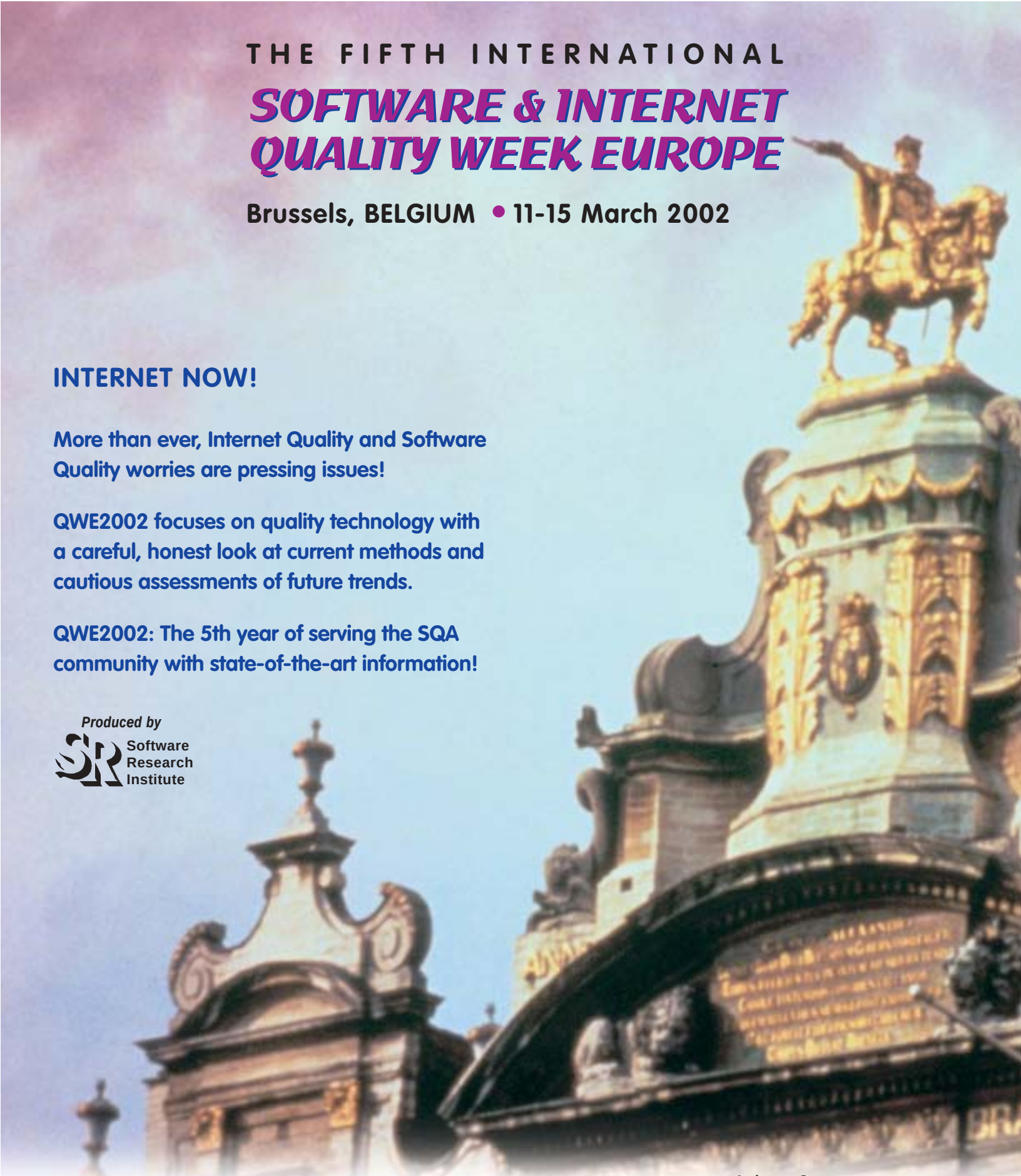
THE FIFTH INTERNATIONAL

SOFTWARE & INTERNET QUALITY WEEK EUROPE

Brussels, BELGIUM • 11-15 March 2002



INTERNET NOW!

More than ever, Internet Quality and Software Quality worries are pressing issues!

QWE2002 focuses on quality technology with a careful, honest look at current methods and cautious assessments of future trends.

QWE2002: The 5th year of serving the SQA community with state-of-the-art information!

Produced by

Software Research Institute

In cooperation with

Centre for SOFTWARE Engineering

DATA TEKNISK FORUM

ESI European Software Institute

GI

OESTERREICHISCHE COMPUTER GESELLSCHAFT AUSTRIAN COMPUTER SOCIETY

SAI

SAST

TestNet

tqi

Industry Sponsors

CMG Information Technology

ca Computer Associates

gitek interaction through software

eValid

ps.testware Software Testing Services

spherion technology architects

VeriTest

QWE2002 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: 31 January 2002.

QUICK ACCESS TO THE FOUR DAY PROGRAM AT QWE2002	
Pre-Conference Tutorial Day 1	Monday 11 March 2002
Pre-Conference Tutorial Day 2	Tuesday 12 March 2002
Conference Day 1	Wednesday 13 March 2002
Conference Day 2	Thursday 14 March 2002
Conference Day 3	Friday 15 March 2002
The QWE2002 program is linked to speaker biographies and presentation abstracts. Click on a ***Speaker*** or on a ***Title*** to read the Presentation Summary and Author Biography.	

REGISTER FOR QWE2002

Monday 11 March 2002
PRE-CONFERENCE TUTORIAL DAY #1

8:30 - 10:00 Coffee 10:30 - 12:00	**Tutorial A1** *Web Testing Master Class* Dr. Gualtierro Bazzana & Mrs. Ioana Ene ONION, S.P.A., Italy (Part I)	**Tutorial B1** *Stepwise Improvement of the Test Process using TPI* Mr. Martin Pol Polteq, Netherlands & Mr. Ruud Teunissen Gitek nv, Belgium	**Tutorial C1** *Managing the Testing Process: Organization, Motivation, and Techniques* Mr. Rex Black RBCS, Inc., USA	**Tutorial D1** *Introduction to the Test Maturity Model* Dr. Erik P. vanVeenendaal Improve Quality Services BV, Netherlands	**Tutorial E1** *Testing Component-Based Systems* Dr. John McGregor Luminary Software, USA
12:00 - 13:30	TUTORIAL DAY LUNCH AND NETWORKING				
13:30 - 15:00 Refreshments 15:15 - 17:00	**Tutorial A2** *Web Testing Master Class* Dr. Gualtierro Bazzana & Mrs. Ioana Ene ONION, S.P.A., Italy (Part II)	**Tutorial B2** *Test Plan 101 Using IEEE 829* Mr. Bernard Homes TESSCO Ltd., France	**Tutorial C2** *Web Testing: A Practical Approach* Dr. Edward Miller eValid, Inc., USA	**Tutorial D2** *Introduction to Defect Analysis* Mr. Otto Vinter Software Engineering Mentor, Denmark	**Tutorial E2** *Guided Inspection of UML Models* Dr. John McGregor Luminary Software, USA

Tuesday 12 March 2002
PRE-CONFERENCE TUTORIAL DAY #2

8:30	**Tutorial F1**	**Tutorial G1**	**Tutorial H1**	**Tutorial J1**	**Tutorial K1**

- 10:00 Coffee 10:30 - 12:00	*Writing Good Requirements* Mr. Erik Simmons Intel Corporation, USA	*Emotional Intelligence as the Key to Software Testing* Mr. Jens Pas I2B, Belgium	*Getting Started - Stressing Web Applications, Stress Early - Stress Often* Mr. Robert Sabourin Amibug.com, Canada	*Managing The Testing Process: Test Estimation* Mr. Rex Black RBCS, Inc., USA	*Model-Based Testing* Mr. Ibrahim K. El-Far Florida Institute of Technology, USA
12:00 - 13:30	TUTORIAL DAY LUNCH AND NETWORKING				
13:30 - 15:00 Refreshments 15:15 - 17:00	**Tutorial F2** *Writing Good Requirements* Mr. Erik Simmons Intel Corporation, USA	**Tutorial G2** *Think Green! Think Different! Think Modular! (The LEGO Principle)* Mr. Olivier Denoo ps_testware, Belgium	**Tutorial H2** *Just In Time Testing - Testing Turbulent Web Based Applications* Mr. Robert Sabourin Amibug.com, Canada	**Tutorial J2** *Structured Testing* Mr. Martin Pol Polteq, Netherlands & Mr. Ruud Teunissen Gitek nv, Belgium	**Tutorial K2** *Software Measurement: The Goal/Question/Metric Method* Dr. Rini vanSolingen CMG, Netherlands
17:30 - 19:45	Welcome Networking Reception at Cantillon Brewery				

Wednesday 13 March 2002
CONFERENCE DAY #1

QWE2002 Exhibition: 10:00 to 18:00

K1 8:30	**PLENARY SESSION**				
	Plenary Session Introduction: Edward Miller Software Research, Inc., USA				
	Keynote K1-1: *Independent Verification And Validation Implementation At NASA (QW2001 Best Presentation)* Dr. Linda Rosenberg GSFC NASA, USA				
	Keynote K1-2: *Creating Quality Web Systems* Mr. Robert A. Sabourin Amibug.com, Canada				
10:00	REFRESHMENTS IN EXHIBIT HALL				
1 10:30	**Vendor Technical**	**Technology** New Tools	**Applications** Case Studies	**Internet** Experience	**Management** Methodology
	Session VT1 *Quality Service Provision* Mr. Peter Sterck ps_testware, Belgium	**Paper 1T** *Validating Quality Requirements of Object Oriented Design* Mr. Jamal Said & Mr. Eric Steegmans Department of Computer Science, K.U. Leuven, Belgium	**Paper 1A** *QA Success Story for Embedded Systems in Real Time Control Systems* Mr. Jon Maurolagoitia SQS S.A., Spain	**Paper 1I** *Providing Automated Support for Web Metrics* Ms. Julia Gonzalez, Mr. Guillermo Lafuente, Mr. Luis Olsina, & Mr. Oscar Pastor	**Paper 1M** *Test Process Improvement -- Theory and Practice* Mr. Stale Amland Amland Consulting, Norway & Mr. Martin Pol Polteq IT Services B.V., Netherlands

