key transactions

This activity is aimed at gaining more in-depth understanding of the system and how it is expected to be used in real-world operations. This is another key area that is critical to the success of C/S and Web applications. Identify the applications real life use and capture this information in operational scenarios. There may be different sets of customers, each with their own distinct operational scenarios. Once the operational scenarios have been clearly identified, then we can determine the key transactions or the mixture of transactions that are being generated by each operational scenario. This activity helps in understanding the technical details of the product's design when each operational scenario is executed, such as objects being invoked, interaction with outside object, calling ODBC/SQL drivers etc. These operational scenarios represent business transactions of some kind, not necessarily SQL transactions being processed on the Server's database.

It is essential to identify the proper operational scenarios that it generates the correct percentage of transactions and the correct mixture for each operational scenario. For example, the baseline for a performance goal on ABC banking system could be 10 concurrent connections doing simultaneous database synchronization, each with unique object instances, broken down as follows:

- 3 clients creating new customer accounts (30% of the transaction mix)
- 4 clients creating new deposits incident objects (40% of the transaction mix)
- 3 clients creating new debit incident objects (30% of the transaction mix)

If the next performance measurement is scaling from 10 to 20 concurrent connections, the new test cases would reflect:

- 6 clients creating new customer accounts (30% of the transaction mix)
- 8 clients creating new deposits incident objects (40% of the transaction mix)
- 6 clients creating new debit incident objects (30% of the transaction mix)

This must be pattern for any number of concurrent users for the given key transactions and transaction mix.

7. System Acceptance Criteria

The Service Level Agreement (SLA) has to be met in all respect. Also justify that the SLAs can be met in actual live operation.

8. Early Component-Level Performance Testing

Any component (third party developed or in-house built) is used in the system will be tested early before the actual performance testing of the system begins.

9. Performance Measurements

While testing, it is proposed to collect the following performance measurement data to get the complete picture:

- Response times of transactions as per the workloads defined.
- Consistency of responses
- Throughput (measured in transactions per second)
- Resource utilization during testing
    - Server and client processor utilization
    - Memory utilization (RAM and Disks)
    - Network utilization (LAN and T1 lines)
    - Thread utilization (the number of concurrent users of critical hardware and software components).
- Availability: ability to access the system resources such as databases, printers and the Internet, while the system is running.
- Bottleneck related
    - High resource consumption by each software and hardware components during the testing
    - Network collision rates
    - Page-in page-out rates
    - Number of hits blocked (blocking the web site visitors)
    - Number of threads blocked
    - Buffer and queue lengths overflows

10. Test Load – Volume and Mix

The load used for measuring the performance is based on:

- Typically daily business workloads,
- Daily peak demand,
- Weekly and monthly peak demand,
- Adhoc demands, if any
- Future growth demand over the next 5 years
- Quarterly, half yearly and yearly demands
- Spike test demand

11. Test Environment

Real environment is ideal to conduct load testing but difficult to have it. In such circumstances, the better option is to have the subset of the real environment. In case of C/S applications, following points to be addressed before finalizing the environment:

- Generic Vs Specific equipments: Some application may not work with all machines and may require specific equipments.
- Network Communication protocols and Network Operating Systems: Some of the automation tools may not work with all protocols and NOS.
- Server Operating system housing the RDBMS: Incompatibilities with certain automation tools or requires a OS (like Linux, Unix, NT) specific tool.
- Specific Browser compatibility
- Determine how many physical client machines are required to simulate virtual clients.

12. Selection of Proper Test Automation Tool

The main objective of this activity is to determine a better automation tool for performance testing. Since the cost of any performance automation tool is high, more attention must be given before finalizing the tool. Following points helps during the selection process:

- Appropriate tool must be selected based on performance goals. The tool must support all measurements that have been envisaged.
- Selected tool must be cost effective.
- The tool must have ability to use virtual databases users. Most significant of this is the database connectivity method or methods, such as ODBC or other RDBMS specific methods like RDO, DBLib, ADO, OLA DB for MS SQL Server.

13. Test Data Sources

The main database will be updated with real environment database. The test scripts captured in functional testing will be modified and used for performance testing. A test data generator will be used to create test data wherever it is necessary.

14. User Think Time

This is the time that a person takes to respond to a system, which can have a significant impact on its performance.

15. Stress and Duration Testing

After the load testing, a final stress and duration test will be executed. A proper workload will be selected and executed for 24 hours. Response time, throughput and resource utilization will be measured continually during testing.

16. Project activities and Schedule

A detailed schedule will be prepared in consultation with the IS department. Approximate duration for the entire test is approximately 4 weeks.

17. Project Risks

The major risks identified in this project are:

- Non-availability of the proper testing tools
- Skill set required for testing and analyzing results
- Cost involved in setting up of the testing environment
- Untimely submission of the final product for the performance testing
- Non-availability of database network administrators.

18. **Conclusion**

Since Senior management is not interested in microscopic details of the strategy, only brief details has been highlighted. A detailed comprehensive document based on this report and feedback from the management, may have to be prepared.

QW2002 Paper W4

Dr. Cem Kaner
(Florida Institute of Technology)

Bug Advocacy: Effective Bug Reporting

Key Points

- Many bugs are not fixed because they weren't well enough reported
- Here are some ideas for making bugs easier to reproduce, or simpler to describe
- Here are some ideas for gaining a better understanding of the severity of a bug.

About the Author

Cem Kaner is Professor of Computer Sciences at the Florida Institute of Technology. Prior to joining Florida Tech, Kaner worked in Silicon Valley for 17 years, doing and managing programming, user interface design, testing, and user documentation. He is the senior author (with Jack Falk and Hung Quoc Nguyen) of TESTING COMPUTER SOFTWARE (2nd Edition) and (with David Pels) of BAD SOFTWARE: WHAT TO DO WHEN SOFTWARE FAILS.

Through his consulting firm, KANER.COM, he teaches courses on black box software testing and consults to software publishers on software testing, documentation, and development management. Kaner is also the co-founder and co-host of the Los Altos Workshop on Software Testing, the Software Test Managers' RoundTable, the Workshop on Heuristic & Exploratory Techniques, and the Florida Workshops on Model-Based Testing.

Kaner is also attorney whose practice is focused on the law of software quality. He is active (as an advocate for customers, authors, and small development shops) in several legislative drafting efforts involving software licensing, software quality regulation, and electronic commerce. Kaner holds a B.A. in Arts & Sciences (Math, Philosophy), a Ph.D. in Experimental Psychology (Human Perception & Performance: Psychophysics), and a J.D. (law degree). He is Certified in Quality Engineering by the American Society for Quality.

Bug Advocacy

Cem Kaner J.D., Ph.D., ASQ-CQE
Florida Institute of Technology

Quality Week 2002

How to

Win Friends,

influence programmers

and

STоMp BUGs.

- (Not necessarily in that order.)

Copyright (c) Cem Kaner, 1999

Bug Advocacy?

- **Have you ever reported a bug that was dismissed as unrealistic or unlikely to be encountered by any normal person?**
- **Have you ever reported a bug that was closed as irreproducible even though you were sure it would show up again in the field?**
- **Have you ever worked on a recall or a crisis dot-release triggered by a bug that you found, but that wasn't fixed, before release?**
- **In mass-market software, most of the bugs customers call to complain about were discovered during testing but not fixed. Why weren't they fixed?**

Bug Advocacy?

- **So can you do anything to improve your company's handling of bug reports?**

  YES

  Think of a bug report as a tool that you use to sell the company on the idea of spending time and energy to fix a bug.

Copyright (c) Cem Kaner, 1999

Bug Advocacy?

1. **The point of testing is to find bugs.**
2. **Bug reports are your primary work product. This is what people outside of the testing group will most notice and most remember of your work.**
3. **The best tester isn't the one who finds the most bugs or who embarrasses the most programmers. The best tester is the one who gets the most bugs fixed.**
4. **Programmers operate under time constraints and competing priorities. For example, outside of the 12-hour workday, some naughty programmers prefer sleeping to fixing bugs.**

Note: When I say "the best tester is the one who gets the most bugs fixed," I'm not encouraging bug counting metrics. These are almost always counterproductive. I'm suggesting that the effective tester looks to the effect of the bug report, and tries to write in a way that gives each bug its best chance of being fixed. Also, a bug report is successful if it enables an informed business decision. Sometimes, the best decision is to not fix the bug. The excellent bug report provides sufficient data for a good decision.

Selling Bugs

- Time is in short supply. If you want to convince the programmer to spend his time fixing your bug, you may have to sell him on it.

  (Your bug? How can it be your bug? The programmer made it, not you, right? It's the programmer's bug.* Well, yes, but you found it so now it's yours *too.)
- Sales revolves around two fundamental objectives:
  - Motivate the buyer (*Make him WANT to fix the bug.)*
  - Overcome objections (*Get past his excuses and reasons for not fixing the bug.)*

Copyright (c) Cem Kaner, 1999

Motivating the Bug Fixer

Some things will often make programmers want to fix the bug:

- **It looks really bad.**
- **It looks like an interesting puzzle and piques the programmer's curiosity.**
- **It will affect lots of people.**
- **Getting to it is trivially easy.**
- **It has embarrassed the company, or a bug like it embarrassed a competitor.**
- **One of its cousins embarrassed the company or a competitor.**
- **Management (that is, someone with influence) has said that they really want it fixed.**
- **You've said that you want the bug fixed, and the programmer likes you, trusts your judgment, is susceptible to flattery from you, owes you a favor or accepted bribes from you.**

Overcoming Objections

These make programmers resist spending time on a bug:

- **The programmer can't replicate the defect.**
- **Strange and complex set of steps required to induce the failure.**
- **Not enough information to know what steps are required, and it will take a lot of work to figure them out.**
- **The programmer doesn't understand the report.**
- **Unrealistic (e.g. "corner case")**
- **It will take a lot of work to fix the defect.**
- **A fix will introduce too much risk into the code.**
- **No perceived customer impact**
- **Unimportant (no one will care if this is wrong: minor error or unused feature.)**
- **That's not a bug, it's a feature.**
- **Management doesn't care about bugs like this.**
- **The programmer doesn't like / trust you (or the customer who is complaining about the bug).**

Copyright (c) Cem Kaner, 1999

Bug Advocacy

Motivating Bug Fixes

By Better Researching

The Failure Conditions

Motivating Bug Fixes: Looking at the Failure

- **Some vocabulary**
  - **An error (or fault) is a design flaw or a deviation from a desired or intended state.**
  - **An error won't yield a failure without the conditions that trigger it. Example, if the program yields 2+2=5 on the 10th time you use it, you won't see the error before or after the 10th use.**
  - **The failure is the program's actual incorrect or missing behavior under the error-triggering conditions.**
  - **A symptom might be a characteristic of a failure that helps you recognize that the program has failed.**
  - **Defect is frequently used to refer to the failure or to the underlying error.**

    • Nancy Leveson (Safeware) draws useful distinctions between errors, hazards, conditions, and failures.

Copyright (c) Cem Kaner, 1999

Motivating Bug Fixes: Looking at the Failure

- **VOCABULARY EXAMPLE**
- **Here's a defective program**
  - INPUT A
  - INPUT B
  - PRINT A/B
- **What is the fault?**
- **What is the critical condition?**
- **What will we see as the failure?**

Motivating Bug Fixes

- **When you run a test and find a failure, you're looking at a symptom, not at the underlying fault. You may or may not have found the best example of a failure that can be caused by the underlying fault.**
- **Therefore you should do some follow-up work to try to prove that a defect:**
  - ***is more serious than it first appears.***
  - ***is more general than it first appears.***

Copyright (c) Cem Kaner, 1999

Motivating Bug Fixes: Make it More Serious

LOOK FOR FOLLOW-UP ERRORS

- When you find a coding error, you have the program in a state that the programmer did not intend and probably did not expect. There might also be data with supposedly impossible values.
- The program is now in a vulnerable state. Keep testing it and you might find that the real impact of the underlying fault is a much worse failure, such as a system crash or corrupted data.
- I do three types of follow-up testing:
  - Vary my behavior (change the conditions by changing what I do)
  - Vary the options and settings of the program (change the conditions by changing something about the program under test).
  - Vary the software and hardware environment.

Follow-Up Testing: Varying Your Behavior

- Keep using the program after you see the problem.
- Bring it to the failure case again (and again). If the program fails when you do X, then do X many times. Is there a cumulative impact?
- Try things that are related to the task that failed. For example, if the program unexpectedly but slightly scrolls the display when you add two numbers, try tests that affect adding or that affect the numbers. Do X, see the scroll. Do Y then do X, see the scroll. Do Z, then do X, see the scroll, etc. (If the scrolling gets worse or better in one of these tests, follow that up, you're getting useful information for debugging.)
- Try things that are related to the failure. If the failure is unexpected scrolling after adding, try scrolling first, then adding. Try repainting the screen, then adding. Try resizing the display of the numbers, then adding.
- Try entering the numbers more quickly or changing the speed of your activity in some other way.
- And try the usual exploratory testing techniques. So, for example, you might try some interference tests. Stop the program or pause it or swap it just as the program is failing. Or try it while the program is doing a background save. Does that cause data loss corruption along with this failure?

Copyright (c) Cem Kaner, 1999

Follow-Up: Vary Options and Settings

In this case, the steps to achieve the failure are taken as given. Try to reproduce the bug when the program is in a different state:

- **Use a different database.**
- **Change the values of persistent variables.**
- **Change how the program uses memory.**
- **Change anything that looks like it might be relevant that allows you to change as an option.**

For example, suppose the program scrolls unexpectedly when you add two numbers. Maybe you can change the size of the program window, or the precision (or displayed number of digits) of the numbers, or background the activity of the spell checker.

Follow-Up: Vary the Configuration

A bug might show a more serious failure if you run the program with less memory, a higher resolution printer, more (or fewer) device interrupts coming in etc.

- **If there is anything involving timing, use a really slow (or very fast) computer, link, modem or printer, etc..**
- **If there is a video problem, try other resolutions on the video card. Try displaying MUCH more (less) complex images.**

Note that we are not:

- **checking standard configurations**
- **asking how broad the circumstances that produces the bug.**

What we're asking is whether there is a particular configuration that will show the bug more spectacularly.

Returning to the example (unexpected scrolling when you add two numbers), try things like:

- **Different video resolutions**
- **Different mouse settings if you have a wheel mouse that does semi-automated scrolling**
- **An NTSC (television) signal output instead of a traditional (XGA or SVGA, etc.) monitor output.**

Copyright (c) Cem Kaner, 1999

Follow-up: Is This Bug New to This Version?

In many projects, an old bug (from a previous shipping release of the program) might not be taken very seriously if there weren't lots of customer complaints.

- **(If you know it's an old bug, check its complaint history.)**
- **The bug will be taken more seriously if it is new.**
- **You can argue that it should be treated as new if you can find a new variation or a new symptom that didn't exist in the previous release. What you are showing is that the new version's code interacts with this error in new ways. That's a new problem.**
- **This type of follow-up testing is especially important during a maintenance release that is just getting rid of a few bugs. Bugs won't be fixed unless they were (a) scheduled to be fixed because they are critical or (b) new side effects of the new bug fixing code.**

Motivating the Bug Fix: Show it is More General

- **LOOK FOR CONFIGURATION DEPENDENCE**

> Question: How many programmers does it take to change a light bulb?
>
> Answer: *What's the problem? The bulb at my desk works fine!*

- **Bugs that don't fail on the programmer's machine are much less credible (to that programmer). If they are configuration dependent, the report will be much more credible if it identifies the configuration dependence directly (and so the programmer starts out with the expectation that it won't fail on all machines.)**

Copyright (c) Cem Kaner, 1999

LOOK FOR CONFIGURATION DEPENDENCE

In the ideal case (standard in many companies), you test on 2 machines

- **Do your main testing on Machine 1. Maybe this is your powerhouse: latest processor, newest updates to the operating system, fancy printer, video card, USB devices, huge hard disk, lots of RAM, cable modem, etc.**
- **When you find a defect, use Machine 1 as your bug reporting machine and replicate on Machine 2. Machine 2 is totally different. Different processor, different keyboard and keyboard driver, different video, barely enough RAM, slow, small hard drive, dial-up connection with a link that makes turtles look fast.**
- **Some people do their main testing on the turtle and use the power machine for replication.**
- **Write the steps, one by one, on the bug form at Machine 1. As you write them, try them on Machine 2. If you get the same failure, you've checked your bug report while you wrote it. (A valuable thing to do.)**
- **If you don't get the same failure, you have a configuration dependent bug. Time to do troubleshooting. But at least you know that you have to.**

AS A MATTER OF GENERAL GOOD PRACTICE, IT PAYS TO REPLICATE EVERY BUG ON A SECOND MACHINE.

Motivating bug fixes: Show it's more general

- **UNCORNER YOUR CORNER CASES**
- **We test at extreme values because these are the most likely places to show a defect. But once we find the defect, we don't have to stick with extreme value tests.**
  - **Try mainstream values. These are easy settings that should pose no problem to the program. Do you replicate the bug? If yes, write it up, referring primarily to these mainstream settings. This will be a very credible bug report.**

Copyright (c) Cem Kaner, 1999

Motivating bug fixes: Show it's more general

- **UNCORNER YOUR CORNER CASES**
  - **If the mainstream values don't yield failure, but the extremes do, then do some troubleshooting around the extremes.**
    - Is the bug tied to a single setting (a true corner case)?
    - Or is there a small range of cases? What is it?
    - In your report, identify the narrow range that yields failures. The range might be so narrow that the bug gets deferred. That might be the right decision. In some companies, the product has several hundred open bugs a few weeks before shipping. They have to decide which 300 to fix (the rest will be deferred). Your reports help the company choose the right 300 bugs to fix, and help people size the risks associated with the remaining ones.