				Extremadura University, Spain	
2 11:30	**Session VT2** *Five years of assesments with the Test Process Scorecard®: the method and the results* Mr. Egbert Bouman Spherion Technology, Netherlands	**Paper 2T** *Performance Testing in Internet Time* Mr. Allan Beumer CMG, Netherlands	**Paper 2A** *Steps to Bring the V- Model Into Real Life: A Case Study* Dr. Rainer Stetter ITQ GmbH & Software Factory GmbH, Germany	**Paper 2I** *Practical Experiences in Bug Cluster Management* Mr. Kim Davis My Virtual Model, Inc., Canada & Mr. Robert Sabourin Amibug.com, Canada	**Paper 2M** *Improving the Software Estimation Process* Mr. Geir Kjetil Hanssen, Mr. Hans Westerheim SINTEF, Norway & Mr. Tor Stalhane NTNU, Norway
12:30	CONFERENCE LUNCH AND NETWORKING IN EXHIBIT HALL				
3 14:00	**Vendor Technical**	**Technology** Web Testing	**Applications** Novel Applications	**Internet** Performance	**Management** Planning
	Session VT3 *Develop High Quality e-business Applications - Test Smarter, Finish Faster* Mr. Kris Dugardyn Rational, Germany	**Paper 3T** *Automated Functional Testing of Web-Based Applications* Mr. Oliver Niese, Dr. Tiziana Margaria & Prof. Bernhard Steffen METAFrame Technologies GmbH, Germany	**Paper 3A** *Choosing an Appropriate Software Testing Method* Mr. Ibrahim K. El-Far, Ms. Florence E. Mottay, Mr. Herbert H. Thompson & Mr. Nikhil Nilakantan Florida Institute of Technology, USA	**Paper 3I** *Performance Testing Applications In Internet Time* Ms. Nancy Landau Alltel Technology Services, USA	**Paper 3M** *Information Systems as an Instrument for Quality Programs* Mr. Jose M. Xexeo, Dr. Ana Regina, Mr. Alvaro Alves, & Dr. Jose Blaschek COPPE/Universidade Federal do Rio de Janeiro, Brazil
4 15:00	**Session VT4** *ConTest: the Automation Architecture for the TestFrame Method* Mr. Martin Gijsen CMG, Netherlands	**Paper 4T** *Traceability and Reproducability in Integrated System Test Environments* Dr. Nancy Eickelmann & Mr. Allan Willey Motorola Labs, USA	**Paper 4A** *Testing of CORBA Products* Ms. Mang Li, Mr. Axel Rennoch, Dr. Ina Schieferdecker & Ms.Dorota Witaszek GMD FOKUS, Germany	**Paper 4I** *Challenges of Automating Performance Tests for New Internet Technologies* Mr. Raymond Rivest Computer Research Institute of Montreal, Canada	**Paper 4M** *Quality Radar: Getting Grip on Customer Expectations* Mr. Ton Dekkers & Mr. Mario Van Os IQUIP Informatica BV, Netherlands
	REFRESHMENTS IN EXHIBIT HALL				
5 16:00	**Session VT5** *Setting up a Test Competence Centre* Dr. Martin Wieczorek SQS, Germany	**Paper 5T** *InBrowser WebSite Testing: The Client-Side Approach* Dr. Edward Miller eValid, Inc., USA	**Paper 5A** *The AGEDIS Software Test Technology Project* Mr. Bernd Mattern imbus AG, Germany	**Paper 5I** *Testing Inside and Outside the Firewall* Mr. Michael Avni Mercury Interactive, UK	**Paper 5M** *Variable Test Strategy: Learn To Only Do What You Need To Do* Mr. Geert Pinxten & Mr. Jens Pas I2B, Belgium
17:00 - 18:00	COCKTAIL PARTY IN EXHIBIT HALL				

Thursday 14 March 2002
CONFERENCE DAY #2

QWE2002 Exhibition: 10:00 to 16:00

K2 8:30	**PLENARY SESSION** **Plenary Session Introduction:** Edward Miller Software Research, Inc., USA **Keynote K2-1:** *Organizing for High Tech Innovation* Prof. Koenraad Debackere KUL, Belgium **Keynote K2-2:** *Power Testing* Mr. Bob Bartlett SIM Group, UK				
10:00	REFRESHMENTS IN EXHIBIT HALL				
	Vendor Technical	**Technology** Models	**Applications** Components	**Internet** New Technology	**Management** Process
6 10:30	**Session VT6** *Opening the J2EE Black Box with PerformaSure: Identifying and Eliminating Performance problems for Java and J2EE* Mr. Brett Hodges Sitraka, Netherlands	**Paper 6T** *Tool Support for Model Based Statistical Testing* Dr. Stacy Prowell The University of Tennessee, USA	**Paper 6A** *Structuring Your Tests in a Component Based Environment* Mr. Chris C. Schotanus CMG, Netherlands	**Paper 6I** *Deployment of Globalised Wireless Internet Applications* Mr. Paul McBride VeriTest, Ireland	**Paper 6M** *Using SPICE as an Internal Software Engineering Process Improvement Tool* Mr. Michael J. Hillelsohn Software Performance Systems, USA
7 11:30	**Session VT7** *Census - Issue Tracking* Mr. Raynald Korchia inQA.labs, Spain	**Paper 7T** *State-of-the Art Information on Petri Nets Applied to Software Quality* Dr. Mihaela Barreau, Dr. Jean-Yves Morel & Dr. Alexis Todoskoff University of Angers, France	**Paper 7A** *Performance Testing: "Step On It"* Ms. Nadine Pelicaen ps_testware, Belgium	**Paper 7I** *Innovative WebSite Mapping Tool* Dr. Edward Miller eValid, Inc., USA	**Paper 7M** *Process Management Maturity* Mr. Peter Gooch Computer Associates, USA
12:30	CONFERENCE LUNCH AND NETWORKING IN EXHIBIT HALL				
	Vendor Technical	**Technology** Hi-Tech Approaches	**Applications** Critical Applications	**Internet** Methods	**Management** Risk
8 14:00	**Session VT8** *Testing to Improve Projects' Productivity* Dr. Prakash Mutalik RelQ, India	**Paper 8T** *Improving Component Quality Through the Systematic Combination of Construction and Analysis* Dr. Christian	**Paper 8A** *The Importance of Data in Functional Testing* Mr. James Lyndsay Workroom Productions,	**Paper 8I** *A Requirements-Based Approach to Delivering E-Business and Enterprise Applications* Mr. Scott Jefferies Starbase	**Paper 8M** *A Risk Based Testing Process* Mr. Anders Claesson Enea Realtime AB, Sweden

		Bunse & Dr. Oliver Laitenberger Fraunhofer Institute for Experimental Software Engineering, Germany	Germany	Corporation, USA	
9 15:00	**Session VT9** *Inter-Operability: the testing issue* Mr. Peter Theunis Integri, Belgium	**Paper 9T** *Systematic Validation of an Interlocking System* Ms. Begona Laibarra SQS SA, Spain & Mr. Francisco Vega Alcatel SEL, Spain	**Paper 9A** *Testing Challenges of Incremental Component Based Development* Mr. Leo VanDerAalst Gitek nv, Belgium	**Paper 9I** *Vulnerabilities and Developing for the Net* Mr. Robert A. Martin The MITRE Corporation, USA	**Paper 9M** *Risk Based Test Strategy* Mr. Rob Baarda IQUIP Informatica BV, Netherlands
	REFRESHMENTS IN EXHIBIT HALL				
10 16:00	**Session VT10** *How to monitor the Quality of Service of your business infrastructure* Mr. Michel Kalis Ubizen Aethis, Belgium	**Paper 10T** *How to Use Scientific Methods In Software Testing* Mr. Anders Claesson Enea Realtime AB, Sweden	**Paper 10A** *Product Triage: A Medical Approach to Predicting and Monitoring Product* Mr. Erik Simmons Intel Corporation, USA	**Paper 10I** *Multifaceted Internet Application Quality Validation Methodology* Mr. Simon J. Hardiman SQS S.A., Spain	**Paper 10M** *Quality Starts by Defining Goals* Dr. Robert Darimont, Ms. Emmanuelle Delor & Mr. Andre Rifaut CEDITI, Belgium
18:00 OR 18:30	The Musical Instruments Museum Visit OR A Small Chocolate Factory Visit				

Friday 15 March 2002
CONFERENCE DAY #3

	Vendor Technical	Technology Advanced Testing	Applications Systematics	Internet QA Issues	Management Real-World
11 8:30	**Session VT11**	**Paper 11T** *Virtual Test Management: Rapid Testing Over Multiple Time Zone* Mr. Jim Bampos Lionbridge, USA & Mr. Eric Patel Nokia, USA	**Paper 11A** *Configuration Management in a Test Centre* Mr. Kie Liang Tan CMG, Netherlands	**Paper 11I** *Checklist for Web Site Quality Assurance* Dr. Klaus Quibeldey-Cirkel TLC GmbH, Germany	**Paper 11M** *The Art of Managing Fixed Price Test Project* Mr. Ruud Teunissen Gitek nv, Belgium
12 9:30	**Session VT12** *Configuration Management Improvement Method (CMIM)* Mr. Eddy Lentze SQS, Netherlands	**Paper 12T** *Experience in Testing Pocket PC Applications* Mr. Ibrahim K. El-Far, Ms. Florence E. Mottay & Mr. Herbert H. Thompson Florida Institute of	**Paper 12A** *Metrics-Cockpit Means of 'Viewing' the Project* Mr. Sridhar Narayanan Cognizant Technology Solutions, USA	**Paper 12I** *Web Development: A New Quality Paradigm* Prof. Antonio de Amescua, Carlos III University of Madrid, Spain & Mr. Guillermo Pastor, INAD, Spain	**Paper 12M** *Survival Guide for Applying a Software Development Process* Mr. Bernd Eberhardt Rational Software, Germany

		Technology, USA			
10:30	REFRESHMENTS				
K3 11:00	**PLENARY SESSION**				
	Plenary Session Introduction: Edward Miller Software Research, Inc., USA				
	Keynote K3-1: *From Requirements to Release Criteria* Mr. Erik Simmons Intel Corporation, USA				
	Keynote K3-2: *Building the Infrastructure for The Future* Mr. Rik Nuytten Cisco Systems, Belgium				
	AWARDS PRESENTATION				
	CONFERENCE CONCLUSION: Edward Miller Software Research, Inc., USA				

	Extra Presentations		
	Applications Track	**Management Track**	**Management Track 2**
SB	*Torturing Technical Software in a Structured Way* Ms. Nadine Pelicaen ps_testware, Belgium	*Organizational Performance Engineering: Quality Assurance For The 21st Century* Mr. Michael J. Hillelsohn Software Performance Systems	*Weight Before It Is Too Late!* Mr. Olivier Denoo ps_testware, Belgium

QWE2002 Tutorial A1, A2

Dr. Gualtierro Bazzana & Mrs. Ioana Ene
(ONION, S.P.A.)

Web Testing Master Class

Key Points

- Peculiarities of Web-based applications will be presented from a technical point of view, explaining their effects on testing practices.
- The tutorial will deal with testing management aspects.
- the tutorial will touch the issues raised by integration testing between ERP and Web and in the validation of E-business solutions.

Presentation Abstract

First of all, peculiarities of Web-based applications will be presented from a technical point of view, explaining their effects on testing practices. Moreover, the tutorial will deal with testing management aspects which are fundamentally affected by the nature of Web applications, including: RAD, regression issues, Testing solution will then be presented, both for static aspects (related to HTML, pictures, XML) and dynamic aspects (ASP, CGI, Proxies, Cookies, etc.).Room will be devoted also to commercial tools available in order to give the audience an overview of the existing technologies, highlighting also experience reports from their introduction, including ROI analysis. Special emphasis will then be given to the Web Accessibility Initiative (WAI) guidelines that have been issued by W3C and can significantly help testing of Web-based applications. Last but not least, the tutorial will touch the issues raised by integration testing between ERP and Web and in the validation of E-business solutions. Case studies will cover: testing of e-commerce sites, testing of commercial Internet Web sites, testing of Intranet sites, testing of home banking/ trading on-line applications.

About the Author

Born in 1966, at university, he graduated with 110/110 and honour in Information Science at the University of Milan, in February 1989. His PhD won the special AICA award for topics related to quality in Information Technology. After working as software developer in a telecommunication company and as consultant/ manager in a consulting company he set-up ONION. His activities cover two areas of interest: consulting - projects in software engineering for various industrial companies and research in the field of software quality and networking (especially in the Internet/ Intranet domain). As far as consulting/ projects in industry are concerned, he matured and exploited know-how in conducting various medium sized and large projects for several companies in various application domains (telecommunications, data processing, MIS, process control, etc.), covering topics like: Internet services, Intranet applications, Supply Chain management, etc. Moreover he has matured significant experiences in the ERP domain, notably with SAP R/3. He has matured significant technical experiences especially in the telecommunications domain (notably: switching systems and GSM mobile radio systems) in CIM and in networking, including Internet/ Intranet/ Extranet services and solutions. He has also dealt with sw development and testing, testing methods and tools, quality planning, test planning, reliability analysis, software product evaluation, process assessment and improvement, definition of quality systems in accordance to ISO 9001 and 9000/3, reviews and inspections, FDA computer system validation and so forth His research activity spanned in various fields of software engineering, ranging from Petri Nets to development methodologies, functional and structural test coverage, metrics and related tools, CAST, reliability evaluation, software development process evaluation and improvement, management by metrics, software product quality evaluation, security technology transfer and total quality management techniques. Moreover, he has co-ordinated several European Research Projects. He has published a book: ("Software Metrics for Product Assessment", McGraw Hill, London, 1995, International Software Quality Assurance Series, ISBN 0-07-707923-X), contributed to 4 other books (in the last one he has been author of four chapters dedicated to Software Process Improvement; it has been published by IEEE Software) and published over 50 papers at international conferences on topics related to software quality and software testing.

QWE2002 Tutorial B1

Mr. Martin Pol & Mr. Ruud Teunissen
(Polteq & Gitek nv)

Stepwise Improvement of the Test Process using TPI

Key Points

- The Context, Why, What, When, Where, Who, etc. to improve?
- TPI, The Test Process Improvement Model.
- Application of the Model, Management of Change.

Presentation Abstract

After a short introduction, the attendees will learn about The Context of testing in the real world and software and test process improvement: why testing, the aims of testing, what to test, required structure, the evolution of testing, the challenges for testing, need of improvement, need of a dedicated test improvement model, other available models, the model requirements, etc. The second part of the tutorial is about the Test Process Improvement Model. The TPIÒ model will be explained in detail: the scope, the characteristics, the key areas, the requirements (checkpoints) for the different levels of maturity, the test maturity matrix, the assessment, the relation between the key areas and how to select priorities, the improvement suggestions, etc. To be successful with any improvement activity an adequate Management of Change approach is required. The third part of the tutorial is about The Application of the Model. The main steps of changing will be taught: how to create awareness, establish goals and scope for change, the assessment process, selection and planning of improvement actions, implementation and evaluation. Delegates will learn about related subjects such as the use of metrics, the requirements for the change team, the human aspects, how to deal with resistance and finally some do's and don'ts will complete the tutorial.

Since the publication of TPI in 1998 and in particular the appearance of the English version ”Test Process Improvement: a practical step-by-step guide to structured testing” (by Tim Koomen and Martin Pol) in May ‘99, the model is rapidly becoming an international standard for improving test processes, used in numerous countries like the United States, most European Community countries, Australia, South Africa, Mexico, Columbia, China and India. The model is frequently used to help an organization meet the test requirements at CMM level 3.

About the Author

Martin Pol, with more than 25 years of experiences in structured testing is the (co-)author of many publications and books on structured testing in different languages. He is the CEO and a senior consultant for Polteq International Testing Services B.V. He was involved in the development of the structured testing approach TMap and the creation of TPI. Also because he's working in the testing practice every day Martin is a highly regarded lecturer and speaker at conferences throughout the world. He twice chaired the EuroSTAR conference and recently he received the European Testing Excellence Award for his contribution to the field of testing.

Since 1989 **Ruud Teunissen** is employed in the testing world. He has been involved in a large number of ICT projects and has performed several functions within the testing organisation: tester, test specialist, test advisor, test manager, etc. Based on his experience, Ruud participated in the development of the structured testing methodology TMap and is co-author of several books on structured testing. The last years Ruud is involved in implementing structured testing within organisations on the Belgian and Dutch market. At this moment Ruud has been working in Belgium for Gitek n.v. as Manager Testen and as such responsible for all testing activities at Gitek (over 45 professional testers). Ruud is frequently speaking at conferences, universities, seminars, workshops etc. (for instance: QWE99, QWE 2000, ICS Test 2000, ICS Test 2001, EuroSTAR 2000, SAI, NOVI, Kluwer Education, IIR).

QWE2002 Tutorial C1

Mr. Rex Black
(RBCS, Inc.)

Managing the Testing Process: Organization, Motivation, and Techniques

Key Points

- Testing must fit within project and organization parameters.
- Testing is done for business reasons, achieving a quantifiable return on investment.
- Test management requires the careful application of several techniques

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 is the President and Principal Consultant of Rex Black Consulting Services, Inc., an international software and hardware testing and quality assurance consultancy. He and his consulting associates help clients such as Bank One, Compaq, Dell, Reef, Schlumberger, and others with implementation, consulting, training, and staffing for testing, test automation, and quality assurance projects. His current book, Managing the Testing Process, was published in June 1999, and popular demand has lead to an expanded and enhanced Second Edition, due to hit the shelves around April . His next book, tentatively titled Critical Testing Processes, will be published by Addison-Wesley in late 2002.

QWE2002 Tutorial D1

Dr. Erik P. vanVeenendaal
(Improve Quality Services BV)

Introduction to the Test Maturity Model

Key Points

- Gain insight, in theory and practice, on structured testing and test process improvement.
- Know about Test Maturity Model (TMM).
- Learn how to prioritise, define and implement those improvements.

Presentation Abstract

This tutorial is to provide the attendees with knowledge and insight regarding structured testing and test process improvement, both from a theoretical and practical perspective. After the tutorial, the participants will: - have knowledge about the basics of test process improvement; - have acquired insight into the most important models; - know about the Test Maturity Model (TMM); - know how to prioritise, define and implement practical test improvements; - know how to define key test metrics

About the Author

Drs. Erik P.W.M. van Veenendaal CISA has been working as a practitioner and manager within the area of software quality for a great number of years carrying out assignments in the field of quality management, project control, EDP-auditing and software testing. He has recently been responsible and involved in a number of European projects within the area of software product quality. Within this area he specializes in software testing and is the author of a number of books, e.g. “Structured testing; an introduction to TMap” and “Software quality from a business perspective”.

As a test manager and test consultant he has been involved in a great number and variety of projects, has implemented structured testing and carried out test process improvements activities in a large number of organisations. He is a regular speaker both at national and international testing conferences and a leading international trainer in the field of software testing. Erik van Veenendaal is the founder and managing director of Improve Quality Services, a company that provides services in the area of quality management and testing.

At the Eindhoven University of Technology, Faculty of Technology Management, Erik is part-time involved in lecturing and research activities. He is on the Dutch standards institute committee for software quality.

QWE2002 Tutorial E1

Dr. John McGregor
(Luminary Software)

Testing Component-Based Systems

Key Points

- Assemble systems from reusable modules, don't built from scratch.
- Component integration creates new testing challenges.
- Learn which integration defects may occur and techniques to deal with each of them.

Presentation Abstract

As software components become more readily available, the dream of assembling systems from the reusable modules rather than building them from scratch each time is becoming a reality. While the components themselves are usually well tested and defect-free, this does not guarantee that they can be integrated together without problems. The component integration process creates new testing challenges that must be addressed. This tutorial describes the kinds of integration defects that may occur and presents techniques to deal with each of them.

Attendees learn how to create and execute effective tests for component-based systems. The tutorial provides an overview of component development concepts and how to effectively test in this environment. The three component standards (DCOM, JavaBeans, and CORBA) are described. Effective integration and testing strategies are presented.

About the Author

Dr. John D. McGregor is a partner in Luminary Software and an associate professor of computer science at Clemson University. Dr. McGregor has developed architecture, design and testing techniques for object-oriented software and developed custom testing processes for a variety of companies. Dr. McGregor is co-author, with David A. Sykes, of "Object-oriented Software Development: Engineering Software for Reuse" published by Van Nostrand Reinhold and "A Practical Guide to Testing Object-Oriented Software" published by Addison-Wesley in 2001. He writes a column on Testing and Quality for the Journal of Object-oriented Programming (JOOP) published by SIGS/101Communications Publishing. He has published numerous articles on software development focusing on design and quality issues. Currently he serves on the editorial board of two journals: JSTP and IJCIS. Dr. McGregor's research interests include software engineering specifically in the areas of process definition, design quality, testing and measurement.

QWE2002 Tutorial B2

Mr. Bernard Homes
(TESSCO Ltd.)

Test Plan 101 Using IEEE 829

Key Points

- Description of different types of documents used in the IEEE 829 standard, and their relation to one another.
- Different steps and aspects of this test plan.
- How to use the plan for different types of tests.

Presentation Abstract

The test plan is the base of all the future tests for an SW application. It is a kind of contract between the test team, the management and the development team.

This presentation first describes the different types of documents used in the IEEE 829 standard and their relations to one another. Building on this information it details the different steps and aspects of the test plan as described in the standard and the relations between the different chapters, how they complement each other in order to provide a complete solution. From there it shows how to use the plan for different types of tests such as functional / regression testing, performance testing and configuration tests. The presentation ends with the application of sub-plans to cover the different aspects of testing for a complex application.

This presentation is based on real life experience and implementation of test plans developed using this and different methods and their comparison.

About the Author

Bernard Homès is founder and principal consultant for TESSCO ltd (Tests Evaluation Service & Solution, Conseil & Organisation limited). After 20 years in software development (banking, ERP, ...) and testing with different consulting firms, he set up the TESSCO group in Europe (France and UK) and Canada. Currently Bernard helps a major Canadian customer in setting up a software test center, and provides trainings and consultancy services worldwide.

QWE2002 Tutorial C2

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.

QWE2002 Tutorial D2

Mr. Otto Vinter
(Independent Consultant)

Introduction to Defect Analysis

Key Points

- How to perform defect analysis.
- Which taxonomies and techniques exist.
- What results can you expect; and how to use them for improvements.

Presentation Abstract

This half-day tutorial will introduce you to the concepts and techniques of defect analysis. You will learn how to analyse defects in your own organisation, and what benefits you can achieve. The tutorial will explain the different defect categories in the currently most comprehensive Bug Taxonomy (by Beizer). I will demonstrate how I used this taxonomy to perform defect analysis in practice on a number of projects, and I will share my results with you. I will also give an overview of other defect classifications and techniques e.g. Orthogonal Defect Classification (by Chillarege e.a.), and show how they might be used.

However, performing defect analysis is of little value if you cannot use the results to correct problems in or improve your software development practices. I will demonstrate how you can use the data from defect analysis to introduce focused improvements in your organisation. I have used defect analysis successfully on a number of projects at Brüel & Kjaer. I have demonstrated improvements to the testing and the requirements engineering processes based on the results of defect analysis. At previous QWE conferences I have reported on these experiences. However, I have not previously presented the defect analysis technique itself.

About the Author

Otto Vinter is an independent software engineering mentor. He is also employed as project manager specialising in software process improvements with the Danish software engineering consulting company DELTA.

Previously, he was responsible for software process improvements at Brüel & Kjær. He has been the driving force in that company's improvement activities in testing, requirements engineering, development models, and introduction of object-oriented development methods. Most of the process improvement activities have been based on findings from analysis of defects.

He has managed software development projects for 30 years. He received his Masters Degree in Computer Science from the Danish Technical University in 1968. He is a regular speaker at conferences and seminars, performs mentoring activities for clients, and is an expert evaluator on the framework programmes of the CEC.