Bug Advocacy

Overcoming

OBJECTIONS

By Better Researching

The Failure Conditions

Copyright (c) Cem Kaner, 1999

Overcoming Objections: Analyze the Failure

- **Things that will make programmers resist spending their time on the bug:**
  - **The programmer can't replicate the defect.**
    - **Strange and complex set of steps required to induce the failure.**
    - **Not enough information to know what steps are required, and it will take a lot of work to figure them out.**
    - **The programmer doesn't understand the report.**
    - **Unrealistic (e.g. "corner case")**
    - **It's a feature.**

Objection, objection: Non-reproducible errors

Always report non-reproducible errors. If you report them well, programmers can often figure out the underlying problem.

To help them, you must describe the failure as precisely as possible. If you can identify a display or a message well enough, the programmer can often identify a specific point in the code that the failure had to pass through.

- **When you realize that you can't reproduce the bug, write down everything you can remember. Do it now, before you forget even more. As you write, ask yourself whether you're sure that you did this step (or saw this thing) exactly as you are describing it. If not, say so. Draw these distinctions right away. The longer you wait, the more you'll forget.**
- **Maybe the failure was a delayed reaction to something you did before starting this test or series of tests. Before you forget, note the tasks you did before running this test.**
- **Check the bug tracking system. Are there similar failures? Maybe you can find a pattern.**
- **Find ways to affect timing of your program or of your devices, Slow down, speed up.**
- *Talk to the programmer* **and/or read the code.**

Copyright (c) Cem Kaner, 1999

Non-Reproducible Errors

- The fact that a bug is not reproducible is data. The program is telling you that you have a hole in your logic. You are not entertaining certain relevant conditions. Why not?
- See Watts Humphrey, Personal Software Process, for recommendations to programmers of a system for discovering and then eliminating characteristic errors from their code. A non-reproducible bug is a tester's error, just like a design bug is a programmer's error. It's valuable to develop a system for discovering your blind spots. To improve over time, keep track of the bugs you're missing and what conditions you are not attending to (or find too hard to manipulate).
- The following pages give a list of some conditions commonly ignored or missed by testers. Your personal list will be different in some ways, but maybe this is a good start. When you run into a irreproducible defect look at this list and ask whether any of these conditions could be the critical one. If it could, vary your tests on that basis and you might reproduce the failure.

(Note: Watts Humphrey suggested to me the idea of keeping a list of commonly missed conditions. It has been a tremendously valuable insight.)

Non-Reproducible Errors: Examples of Conditions Often Missed

- Some problems have delayed effects:
  - a memory leak might not show up until after you cut and paste 20 times.
  - stack corruption might not turn into a stack overflow until you do the same task many times.
  - a wild pointer might not have an easily observable effect until hours after it was mis-set.
- If you suspect that you have time-delayed failures, use tools such as videotape, capture programs, debuggers, debug-loggers, or memory meters to record a long series of events over time.

Copyright (c) Cem Kaner, 1999

Non-Reproducible Errors: Examples of Conditions Often Missed

- I highlighted the first three in lecture because so many people have trouble with time-delayed bugs. Until you think backwards in time and ask how you could find a defect that has a delayed reaction effect, you won't be able to easily recreate these problems.
- The following pages give additional examples. There are plenty of other conditions that are relevant in your environment. Start with these but add others as you learn of them. How do you learn? Sometimes, someone will fix a bug that you reported as non-reproducible. Call the programmer, ask him how to reproduce it, what are the critical steps that you have to take? You need to know this anyway, so that you can confirm that a bug fix actually worked.

Non-Reproducible Errors: Examples of Conditions Often Missed

- The bug depends on the value of a hidden input variable. (Bob Stahl teaches this well.) In any test, there are the variables that we think are relevant and then there is everything else. If the data that you think are relevant don't help you reproduce the bug, ask what other variables were set, and what their values were, in the course of running or preparing this test.
- Some conditions are hidden and others are invisible. You cannot manipulate them and so it is harder to recognize that they're present. You might have to talk with the programmer about what state variables or flags get set in the course of using a particular feature.
- Some conditions are catalysts. They make failures more likely to be seen. Example: low memory for a leak; slow machine for a race. But sometimes catalysts are more subtle, such as use of one feature that has a subtle interaction with another.
- Some bugs are predicated on corrupted data. They don't appear unless there are impossible configuration settings in the config files or impossible values in the database. What could you have done earlier today to corrupt this data?

Copyright (c) Cem Kaner, 1999

Non-Reproducible Errors: Examples of Conditions Often Missed

- The bug might appear only at a specific time of day or day of the month or year. Look for week-end, month-end, quarter-end and year-end bugs, for example.
- Programs have various degrees of data coupling. When two modules use the same variable, oddness can happen in the second module after the variable is changed by the first. (Books on structured design, such as Yourdon/Constantine often analyze different types of coupling in programs and discuss strengths and vulnerabilities that these can create.) In some programs, interrupts share data with main routines in ways that cause bugs that will only show up after a specific interrupt.
- Special cases appear in the code because of time or space optimizations or because the underlying algorithm for a function depends on the specific values fed to the function (talk to your programmer).
- The bug depends on you doing related tasks in a specific order.

Non-Reproducible Errors: Examples of Conditions Often Missed

- The bug is caused by a race condition or other time-dependent event, such as:
  - An interrupt was received at an unexpected time.
  - The program received a message from another device or system at an inappropriate time (e.g. after a time-out.)
  - Data was received or changed at an unexpected time.
- The bug is caused by an error in error-handling. You have to generate a previous error message or bug to set up the program for this one.
- Time-outs trigger a special class of multiprocessing error handling failures. These used to be mainly of interest to real-time applications, but they come up in client/server work and are very pesky.
- Process A sends a message to Process B and expects a response. B fails to respond. What should A do? What if B responds later?

Copyright (c) Cem Kaner, 1999

Non-Reproducible Errors: Examples of Conditions Often Missed

- **Another inter-process error handling failure -- Process A sends a message to B and expects a response. B sends a response to a different message, or a new message of its own. What does A do?**
- **You're being careful in your attempt to reproduce the bug, and you're typing too slowly to recreate it.**
- **The program might be showing an initial state bug, such as:**
  - **The bug appears only the first time after you install the program (so it happens once on every machine.)**
  - **The bug appears once after you load the program but won't appear again until you exit and reload the program.**
    - (See Testing Computer Software's Appendix's discussion of initial state bugs.)
- **The program may depend on one version of a DLL. A different program loads a different version of the same DLL into memory. Depending on which program is run first, the bug appears or doesn't.**

Non-Reproducible Errors: Examples of Conditions Often Missed

- **The problem depends on a file that you think you've thrown away, but it's actually still in the Trash (where the system can still find it).**
- **A program was incompletely deleted, or one of the current program's files was accidentally deleted when that other program was deleted. (Now that you've reloaded the program, the problem is gone.)**
- **The program was installed by being copied from a network drive, and the drive settings were inappropriate or some files were missing. (This is an invalid installation, but it happens on many customer sites.)**
- **The bug depends on co-resident software, such as a virus checker or some other process, running in the background. Some programs run in the background to intercept foreground programs' failures. These may sometimes trigger failures (make errors appear more quickly).**

Copyright (c) Cem Kaner, 1999

Non-Reproducible Errors: Examples of Conditions Often Missed

- You forgot some of the details of the test you ran, including the critical one(s) or you ran an automated test that lets you see that a crash occurred but doesn't tell you what happened.
- The bug depends on a crash or exit of an associated process.
- The program might appear only under a peak load, and be hard to reproduce because you can't bring the heavily loaded machine under debug control (perhaps it's a customer's system).
- On a multi-tasking or multi-user system, look for spikes in background activity.
- The bug occurred because a device that it was attempting to write to or read from was busy or unavailable.
- It might be caused by keyboard keybounce or by other hardware noise.

Non-Reproducible Errors: Examples of Conditions Often Missed

- Code written for a cooperative multitasking system can be thoroughly confused, sometimes, when running on a preemptive multitasking system. (In the cooperative case, the foreground task surrenders control when it is ready. In the preemptive case, the operating system allocates time slices to processes. Control switches automatically when the foreground task has used up its time. The application is suspended until its next time slice. This switch occurs at an arbitrary point in the application's code, and that can cause failures.
- The bug occurs only the first time you run the program or the first time you do a task after booting the program. To recreate the bug, you might have to reinstall the program. If the program doesn't uninstall cleanly, you might have to install on a fresh machine (or restore a copy of your system taken before you installed this software) before you can see the problem.

Copyright (c) Cem Kaner, 1999

Non-Reproducible Errors: Examples of Conditions Often Missed

- The bug is specific to your machine's hardware and system software configuration. (This common problem is hard to track down later, after you've changed something on your machine. That's why good reporting practice involves replicating the bug on a second configuration.)
- The bug was a side-effect of a hardware failure. This is rarely the problem, but sometimes it is. A flaky power supply creates irreproducible failures, for example. Another example: One prototype system had a high rate of irreproducible firmware failures. Eventually, these were traced to a problem in the building's air conditioning. The test lab wasn't being cooled, no fan was blowing on the unit under test, and several of the prototype boards in the machine ran very hot. (Later versions would run cooler, but these were early prototypes.) The machine was failing at high temperatures.
- Elves tinkered with your machine when you weren't looking.
- There are several other ideas (focused on web testing) at http://www.logigear.com/whats_new.html#article

Putting Bugs in the Dumpster

Problem:

- Non-reproducible bugs burn a huge amount of programmer troubleshooting time, then get closed (usually abandoned). Until they're closed, they show up in open-bug statistics. In companies that manage more by bug numbers than by good sense, there is tremendous pressure to close irreproducible bugs quickly.

The Dumpster:

- A resolution code that puts the bug into an ignored storage place. The bug shows up as resolved (or is just never counted) in the bug statistics, but it is not closed. It is in a holding pattern.
- Assign a non-reproducible bug to the dumpster whenever you (testers and programmers) spend enough time on it that you don't think that more work on the bug will be fruitful at this time.

Dumpster Diving:

- Every week or two, (testers and/or programmers) go through the dumpster bugs looking for similar failures. At some point, you'll find a collection of several similar reports. If you (or the programmer) think there are enough variations in the reports to provide useful hints on how to repro the bug, spend time on the collection. If you (or the programmer) can repro the bugs, reopen them with the extra info (status is now open, resolution is pending)
- Near the end of the project, do a final review of bugs in the dumpster. These will either close non-repro or be put through one last scrutiny

(This is an unusual practical suggestion, but it has worked for clients of mine.)

Copyright (c) Cem Kaner, 1999

Overcoming Objections: Analyze the Failure

- Things that will make programmers resist spending their time on the bug:
  - The programmer can't replicate the defect.
  - **Strange and complex set of steps required to induce the failure.**
  - **Not enough information to know what steps are required, and it will take a lot of work to figure them out.**
  - **The programmer doesn't understand the report.**
  - Unrealistic (e.g. "corner case")
  - It's a feature!

Bug Advocacy

Writing the Bug Report

Copyright (c) Cem Kaner, 1999

Reporting Errors

- As soon as you run into a problem in the software, fill out a Problem Report form.
- In the well written report, you:
  - Explain how to reproduce the problem.
  - Analyze the error so you can describe it in a minimum number of steps.
  - Include all the steps.
  - Make the report easy to understand.
  - Keep your tone neutral and non-antagonistic.
  - Keep it simple: one bug per report.
  - If a sample test file is essential to reproducing a problem, reference it and attach the test file.
  - To the extent that you have time, describe the dimensions of the bug and characterize it. Describe what events are and are not relevant to the bug. And what the results are (any characteristics of the failure) and how they varied across tests.

The Problem Report Form

A typical form includes many of the following fields

- Problem report number: must be unique
- Reported by: original reporter's name. Some forms add an editor's name.
- Date reported: date of initial report
- Program (or component) name: the visible item under test
- Release number: like Release 2.0
- Version (build) identifier: like version C or version 20000802a
- Configuration(s): h/w and s/w configs under which the bug was found and replicated
- Report type: e.g. coding error, design issue, documentation mismatch, suggestion, query
- Can reproduce: yes / no / sometimes / unknown. (Unknown can arise, for example, when the repro configuration is at a customer site and not available to the lab.)

Copyright (c) Cem Kaner, 1999

The Problem Report Form

A typical form includes many of the following fields

- Severity: assigned by tester. Some variation on small / medium / large
- Priority: assigned by programmer/project manager
- Customer impact: often left blank. When used, typically filled in by tech support or someone else predicting actual customer reaction (such as support cost or sales impact)
- Problem summary: 1-line summary of the problem
- Key words: use these for searching later, anyone can add to key words at any time
- Problem description and how to reproduce it: step by step repro description
- Suggested fix: leave it blank unless you have something useful to say
- Assigned to: typically used by project manager to identify who has responsibility for researching/fixing the problem

The Problem Report Form

A typical form includes many of the following fields

- Status: Tester fills this in. Open / closed / dumpster—see prev slide on dumpster.
- Resolution: The project manager owns this field. Common resolutions include:
  - Pending: the bug is still being worked on.
  - Fixed: the programmer says it's fixed. Now you should check it.
  - Cannot reproduce: The programmer can't make the failure happen. You must add details, reset the resolution to Pending, and notify the programmer.
  - Deferred: It's a bug, but we'll fix it later.
  - As Designed: The program works as it's supposed to.
  - Need Info: The programmer needs more info from you. She has probably asked a question in the comments.
  - Duplicate: This is just a repeat of another bug report (XREF it on this report.) Duplicates should not close until the duplicated bug closes.
  - Withdrawn: The tester who reported this bug is withdrawing the report.

Copyright (c) Cem Kaner, 1999

The Problem Report Form

A typical form includes many of the following fields

- **Resolution version: build identifier**
- **Resolved by: programmer, project manager, tester (if withdrawn by tester), etc.**
- **Resolution tested by: originating tester, or a tester if originator was a non-tester**
- **Change history: datestamped list of all changes to the record, including name and fields changed.**
- **Comments: free-form, arbitrarily long field, typically accepts comments from anyone on the project. Testers programmers, tech support (in some companies) and others have an ongoing discussion of repro conditions, etc., until the bug is resolved. Closing comments (why a deferral is OK, or how it was fixed for example) go here.**
  - **This field is especially valuable for recording progress and difficulties with difficult or politically charged bugs.**
  - **Write carefully. Just like e-mail and usenet postings, it's easy to read a joke or a remark as a flame. Never flame.**

The Problem Report Form

- **The best discussion in print of bug reporting and bug tracking system design is probably still the one in my book, Testing Computer Software, chapters 5 & 6. (Not because it's so wonderful but because not enough good stuff has been written since. For more, see Rex Black's *Managing the Testing Process.)***
- **Brian Marick has captured some useful material at his site, www.testingcraft.com. (You should get to know this site, and ideally, contribute to it. This is a collection point for examples.)**
- **Hung Quoc Nguyen (who co-authored TCS 2.0 and is working with us on 3.0) published TrackGear, a web based bug tracking system that has a lot of thought behind it. You can get a 30-day free eval.**
- **The Testing Tools FAQ lists other bug tracking software that you can get eval copies. The FAQ is linked from the main comp.software.testing FAQ at http://www.rstcorp.com/resources/hosted.html**

Copyright (c) Cem Kaner, 1999

Important Parts of the Report: Problem Summary

- **This one-line description of the problem is the most important part of the report.**
  - **The project manager will use it in when reviewing the list of bugs that haven't been fixed.**
  - **Executives will read it when reviewing the list of bugs that won't be fixed. They might only spend additional time on bugs with "interesting" summaries.**
- **The ideal summary gives the reader enough information to help her decide whether to ask for more information. It should include:**
  - **A brief description that is specific enough that the reader can visualize the failure.**
  - **A brief indication of the limits or dependencies of the bug (how narrow or broad are the circumstances involved in this bug)?**
  - **Some other indication of the severity (not a rating but helping the reader envision the consequences of the bug.)**

The Report: Can you reproduce the problem?

- **You may not see this on your form, but you should always provide this information.**
  - **Never say it's reproducible unless you have recreated the bug. (Always try to recreate the bug before writing the report.)**
  - **If you've tried and tried but you can't recreate the bug, say "No". Then explain what steps you tried in your attempt to recreate it.**
  - **If the bug appears sporadically and you don't yet know why, say "Sometimes" and explain.**
  - **You may not be able to try to replicate some bugs. Example: customer-reported bugs where the setup is too hard to recreate.**
- **The following policy is not uncommon:**
  - ***If the tester says that a bug is reproducible and the programmer says it is not, then the tester has to recreate it in the presence of the programmer.***

Copyright (c) Cem Kaner, 1999

The Report—Description: How to reproduce it.

- First, describe the problem. What's the bug? Don't rely on the summary to do this -- some reports will print this field without the summary.
- Next, go through the steps that you use to recreate this bug.
  - Start from a known place (e.g. boot the program) and
  - Then describe each step until you hit the bug.
  - NUMBER THE STEPS. Take it one step at a time.
  - If anything interesting happens on the way, describe it. (You are giving people directions to a bug. Especially in long reports, people need landmarks.)
- Describe the erroneous behavior and, if necessary, explain what should have happened. (Why is this a bug? Be clear.)
- List the environmental variables (config, etc.) that are not covered elsewhere in the bug tracking form.
- If you expect the reader to have any trouble reproducing the bug (special circumstances are required), be clear about them.

The Report—Description: How to reproduce it.

It is essential keep the description focused:

- The first part of the description should be the shortest step-by-step statement of how to get to the problem.
- Add "Notes" after the description if you have them. Typical notes include:
  - Comment that the bug won't show up if you do step X between step Y and step Z.
  - Comment explaining your reasoning for running this test.
  - Comment explaining why you think this is an interesting bug.
  - Comment describing other variants of the bug.