More information can be found at: http://inet.uni2.dk/~vinter/engindex.htm

QWE2002 Tutorial E2

Dr. John McGregor
(Luminary Software)

Guided Inspection of UML Models

Key Points

- Describe guided inspections.
- Create test cases to be used in guided inspections.
- Choose test cases that maximize defect discovery while minimizing thier cost.

Presentation Abstract

Early detection of faults is a cost-effective technique for ensuring quality. This tutorial presents the guided inspection technique for analyzing UML models. This technique uses explicit test cases to guide the inspection process rather than leaving the coverage of the model to chance. The technique systematically determines whether the model is complete, correct and consistent. Participants will actively participate in an exercise that examines the model for a small distributed system. Participants will also gain an understanding of how to integrate the technique into the typical interactive, incremental process.

About the Author

Dr. John D. McGregor is a partner in Luminary Software and an associate professor of computer science at Clemson University. Dr. McGregor has developed architecture, design and testing techniques for object-oriented software and developed custom testing processes for a variety of companies. Dr. McGregor is co-author, with David A. Sykes, of "Object-oriented Software Development: Engineering Software for Reuse" published by Van Nostrand Reinhold and "A Practical Guide to Testing Object-Oriented Software" published by Addison-Wesley in 2001. He writes a column on Testing and Quality for the Journal of Object-oriented Programming (JOOP) published by SIGS/101Communications Publishing. He has published numerous articles on software development focusing on design and quality issues. Currently he serves on the editorial board of two journals: JSTP and IJCIS. Dr. McGregor's research interests include software engineering specifically in the areas of process definition, design quality, testing and measurement.

QWE2002 Tutorial F1 & F2

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

Presented at PNSQC 2000, attended by 54 people. First presentation outside of Intel, where it was taught to more than 1,000 students around the world this year. Of the 45 PNSQC evaluations returned, 44 would recommend the workshop to others and found it either "valuable" or "very valuable". People like the fast pace and depth of information presented. 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.

QWE2002 Tutorial G1

Mr. Jens Pas
(I2B)

Emotional Intelligence as the Key to Software Testing

Key Points

- Understanding Emotional Intelligence (EQ).
- Understanding the Test Requirements Hierarchy.
- Applying Strategic Driven thinking to evelop test requirements.
- Enhancing test requirements with emotions and instincts.

Presentation Abstract

We explain what Emotional Intelligence is about and what the relation towards software quality is. Before the use of EQ in software quality is explained, a technique called Test Requirements Hierarchy (or Test Requirement Matrix) (TRH or TRM) will be explained. This classic technique, though too little applied correctly, allows the objective measurement of quality against an agreed baseline. The conservative application of the technique only measures quality in a rational dimension. With EQ we will add two more dimensions to the TRH or TRM.

A real case will be used to develop a TRH and to develop the EQ-dimensions for it. Anonymous real life data, such as interview notes, requirements documents and progress and test reports will be used.

About the Author

Jens Pas is Managing Director and founding Partner of I2B and President of the Board of Directors. Jens Pas is an authority in the business of Software Testing. He developed a unique method for structured software testing. His method was introduced in and adopted by major Belgian and Dutch companies. He has also development Test Assessment programs and has conducted them at directors and top-management level at various sites in larger IT environments. He published various articles regarding the methodology of Software Testing and is the author of the textbook that accompanies the course “Introduction to Structured Software Testing”. He currently publishes his views on software engineering monthly in three area related magazines and newspapers. He is a (keynote) speaker and track chairman on many Software Engineering and Quality Conferences (Eurostar 98 Munich, 99 Barcelona & 2000 Copenhagen, Quality Week Europe, Fesma Brussels, Software Automation, Novi Amsterdam, Technological Institute of the Royal Society of Belgian Civil Engineers, SQE Europe Test Congress 2000). Currently he is specialising in Innovation Management services.

Jens Pas was co-founder and General Manager of a Belgian consultancy company specialised in testing. Previously he was responsible for the re-engineering of the pan-European IT-support department of Quaker Oats Europe and for setting up the Application Support department of the ETS division of Barco Graphics. Jens Pas is an engineer and holds an Executive International Master in Business Administration degree from the Vlerick Leuven Gent Management School.

QWE2002 Tutorial H1

Mr. Robert Sabourin
(Amibug.com)

Getting Started - Stressing Web Applications, Stress Early - Stress Often

Key Points

This tutorial outlines very practical steps to follow, with real exemples from recent e-commerce projects:

- Test planning: What works or does not work, when to use what tool and technique
- How to avoid blowing the budget for Stress Testing, Load Testing, Performance Testing
- How to set up a test lab which allows for stress testing at the Unit, Integration and System Testing phases.

Presentation Abstract

This tutorial outlines very practical steps to follow when setting out to stress web applications - preferably well before they actually go live. Real examples from recent e-commerce projects are described - what works - what does not work - when to use what tools or technique? How to avoid blowing the budget! What can we tell about the way our application will react to success? Will the application scale? How well? Can we tell? What can we tell? What can we simulate? Can we set up a test lab organization which allows for stress testing at the Unit, Integration and System testing phases?

Techniques used in live e-commercce development sites are described. Where can and should stress testing be implemented, at what stage of development? What do we want to measure, how and why? Practical tools and techniques are reviewed! Examples are from the author's current experience related to testing some key e-commerce sites - noteably the Virtual Model Shopping Experience at www.landsend.com - www.jcpenny.com and many more extremely popular e-commerce sites!

The tutorial also includes elements of test planning and some practical examples of "risk analysis" applied to internet testing!

About the Author

Robert Sabourin has been involved in all aspects of development, testing and management of software engineering projects. Robert graduated from McGill University in 1982. Since writing his first program in 1972, Robert has become an accomplished software engineering and SQA management expert and evangelist (don't tell me it can't de done!). He is presently the President of AmiBug.Com Inc; a Montreal-based international management consulting firm specializing in the implementation of "light effective process" to achieve excellence in delivering on-time, on-quality, on-budget commercial software solutions. AmiBug.Com provides management consulting, training and professional development directly and with various business partners.

QWE2002 Tutorial J1

Mr. Rex Black
(RBCS, Inc.)

Managing The Testing Process: Test Estimation

Key Points

- Test Estimating Techniques
- What you Need to Know to Estimate the Testing Effort
- Guidelines for the Estimating Process

Presentation Abstract

Based on the upcoming Second Edition of his 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 is the President and Principal Consultant of Rex Black Consulting Services, Inc., an international software and hardware testing and quality assurance consultancy. He and his consulting associates help clients such as Bank One, Compaq, Dell, Reef, Schlumberger, and others with implementation, consulting, training, and staffing for testing, test automation, and quality assurance projects. His current book, Managing the Testing Process, was published in June 1999, and popular demand has lead to an expanded and enhanced Second Edition, due to hit the shelves around April . His next book, tentatively titled Critical Testing Processes, will be published by Addison-Wesley in late 2002.

QWE2002 Tutorial K1

Mr. Ibrahim K. El-Far
(Florida Institute of Technology)

Model-Based Testing

Key Points

- Graphs and Finite state machine models for software testing
- Methods for modeling software using state machines: simple graphs, state variables, hierarchical state models construction
- Generating tests based on state models

Presentation Abstract

Model-Based Testing has its roots in hardware testing applications - most notably telephone switches - and recently has spread to a reasonably broad variety of software domains. The wealth of published work portraying case studies in both academic and industrial settings is a sign of growing interest in this style of testing. This presentation discusses model-based testing in terms of finite state machines, one of the more popular software models that are employed in testing. The audience will be exposed to basic skills in theories of graphs and state machines that are useful in understanding and building state models. Various methods to build models varying in their difficulty and in their dependency on tools are presented. Next, the audience will be shown basic test generation techniques with state models. Throughout the tutorial, there will be working examples illustrating the techniques being taught. Finally, there will be some exercises for the audience to acquire an initial hands-on experience with this style of testing.

About the Author

Ibrahim K. El-Far is a doctoral student in computer science under James A. Whittaker at the Florida Institute of Technology, Melbourne, Florida. His research interests are in investigating software models, test automation and tools, adequacy criteria, and software testing education. In 2000, El-Far received an IBM CAS Fellowship supporting his research in software testing. He has over four years of experience in model-based testing using finite state machines at the Center for Software Engineering Research. He has acted in the capacity of an advisor to model-based testing groups in the Center, and he has supervised the development of experimental state-model testing tools. He has also taught this material in various formats to a variety of students both as part of a normal university course and as part of a program to train students who wish to participate in related industry-sponsored research.

QWE2002 Tutorial G2

Mr. Olivier Denoo
(ps_testware)

Think Green! Think Different! Think Modular! (The LEGO Principle)

Key Points

- Efficient solution for web and e-commerce testing
- Web requirement hierarchy.
- Efficiency, effectiveness, improvement.

Presentation Abstract

Testing E-commerce web sites and virtual shops requires the right approach. Neglecting business (or better, consumer)-driven tests when your web site is the only interface that separates you from your customers and their money could obviously lead to a major business failure if not a crash. Fine, but how can we come to a decent result without messing around such a huge set of tests, such a huge bunch of information? The answer is simple: structure your test requirements in an appropriate structure: TRH (Test Requirement Hierarchy). The core of the presentation shows how to implement a TRH approach within e-commerce applications, knowing that the new (Net) economy is not subject to any speed limits.

Our method showed that it was possible to gain sometimes up to 70-80% time with no significant quality decrease. Because it is modular, we called it The LEGO principle.

About the Author

Olivier Denoo is a consultant active in the Business of Structured Software Testing since 1997. He was the key-developer of ps_testware's Y2K testing techniques. Nowadays, he still is a highly respected trainer of this methodology. Continuously looking for new challenges, Olivier started to investigate the possibilities of website and e-commerce testing and became in charge of ps_testware's e-commerce/WWW-testing knowledge base. Currently, he is working in a project at one of the largest Belgian Telecom companies. To be supplied.

QWE2002 Tutorial H2

Mr. Robert Sabourin
(Amibug.com)

Just In Time Testing - Testing Turbulent Web Based Applications

Key Points

Some real examples, from recent e-commerce projects, are described - what works - what does not work - when to use what tools or technique? How to avoid blowing the budget! Practical tools and techniques are reviewed! Examples are from the author's current experience related to testing some key e-commerce sites - noteably the Virtual Model Shopping Experience at www.landsend.com - www.jcpenny.com and many more extremely popular e-commerce sites!

- Risk based test management.
- Data driven testing.
- Web application testing.

Presentation Abstract

With development cycles becoming extremely short it is important to "plan for", "prepare for" and "react to" risks as they come up during the project.

This tutorial outlines practical steps to follow when setting out to test turbulent web applications. Turbulent web applications experience almost daily requirement changes coupled with frequent changes to existing user interfaces and the continual integration of new functions and features.

Here are some of the points this tutorial looks at.

- How do you get the most out of limited testing resources available?
- How to perform GUI testing and Usability testing in parallel with functional testing?
- How to identify testing objectives?
- How to implement testability hooks to facilitate automation and regression testing?

About the Author

Robert Sabourin has been involved in all aspects of development, testing and management of software engineering projects. Robert graduated from McGill University in 1982. Since writing his first program in 1972, Robert has become an accomplished software engineering and SQA management expert and evangelist (don't tell me it can't de done!). He is presently the President of AmiBug.Com Inc; a Montreal-based international management consulting firm specializing in the implementation of "light effective process" to achieve excellence in delivering on-time, on-quality, on-budget commercial software solutions. AmiBug.Com provides management consulting, training and professional development directly and with various business partners.

QWE2002 Tutorial J2

Mr. Martin Pol & Mr. Ruud Teunissen
(Polteq, Gitek nv)

Structured Testing

Key Points

- The Context of Testing
- Why Testing?
- Structured testing: What, When, Where, Who?

Presentation Abstract

This tutorial provides general answers to the questions of what, when, how, with what, where and who in matters of testing. The answers to the questions what and when are given in a life cycle model, which provides the different test phases and activities for all testing levels (from unit test to user acceptance test). The answers to how are given by techniques available for planning, preparing, specification and executing the various tests. The questions with what and where are considered in the description of how to implement the required infrastructure and how to automate the test process adequately. The who aspects deal with the organisation of the test process, e.g. the required expertise, the people and team aspects, resource management, the responsibilities, testware management, etc. Within the tutorial special attention is paid to the application of systematic software testing in different projects, e.g.: Maintenance; RAD; Client/Server & GUI; OO and CBD; ERP; Web & E-BusinessTesting; The Test Factory, etc.

The foundations for the tutorial are taken from TMap, a comprehensive, generic test approach that is applicable for every test level and test activity. It's generic structure enables testers to link their know-how to any innovation in the industry and development method. TMap is a preferred standard in more that 300 companies, including banks, insurance companies, pension funds and government agencies as well in smaller organizations and sectors such as embedded software, telecommunications and logistics.

About the Author

Martin Pol, with more than 25 years of experiences in structured testing is the (co-)author of many publications and books on structured testing in different languages. He is the CEO and a senior consultant for Polteq International Testing Services B.V. He was involved in the development of the structured testing approach TMap and the creation of TPI. Also because he's working in the testing practice every day Martin is a highly regarded lecturer and speaker at conferences throughout the world. He twice chaired the EuroSTAR conference and recently he received the European Testing Excellence Award for his contribution to the field of testing.

Ruud Teunissen is employed in the testing world since 1989. He has been involved in a large number of ICT projects and has performed several functions within the testing organisation: tester, test specialist, test advisor, test manager, etc. Based on his experience, Ruud participated in the development of the structured testing methodology TMap and is co-author of several books on structured testing. The last years Ruud is involved in implementing structured testing within organisations on the Belgian and Dutch market. At this moment Ruud has been working in Belgium for Gitek n.v. as Manager Testen and as such responsible for all testing activities at Gitek (over 45 professional testers). Ruud is frequently speaking at conferences, universities, seminars, workshops etc. (for instance: QWE99, QWE 2000, ICS Test 2000, ICS Test 2001, EuroSTAR 2000, SAI, NOVI, Kluwer Education, IIR).

QWE2002 Tutorial K2

Dr. Rini vanSolingen
(CMG)

Software Measurement: The Goal/Question/Metric Method

Key Points

- Goal-oriented Software Measurement.
- Real-life Industrial Examples.
- Excercises to learn basic principles.

Presentation Abstract

Tutorial on the Goal/Question/Metric methode (GQM), the de-facto standard for measurement of software practices. By measuring software processes and products, company specific knowledge on software practices becomes available. This knowledge can be used for decision making, planning, or estimating project cost or duration. The main subject of the tutorial is a method for software measurement: the Goal/Question/Metric method. This method originates from the USA; however, this method is also increasingly applied in Europe. The GQM method is largely expanded during the last years. Beside much theoretical expansions, a practical handguide has recently become available.

About the Author

Dr. Rini vanSolingen is a Principal Consultant at CMG, the Netherlands. Within this role, he is specialised on SW Product and Process Improvement. In the past years, he has been a senior quality engineer at Schlumberger/Tokheim and has been head of the Quality and Process Engineering department at the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern. Rini's activities focus on practical software product quality and process improvement for industry. Within this topic, he is specialised in industrial software measurement and product-focused SPI. He has published more than 50 papers in international journals and conference proceedings, among which the McGraw-Hill software measurement book: 'The Goal/Question/Metric method: a practical guide for quality improvement of software development'. See: Http://www.gqm.nl/.

QWE2002 Paper K1-1

Dr. Linda Rosenberg
(GSFC NASA)

Independent Verification And Validation Implementation At NASA

Key Points

- IV&V is a valuable tool for increasing sofware quality and reliability
- IV&V needs selection criteria for which projects should apply it
- NASA is now requiring IV&V evaluation for all projects and application on some

Presentation Abstract

This paper will discuss the management approach taken to consistently apply IV&V on NASA projects. We will discuss the development of the NASA policy stating that all projects and programs with few exceptions shall employ software V&V techniques for risk mitigation; criteria shall be applied to determined if IV&V is warranted. The IV&V selection criteria are based on determining the extent of risk. Developed by an Agency wide group, the criteria provide an algorithm for estimating probability of software failure based on quantitative software complexity, development team, management, and other factors. Factors for characterizing the consequences of software failure are determined, then thresholds for performing IV&V based on probability of failure and severity of impact (consequences) are applied. In this paper, we will discuss how these guidelines were developed and currently being applied. We will conclude with a discussion on the approach to projects, how do you convince the project manager to implement IV&V, and what is the cost to the project.

About the Author

Dr. Rosenberg is the division chief of the Software Assurance Technology Office (SATO) in the Office of Systems Safety and Mission Assurance Directorate at Goddard Space Flight Center (GSFC), NASA. She also serves as the Software Assurance technologist for GSFC and NASA. Dr. Rosenberg is a recognized International expert in the areas of software assurance, software metrics, requirements and reliability. She serves on IEEE program committees for software reliability, software metrics, and software requirements. She has chaired sessions at many international conferences, included those sponsored by NASA and AIAA. Dr. Rosenberg also reviews papers for the Department of Defense sponsored conferences, and other industrial organizations for software quality. She has presented papers and tutorials in many areas of software assurance, including reliability, at IEEE and ACM international conferences and the International Astronautical Congress. Dr. Rosenberg is currentl! y on the steering committee to evaluate the metrics in the IEEE 982 Reliability Standard. Dr. Rosenberg is also an adjunct professor at University of Maryland, Baltimore for the Masters/Doctoral Program.

Dr. Rosenberg holds a Ph.D. in Computer Science from the University of Maryland, an M.E.S. in Computer Science from Loyola College, and a B.S. in Mathematics from Towson University. She is a member of Electrical and Electronic Engineers (IEEE), the IEEE Computer Society, the Association for Computing Machinery (ACM) and Upsilon Pi Epsilon.

Independent Verification and Validation Implementation at NASA

Dr. Linda H. Rosenberg
Chief Scientist for Software Assurance Technologies
Goddard Space Flight Center, NASA
301-286-0087
Linda.Rosenberg@gsfc.nasa.gov

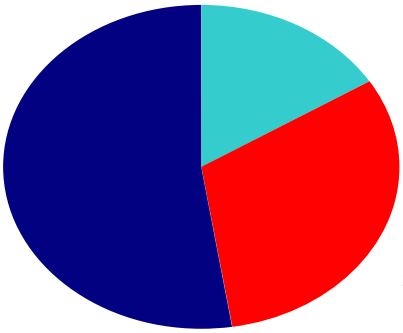
Overview

- WHY IV&V
- WHAT is IV&V
  - Definition
  - Approach
  - Activities
- HOW to implement IV&V
  - Policy
  - Criteria
  - Example
- Implementation at NASA
- Benefits
- Summary

WHY IV&V - Software Chaos

The Standish Group examined 8,380 Software Projects.

53% "Challenged"
- Over budget by 189%
- Late by 222%
- Missing 39% of Capabilities

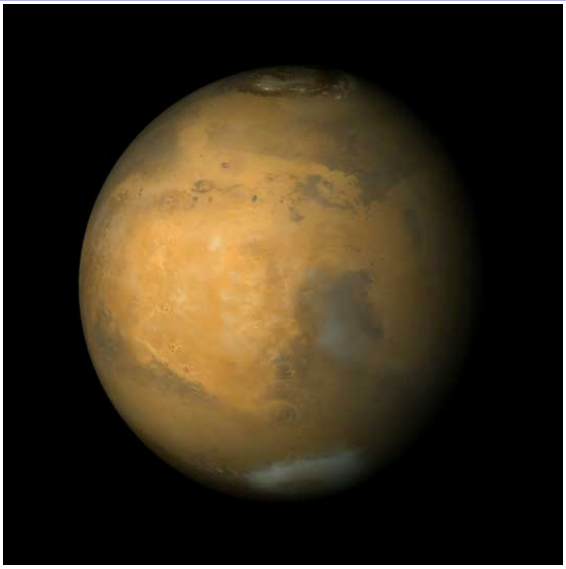


16% Successful
- In Budget
- On Time
- Meets Requirements
- User involved

Note: For Large Companies
- 9% were Successful
- 61.5% Challenged
  - Over budget by 178%
  - Late by 230%
  - Missing 58% of Capabilities
- 29.5% were Cancelled

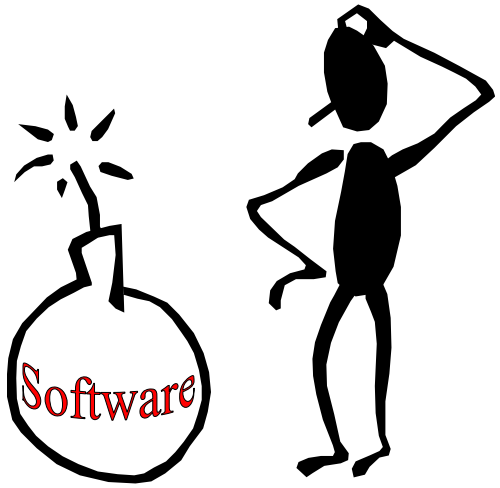
31% Cancelled
- Development

Why IV&V at NASA



MARS

WHAT is IV&V



Independent Verification & Validation

Independent

- Technical: IV&V prioritizes its own efforts
- Managerial: Independent reporting route to Program Management
- Financial: Budget is allocated by program and controlled at high level such that IV&V effectiveness is not compromised

Verification ***(Are we building the product right?)***

- The process of determining whether or not the products of a given phase of the software development cycle fulfill the requirements established during the previous phase
- Is internally complete, consistent and correct enough to support next phase

Validation ***(Are we building the right product?)***

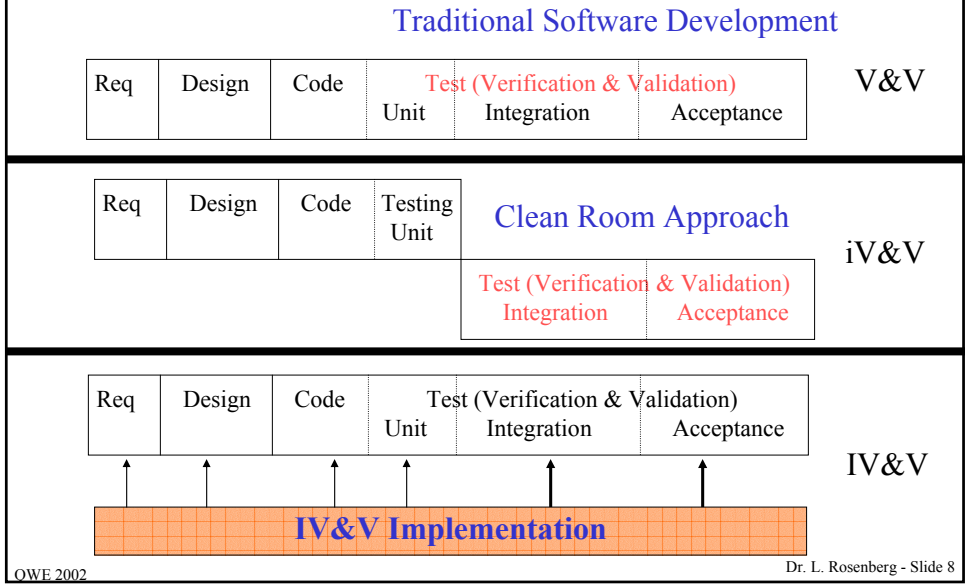
- The process of evaluating software throughout its development process to ensure compliance with software requirements. This process ensures:
  - Expected behavior when subjected to anticipated events
  - No unexpected behavior when subjected to unanticipated events
  - System performs to the customer's expectations under all operational conditions