Copyright (c) Cem Kaner, 1999

Keeping the Report Simple

- **If you see two failures, write two reports.**
- **Combining failures on one report creates problems:**
  - **The summary description is typically vague. You say words like "fails" or "doesn't work" instead of describing the failure more vividly. This weakens the impact of the summary.**
  - **The detailed report is typically lengthened. It's common to see bug reports that read like something written by an inept lawyer. Do this unless that happens in which case don't do this unless the first thing and then the testcase of the second part and sometimes you see this but if not then that.**
  - **Even if the detailed report is rationally organized, it is longer (there are two failures and two sets of conditions, even if they are related) and therefore more intimidating.**
  - **You'll often see one bug get fixed but not the other.**
  - **When you report related problems on separate reports, it is a courtesy to cross-reference them.**

Keeping it Simple: Eliminate Unnecessary Steps (1)

- **Sometimes it's not immediately obvious what steps can be dropped from a long sequence of steps in a bug.**
  - **Look for critical steps -- Sometimes the first symptoms of an error are subtle.**
- **You have a list of all the steps that you took to show the error. You're now trying to shorten the list. Look carefully for any hint of an error as you take each step -- A few things to look for:**
  - **Error messages (you got a message 10 minutes ago. The program didn't fully recover from the error, and the problem you see now is caused by that poor recovery.)**
  - **Delays or unexpectedly fast responses.**
  - **Display oddities, such as a flash, a repainted screen, a cursor that jumps back and forth, multiple cursors, misaligned text, slightly distorted graphics, doubled characters, omitted characters, or display droppings (pixels that are still colored even though the character or graphic that contained them was erased or moved).**

Copyright (c) Cem Kaner, 1999

Keeping it Simple: Eliminate Unnecessary Steps (2)

- Sometimes the first indicator that the system is working differently is that it sounds a little different than normal.
- An in-use light or other indicator that a device is in use when nothing is being sent to it (or a light that is off when it shouldn't be).
- Debug messages—turn on the debug monitor on your system (if you have one) and see if/when a message is sent to it.

- If you've found what looks like a critical step, try to eliminate almost everything else from the bug report. Go directly from that step to the last one (or few) that shows the bug. If this doesn't work, try taking out individual steps or small groups of steps.

Keep it Simple: Put Variations After the Main Report

- Suppose that the failure looks different under slightly different circumstances. For example:
  - The timing changes if you do a additional two sub-tasks before hitting the final reproduction step
  - The failure won't show up at all or is much less serious if you put something else at a specific place on the screen
  - The printer prints different garbage (instead of the garbage you describe) if you make the file a few bytes longer
- This is all useful information for the programmer and you should include it. But to make the report clear:
  - Start the report with a simple, step-by-step description of the shortest series of steps that you need to produce the failure.
  - Identify the failure. (Say whatever you have to say about it, such as what it looks like or what impact it will have.)
  - Then add a section that says "ADDITIONAL CONDITIONS" and describe, one by one, in this section the additional variations and the effect on the observed failure.

Copyright (c) Cem Kaner, 1999

Overcoming Objections: Analyze the Failure

- **Things that will make programmers resist spending their time on the bug:**
  - **The programmer can't replicate the defect.**
  - **Strange and complex set of steps required to induce the failure.**
  - **Not enough information to know what steps are required, and it will take a lot of work to figure them out.**
  - **The programmer doesn't understand the report.**
  - **Unrealistic (e.g. "corner case")**
  - **It's a feature!**

Overcoming Objections: Unrealistic (e.g., Corner Conditions)

Some reports are inevitably dismissed as unrealistic (having no importance in real use).

- **If you're dealing with an extreme value, do follow-up testing with less extreme values.**
- **If you're protesting a bug that has been left unfixed for several versions, realized that it has earned tenure in some people's minds. Perhaps customer complaints about this bug have simply never filtered through to developers.**
- **If your report of some other type of defect or design issue is dismissed as having "no customer impact," ask yourself:**

Hey, how do they know the customer impact?

- **Then check with people who might know:**
  - -- Technical marketing
  - -- Technical support
  - -- Human factors
  - -- Documentation
  - -- Network administrators
  - -- Training
  - -- In-house power users
  - -- Maybe sales

Copyright (c) Cem Kaner, 1999

Overcoming Objections: Analyze the Failure

- Things that will make programmers resist spending their time on the bug:
  - The programmer can't replicate the defect.
  - Strange and complex set of steps required to induce the failure.
  - Not enough information to know what steps are required, and it will take a lot of work to figure them out.
  - The programmer doesn't understand the report.
  - Unrealistic (e.g. "corner case")
  - **It's a feature!**

    Later in the course, we'll think about this. The usual issues involve the costs of fixing bugs, the company's understanding of the definitions of bugs, and your personal credibility.

Bug Advocacy

Editing the Bug Report

Copyright (c) Cem Kaner, 1999

Editing Bug Reports

- Some groups have a second tester (usually a senior tester) review reported defects before they go to the programmer. The second tester:
  - checks that critical information is present and intelligible
  - checks whether she can reproduce the bug
  - asks whether the report might be simplified, generalized or strengthened.
- If there are problems, she takes the bug back to the original reporter.
  - If the reporter was outside the test group, she simply checks basic facts with him.
  - If the reporter was a tester, she points out problems with an objective of furthering the tester's training.

Editing Bug Reports

- This tester might review:
  - all defects
  - all defects in her area
  - all of her buddy's defects.
- In designing a system like this, beware of overburdening the reviewing testers. The reviewer will often go through a learning curve (learning about parts of the system or types of tests that she hasn't studied before). This takes time. Additionally, you have to decide whether the reviewer is doing an actual reproduction of the test or thinking about the plausibility and understandability of the report when she reads it.

Copyright (c) Cem Kaner, 1999

Editing Bugs--Practice at Home

Go through your bug database and find some bugs that look interesting

- **Do an initial review of them**
- **Replicate them**
- **Revise the descriptions to make them clearer and more useful.**

Assignment:

- **Give two improved bugs to a co-worker**
- **Review two improved bugs from a co-worker**
- **Compare notes**

(Note: When I teach this course to undergraduates, I require them to successfully edit bugs before they can write any. It is effective training.)

Editing Bugs: Assignment Procedure

First times:

The tester gives you the bug report before entering it into the bug tracking system.

- **The reporter should give you a hard copy of the proposed bug report or a file in a format you can read. If you can't read the reporter's file format, the reporter has to give you the bug in some other format. This is the reporter's responsibility, not yours.**
- **Read over the report. If you can't understand it or if there are obvious problems, note those problems and return it to the reporter. If there are significant problems when you try to read the report, don't spend any time trying to replicate it. Just give it back and deal with it again later, when it has been fixed.**
- **If the report is OK to read (not perfect, but OK), make some comments (maybe on a printout, maybe in the text file that the reporter gave you) and then try to replicate the bug.**

Copyright (c) Cem Kaner, 1999

Editing Bugs: Assignment Procedure

First times:

Make comments as appropriate. Then hand the commented report back to the reporter. The reporter can review your comments, decide what to change, and then either:

- Submit the bug directly into the bug tracking system, or
- Give the bug back to you for a second review.

You are only obligated to review a bug once. If you review the bug and bounce it because it is unintelligible, you don't have to accept it back for replication. If you replicated it and gave feedback, you don't have to review the improved version.

If the reporter is submitting a bug to you that was previously reviewed by someone else, she MUST give you a copy of the report that she gave to that other person and their comments, along with the new improved report.

Editing Bugs: Assignment Procedure

Later times:

If the tester gives you the bug report before entering it into the bug tracking system.

- **Same procedure as before**

If the tester gives you the report AFTER entering it into the bug tracking system

- **Review the report for clarity and tone (see "first impressions", next slide) and send comments back to the reporter by email**
- **Attempt to replicate the bug and send comments back to the reporter by email on the replication steps, your overall impressions, and any follow-up tests you recommend**
- **You may edit the bug report yourself, but ONLY in the following ways.**
  - Add a comment indicating that you successfully replicated the bug on XXX configuration in YYY build. (This is only valuable if the configuration or build is different from the reporter's.)
  - Add a comment describing a simpler set of replication steps. Make sure these are clear and accurate. (Continued . . .)

Copyright (c) Cem Kaner, 1999

Editing Bugs: Assignment Procedure

Later times:

You may edit the bug report yourself, but ONLY in the following ways. (. . . Continued)

- Add a comment describing why this bug would be important to customers (this is only needed if the bug looks minor or like it won't be fixed. It is only useful if you clearly know what you are talking about, your tone is respectful).
- Your comments should NEVER appear critical or disrespectful of the original report or of the person who wrote it. You are adding information, not criticizing what was there.

- **If you edit the report in the database,** never change what the reporter has actually written**. You are not changing his work, you are adding comments to it at the end of the report**
- **Your comments should have your name and the comment date, usually at the start of the comment, for example: "(Cem Kaner, 12/14/01) Here is an alternative set of replication steps:")**
- **Send the reporter an email, telling her that you have reviewed the report and made changes.**

Editing Bugs—A Checklist

The bug editor should check the bug report for the following characteristics:

First impressions—when you first read the report:

- Is the summary short (about 50-70 characters) and descriptive? (see the slide: *Important Parts of the Report: Problem Summaries*)
- Can you understand the report? As you read the description, do you understand what the reporter did? Can you envision what the program did in response? Do you understand what the failure was?
- Is it obvious where to start (what state to bring the program to, to replicate the bug)?
- Is it obvious what files to use (if any)? Is it obvious what you would type?

Copyright (c) Cem Kaner, 1999

Editing Bugs—A Checklist

The bug editor should check the bug report for the following characteristics:

First impressions—when you first read the report:

- Is the replication sequence provided as a numbered set of steps, which tell you exactly what to do and, when useful, what you will see?
- Does the report include unnecessary information, personal opinions or anecdotes that seem out of place?
- Is the tone of the report insulting? Are any words in the report potentially insulting?
- Does the report seem too long? Too short? Does it seem to have a lot of unnecessary steps? (This is your first impression—you might be mistaken. After all, you haven't replicated it yet. But does it LOOK like there's a lot of excess in the report?)
- Does the report seem overly general ("Insert a file and you will see" – what file? What kind of file? Is there an example, like "Insert a file like blah.foo or blah2.fee"?)

Editing Bugs—A Checklist

The bug editor should check the bug report for the following characteristics:

When you replicate the report:

- Can you replicate the bug?
- Did you need additional information?
- Did you get lost or wonder whether you had done a step correctly? Would additional feedback (like, "the program will respond like this...") have helped?
- Did you have to guess about what to do next?
- Did you have to change your configuration or environment in any way that wasn't specified in the report?
- Did some steps appear unnecessary? Were they unnecessary?
- Did the description accurately describe the failure?
- Did the summary accurate describe the failure?
- Does the description include non-factual information (such as the tester's guesses about the underlying fault) and if so, does this information seem credible and useful or not?

Copyright (c) Cem Kaner, 1999

Editing Bugs—A Checklist

The bug editor should check the bug report for the following characteristics:

3. Closing impressions:

- Does the description include non-factual information (such as the tester's guesses about the underlying fault) and if so, does this information seem credible and useful or not? (The report need not include information like this. But it should not include non-credible or non-useful speculation.)
- Does the description include statements about why this bug would be important to the customer or to someone else? (The report need not include such information, but if it does, it should be credible, accurate, and useful.)

Editing Bugs—A Checklist

The bug editor should check the bug report for the following characteristics:

4. Follow-up tests:

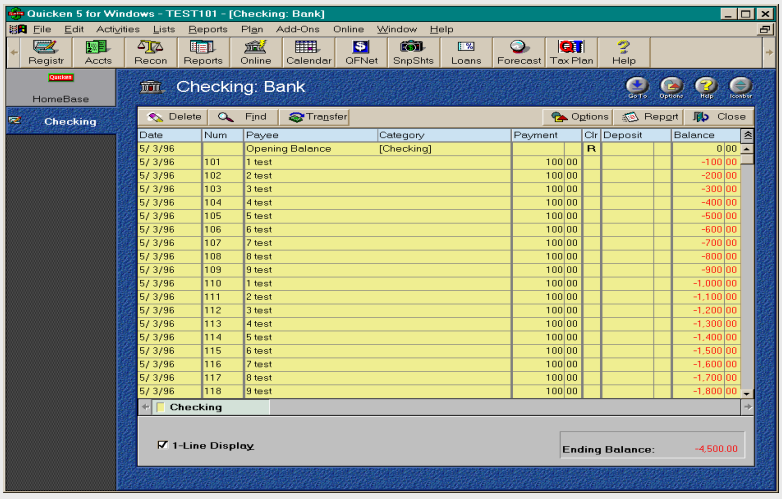
- Are there follow-up tests that you would run on this report if you had the time? (*Refer to the slides on follow-up testing)?*
- What would you hope to learn from these tests?
- How important would these tests be?
- You will probably NOT have time to run follow-up tests yourself, or if you run any, you will not / should not take the time to run more than 1 or 3 such tests.
- Are some tests so obvious that you feel the reporter should run them before resubmitting the bug? Can you briefly describe them to the reporter?
- Some obvious style issues that call for follow-up tests. The report describes a corner case without apparently having checked non-extreme values. Or the report relies on other specific values, with no indication about whether the program just fails on those or on anything in the same class (what is the class?) Or the report is so general that you doubt its accuracy ("Insert any file at this point" – *really? Any file? Any type of file? Any size?* Maybe this is accurate, but are there examples or other reasons for you to believe this generalization is credible?)

Copyright (c) Cem Kaner, 1999

Black Box Testing

Bug Reporting Exercises

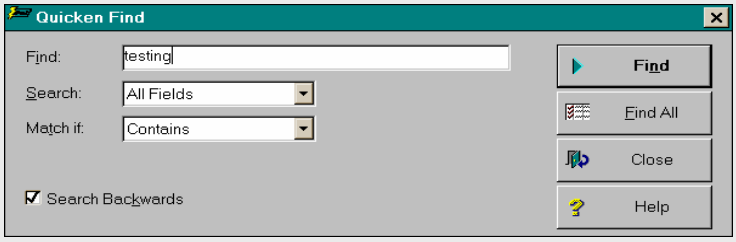
Bug Reporting Exercise 1 (1)



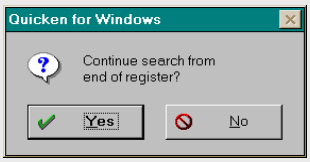
Create a sample database of cheques. Enter many new cheques.

Copyright (c) Cem Kaner, 1999

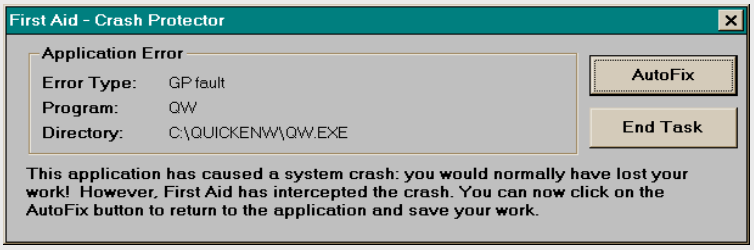
Bug Reporting Exercise 1 (2)



Now search the cheques to find one. Here, I searched for the word "testing". The program searches backwards, from the currently selected cheque to the start of the register. It doesn't find any instances of "testing" so it asks whether it should keep searching from the end of the register backwards.



Bug Reporting Exercise 1 (3)

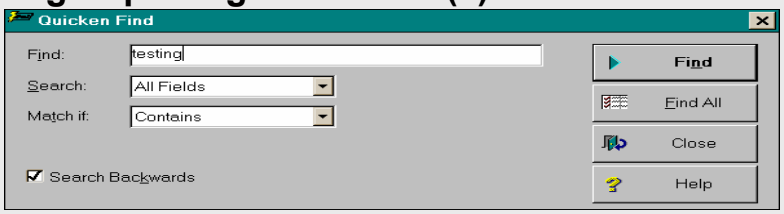


Kaboom! A General Protection Fault!

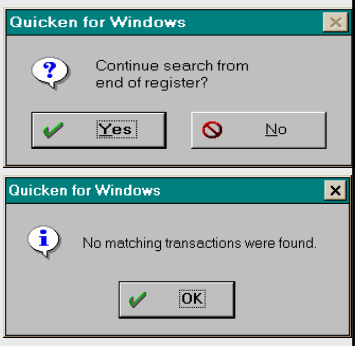
- The "First Aid" application tries to protect the customer from losing data when there is a GP fault. It's always *possible* that the crash was *caused* by an interaction between Quicken and First Aid, so try the test again after turning off First Aid.
- When I re-ran the test, Quicken crashed again, with a Win 95 system window that identifies a GP Fault. (These are harder to screen shoot, so it's not here.) Therefore the bug was not due to First Aid.

Copyright (c) Cem Kaner, 1999

Bug Reporting Exercise 1 (4)



When analyzing a bug, it's wise to try to recreate it on another computer. I did that. This time, the search didn't crash. The crashing computer is a Pentium with 32 megs RAM, a Logitech trackball, the MS keyboard, a 1.6 gig hard drive, no disk compression, a 4 meg high res MPEG video card and a big monitor. The other is an 8 meg 486 with an MS Mouse, an old standard keyboard, a 540 meg hard drive (compressed) and basic SVGA video.



Bug Reporting Exercise 1 (5)

Because this is a crash, you decide to get it into the tracking system right away. You'll do more troubleshooting later. So here is your assignment.

1. **Write these two sections of the bug report:**
   - Problem Summary
   - Problem Description
2. **What other tests should you run? Why? Write down your list.**
3. **Meet with your group to read each other's reports.**
   - How good is the summary?
   - How clear is the description?
   - How complete is the description?
   - How accurate is the description?
   - How promising is your list of ideas?