IV&V

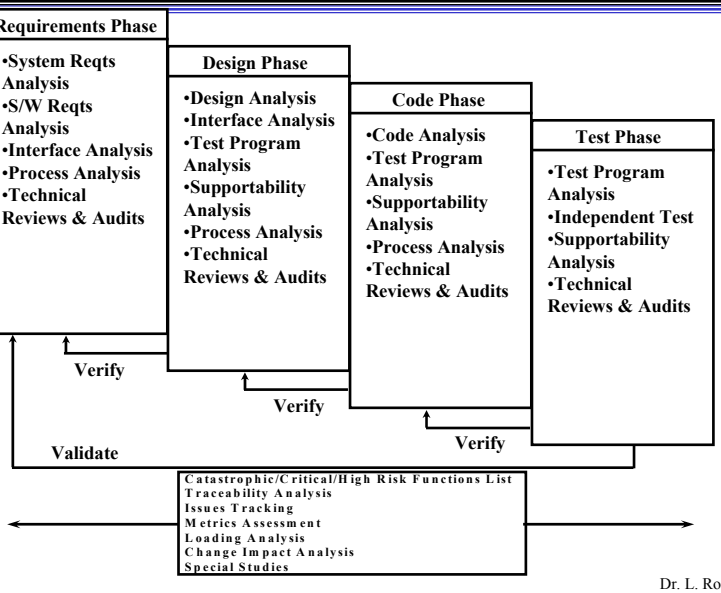
Software IV&V is a systems engineering process employing rigorous methodologies for evaluating the correctness and quality of the software product throughout the software life cycle

Adapted to characteristics of the target program

IV&V Approach



IV&V Activities



HOW to Implement IV&V

1 - Policy
2 - Criteria
  - Development Factors
  - Consequence Factors
  - Example
3 - Implementation

NASA IV&V Policy

"It is NASA policy that all efforts covered by NPG 7120.5 and other software programs and projects shall employ software V&V techniques for risk mitigation. Each project is required to produce and document a plan that addresses the performance of V&V over the life cycle of the software design and development."

"…the IV&V Facility is responsible for the management of all software IV&V within the Agency"

" Unless justified otherwise, the IV&V Facility shall execute this role by directly performing all IV&V at the WV Facility"

NASA IV&V Policy

Steps:

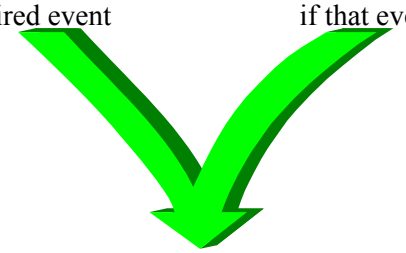
- Program/Project with input from the IV&V Facility shall recommend the selection of whether IV&V shall be performed on any particular software effort
- Governing Program Management Council responsible for the Project is charged with approving this recommendation
- Selection of software for IV&V is subject to review of the IV&V Facility
- All IV&V is managed by the Facility regardless of being done at the Facility or project site

IV&V Criteria

IV&V is intended to mitigate risk

Probability
of an undesired event

Consequences
if that event should occur



Risk = Probability * Consequence

∴ IV&V must be based on Risk Probability & Consequence

Risk & IV&V

- Interim IV&V Criteria were developed by the inter-Center Software Working Group (SWG)
- Criteria provide:
  - Algorithm for estimating probability of software failure based on quantitative software complexity, development team, management, and other factors
  - Factors for characterizing the consequences of software failure
  - Thresholds for performing Independent Assessment or IV&V based on probability of failure and severity of impact (consequences)

IV&V Criteria Risk Factors

Factors that impact the difficulty of the development

- Software Team Complexity
- Contractor Support
- Organization Complexity
- Schedule Pressure
- Process Maturity of Software Provider
- Degree of Innovation
- Level of Integration
- Requirement Maturity
- Software Lines of Code

Risk Factors

Software team complexity –The larger the team the more complex and formal the communication and more points of failure. Smaller teams are generally co-located, making communication a common occurrence.

Contractor support –The presence of a contractor (other than the prime system developer) introduces a layer of contract management and communication between the government and the actual software developer.

Organization complexity –A single organization working from multiple locations faces a slightly greater challenge than an organization in one location. When the software development is accomplished by multiple organizations working for a single integrator, the development is significantly complicated.

Risk Factors (cont.)

Schedule pressure - A deadline is negotiable if changing the deadline is possible although it may result in slightly increased cost, schedule delays, or negative publicity. A deadline is non-negotiable if it is driven by immovable event such as an upcoming launch window. Project working under deadlines are more likely to take short cuts such as decrease testing, hence potentially decreasing the reliability and safety, and increasing the risk.

Process maturity of software provider - Application of the Software Engineering Institute's Capability Maturity Model (CMM) increases the quality of the software developed, resulting in higher reliability. This model has 5 levels of attainment, 3 being the most common, and all projects start out at Level 1.

Risk Factors (cont.)

Degree of innovation – When attempting to formulate a new equation or concept, there is a higher risk than if the software has been proven in previous programs.

Level of integration – Programs that stand alone and are not integrated with any other components have a higher probability of success due to simplicity of their interfaces. The more extensive integration of multiple components, the higher the risk.

Requirement maturity – Requirements that are constantly changing, ambiguous or untestable are at a higher risk than those requirements whose objectives are well defined with few or no unknowns.

Software Lines of Code – The larger the project, the higher the risks. Lines of code is used as a size measurement and is defined to be all code in the programs, including software that is reused and auto generated code.

IV&V Criteria Risk Factors Weighting

- Software Team Complexity x2
- Contractor Support x2
- Organization Complexity x1
- Schedule Pressure x2
- Process Maturity of Software Provider x2
- Degree of Innovation x1
- Level of Integration x2
- Requirement Maturity x2
- Software Lines of Code x2

IV&V Criteria Risk Factors

Factors contributing to probability of software failure	Un-weighted probability of failure score					Weighting Factor	Likely-hood of failure rating
	1	2	4	8	16		
Software team complexity	Up to 5 people at one location	Up to 10 people at one location	Up to 20 people at one location or 10 people with external support	Up to 50 people at one location or 20 people with external support	More than 50 people at one location or 20 people with external support	X2	
Contractor Support	None	Contractor with minor tasks		Contractor with major tasks	Contractor with major tasks critical to project success	X2	
Organization Complexity*	One location	Two locations but same reporting chain	Multiple locations but same reporting chain	Multiple providers with prime sub relationship	Multiple providers with associate relationship	X1	
Schedule Pressure**	No deadline		Deadline is negotiable		Non-negotiable deadline	X2	
Process Maturity of Software Provider	Independent assessment of Capability Maturity Model (CMM) Level 4, 5	Independent assessment of CMM Level 3	Independent assessment of CMM Level 2	CMM Level 1 with record of repeated mission success	CMM Level 1 or equivalent	X2	
Degree of Innovation	Proven and accepted		Proven but new to the development organization		Cutting edge	X1	
Level of Integration	Simple - Stand alone				Extensive Integration Required	X2	
Requirement Maturity	Well defined objectives - No unknowns	Well defined objectives - Few unknowns		Preliminary objectives	Changing, ambiguous, or untestable objectives	X2	
Software Lines of Code***	Less than 50K		Over 500K		Over 1000K	X2	
Total							

Table 1 Likelihood of Failures Based on Software Environment

Criteria Application Example

- Software team complexity – "up to 20 people at one location" = 4 * 2 = 8
- Contractor support – "with minor tasks" = 2 * 2 = 4
- Organizational complexity – "two locations but with same reporting chain" = 2 * 1 = 2
- Schedule pressure – "non-negotiable" = 16 * 2 = 32
- Process maturity – " CMM Level 1 but with a successful history" = 8 * 2 = 16
- Innovation – between proven but new and cutting edge = 8 * 1 = 8
- Integration – almost stand alone = 2 * 2 = 4
- Requirement maturity – "preliminary objectives" = 8 * 2 = 16
- Lines of code = ~ 300K = 2 * 2 = 4

TOTAL = 8+4+2+32+16+8+4 +16+4= 94

Consequence Factors

GRAVE

SUBSTANTIAL

MARGINAL

INSIGNIFICANT

- Potential for loss of life
- Potential for serious injury
- Potential for catastrophic mission failure
- Potential for partial mission failure
- Potential for loss of equipment
- Potential for waste of software resource investment-
- Potential for adverse visibility
- Potential effect on routine operations

Consequence Factors - Sample

Grave

- Potential for loss of life - Yes
- Potential for loss of equipment – Greater than $100,000,000
- Potential for waste of resource investment – Greater than 200 work-years on software
- Potential for adverse visibility - International

Substantial

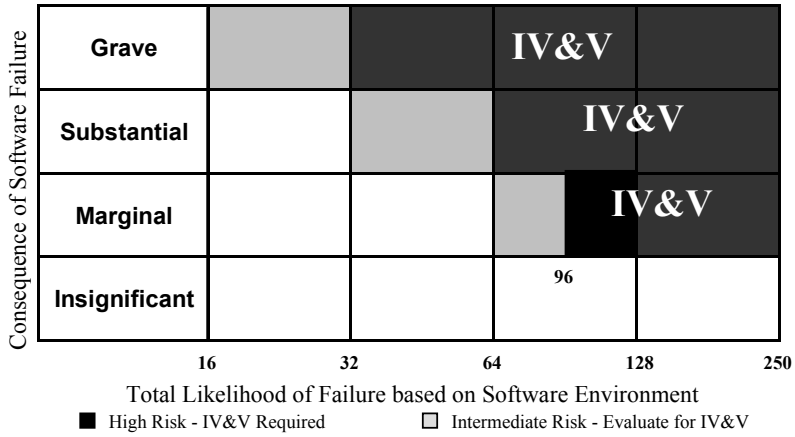
- Potential for serious injury – Yes
- Potential for catastrophic mission failure – Yes
- Potential for loss of equipment – Greater than $20,000,000
- Potential for waste of resource investment – Greater than 100 work-years on software
- Potential for adverse visibility - National
- Potential effect on routine operations – Agency work stoppage

Consequence Factors - Example

- Software controls life-support systems (potential loss of life)
- There is no potential for loss of equipment
- There is potential for adverse visibility at the Agency level
- There would be no effect on routine operations