Copyright (c) Cem Kaner, 1999

Notes

Notes

Copyright (c) Cem Kaner, 1999

Notes on Exercise

I do some analysis before writing. Here's a structure for making your notes:

OBSERVED FAILURES	**CONDITIONS**
■ *General protection fault*	■ *search for non-existent text* ■ *search backwards* ■ *Yes to query, search from end of register*
OTHER CONDITIONS (maybe irrelevant) ■ *Configurations (list them all)*	**NOTES**

Notes on Exercise

MY SUMMARY

GPF on search for non-existent text. (Configuration dependent.)

MY PROBLEM DESCRIPTION

1. *Start the program*
2. *Open a database (I used the TESTING file)*
3. *Search (backwards) for a string that doesn't appear in the database*
4. *When Quicken asks whether to search from end of register, click YES*
5. *Result = GPF*

NOTES: This bug is configuration dependent. The two machines involved are the two at my desk, if you need to replicate while I'm gone. I'll do further analysis later, but I put this into the database now in order to give you an early warning of a serious bug. The configurations of the two machines are:

Replicates	*Fails to Replicate*
Pentium	*486*
etc	*etc*

Copyright (c) Cem Kaner, 1999

Notes

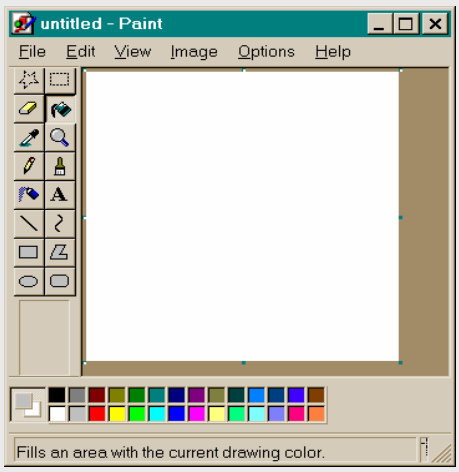
Bug Reporting Exercise 2 (1)

- **The following group of slides are from Windows Paint 95. Please don't spend your time replicating the steps or the bug. (You're welcome to do so if you are curious, but it is not necessary for analysis of this exercise.)**
- **Treat the steps that follow as fully reproducible. If you go back to ANY step, you can reproduce it.**
- **In case you aren't familiar with paint programs, the key idea is that you lay down dots. For example, when you draw a circle, the result is a set of dots, not an object. If you were using a draw program, you could draw the circle and then later select the circle, move it, cut it, etc. In a paint program, you cannot select the circle once you've drawn it. You can select an area that includes the dots that make up the circle, but that area is simply a bitmap and none of the dots in it have any relationship to any of the others.**

Copyright (c) Cem Kaner, 1999

Bug Exercise 2 Continued

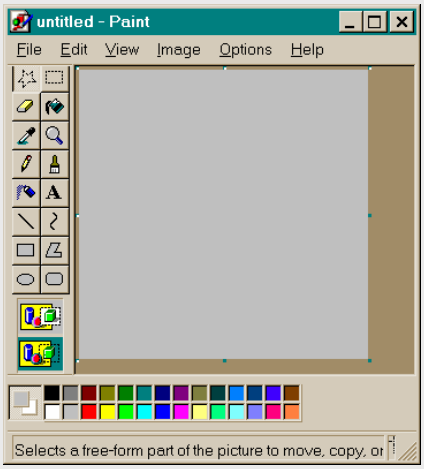
Here's the opening screen. The background is white. The first thing that we'll do is select the Paint Can
We'll use this to lay down a layer of grey paint on top of the background. Then, when we cut or move an area, we'll see the white background behind what was moved.



Bug Exercise 2 Continued

Here's the screen again, but the background has been painted gray.

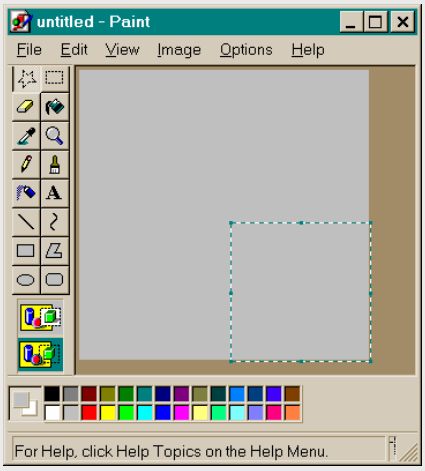
The star in the upper left corner is a freehand selection tool. After you click on it, you can trace around any part of the picture. The tracing selects that part of the picture. Then you can cut it, copy it, move it, etc.



Copyright (c) Cem Kaner, 1999

Bug Exercise 2 Continued

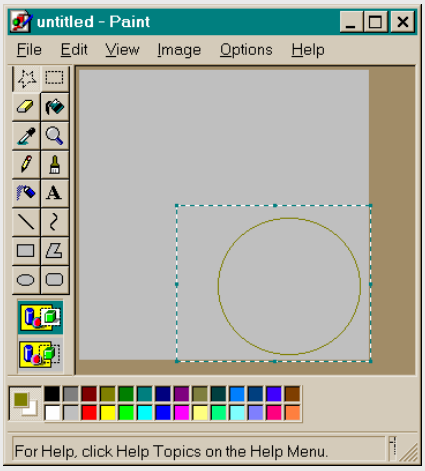
This shows an area selected with the freehand selection tool. The bottom right corner is selected. (The dashed line surrounds the selected area.)
NOTE: The actual area selected might not be perfectly rectangular. The freehand tool shows a rectangle that is just big enough to enclose the selected area. For our purposes, this is not a bug. This is a design decision by Microsoft.



Bug Exercise 2 Continued

Next, we'll draw a circle (so you can see what's selected), then use the freehand select tool to select the area around it.

When you use the freehand selection tool, you select an area by moving the mouse. The real area selected is not a perfect rectangle. The rectangle just shows us where the selected area is.

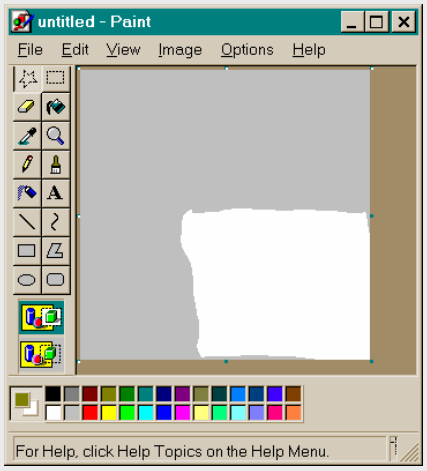


Copyright (c) Cem Kaner, 1999

Bug Exercise 2 Continued

Now we cut the selection. (To do this, press Ctrl-X.)

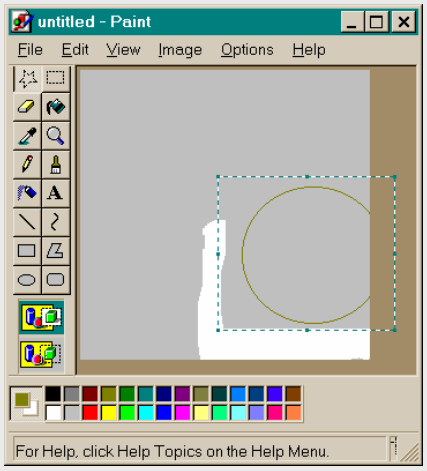
The jagged border shows exactly the area that was selected.



Bug Exercise 2 Continued

Next, select the area around the circle and drag it up and to the right.

This works.



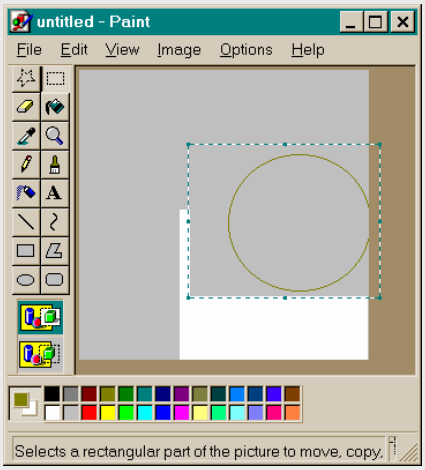
Copyright (c) Cem Kaner, 1999

Bug Exercise 2 Continued

This time, we'll try the Rectangular Selection tool.

With this one, if you move the mouse to select an area, the area that is actually selected is the smallest rectangle that encloses the path that your mouse drew.

So, draw a circle, click the Rectangular Selection tool, select the area around the circle and move it up. It works.



Bug Exercise 2 Continued

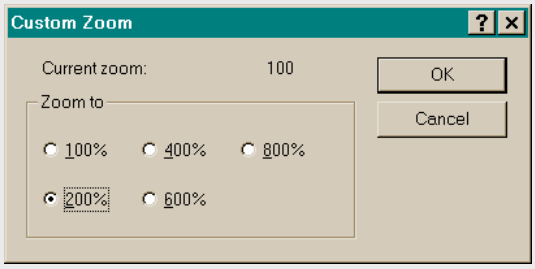
Well, this was just too boring, because everything is working. When you don't find a bug while testing a feature, one tactic is to keep testing the feature but combine it with some other test.

In this case, we'll try Zooming the image. When you zoom 200%, the picture itself doesn't change size, but the display doubles in size. Every dot is displayed as twice as tall and twice as wide.

Copyright (c) Cem Kaner, 1999

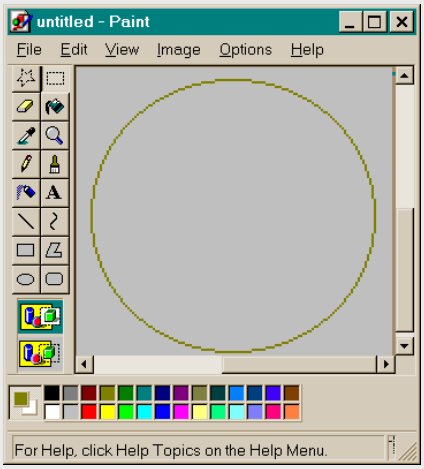
Bug Exercise 2 Continued

Bring up the Custom Zoom dialog, and select 200% zoom, click OK.



Bug Exercise 2 Continued

It worked. The paint area is displayed twice as tall and twice as wide. We're looking at the bottom right corner. To see the rest, we could move the scroll bars up or left.



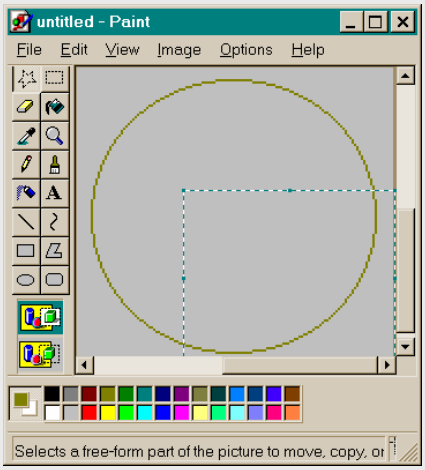
Copyright (c) Cem Kaner, 1999

Bug Exercise 2 Continued

So, we select part of the circle using the freehand selection tool. We'll try the move and cut features.

Cutting fails.

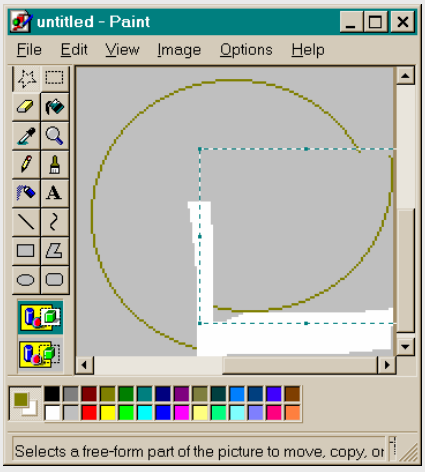
When we try to cut the selection, the dashed line disappears, but nothing goes away.



Bug Exercise 2 Continued

Draw the circle, zoom to 200%, select the area.

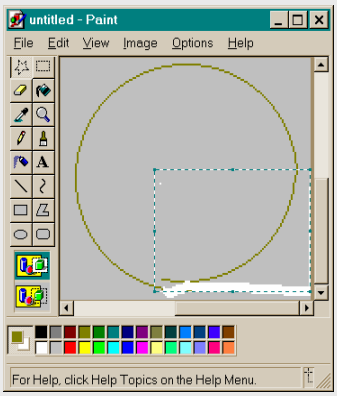
Drag the area up and to the right. It works.

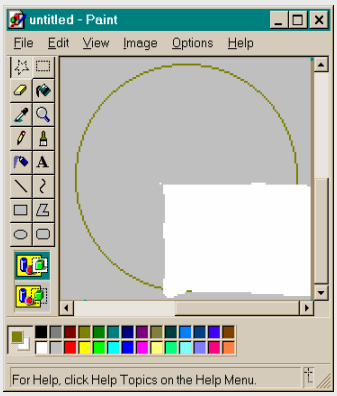


Copyright (c) Cem Kaner, 1999

Bug Exercise 2 Continued

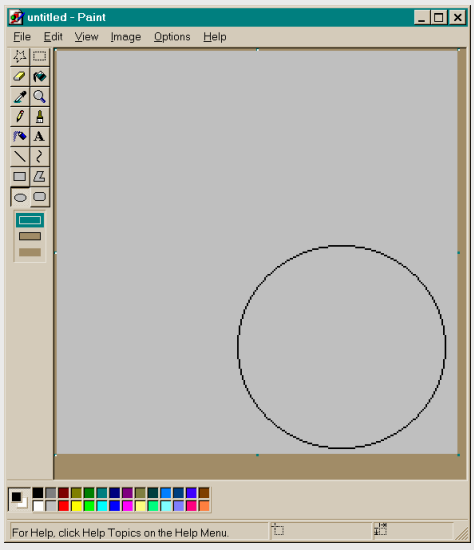
Draw the circle, zoom to 200%, select the area. Now try this. Select the area and move it a bit. THEN press Ctrl-X to cut. This time, cutting works.





Bug Exercise 2 Continued

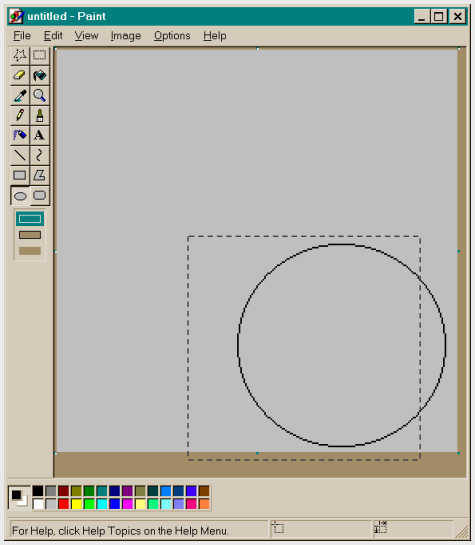
Draw the circle, zoom to 200%, and this time, *grow the window* so you can see the whole drawing area.



Copyright (c) Cem Kaner, 1999

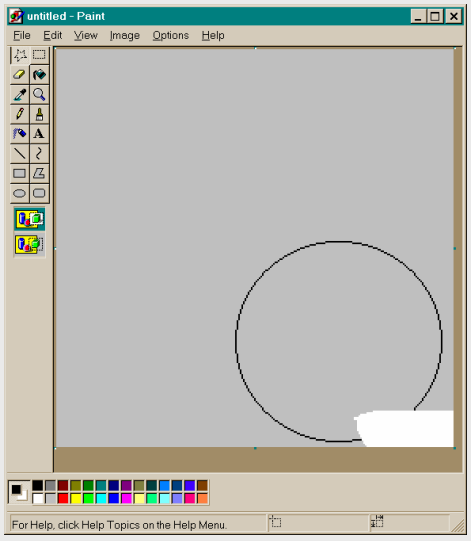
Bug Exercise 2 Continued

Now, select the circle. That seems to work.



Bug Exercise 2 Continued

But when you press Ctrl-X to cut the circle, the program cuts the wrong area.



Copyright (c) Cem Kaner, 1999

Bug Exercise 2 Continued

Now, write a bug report. I want two sections:

- **The Problem summary (or title)**
- **The Problem Description (how to reproduce the problem)**

Additionally, please describe three follow-up tests that you would run with this bug

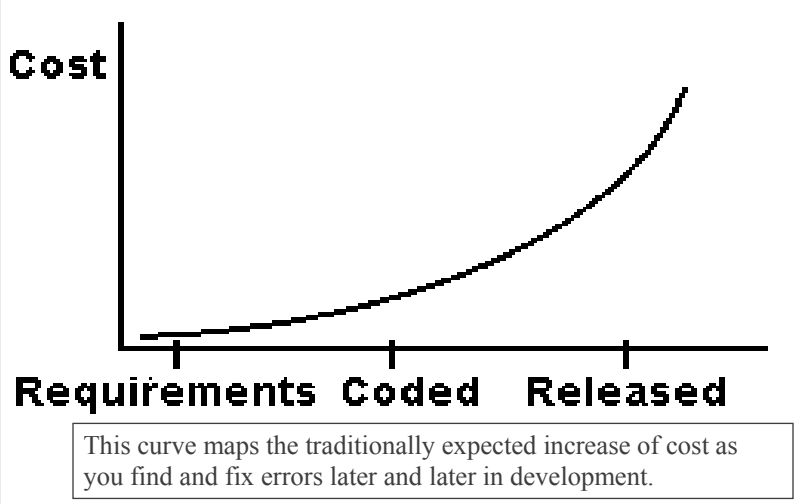
Notes

Bug Advocacy

Advocating for bug fixes by alerting people to costs.

Supplementary Reading:
Kaner, *Quality Cost Analysis: Benefits & Risks.*

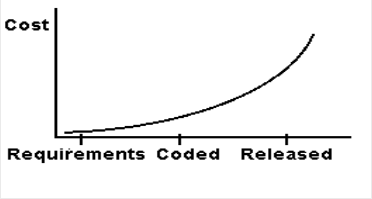
Money Talks:
Cost of Finding and Fixing Software Errors



This curve maps the traditionally expected increase of cost as you find and fix errors later and later in development.

Copyright (c) Cem Kaner, 1999

Money Talks:
Cost of Finding and Fixing Software Errors



This is the most commonly taught cost curve in software engineering.
Usually people describe it from the developers-eye view. That is, the discussion centers around

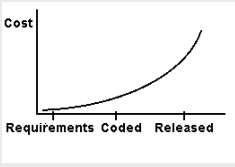
- how much it costs to find the bug
- how much it costs to fix the bug
- and how much it costs to distribute the bug fix.

But sometimes, it pays to adopt the viewpoints of other stakeholders, who might stand to lose more money than the development and support organizations.

Money Talks:
Cost of Finding and Fixing Software Errors

Costs escalate because more people in and out of the company are affected by bugs, and more severely affected, as the product gets closer to release. We all know the obvious stuff

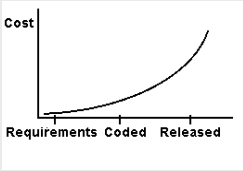
- if we find bugs in requirements, we can fix them without having to recode anything;
- programmers who find their own bugs can fix them without taking time to file bug reports or explain them to someone else;
- it is hugely expensive to deal with bugs in the field (in customers' hands).



Along with this, there are many effects on other stakeholders in the company. For example, think of the marketing assistant who wastes days trying to create a demo, but can't because of bugs.

Copyright (c) Cem Kaner, 1999

Money Talks: Cost of Finding and Fixing Errors



It is important to recognize that this cost curve is predicated on a family of development practices.

When you see a curve that says,

"Late changes are expensive"

you can reasonably respond in either of two ways:

- Make fewer late changes.
  - *This is the traditional recommendation*
- **Make it cheaper to make late changes.**
  - *This is a key value of the agile development movement (see Beck's Extreme Programming Explained, or go to www.agilealliance.org)*

In this testing course, I will push you to find ways to find bugs earlier, but my development philosophy is agile.

Quality Cost Analysis

Quality Cost Measurement is a cost control system used to identify opportunities for reducing the controllable quality-related costs

The *Cost of Quality* is the total amount the company spends to achieve and cope with the quality of its product.

This includes the company's investments in improving quality, and its expenses arising from inadequate quality.

A key goal of the quality engineer is to help the company minimize its cost of quality.

- Refer to my paper, "Quality Cost Analysis: Benefits & Risks" available at www.kaner.com.

Copyright (c) Cem Kaner, 1999

Quality-Related Costs

Prevention	*Appraisal*
Cost of preventing customer dissatisfaction, including errors or weaknesses in software, design, documentation, and support.	Cost of inspection (testing, reviews, etc.).
Internal Failure	***External Failure***
Cost of dealing with errors discovered during development and testing. Note that the company loses money as a user (who can't make the product work) and as a developer (who has to investigate, and possibly fix and retest it).	Cost of dealing with errors that affect your customers, after the product is released.

Examples of Quality Costs

Prevention	Appraisal
■ Staff training ■ Requirements analysis & early prototyping ■ Fault-tolerant design ■ Defensive programming ■ Usability analysis ■ Clear specification ■ Accurate internal documentation ■ Pre-purchase evaluation of the reliability of development tools	■ Design review ■ Code inspection ■ Glass box testing ■ Black box testing ■ Training testers ■ Beta testing ■ Usability testing ■ Pre-release out-of-box testing by customer service staff
Internal Failure	**External Failure**
■ Bug fixes ■ Regression testing ■ Wasted in-house user time ■ Wasted tester time ■ Wasted writer time ■ Wasted marketer time ■ Wasted advertisements ■ Direct cost of late shipment ■ Opportunity cost of late shipment	■ Lost sales and lost customer goodwill ■ Technical support calls ■ Writing answer books (for Support) ■ Investigating complaints ■ Supporting multiple versions in the field ■ Refunds, recalls, warranty, liability costs ■ Interim bug fix releases ■ Shipping updated product ■ PR to soften bad reviews ■ Discounts to resellers

Copyright (c) Cem Kaner, 1999

Customers' Quality Costs

Seller: external costs	Customer: failure costs (seller's externalized costs)
These illustrate costs absorbed by the seller that releases a defective product.	***These illustrate costs absorbed by the customer who buys a defective product.***
■Lost sales and lost customer goodwill ■Technical support calls ■Writing answer books (for Support) ■Investigating complaints ■Refunds, recalls, warranty, liability costs ■Government investigations ■Supporting multiple versions in the field ■Interim bug fix releases ■Shipping updated product ■PR to soften bad reviews ■Discounts to resellers	■Wasted time ■Lost data ■Lost business ■Embarrassment ■Frustrated employees quit ■Failure during one-time-only tasks, e.g. demos to prospective customers ■Cost of replacing product ■Reconfiguring the system ■Cost of recovery software ■Tech support fees ■Injury / death

Influencing Others Based on Costs

It's often impossible to fix every bug. But sometimes the development team will choose to not fix a bug based on their assessment of its risks for them, without thinking of the costs to other stakeholders in the company.

- **Probable tech support cost.**
- **Risk to the customer.**
- **Risk to the customer's data or equipment.**
- **Visibility in an area of interest to reviewers.**
- **Extent to which the bug detracts from the use of the program.**
- **How often will a customer see it?**
- **How many customers will see it?**
- **Does it block any testing tasks?**
- **Degree to which it will block OEM deals or other sales.**

To argue against a deferral, ask yourself which stakeholder(s) will pay the cost of keeping this bug. Flag the bug to them.

Copyright (c) Cem Kaner, 1999

Bug Advocacy

What About the Objection that it's Not Really a Bug?

Really, it's a feature.

Or, at least, it's not a problem for my release so I don't have to fix it.

It won't matter until we ship it to Germany. Let them fix it.

Supplemental reading: Kaner, What is a Software Defect?

Software Errors: What is Quality?

Here are some of the traditional definitions:

- **Fitness for use (Dr. Joseph M. Juran)**
- **The totality of features and characteristics of a product that bear on its ability to satisfy a given need (ASQ)**
- **Conformance with requirements (Philip Cosby)**
- **The total composite product and service characteristics of marketing, engineering, manufacturing and maintenance through which the product and service in use will meet expectations of the customer (Armand V. Feigenbaum)**

Note the absence of "conforms to specifications."

Copyright (c) Cem Kaner, 1999

Software Errors: What is Quality?

Juran distinguishes between *Customer Satisfiers* and *Dissatisfiers* as key dimensions of quality:

- ***Customer Satisfiers***
  - the right features
  - adequate instruction
- ***Dissatisfiers***
  - unreliable
  - hard to use
  - too slow
  - incompatible with the customer's equipment

Software Errors: What Should We Report?

I like Gerald Weinberg's definition:

Quality is value to some person

But consider the implication:

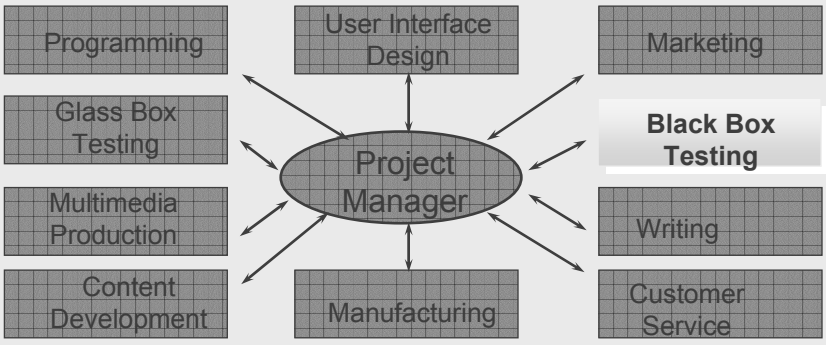
- It's appropriate to report any deviation from high quality as a software error.
- Therefore many issues will be reported that will be errors to some and non-errors to others.

Glen Myers' definition:

- **A software error is present when the program does not do what its user reasonably expects it to do.**

Copyright (c) Cem Kaner, 1999

Quality is Multidimensional



When you sit in a project team meeting, discussing a bug, a new feature, or some other issue in the project, you must understand that each person in the room has a different vision of what a "quality" product would be. Fixing bugs is just one issue.
The next slide gives some examples.

Quality is Multidimensional: Different People, Different Visions

- **Localization Manager:** *A good product is easy to translate and to modify to make it suitable for another country and culture. Few expereienced localization managers would consider acceptable a product that must be recompiled or relinked to be localized.*
- **Tech Writers:** *A high quality program is easily explainable. Aspects of the design that are confusing, unnecessarily inconsistent, or hard to describe are marks of bad quality.*
- **Marketing:** *Customer satisfiers are the things that drive people to buy the product and to tell their friends about it. A Marketing Manager who is trying to add new features to the product generally believes that he is trying to improve the product.*
- **Customer Service:** *Good products are supportable. They have been designed to help people solve their own problems or to get help quickly.*
- **Programmers:** *Great code is maintainable, well documented, easy to understand, well organized, fast and compact.*

Copyright (c) Cem Kaner, 1999

Software Errors: What Kind of Error?

You will report all of these types of problems, but it's valuable to keep straight in your mind, and on the bug report, which type you're reporting.

- **Coding Error**: ***The program doesn't do what the programmer would expect it to do.***
- **Design Issue**: ***It's doing what the programmer intended, but a reasonable customer would be confused or unhappy with it.***
- **Requirements Issue**: ***The program is well designed and well implemented, but it won't meet one of the customer's requirements.***
- **Documentation / Code Mismatch**: ***Report this to the programmer (via a bug report) and to the writer (usually via a memo or a comment on the manuscript).***
- **Specification / Code Mismatch**: ***Sometimes the spec is right; sometimes the code is right and the spec should be changed.***

Software Errors: Why are there Errors?

New testers often conclude that the programmers on their project are incompetent or unprofessional.

- **This is counterproductive. It leads to infighting instead of communication, and it leads to squabbling over bugs instead of research and bug fixing.**
- **And as we saw when we discussed private bug rates, programmers actually find and fix the large majority of their own bugs.**
- **Bugs come into the code for many reasons. It's worth considering some common systematic (as distinct from poor individual performance) factors. You will learn to vary your strategic approaches as you learn your companies' weaknesses.**

Copyright (c) Cem Kaner, 1999

Software Errors: Why are there Errors?

Bugs come into the code for many reasons:

- **The major cause of error is that programmers deal with tasks that aren't fully understood or fully defined. This is said in many different ways. For example:**
  - Tom Gilb and Dick Bender quote industry-summary statistics that 80% of the errors, or 80% of the effort required to fix the errors, are caused by bad requirements;
  - Roger Sherman recently summarized research at Microsoft that the most common underlying issue in bug reports involved a need for new code.

  If you graduated from a Computer Science program, how much training did you have in task analysis? Requirements definition? Usability analysis? Negotiation and clear communication of negotiated agreements? Not much? Hmmmm . . . .

Software Errors: Why are there Errors?

- **Some companies drive their programmers too hard. They don't have enough time to design, bulletproof, or test their code. Another Sherman quote: "Bad schedules are responsible for most quality problems."**
- **Late design changes result in last minute code changes, which are likely to have errors.**
- **Some third-party components introduce bugs. Your program might rely on a large set of components that display a specific type of object, filter data in a special way, drive a specific printer, etc. Many of these, bought from tool vendors or hardware vendors, are surprisingly buggy. Others work, but aren't compatible with common test automation tools.**
- **Failure to use source control tools creates characteristic bugs. For example, if a bug goes away, comes back, goes away, comes back, goes away, then ask about your company's source control or configuration management process.**

Copyright (c) Cem Kaner, 1999

Software Errors: Why are there Errors?

- **Some programs or tasks are inherently complex. Boris Beizer talks perceptively about the locality problem in software. Think about an underlying bug, and then about symptoms caused by the bug. When symptoms appear, there's no assurance that they'll be close in time, space, or severity to the underlying bug. They may appear much later, or when working with a different part of the program, and they may seem much more or much less serious than the bug.**
- **Some programmers (some platforms) work with poor tools. Weak compilers, style checkers, debuggers, profilers, etc. make it too easy to get bugs or too hard to find bugs.**
- **Similarly, some third party hardware, or its drivers, are non-standard and don't respond properly to standard system calls. Incompatibility with hardware is often cited as the largest single source of customer complaints into technical support groups.**
- **When one programmer tries to fix bugs, or otherwise modify another programmer's code, there's lots of room for miscommunication and error.**
- **And, sometimes people just make mistakes.**

Family Drug Store v. Gulf States Computer

(563 So.2d 1324, Louisiana Court of Appeal, 1990). *The basic holding of this case is that a computer program that is honestly marketed can be extremely awkward to use without imposing liability on the seller.*

Two pharmacists bought a computer program known as the Medical Supply System from Gulf States. After they realized what they had bought, they asked for, and then sued for, a refund. Here were some of the problems of the system:

"1 all data had to be printed out, and could not be viewed on the monitor;

2 the information on the monitor would appear in code;

3 numerical codes were needed in order to open a new patient file

4 the system was unable to scroll."

The court found that the seller had not in any way misrepresented the system, and that it was not useless even though it was awkward to use. Further, the price of the software was about $2500 compared to $10,000 for other packages. *The plaintiffs had gotten what they'd paid for.*

Copyright (c) Cem Kaner, 1999

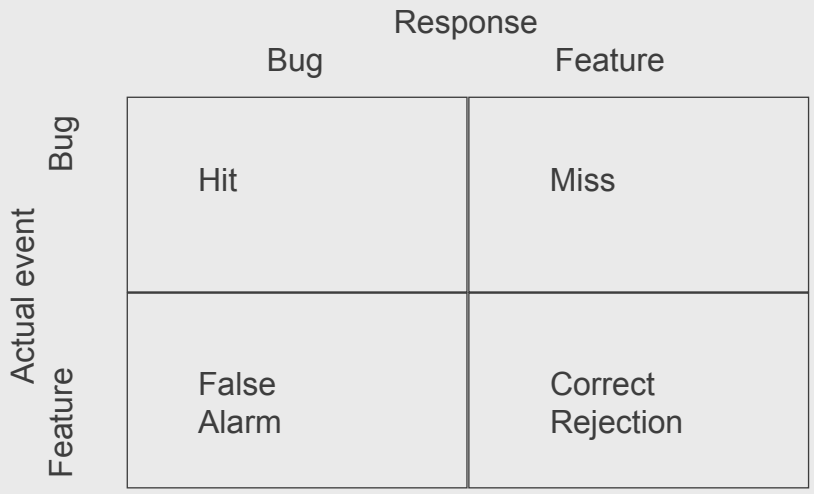
Notes

Bug Advocacy

Decision Making, Information Flow, and Credibility

Copyright (c) Cem Kaner, 1999

The Signal Detection & Recognition Problem



Refer to Testing Computer Software, pages 24, 116-118

Lessons From Signal Detection: We Make Decisions Under Uncertainty

When you try to decide whether an item belongs to one category or the other (bug or feature), your decision will be influenced by your expectations and your motivation.

- **Can you cut down on the number of false alarms without increasing the number of misses?**
- **Can you increase the number of hits without increasing the number of false alarms?**
- **Pushing people to make fewer of one type of reporting error will inevitably result in an increase in another type of reporting error.**
- **Training, specs, etc. help, but the basic problem remains.**

Copyright (c) Cem Kaner, 1999

Lessons From Signal Detection: Decisions Are Subject To Bias

We make decisions under uncertainty.

Decisions are subject to bias, and much of this is unconscious.

The prime biasing variables are:

- *perceived probability:*

  If you think that an event is unlikely, you will be *substantially* less likely (beyond the actual probability) to report it.
- *perceived consequence of a decision:*

  What happens if you make a False Alarm? Is this worse than a Miss or less serious?
- *perceived importance of the task:*

  The degree to which you care / don't care can affect your willingness to adopt a decision rule that you might otherwise be more skeptical about

Lessons From Signal Detection: Decisions Are Subject To Bias

Decisions are made by a series of people.

- Bug reporting policies must consider the effects on the overall decision-making system, not just on the tester and first-level bug reader.

Trace these factors through the decisions and decision-makers (next slides). For example, what happens to your reputation if you

- **Report every bug, no matter how minor, in order to make sure that no bug is ever missed?**
- **Report only the serious problems (the "good bugs")?**
- **Fully analyze each bug?**
- **Only lightly analyze bugs?**
- **Insist that every bug get fixed?**

Copyright (c) Cem Kaner, 1999

Decisions Made During Bug Processing

Bug handling involves many decisions by different people, such as:

Tester:

- **Should I report this bug?**
- **Should I report these similar bugs as one bug or many?**
- **Should I report this awkwardness in the user interface?**
- **Should I stop reporting bugs that look minor?**
- **How much time should I spend on analysis and styling of this report?**

Your decisions will reflect on you. They will cumulatively have an effect on your credibility, because they reflect your judgment.

The comprehensibility of your reports and the extent and skill of your analysis will also have a substantial impact on your credibility.

Refer to Testing Computer Software, pages 90-97, 115-118

Decisions Made During Bug Processing-2

Bug handling involves many decisions by different people, such as

- ***Programmer*:**
  - **Should I fix this bug or defer it?**
- ***Project Manager:***
  - **Should I approve the deferral of this bug?**
- ***Tester*:**
  - **Should I appeal the deferral of this bug?**
  - **How much time should I spend analyzing this bug further?**
- ***Test Group Manager:***
  - **Should I make an issue about this bug?**
  - **Should I encourage my tester to**
    - investigate the bug further
    - argue the bug further,
    - or to quit worrying about this one,
    - or should I just keep out of the discussion this time?