➔ **"GRAVE"**

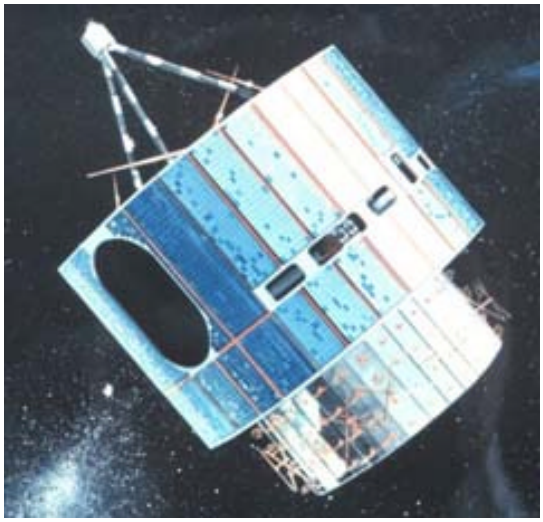
Criteria Determination for IV&V



Example: 94 Development Factors
Grave Consequence

IV&V Required

Criteria Application at NASA



Tale of Two Implementations

NASA = 10 Centers
100 projects evaluated (Approximately)

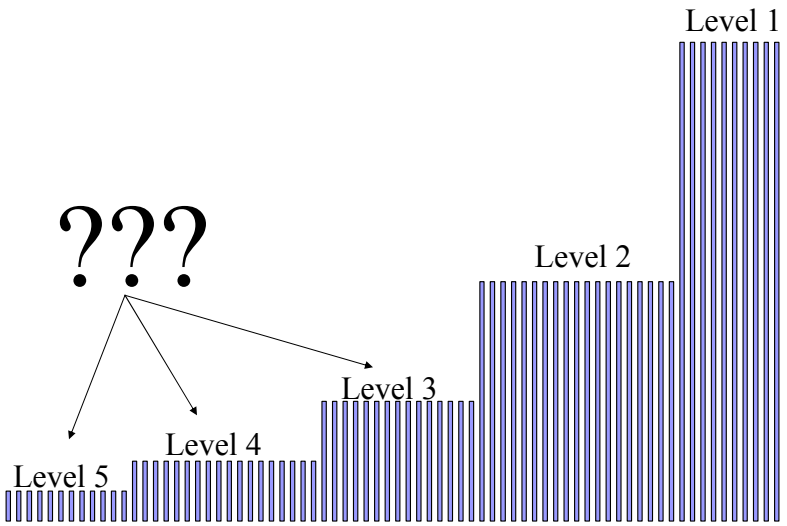
Projects potential for IV&V = 70

Distribution/ data collection
- e-mail
- interview

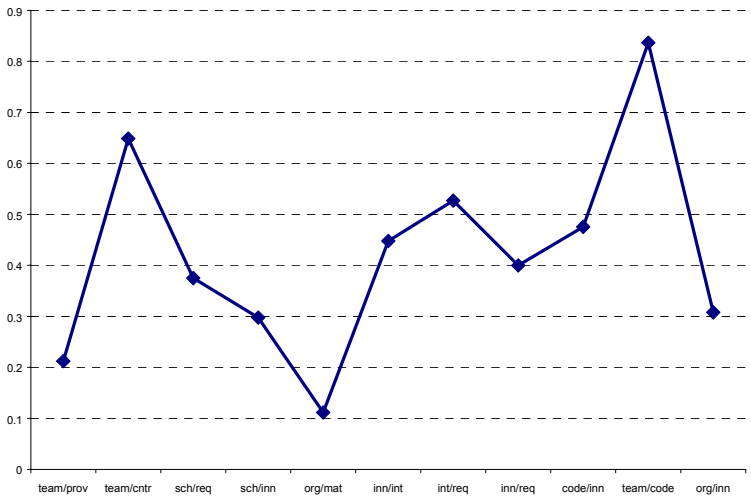
Results

Score	Count	Scores	
> 32	24	24 - 232	**Grave** Potential for loss of life - Yes Potential for loss of equipment – Greater than $100,000,000 Potential for waste of resource investment – Greater than 200 work-years on software Potential for adverse visibility - International
> 64	23	24 - 142	**Substantial** Potential for serious injury – Yes Potential for catastrophic mission failure – Yes Potential for loss of equipment – Greater than $20,000,000 Potential for waste of resource investment – Greater than 100 work-years on software Potential for adverse visibility - National Potential effect on routine operations – Agency work stoppage
> 96	23	24 - 149	**Marginal** Potential for partial mission failure - Yes Potential for loss of equipment – Greater than $2,000,000 Potential for waste of resource investment – Greater than 20 work-years on software Potential for adverse visibility - Agency Potential effect on routine operations – Center work stoppage or Agency inconvenience
	3	24	**Insignificant** Potential for loss of life - No Potential for serious injury – No Potential for catastrophic mission failure – No Potential for partial mission failure – No Potential for loss of equipment – Less than $2,000,000 Potential for waste of resource investment – Less than 20 work-years on software Potential for adverse visibility – No more than local visibility Potential effect on routine operations – No more than a Center inconvenience

Level of Maturity Data



Correlations



Software Team Complexity	Contractor Support	Organization Complexity	Schedule Pressure	Process Maturity of Software Provider	Degree of Innovation	Level of Integration	Requirement Maturity	Software Lines of Code

IV&V Roll Out

GRAVE addressed first

SUBSTANTIAL with risk > 32 in requirements or design phase

SUBSTANTIAL with risk > 32 past design

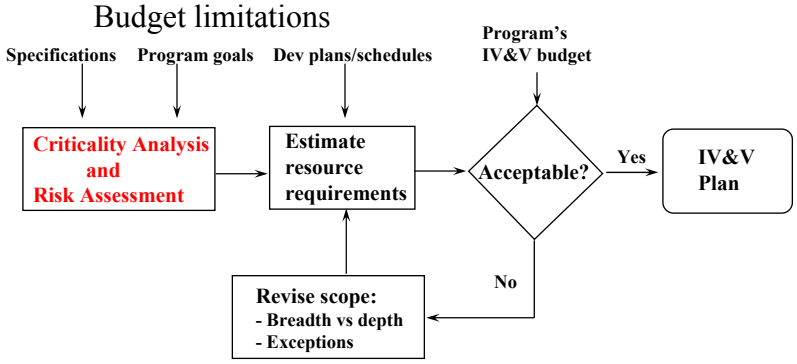
MARGINAL with risk >96 started with those closest to inception

IV&V Scope

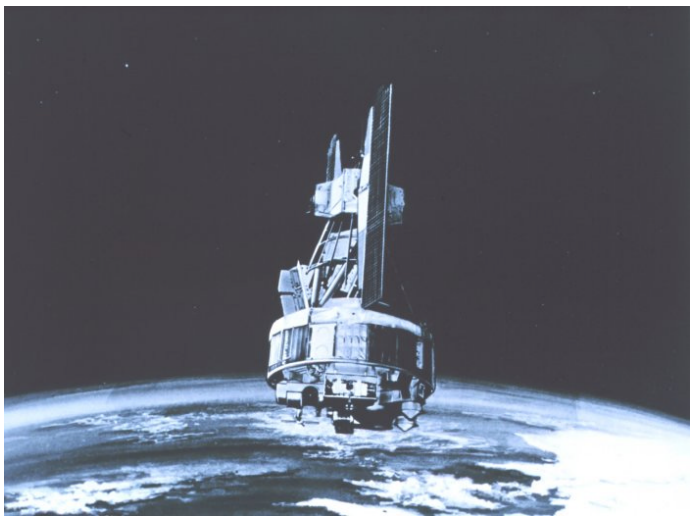
Scope is determined so as to minimize the risk within the Program's IV&V budget. Effort is based on:

Criticality and risk of system functions performed/managed by software

Budget limitations



Summary



IV&V

Software IV&V, as practiced by the NASA Software IV&V Facility, is a well-defined, proven, systems engineering discipline designed to reduce the risk in major software developments.

IV&V Benefits

Technical	Management
•Better software/system Performance	•Better Visibility into Development
•Higher Confidence in Software Reliability	•Better Decision Criteria
•Compliance between Specs & Code	•Second Source Technical Alternative
•Criteria for Program Acceptance	•Reduced maintenance cost
	•Reduced Frequency of Operational Change

NASA IV&V Benefits

Across different software domains

Manned Space - > 4,000 problems identified
10 highest criticality (loss of mission or life)

Experimental Flight vehicles > 300 requirements and design problems

Ground systems: > 250 requirements and mitigation problems

Conclusion

- Applied early in the software development process, IV&V can reduce overall Project cost.
- NASA policy provides the management process for assuring that the right level of IV&V is applied.
- IV&V Implementation Criteria provide a quantitative approach for determining the right level based on mission risk

References

IV&V Facility

Director - Judy Bruner
jbruner@pop500.gsfc.nasa.gov

Deputy Director - Bill Jackson
William.L.Jackson@ivv.nasa.gov

Criteria Development - Ken McGill
mcgill@orion.ivv.nasa.gov

Independent Verification and Validation Implementation at NASA

Linda H. Rosenberg, Ph.D.
Software Assurance Technology Office, Code 302
Goddard Space Flight Center, NASA
Greenbelt, MD 20771 USA
301-286-0087
Linda.Rosenberg@gsfc.nasa.gov

ABSTRACT
NASA is recognized as a leader in space technology, with cutting edge science probing galaxies never before seen by mankind. In keeping with this cutting edge technology, much of the functionality previously done through hardware has transferred to software, including mission critical functions. But technology implementation is moving so fast, that at times quality assurance cannot keep up, although we try. The NASA Independent Verification and Validation (IV&V) Facility was established in 1993 in West Virginia and tasked to provide the highest achievable levels of safety and cost effectiveness for mission-critical software. Despite this, NASA has experienced some failures recently that traced, in part, to less than adequate implementation of mission software. In response in part to these failures, NASA has taken steps to place an emphasis on implementing improvements in the Agency's software process. One of these steps relates to the increased focus on IV&V.

Every industry has data to support the statement that IV&V is cost effective, a very positive return on investment, yet most projects don't employ it. NASA decided to investigate the current use of IV&V, identify projects that should be using IV&V, and start using the expertise of the NASA IV&V Facility to improve the quality assurance on NASA projects through the appropriate implementation of IV&V. This paper is not about IV&V, but about NASA's new approach to the implementation of IV&V on all software development throughout the Agency.

Keywords
Independent Verification and Validation, IV&V, risk mitigation

1 INTRODUCTION

NASA determined the need for software Independent Verification and Validation (IV&V) after evaluating the causes of recent mission failures. These failures were due, in part, to software issues that should have been identified during development or testing. The NASA IV&V Facility in West Virginia was developed to be a Center of Excellence, but was under utilized. Projects that did use the Facility had proven benefits. The focus for improvement within NASA has turned to the application of Independent Verification and Validation of the software.

This approach started by looking at the resources available and projects that have applied IV&Vto determine if there were benefits in the NASA environment. It was quickly determined that only a few projects were implementing IV&V, not all were taking advantage of the Facility's expertise, and there were very definite proven benefits to NASA when IV&V was applied. These benefits ranged from cost savings to identifying mission critical errors not previously identified through testing; the application of IV&V on software resulted in increased safety and reliability of the mission. But deficiencies in the application of IV&V were also identified, specifically it was randomly applied by projects, with no consistency.

This paper will discuss the approach taken to increase the use of IV&V within NASA. We will start by defining independent, verification, and validation, what they mean. We will then discuss the policy written relative to the performance of software IV&V and the criteria developed to help projects quantify the need for IV&V. We will conclude with a discussion on the approach and how to convince the project manager to implement IV&V.