Refer to Testing Computer Software, pages 90-97, 115-118

Copyright (c) Cem Kaner, 1999

Decisions Made During Bug Processing - 3

- *Customer Service, Marketing, Documentation:*
  - **Should I ask the project manager to reopen this bug?**
  - **(The tester appealed the deferral) Should I support the tester this time?**
  - **Should I spend time trying to figure this thing out?**
  - **Will this call for extra work in the answer book / advertising / manual / help?**
- *Director, Vice President, other senior staff*:
  - **Should I override the project manager's deferral of this bug?**

Decisions Made During Bug Processing - 4

Who else is in your decision loop?

Copyright (c) Cem Kaner, 1999

Issues That Bias People Who Evaluate Bug Reports

These reduce the probability that the bug will be taken seriously and fixed.

- Language critical of the programmer.
- Severity inflation.
- Pestering & refusing to ever take "No" for an answer.
- Tight schedule.
- Incomprehensibility, excessive detail, or apparent narrowness of the report.
- Weak reputation of the reporter.

Issues That Bias People Who Evaluate Bug Reports

These increase the probability that the bug will be taken seriously and fixed.

- Reliability requirements in this market.
- Ties to real-world applications.
- Report from customer/beta rather than from development.
- Strong reputation of the reporter.
- Weak reputation of the programmer.
- Poor quality/performance comparing to competitive product(s).
- News of litigation in the press.

Copyright (c) Cem Kaner, 1999

Clarify Expectations

One of the important tasks of a test manager is to clarify everyone's understanding of the use of the bug tracking database and to facilitate agreements that this approach is acceptable to the stakeholders.

- **Track open issues / tasks or just bugs?**
- **Track documentation issues or just code?**
- **Track minor issues late in the schedule or not?**
- **Track issues outside of the published spec and requirements or not?**
- **How to deal with similarity?**

Make the rules explicit.

Biasing People Who Report Bugs

These will reduce the probability that bugs will be reported, by discouraging reporters, by convincing them that their work is pointless or will be filtered out, or by creating incentives for other people to pressure people not to report bugs.

- **Never use bug statistics for employee bonus or discipline.**
- **Never use bug statistics to embarrass people.**
- **Never filter reports that you disagree with.**
- **Never change an in-house bug reporter's language, or at least not without free permission. Add your comments as additional notes, not as replacement text.**
- **Monitor language in the reports that is critical of the programmer or the tester.**
- **Beware of accepting lowball estimates of bug probabilities.**

Copyright (c) Cem Kaner, 1999

Biasing People Who Report Bugs

These help increase the probability that people will report bugs.

- **Give results feedback to non-testers who report bugs.**
- **Encourage testers to report all anomalies.**
- **Adopt a formal system for challenging bug deferrals.**
- **Weigh schedule urgency consequences against an appraisal of quality costs. (Early in the schedule, people will report more bugs; later people will be more hesitant to report minor problems).**
- **Late in the schedule, set up a separate database for design issues (which will be evaluated for the start of the next release).**

Copyright (c) Cem Kaner, 1999

QW2002 Paper SB4

Hans Buwalda
(LogiGear Corporation)

The Essentials of Testing and Test Automation

Key Points

- Typical challenges of testing and test automation
- "Essentials", key considerations for success
- Benefits and consequences of applying or ignoring essentials

Presentation Abstract

Testing and test automation are among the important and at the same time most difficult tasks in IT. Effective test design and efficient test automation are key to the success of many organizations, even at the business level. However, they are often a constant source of costly nightmares and are often neglected by management sometimes purely out of frustration.

If you look carefully at many of the projects that go on it turns out that the a relatively small set of key issues re-appear over and over again, in many disguises. One could call them the "essentials". These essentials are equally divided among:
• Test development
• Test automation
• Test management

This talk will present what, from the perspective of the speaker, are the essential considerations to be successful with testing and test automation. For each of the considerations attention will be paid to what the benefits are in applying it and the typical pitfalls if you don't!

The considerations are based on 8 years of experiences in numerous projects, many of them using earlier presented concepts like the actions words and soap opera testing. The audience is invited to take the essentials home and use them to become more successful.

About the Author

Hans Buwalda, ABT Chief Architect, leads LogiGear's Action Based Testing™ research and development, including ABT Toolset™ operations, and oversees the practice of ABT methodology. Prior to joining LogiGear, Hans served as Project Director at CMG The Netherlands where he was the original architect behind the Action Words approach - an integrated method for planning, managing, and deploying software testing and test automation. Hans is an internationally recognized expert specializing in action-based test automation, test development, and test-technology management. He speaks and presents workshops at international conferences on testing concepts such as Action Based Testing, The Three Holy Grails of Test Development, Soap Opera Testing, and Testing in the Cold. Hans authored (along with Dennis Janssen and Iris Pinkster) Integrated Test Design and Automation (Addison Wesley 2001). He holds a Master of Science degree in Computer Science from Free University, Amsterdam.

LogiGear Expert Testing... Real World Solutions.

QualityWeek 2002

Process Track Standby Presentation

Essentials of Testing and Test Automation

Hans Buwalda

LogiGear® Corporation

© 2002 Logigear Corporation, All Rights Reserved.

Objectives of this Presentation

- To share some (personal) views
- Not necessarily the "final answers"
  - practice makes modest
- No pretension of completeness
- Little details (see also the book refs at the end)
- Discussion invited (now or later)

© 2002 LogiGear Corporation. All Rights Reserved.

Challenges for a Test Process

- Testing should be effective
- Testing should be efficient
- Testing should be fun
- Testing should be under control

© 2002 LogiGear Corporation. All Rights Reserved. 3

Questions to Answer With the Tests

1. Does the system comply to the requirements
2. Are there any problems (defects and/or failures) we should know about
3. Will the system work in practice

© 2002 LogiGear Corporation. All Rights Reserved. 4

Three Dimensions of Testing

- Test Development
- Test Automation
- Test Organization and Management
  - including people

© 2002 LogiGear Corporation. All Rights Reserved.

Test Development Essentials

- Good breakdown of the tests ("clustering")
- Smart and effective testing techniques
- Efficient ways to express the tests

© 2002 LogiGear Corporation. All Rights Reserved.

Typical Price of not Understanding

- No (sense of) control
- Tests are not smart and only moderate effective
  - ROD Testing ("Requirements driven Open Door")
- Complaining people, no fun
  - also low in the food chain
- Unnecessary high test maintenance
  - high impact of system changes on the test

© 2002 LogiGear Corporation. All Rights Reserved.

Examples of Clustering Criteria

STRAIGHTFORWARD

- Architecture of the system under test
- Functionality and other requirements
- Quality attributes
- Level of detail

ADDITIONAL

- Planning and control
- Level of risks involved
- Complexity of the test
- Technical aspects of test execution
- Stake holders
- Code hand-offs (Brian Marick)

© 2002 LogiGear Corporation. All Rights Reserved.

Test Automation Essentials

- Separation between test development and test automation
- Automate at action level, keep the "test logic" out of the scripts
- Place maximum attention to re-usability
- Keep testers in the lead
  - avoid domination by engineers

© 2002 LogiGear Corporation. All Rights Reserved.

Typical Price of not Understanding

- Testers are out of control
- Tests are hard to access and assess
- Either test automation percentage is low or the tests are uninteresting
  - like just following all controls in the GUI..

© 2002 LogiGear Corporation. All Rights Reserved.

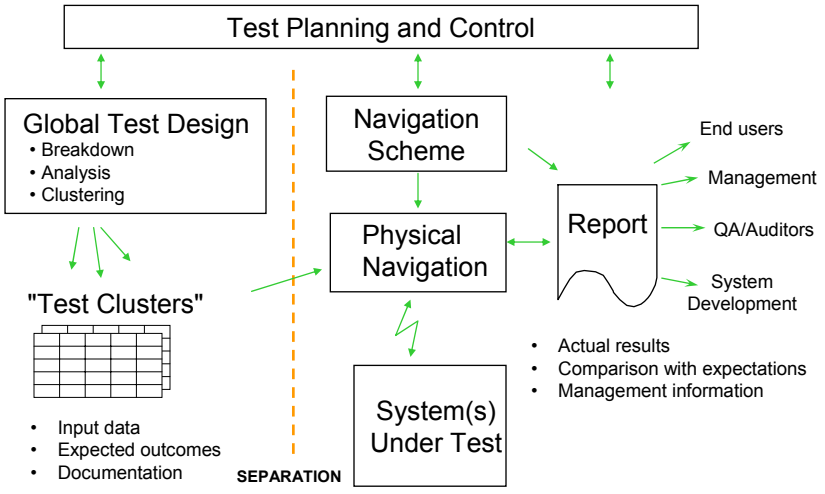
Action Based Testing™

- Based on a notion that a test can be broken down in a number of consecutive actions
- Not only the test data but also the actions are in the test file, they are defined by the tester
- Instead of implementing test cases, the engineer concentrates on the programming individual actions
  - we call this person the "navigation engineer" or "navigator"
- Reporting is done at the level of the test
- Most actions are high-level
  - test oriented instead of object oriented

enter customer	Jones
instead of	
push_button	New
type_keys	Jones
push_button	Process

© 2002 LogiGear Corporation. All Rights Reserved.

ABT Product Life Cycle



© 2002 LogiGear Corporation. All Rights Reserved.

Example of an ABT Test Cluster

cluster	**EXAMPLE OF A TEST CLUSTER**			
version	1.0			
author	Hans Buwalda			
section	**1. Entering clients and balances**			
	last name	first name	account nr	balance
enter client	Green	John	458473948	1500
enter client	Wood	Anna	422087596	2100
section	**2. Money Transfers**			
	from	to	sum	
transfer	458473948	422087596	500	
transfer	422087596	785793025	1201	
section	**3. Checking names and numbers**			
	account nr	last name	first name	
check name	458473948	Green	John	
check name	422087596	Wood	Anna	
	account nr	sum		
check balance	458473948	1000		
check balance	422087596	1399		

HEADER

INPUT

EXPECTED OUTCOMES

© 2002 LogiGear Corporation. All Rights Reserved.

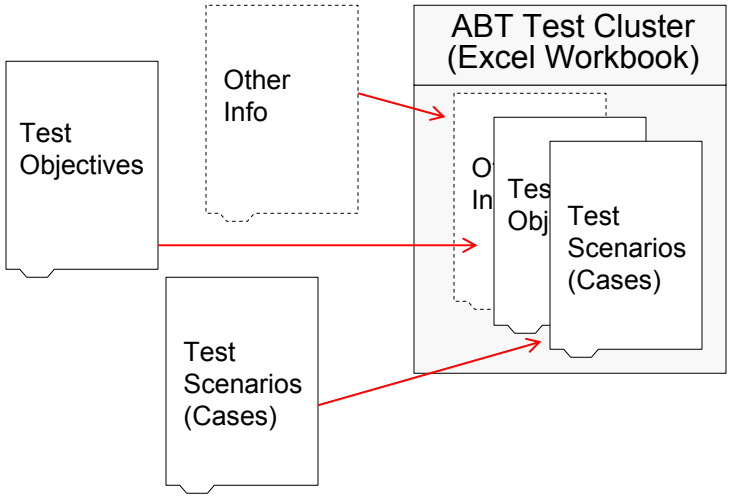
Example of a Test Objective

…
TO-3.51 The exit date must be after the entry date
…

test objective	TO-3.51		
	name	entry date	exit date
enter employment	Bill Goodfellow	2002-10-02	2002-10-01
check error message	The exit date must be after the entry date.		

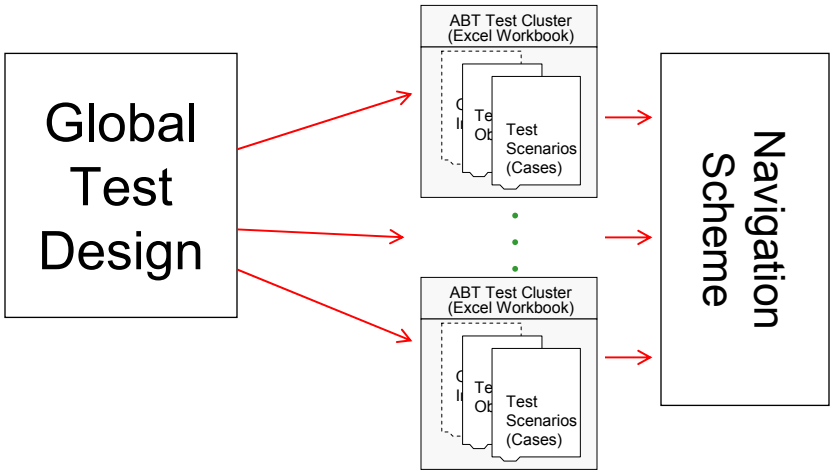
© 2002 LogiGear Corporation. All Rights Reserved.

Build Up of an ABT Test Cluster



© 2002 LogiGear Corporation. All Rights Reserved. 15

ABT Test Project



© 2002 LogiGear Corporation. All Rights Reserved. 16

Test Organization Essentials

- Use a product approach for both test development and test automation
  - see my other presentation
- Keep project level and organization level separate
  - projects need solutions
  - the organization needs improvements

© 2002 LogiGear Corporation. All Rights Reserved.

Typical Price of not Understanding

- Test product grow out of control
- Project priorities frustrate improvement processes
  - and improvement processes frustrate projects
- Legitimate Project interests are not adequately serviced

© 2002 LogiGear Corporation. All Rights Reserved.

Test Organization Essentials

- Treat test projects as difficult
  - high risk by definition
  - much external dependencies, that are hard to control
- Be suspicious about standard life-cycles
  - tend to be to simple and not fitting
- Developers and other stakeholders are not the enemy
  - neither are the testers
  - present and document tests and outcomes well

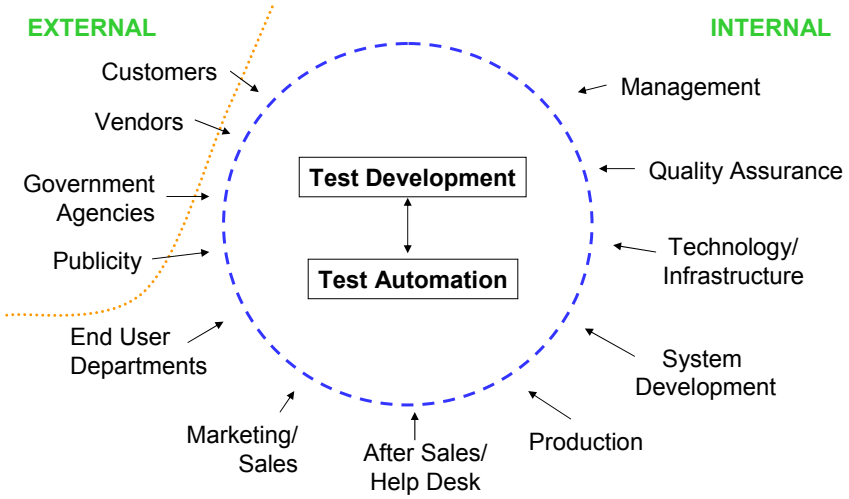
© 2002 LogiGear Corporation. All Rights Reserved.

Typical Price of not Understanding

- Quality defects remain into production
- Testing causes delays, conflicts and frustration
- Other project priorities go at the expense of testing
  - resources, time, but also environments
- Unfitting and unnecessary activities eat time and money

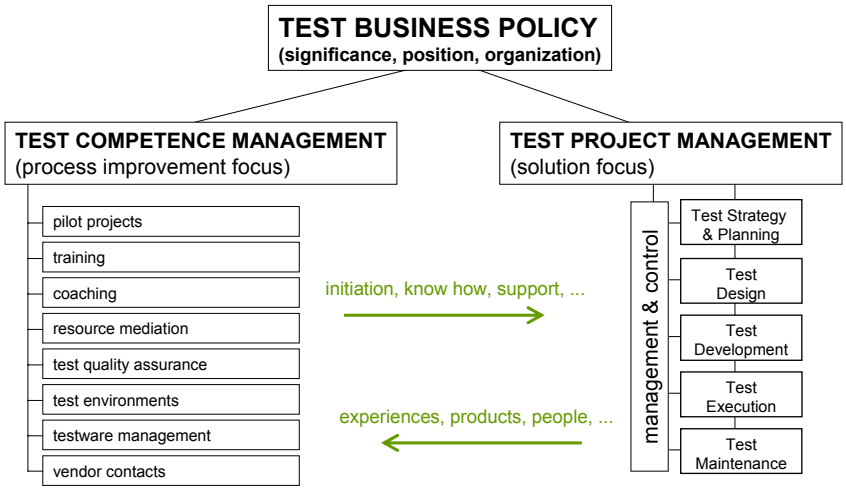
© 2002 LogiGear Corporation. All Rights Reserved.

Strategic Position of Test Processes



© 2002 LogiGear Corporation. All Rights Reserved.

ABT Test Governance Model



© 2002 LogiGear Corporation. All Rights Reserved.

Testing People

- Understanding testing
- Understanding the system under test
  - at least some of them...
- Smart, independent thinkers, able to find bugs
  - at least some of them...
- Test automators might not be the best
  - tend to be smart at automation, not at the tests

© 2002 LogiGear Corporation. All Rights Reserved.

Typical Price of not Understanding

- Test teams seem busy, but are not effective
- Outcomes are not followed up, because developers dominate
- Management looses grip on quality
  - typical symptom: extra money doesn't help

© 2002 LogiGear Corporation. All Rights Reserved.

Getting the Right Teams

- If possible use a mix of different skills
- Education might help:
  - training in testing, subject matter expertise, IT, ...
  - visits to conferences
- Introduce test leads
  - control quality
  - pull up the others
- Let people share knowledge and experiences
  - for example: setup/stimulate Special Interest Groups
- Promote cooperation with developers, users and other stakeholders

© 2002 LogiGear Corporation. All Rights Reserved.

Some Books for Further Reading

- *Lessons Learned in Software Testing*, Cem Kaner, James Bach, Bret Pettichord
  - much more lessons than in this presentation
- *Integrated Test Design and Automation*, Hans Buwalda, Dennis Janssen, Iris Pinkster
  - how to integrate action words into your approach
- *Testing Computer Software*, Cem Kaner, Jack Falk, Hung Nguyen
  - one of the "bibles" on software testing

© 2002 LogiGear Corporation. All Rights Reserved.

IN COOPERATION WITH...

The American Society for Quality is a membership organization dedicated to promoting the principles and practices of quality improvement, with a mission to "be recognized throughout the world as the leading authority on, and champion for, quality." The ASQ Software Division, which represents over 5000 professionals worldwide, is dedicated to "improve the ability of individuals and organizations to satisfy their customers with quality software products and services through education, communication, research, outreach, and professional development. (http://www.asq.org)

World Organization of Webmasters Description to be supplies.

ORGANIZERS...

Software Research Institute (SR/Institute), a not-for-profit subsidiary of Software Research, Inc., was founded to promote the issues of Software Quality throughout the software development community. In addition to the Quality Week Conference series SR/Institute sponsors continuing education seminars in the general area of software quality and software engineering, and Software Quality Forums at which software quality industry leaders provide state of the art technology transfer to industry executives about how to best apply current technology to immediate software quality needs.

MEDIA SPONSORS...

SD Times is filled with authoritative reporting written specifically for software development managers. It covers the entire industry to keep you informed and up to date on Java, Linux, XML, embedded wireless, components, e-business and much more! We cover the things the technical monthlies rarely touch: the trade shows, the vendor roadmaps, the trends, the tools, the alliances, the people, the NEWS! Subscribe today. SD Times is free to qualified professionals like yourself. BUT you must sign up now!

Software Business is The Magazine for Software Executives! Every issue of Software Business Magazine includes the following editorial features: . CEO Strategies . Electronic Distribution . ASP Reports . Financial Reports . Business Automation . Software Replication . Product Development . Packaging/Fulfillment . Support/Customer Service . International Opportunities . And More...!

Software Research, Inc.'s TestWorks, an integrated suite of software test tools, is the broadest test tool suite available. TestWorks tools help automate and streamline the software development and testing process with product lines that work independently or as an integrated toolsuite. TestWorks is the only tool suite that offers Regression Testing, Test Suite Management, and Test Coverage support for Web **and** Windows **and** UNIX Platforms.

TestWorks Products

Windows/Coverage:

TCAT/Java
DEMO
EVAL
TCAT/C-C++
DEMO
EVAL
Product Comps

Windows/Regression:

CAPBAK/MSW
SMARTS

UNIX Evals:

SPARC/Solaris
x86/Solaris
RS6000/AIX
HP9000 HPUX
DEC-Alpha
SGI/Irix
SCO ODT5

Quality Techniques Newsletter QTN
Subscribe

TestWorks For Windows

TestWorks for Windows, an integrated suite of automated testing tools, is the broadest suite of tools available to test applications running under MS/Windows (Win3.1), MS/Windows 9x/Me or MS/Windows NT/2000.

Testworks for Windows has two main bundles of tools:

- TestWorks/Regression
  - CAPBAK/MSW
  - SMARTS/MSW
- TestWorks/Coverage
  - TCAT C/C++
  - TCAT for Java/Windows

TestWorks For UNIX

TestWorks for UNIX is designed to work independently or as an integrated tool suite to provide an efficient, automated testing environment for most UNIX-based platforms.

TestWorks for UNIX consists of three product lines:

- TestWorks/Regression
  - CAPBAK
  - SMARTS
  - EXDIFF
- TestWorks/Advisor
  - METRIC
  - TDGEN
  - STATIC
- TestWorks/Coverage
  - TCAT C/C++
  - TCAT for Java/UNIX
  - TCAT/S-TCAT Ada/f77

Downloading Products

DOWNLOAD PRODUCTS **Download Datasheets**

License Key Request **QuickStart Manuals**

User Manuals

Tell A Friend About This Site

Go!

Software Research, Inc.
901 Minnesota Street
San Francisco CA 94107 USA

PHONE: [+1] (415) 550-3020 FAX: [+1] (415) 550-3030
Comments & Suggestions: suggest@soft.com
© Copyright 2001 by Software Research, Inc.

QUALITY ASSURANCE

SD Times
SOFTWARE DEVELOPMENT
The Industry Newspaper for Software Development Managers
AUGUST 1, 2002
ISSUE NO. 059

Glue Holds Web Services Together 5
CollabNet Is Turning Japanese, Korean 5
BetaSphere Involves Customers Earlier in Product Development 6
Neon Links Mainframe Data Via J2EE 6
Intalio Takes Steps Toward Web Services Choreography ..8
VA Software's Augustin Steps Down as CEO 8
Liberty Alliance Promises Unified Security 12
Red Gate Marches ANTS Into .NET Web Services ...14
Telelogic Enhances Tau With Scenario Synthesis ...14
SpeeDev Adds Visual Designer to Tool Suite 16
IBM Delivers Directory Server, Integrator 18
Automated WebQA Crawls Over Competitors 20
Starbase Launches Multicast Server 22
Mercury Pushes Post-Deployment Tuning ..24
Altia Design 5.0 Adds Object Browser 26
SmartPhone 2002 Close to Release 26
SPECIAL REPORT: Minding Your Language

Loyalties In Sun and Microsoft Camps Run Strong 29

A BZ MEDIA PUBLICATION $7.95
www.sdtimes.com

WITH OPTIMIZED KVM, SUN SEES CLDC HOTSPOT IN ARM DEVICES

Company also updates J2ME wireless toolkit

BY EDWARD J. CORREIA

Sun Microsystems Inc. is covered until 2007, at least as far as its K Virtual Machine is concerned. The company has released CLDC HotSpot, an implementation of its J2ME Virtual Machine for the Connected Limited Device Configuration that is optimized for processors using ARM's 32-bit cores. Sun claims that it outperforms its predecessor by a factor of 10, and should fill developers' needs for the next half-decade.

Intelligence in the adaptive compiler is the key to its speed, says Sun's Tallman.

"We're thinking about CLDC HotSpot enabling the next five years of J2ME handsets," said Nicholas Lorain, Sun's senior product manager for wireless Java technology, as he described devices that increasingly will contain color screens, use Bluetooth communications and require longer battery life. CLDC HotSpot, which Lorain claims offers greatly increased performance, addresses these needs. "J2ME applications will run about 10 times faster on CLDC HotSpot," than on the previous CLDC

▸ continued on page 26

Microsoft Blends Java Into VS.NET

BY ALAN ZEICHICK

Java is now officially Microsoft's fourth .NET language. Last month, the company completed its Visual J# compiler, and is offering it for use within the Visual Studio .NET environment, alongside Visual Basic, Visual C# and Visual C++. The company also unveiled an enterprise SDK for integrating custom-written tools into VS.NET and a new Web site for software architects.

On the Java front, Microsoft has been working on a two-pronged approach for wooing Sun's developers to .NET. One initiative, called JUMP to .NET, presents a code translator that converts J2SE code to Microsoft's C# language. The other involves J#, an implementation of the Java syntax designed to use .NET APIs rather than J2SE features.

Patterns and practices will be available, says Microsoft's Rubinstein.

The first beta of VJ#.NET, the compiler and development tool for the language, appeared in 2001; the company released version 1.0 in late June, making it available for free download from http://msdn.microsoft.com/vjsharp. The requirement is that developers have any edition of VS.NET, or of Visual Basic .NET or Visual C# .NET, to provide the IDE. It will also be included in the VS.NET package, said Tony Goodhew, product manager for both Visual C# and Visual J#.

The important fact, stressed Goodhew, is "we are not building a Java platform. We are delivering the Java language for the .NET Framework. We don't support RMI or the CORBA extensions. But in terms of the real value of your application, the ex-

▸ continued on page 24

HP Dumps J2EE Software Stack

Will partner with BEA for middleware under new strategy

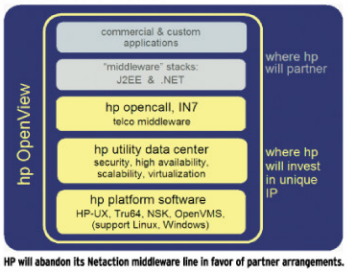
BY ALAN ZEICHICK

It's been several months since Hewlett-Packard Co.'s controversial acquisition of Compaq Computer Corp. While Compaq brought new hardware lines and consulting services strengths to HP, the new, larger company intends to turn its attention on the software side to its systems-management tools and telephony products—not developer tools and general enterprise platforms. And to that end, it is killing its in-house J2EE application platforms.

In an analyst briefing, Nora Denzel, senior vice president for HP's software global business unit, said, "HP will refocus its post-Compaq merger software strategy heavily around HP OpenView, for managing and optimizing services for IT over voice and data infrastructures, and UDC, for optimizing IT assets." UDC, or Utility Data Center, provides a layer of network and storage management software that the company says will allow virtualization of a data center.

The emphasis on the developer side will be on third-party partnerships, according to Den-

▸ continued on page 24



HP will abandon its Netaction middleware line in favor of partner arrangements.

CEOs, CFOs MUST SWEAR TO FINANCIALS

New SEC rule affects 11 software giants

BY DAVID RUBINSTEIN

Under a U.S. Securities and Exchange Commission rule adopted in late June, investors in the country's largest publicly traded companies—including 11 related to software development—will start to know by Aug. 15 whether or not the financial statements offered up by those companies are accurate and complete.

The rule requires the principal executive officer and financial officer of 947 companies with revenues at or greater than $1.2 billion to personally swear under oath to the company's financial reporting. The action is part of a broader government

▸ continued on page 24

www.sdtimes.com

Not A Member Yet?

Do it now! Join the IEEE Computer Society today for many valuable benefits including...

Distance Learning

Members of the IEEE Computer Society enjoy FREE access to a comprehensive online training program for computer professionals. Join today and take any or all of 100 online course titles through our Distance Learning Campus. Subjects include Java, project management, Cisco networks, UNIX, Windows, HTML, Oracle, SQL, and more.

Magazines and Journals

To help you stay informed and competitive, the IEEE Computer Society publishes numerous specialized magazines and journals that are available to members at deeply discounted rates. Plus, you get a FREE subscription to *Computer* magazine included as part of your membership.

Conferences and Workshops

Enhance your knowledge and share practical experiences at more than 140 conferences, workshops, and symposia held each year worldwide. Members save at least 25% on registration fees and get advance notice of the meetings.

Other Valuable Benefits

- FREE membership in your local chapter
- FREE e-mail alias of Your.Name@computer.org
- FREE membership in up to four of 40+ Technical Committees
- FREE membership in 200+ Standards Working Groups
- Member discounts on hundreds of books and conference proceedings

IEEE COMPUTER SOCIETY

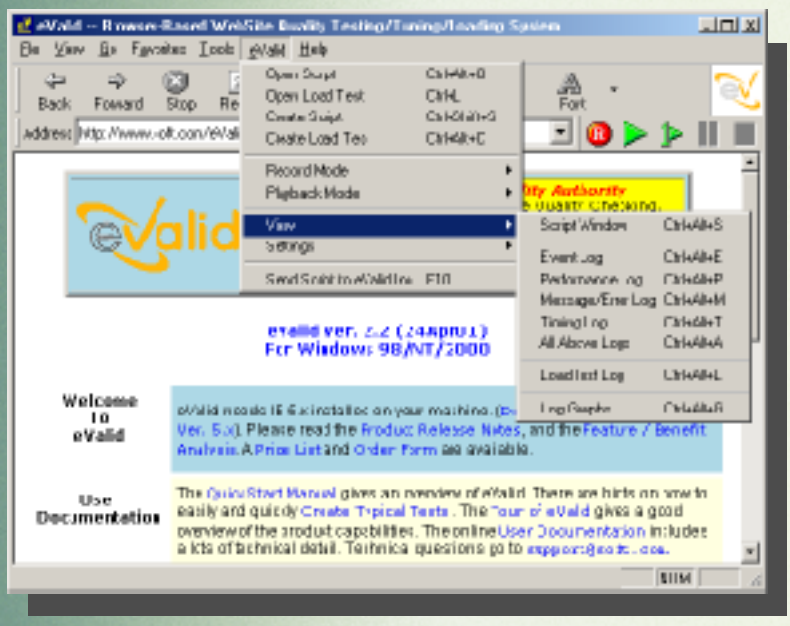
IEEE

Join Today!

Apply online at http://computer.org/join

Quality Testing of WebSites

eValid: Testing from the Web Browser



If you can browse it,

eValid can test it, at

the click of a mouse.

It's . . .

✓	Easy	Up and running in minutes
✓	Fast	Immediate test results
✓	Real	Shows the users' real experience – no simulations
✓	Deep	Handles complex transactions
✓	Precise	Finds load bottlenecks
✓	Thorough	Visits every page – reveals site tree
✓	Visual	Reports problems in graphs, tables or spreadsheets
✓	Value	Superior return on investment

The most complete WebSite performance solution today – at a fraction of the price!

eValid

Your e-business partner

Free Download

www.e-valid.com

1-800-942-SOFT

eValid™ -- The Internet Quality Authority™
Client-Side Browser-Based WebSite Quality Checking,
Testing, Validation, Tuning, Loading, 24x7 Monitoring
Training, Consulting, Seminars
© Copyright 2002 by eValid, Inc.

About eValid -- The Internet Quality Authority

eValid Home

eValid enhances your e-business success by assuring that your WebSite is trouble-free, reliable, speedy, and available 24x7. In a Web-paced world your WebSite is your key asset. eValid checks, protects and insures.

eValid Products

eValid's Test Enabled Web Browser™ is a test engine that provides you with browser based 100% client side quality checking, dynamic testing, content validation, page performance tuning, and webserver loading and capacity analysis.

This new cutting-edge technology is 100% client side based and is completely object-oriented. eValid offers a unified approach to WebSite testing that is unique in its simplicity, power, efficiency, effectiveness, and superior ease of use.

By focusing entirely on the users' view of WebSite quality, eValid results are accurate, complete, repeatable, and highly effective -- all as experienced by your users. The eValid test engine is available in several product configurations.

- **Mapping:** eValid Site Analysis Mode will systematically examine a WebSite or a sub-WebSite for critical quality factors such as slow downloading pages, unavailable links, too-long pages.
- **Testing:** eValid test scripts can exercise the key parts of your site, confirm links, check content, and simulate users' activities. *Make sure your customers get the right message!* More...
- **Validation:** eValid can confirm selected content, validate document properties, images and applets. *Have confidence that you are delivering correct information!* More...
- **Loading:** eValid load testing scenarios can simulate 100's or 1000's of users. *Can your WebSite handle the traffic when a serious crunch comes?* More...
- **Tuning:** eValid timing capabilities let you identify slow-loading pages so you can "tune up" your site for optimum performance. *Keep customers from clicking away!* More...

eValid Services

eValid website quality services are all based on the eValid test engine, and are supported through training, consulting, and technical seminars.

- **Standard Monitoring:** eValid monitoring, based on the eValid test engine, runs standard tests on your site. eValid's 24x7 website performance monitoring provides for email and/or pager/beeper alert service, plus customer access on our WebSite to historic testing and monitoring data. *Be the first to know whenever your site is misbehaving.* More...
- **Custom Monitoring:** Use eValid test services to contract us to run tests you have recorded and proved out yourself using the standard eValid test engine. Custom eValid test executions run on standard intervals, in varying time zones, and are all 24x7. *Make sure your **own** tests run successfully all the time.* More...
- **WebSite Testing, Qualification, Verification, Loading:** eValid consulting services include WebSite testing, test suite development, WebSite qualification, e-commerce verification, and WebSite loading and capacity checking exercises. All work is based on application of the eValid test engine plus other non-released WebSite analysis facilities. More...
- **WebSite Quality Consulting & Seminars:** eValid website quality experts can work along side your web developers to make sure your site meets the highest reliability, quality, performance, and capacity standards. eValid seminars and workshops are aimed at bringing your own team up to speed. More...

eValid -- Your E-Business Partner

eValid -- offering products and custom services -- is your one stop solution provider for WebSite quality. eValid is your true e-business partner.

Tell A Friend About This Site

Go!

eValid, Inc.
901 Minnesota Street
San Francisco, CA 94107 USA

Phone [+1] 415.550.3020
FAX [+1] 415.550.3030
info@soft.com.

You know how to run your testing.
We can help you do it *BETTER and FASTER.*

LogiGear®
Expert Testing...Real World Solutions

*Logi*Gear Corporation
555 Pilgrim Drive, Suite B
Foster City, CA 94404

1.800.322.0333
www.logigear.com/information/

LogiGear Corporation® is the first Silicon Valley-based software testing company to offer a full range of services, including technical and management expertise in software quality engineering, comprehensive outsource testing solutions, a skill-based training curriculum for software testing professionals, and world-class testing support products. Our value-added services include application testing, automated testing, white-box testing, and load/performance testing for business and consumer applications; TRACKGEAR™, a web-based defect-tracking solution; and QA Training through LogiGear University's Software Testing Training Series. LogiGear specializes in testing web and client-server systems, software applications, hand-held communication devices, and consumer electronic products. For more information, please visit our website at www.logigear.com.

Some things just have to work right the first time...

Especially your software applications.

For Configuration Management,
Testing and QA products

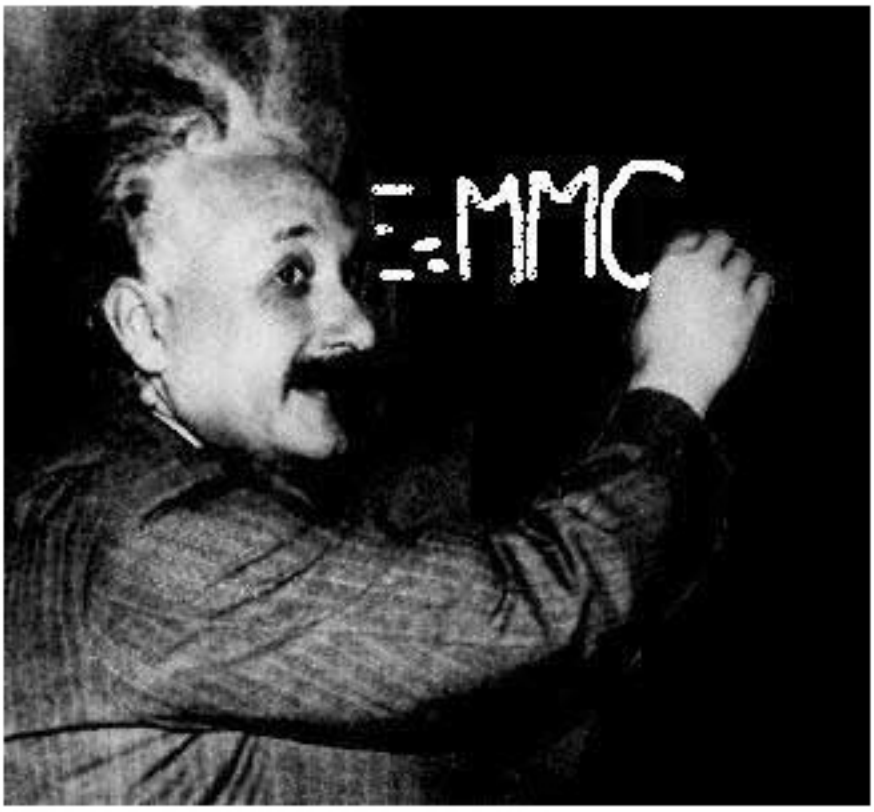
1-800-638-6316
www.mccabe.com

McCabe & Associates®

9861 Broken Land Parkway • Fourth Floor • Columbia, MD 21046
(410) 381-0036 • Toll Free 800 638-6316 • FAX (410) 381-7912•
URL - http://www.mccabe.com

McCabe & Associates enables IT to deliver better applications by providing products and process that implement a relevant, repeatable, and measurable approach to managing software changes and their effects on the testing and quality of applications. McCabe products include McCabe QA, McCabe Test, and McCabe TRUEchange and Release Rocket Verify. Release Rocket Verify identifies, tracks, and reports on software-testing efforts at the ***change*** level. Verify focuses on validating changes and testing results across an application via real-time feedback, enabling rapid delivery of releases with high confidence. Verify reduces testing time, maximizes testing resources, and increases the manageability of application releases.

How Complicated Is It To Plan Your Testing?



Simplify.

Contact The Leaders In Uncomplicated Outsource Testing:

RECOMMENDED TEST LABS, INC.

800 464-LABS (5227)
www.testlabs.com

Recommended Test Labs, Inc. (RTL) is an outsource testing services vendor with offices in San Francisco, CA Portland, OR, Ashburn, VA and Pittsburgh, PA. RTL has been helping top software and hardware companies ship high quality products on–time and within budget since 1989. Whether to augment their internal QA staff or fully manage their testing projects, Fortune 500 companies have turned to RTL's ***full-time***, ***experienced***, and highly ***trained staff*** for their off-site testing needs.

RTL provides a comprehensive range of testing services from compatibility testing to web services testing to in-depth feature conformance testing:

- Network Testing
- Windows testing on 95/98/ME/NT/2K and XP
- Web Site and Application Functional and Performance Testing
- Test Planning and Design
- Wireless Testing (802.11b, 802.1x, Bluetooth)
- Macintosh OS with access to Apple's ADR Labs
- Localization & Translation Validation Testing
- Driver testing

To find out more about RTL and learn how our clients have been able to meet their testing needs and deliver their products on time, stop by our booth or contact RTL at 800-464-LABS (5227), http://www.testlabs.com, or info@testlabs.com.

Company Description:

Semantic Designs

Semantic Designs offers off-the-shelf and customizable tools to mechanize the analysis, modification, enhancement, translation or generation of large-scale software systems in virtually any language. The company also provides services for building custom testing and quality enhancment tools.

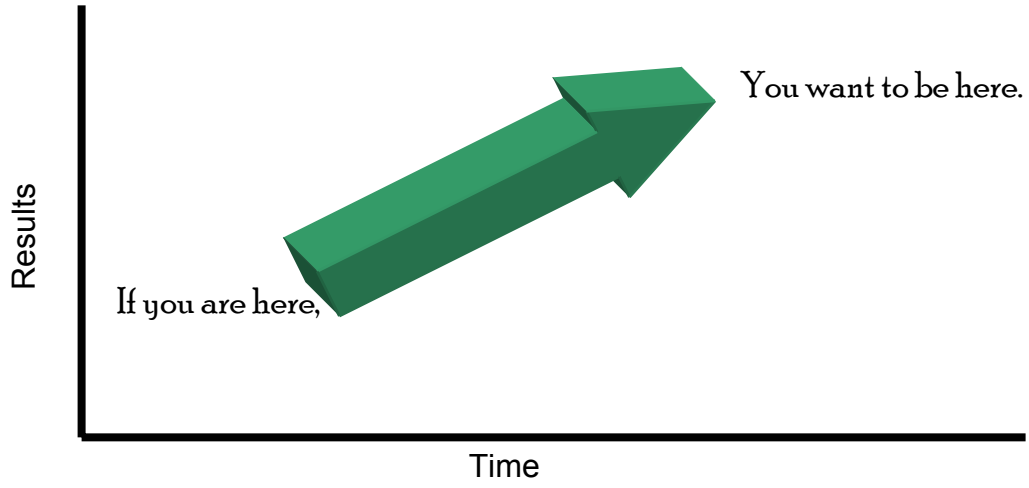
The DMS Software Reengineering Toolkit, based on generalized compiler technology, can parse/analyze/transform/prettyprint software. It is absolutely different from any other tool available for handling automated analysis and software modifications. DMS handles multiple and mixed programming language, for systems of several million lines. It has language modules for C, C++, Java, C#, VisualBasic, COBOL, Fortran, Ada, XML, and others, and is designed to accept new and proprietary language definitions very quickly. DMS has been used for many custom tasks, including building HTMLized cross references for complete systems, finding dead code, removing useless preprocessor directives, automatic synthesis of target-language fast XML readers and writers, and full translation of legacy languages such as JOVIAL to C.

Off-the-shelf products are derived from DMS. They include CloneDR, a tool that finds the typically 10-20% redundant code in every system, and TestCoverage tools provide extremely low-overhead ways to determine and interactively display code coverage for standard and custom langauges, along with full summary reports.
<http://www.semdesigns.com>

Software Quality First

Your source for solutions in software quality systems, testing, process improvement, metrics, and project management.

We take a very practical and do-able approach to software quality assurance and quality improvement. Our focus is on those things that will be effective in a particular organization given its current situation and future plans. Our goal is to obtain measurable improvements in the product and process which in turn lead to improved business results. Our extensive experience in software development, testing, project management, and software quality assurance allow us to quickly determine the highest priority issues and to develop an effective approach for dealing with them.



Software Quality First
Jessee Ring, Principal Consultant
40119 San Carlos Place
Fremont, CA 94539
sqa1st@attbi.com
www.sqa1st.com
510-915-2353
fax: 510-573-7464

Software Quality First provides consulting and/or contract services in software quality assurance, testing, metrics, process improvement and project management. We can handle short term or long term assignments that may require less than one full time person or several people. Our principle consultant is Jessee Ring, who has a masters degree in electrical engineering and over twenty-five years experience in software development, management, SQA, and process improvement in diverse environments and application areas. We can help companies that are just getting started on the quality or process improvement journey or ones who have been working on it for awhile, since we tailor our services to the specific needs of each individual organization. We draw upon a wide variety of industry best known methods that have been shown to be effective. Our approach is based upon sound fundamentals while at the same time being practical and results oriented.

Please visit our website at www.sqa1st.com.

We may be reached at: 510-915-2353, sqa1st@attbi.com, fax: 510-573-7464.

"TeamShare® had just what we wanted...

functionality, scalability, the right price, Web architecture, integration with other vendors, great support staff and a product management group that really listens to its customers."

David S. Lee,
Manager, Engineering Operations
Managed Objects

The Client.

Managed Objects produces a software platform called Formula®, which helps companies understand how their underlying technology impacts their top-line business.

The Challenge.

To find a solution that would best integrate customer enhancement requests into Formula® product lifecycle.

The Solution.

TeamTrack's® feature-rich and highly configurable Web-architected solution gave Managed Objects the ability to streamline and enforce their processes while integrating customer feedback into the development loop.

For more information or for an online demonstration, call:

1.888.832.6742

www.teamshare.com

TeamShare® and TeamTrack® are registered trademarks of TeamShare, Inc.
All other trademarks used are the property of their respective owners.

TeamTrack®

from teamshare®

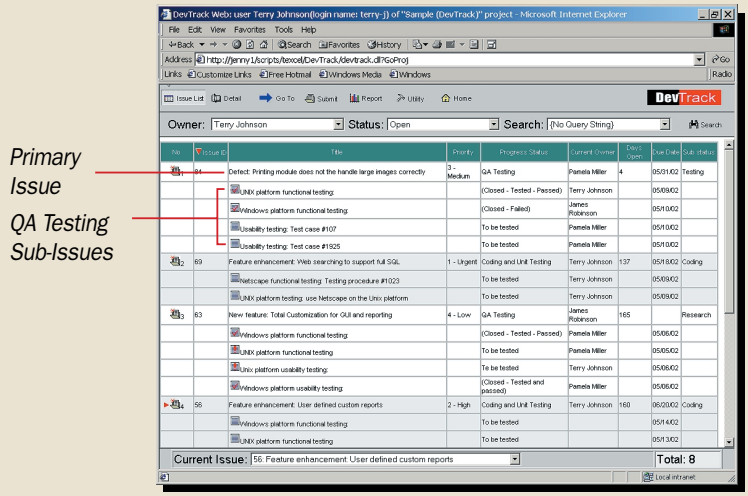
Founded in 1996, TeamShare, Inc. delivers Web-architected collaborative software solutions. The company's product line, powered by the TeamTrack workflow engine, enhances process management, speeds resolution and encourages collaboration within and across enterprises. TeamTrack provides enterprise business process management and enables collaborative product development with business customers and partners. TeamTrack Workgroup is the robust defect and issue management solution for teams on a limited budget. TeamShare solutions are highly configurable, simple to implement and maintain a low cost of ownership. TeamShare's customers and partners include Dell, Hewlett-Packard, KPMG, CitiGroup, 3-Com, Siemens, and ADP. TeamShare has been named to Computerworld's "'Top 100 Emerging Companies" and the "SoftLetter 100" lists. For more information, contact TeamShare, Inc. by phone at 888-TEAMSHARE (832-6742), via e-mail at inquiries@teamshare.com or on the Web at www.teamshare.com.

Announcing

DevTrack 5.0

The New Standard for Defect and Project Tracking Software

In Your Quest To Find The Best Defect Tracking Software, You'll Find DevTrack 5.0 Is The Only Clear, Rational Choice.



DevTrack 5.0 is the only product on the market that comprehensively manages and automates your development management, defect tracking, and test plan automation processes within one seamlessly integrated application.

DevTrack 5.0 is designed to meet your specific business needs.

DevTrack 5.0 Enterprise Edition

NEW!

- ► Advanced workflow & issue/sub-issue integration for multitasking
- ► Test plan automation and management
- ► Web interface for beta customers

DevTrack 5.0 Standard Edition

NEW!

- ► Comprehensive, single-page issue submission
- ► Integrated, user-defined custom reports
- ► Public and private issue templates for default values
- ► Quick selection of saved search queries

DevTrack 5.0 Small Business Edition

NEW!

- ► Same feature-rich product as DevTrack 5.0 Standard Edition priced specifically for small businesses

See for yourself!
View a brief, pre-recorded demonstration on DevTrack 5.0's new features.
www.devtrack.com

TechExcel™

Toll-free (800) 439-7782
International (925) 871-3900

DevTrack 5.0 – The New Standard for Defect and Project Tracking Software

DevTrack 5.0 comprehensively manages and automates development management, defect tracking, and test plan management processes within one seamlessly integrated application. It provides powerful workflow and process automation features, seamless source code control integration with Microsoft Visual SourceSafe (VSS) and Perforce, robust searching and reporting, and comprehensive point-and-click customization. New features include advanced workflow & issue/ sub-issue integration for multitasking and a Web interface for beta customers.

Since its introduction in 1997, DevTrack has become one of the top defect- and project-tracking tools on the market. Industry giants such as Hewlett Packard, Texas Instruments, and Honeywell have all deployed DevTrack with teams of over one hundred developers. Many smaller software companies have deployed DevTrack as well because of its unique combination of power, flexibility, and ease of use.

With DevTrack 5.0, TechExcel now offers three new editions of DevTrack designed specifically to meet any company's product development and engineering process automation.

DevTrack 5.0 Enterprise Edition

- Advanced workflow & issue/sub-issue integration for multitasking
- Test plan automation and management
- Web Interface for beta customers

DevTrack 5.0 Standard Edition

- Comprehensive, single-page issue submission
- Integrated, user-defined custom reports
- Public and private issue templates for default values
- Quick selection of saved search queries

DevTrack 5.0 Small Business Edition

- Same feature-rich product as DevTrack 5.0 Standard Edition priced specifically for small businesses

For more information, please call 800-439-7782 or visit us on the Web at www.techexcel.com.

TestQuest, Inc.

TestQuest™ Pro is the only non-intrusive automated test solution that provides comprehensive support for a wide range of electronic devices including embedded systems, computer systems, handheld devices, cell phones and Interactive TV set-top boxes. Simulating the presence of a "virtual" user, TestQuest Pro executes pre-defined streams of actions, and compares the output to valid states to determine whether the test was successful. A scripting facility provides a foundation for consistent, reliable and repeatable testing.

18976 Lake Drive East
Chanhassen, MN 55317
Phone: 952.936.7887
www.testquest.com
info@testquest.com

TestQuest, Inc.
18976 Lake Drive East
Chanhassen, MN 55317
+1.952.936.7887
info@testquest.com
www.testquest.com

TestQuest™ Pro is the only non-intrusive automated test solution that provides comprehensive support for a wide range of electronic devices including embedded systems, computer systems, handheld devices, cell phones and Interactive TV set-top boxes. Simulating the presence of a "virtual" user, TestQuest Pro executes pre-defined streams of actions, and compares the output to valid states to determine whether the test was successful. A scripting facility provides a foundation for consistent, reliable and repeatable testing. The benefits of using TestQuest Pro include:

Reduced test cycle time: Complete test cycles faster – customers report up to 90% time savings compared to manual methods. TestQuest Pro can run tests 24 hours a day, 7 days a week, dramatically expanding the time available for testing. This means dramatic savings in the time it takes to get products tested and into the market.

Reduced test cost: Customers report rapid ROI and dramatic cost savings by eliminating the need to dedicate staff to testing or outsource testing to 3rd parties.

Improved product quality: Build sophisticated test scripts that thoroughly exercise your products and reliably uncover defects. With TestQuest, repeating an advanced regression test is as easy as running a script.

TESTING TESTING 1 2 3

4124 148th Ave NE Redmond WA 98052 Email: info@TT123.com Website: TT123.com Phone: 888.718.0837

OUTSOURCE TESTING SERVICES

- -Functionality
- -Installation
- -Regression
- -PC/Console Games
- -Internet/Website
- -Localization
- -Hardware
- -Consulting Services
- -PC/Mac Compatibility Lab
- -Performance/Stress

25% OFF

ASK ABOUT OUR **NEW CUSTOMER DISCOUNT**

- -increase profitability
- -reduce costs
- -meet deadlines
- -ensure quality

"Testing Testing 123 has been invaluable to me in several emergency situations. Several times I've called them to have a problem testing for my Web site. They have been able to configure a machine, test the site, and provide the results with surprising speed. For a Web site, it's critical to have someone with the hardware, software and people resources to test a specific problem so quickly."

Brian Copps, Release Program Manager
Microsoft, TechNet

Testing Testing 123 was founded in May of 1998, with a focus on building strong client relationships by providing excellent service, flexibility and value. Our outsource testing solutions are cost conscious, responsible and flexible. We have established working relationships with Microsoft, Citibank, Nintendo, Macromedia, Spacelabs, Corbis and others.

Testing Testing 123 is a client driven company with the goal to provide our customers with services that fit their specific needs. Our company mission is to empower our clients to produce the highest quality software possible. Our experienced personnel specialize in time critical performance, delivered with exceptional value and service.

Testing Testing 123's outsourced testing allows your company to reduce or eliminate the overhead required to maintain testing equipment or the additional cost of adding new staff for your project. Our testing facility is designed to change, which allows us to "ramp-up" quickly for new projects and new equipment configurations. We can test your project in our lab or provide personnel for testing at your site. Our staff can also provide consulting, project management and test planning services for your project.

Quality Assurance is crucial to the overall success of your business. When additional expertise is needed or when time is short and resources are few, let **Testing Testing 123** show you how we can help your business increase efficiency and improve your bottom line.

If you have any questions, please contact us.