2 INDEPENDENT VERIFICATION AND VALIDATION (IV&V)

Although this is not a paper on IV&V, we do need to start with a basic understanding of what constitutes IV&V, starting first with the definitions of verification and validation, then determining what is required for "I" – independence. IEEE 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, defines verification and validation as follows.[1] Verification is defined as the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. Validation is defined as the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. Another way of stating verification is "Did we built the system right?" and validation as "Did we build the right system?" IV&V is implement at all phases of the software development life cycle, not just in testing.

Independence in IV&V is defined by IEEE as three parameters: technical independence, managerial independence, and financial independence. Technical independence is achieved by personnel who are not involved in the development of the software. IV&V personnel use their expertise to assess development processes and products independent of the developer. They formulate their own understanding of potential problems and how the proposed system is solving them. Managerial independence requires responsibility for the IV&V effort to be vested in an organization separate from the organization responsible for performing the implementation of the system. The IV&V effort independently selects the segments of the software and system to analyze and test, chooses the IV&V techniques, defines the schedule of IV&V activities, and selects the specific technical issues and problems to act upon. Most projects view V&V sufficient and do not recognize the added value the independence brings. Finance independence has been harder to attain. All work on the project, including quality assurance, is funded directly by the project, hence, IV&V is also funded directly by the project. In theory, the project could remove IV&V funding if they are not satisfied with the findings, but with the implementation of the IV&V policy, the projects are now required to work with the IV&V Facility to reach an agreement on the amount of IV&V and funding.

Any changes to this must be agreed to by the Center Director with strong justification by the project manager.

3 IMPLEMENTATION APPROACH

All software project managers, whether government or industry, never have time or money to spare, so when NASA identified the need for IV&V, it was met with a cry of "Not on my project!" But the implementation of IV&V was part of a larger effort to improve the software developed at NASA, to achieve the highest levels of safety and cost effectiveness possible for mission critical software. To accomplish this, NASA established a group of software experts from all NASA Centers to advise and develop software policies and standards.[1] Each Center provided two experts in different aspects of software development to form the NASA Software Working Group (SWG). Their charter is to advise the Agency on software related matters and recommend software management, engineering and assurance policies, standards, best practices and guidance, quite a task. They have been instrumental in the implementation of IV&V at NASA as will be demonstrated in this paper.

In 1993, NASA established the IV&V Facility in Fairmont West Virginia to provide the highest achievable levels of safety and cost-effectiveness for mission-critical software. The mission of the IV&V Facility is to become the NASA Center of Expertise for the application of software verification and validation technology to the development of high quality, highly reliable software systems to support NASA missions. The IV&V Facility provides tailored technical, project management and financial analyses for NASA projects, industry, and other Government agencies, by applying software engineering "best practices" to evaluate software risk and criticality assessments throughout the system development life cycle. Through the Facility, NASA has the means for implementing IV&V, but it is under utilized. Only a few very large NASA projects, such as Space Station, chose to apply IV&V through the Facility. A few other projects applied IV&V but through another source, usually a contractor.

NASA now has the ability to implement software improvement throughout the Agency through the Office of the Chief Engineer. It has the body of software experts to write policy through the Software Working Group. It has the experts in IV&V at the Facility. The infrastructure is in place, now the process had to begin.

The first step was to write a policy requiring projects to investigate the necessity of doing IV&V. IV&V is an effective risk mitigation strategy, and since most NASA missions are cutting edge technology, they also are at high risk. But NASA develops many types of projects, from ground and flight systems, to instruments and data collection systems. The specific types of missions potentially requiring IV&V had to be identified. Recognizing that cost must be balanced against potential benefits, the "amount" of risk incurred by a project had to be calculated. In order to require projects to investigate the application of IV&V, a policy was developed identifying the types of projects that must potentially apply IV&V was developed. The process of determining the need for IV&V, the extent and approach are

[1] The author is Goddard Space Flight Center's primary representative to this group.

specified in the policy. The policy also states all IV&V will be done under the management of the NASA IV&V Facility, centralizing expertise and ensuring consistency.

The next step was to develop criteria for determining when IV&V must be considered - quantifying project risk for an initial assessment on which projects may require IV&V. Project risk is defined as a combination of the probability that an undesirable event will occur, and the consequence if the event does occur. The IV&V Criteria were written using probability and consequence. Factors influencing software development were identified, and risk factors associated with them for a calculation of the probability. Consequences of failure were classified as Grave, Substantial, Marginal, Insignificant. These are combined for a determination of the necessity of IV&V.

The final step, and the hardest in some respects, was to identify the projects that potentially required IV&V and determine to what level IV&V was needed. Money is always an issue, there is never enough in any software development, so what was the cost and what are the balancing benefits.

Therefore, the approach to implement IV&V consistently and logically on all NASA software was broken into 3 steps:

1 – Write a policy for the requirement of IV&V implementation
2 – Write the criteria for an initial determination of IV&V necessity based on project risk evaluation
3 – Work with projects to implement IV&V

Step 1 – IV&V Implementation Policy

The policy for IV&V implementation was to clearly specify the process of determining when a program must apply IV&V under the management of the NASA IV&V Facility. The initial version of the IV&V policy was stated in the IV&V Facility Business Plan in June 2000.[2] The final version of the policy will be distributed as a NASA policy by the end of 2001, giving it the authority of being required for all NASA software development.

One strength of the policy is the specification that the NASA IV&V Facility is responsible for the management of all software IV&V efforts within the Agency. The Facility's role is to provide a value-added service to the Agency's software development efforts. The cost to perform tailoring and implementation of IV&V is expected to be borne by the project or the sponsoring HQ Enterprise. All results are to be reported to the project manager and the cognizant Center Director. This creates a central repository of knowledge, tools, metrics and lessons learned that can be used to improve the IV&V efforts on future projects throughout NASA.

The policy identifies all projects that this policy pertains to as those that are covered in NASA Policy Guideline (NPG) 7120.5 "NASA Program and Project Management Processes and Requirements Highlight Code", or other projects within NASA with significant software effort as determined by the NASA Chief Information Officer (CIO), the NASA Office of the Chief Engineer (OCE), and the NASA Office of Safety and Mission Assurance (Code Q) or

Center Safety and Mission Assurance. Projects covered in NPG 7120.5 include "all programs/projects that provide aerospace products or capabilities, i.e., provide space and aeronautics, flight and ground systems, technologies, and operations."[3] This covers most of the projects but may exclude projects where NASA is a minor partner and those developed by Universities for educational purposes and considered at minor risk due to limited investment.

The policy states that each project must produce, document, and implement a plan that addresses the performance of V&V, and if appropriate, IV&V, over the life cycle of the software, from requirements through delivery and maintenance. The level of IV&V of software that is performed is based on the cost, size, complexity, life span, risk, and consequences of failure as defined using the Criteria (explained in the following section).

The policy specifies the following steps for IV&V determination and application:

a. The project manager is to evaluate their project against the criteria (described in Step 2) to determine if IV&V is indicated.
b. For projects where the criteria indicates software IV&V is warranted, the project manager is to discuss the results with a representative of the NASA IV&V Facility. Application of the IV&V criteria simply determines if a project is a candidate for IV&V – not the level of IV&V nor the resources associated with the IV&V. The Facility personnel will work jointly with the project office to provide recommendations tailored to the project on what sections and to what extent IV&V should be performed.
c. With this input from the Facility, the project manager will document in the project plan what IV&V is intended to be performed. Since IV&V compliments and enhances risk mitigation, projects are encouraged to achieve the most effective balance of risk mitigation strategies.
d. The level of IV&V activities selected in the project plan is subject to Facility review. The Center Director is responsible for resolving differences between the Facility and Project Office.
e. When IV&V is selected for a project, the Facility, with information from the project, will make the determination of which IV&V work will be done at the Facility location in WV, and which work will be done local to the project. The objective over the life of the project is that at least 50% of the work be done at the Facility.
f. When the project undergoes significant changes that impact the software subsystems, the project manager must revisit the criteria.

Step 2 - IV&V Criteria

In order to accomplish of goal of increasing the application of IV&V on all appropriate projects, it had to be determined what was meant by "appropriate" projects, hence the development of IV&V criteria. Again the members of the NASA Software Working Group were task with defining when IV&V should be applied to a project. Looking back on the objective of IV&V for risk mitigation, those projects with high risk had to be identified, but first, the criteria for what makes a software project high risk had to be defined and quantified. The quantification was the hardest part.

IV&V is intended to assist mitigating risk, hence, the decision to do IV&V must be risk based. NASA policy NPD 7120.5 defines risk as the "combination of 1) the probability (qualitative or quantitative) that a program or project will experience an undesired event such as cost overrun, schedule slippage, safety mishap, or failure to achieve a needed breakthrough; and 2) the consequences, impact, or severity of the undesired event were it to occur." [3] The exact probability of occurrence and consequences of a given software failure cannot be calculated early in the software lifecycle. However, there are realistically available metrics which give good general approximations of the consequences as well as the likelihood of failures.

Probability Evaluation

The probability of failure for software is difficult to determine at any phase in the software development life cycle. The NASA Software Working Group (SWG) has identified factors that impact the difficulty of the development. These factors were then calibrated to determine the extent of risk for successful software development. While the indicators are not precise and are currently in Beta testing by NASA software development projects and the IV&V determination team, they do provide order of magnitude estimates, which are adequate for assessing the need for IV&V. These factors are described below.

a. Software team complexity – Industry research has shown that the larger the team the more complex the communication and more points of failure. Smaller teams are generally co-located, making communication a common occurrence. The larger the development teams, the more formal communication, often loosing some subtle communication activities.
b. Contractor support – NASA software is developed by civil servants, contractors or a combination. The presence of a contractor (other than the prime system developer) introduces a layer of contract management between the government and the actual software developer.
c. Organization complexity – This is an indirect measure of communications challenges inherit in the software developer. A single organization working from multiple locations faces a slightly greater challenge than an organization in one location. When the software development is accomplished by multiple organizations working for a single integrator, the development is significantly complicated. If the developing organizations are coequal such as in an associate contractor relationship (or a similar relationship between government entities) then there is no integrator. Experience has shown this arrangement to be extremely challenging as no one is in charge.
d. Schedule pressure - A deadline is negotiable if changing the deadline is possible although it may result in slightly increased cost, schedule delays, or negative publicity. A deadline is non-negotiable if it is driven by immovable event such as an upcoming launch window. Project working under deadlines are more likely to decrease testing, hence potentially decreasing the reliability and safety, and increasing the risk.
e. Process maturity of software provider – It has been demonstrated in almost all cases, application of the Software Engineering Institute's Capability Maturity Model (CMM) increases the quality of the software developed, resulting in higher reliability. This model has 5 levels of attainment, 3 being the most common, and all projects start out at Level 1. While striving to attain Level 2, projects are implementing some of the

techniques that reduce project risk. The higher the CMM Level achieved, the lower the risks. A formal assessment is required to determine the CMM Level.

f. Degree of innovation – Innovation is usually found in most of NASA's software, as expected. But when attempting to formulate a new equation or concept, there is a higher risk than if the software has been proven in previous programs.
g. Level of integration – Programs that stand alone and are not integrated with any other components have a higher probability of success due to simplicity of their interfaces. The more extensive integration of multiple components, the higher the risk.
h. Requirement maturity – Requirements that are constantly change or change late, ambiguous or untestable are at a higher risk than those requirements whose objectives are well defined with few or no unknowns.
i. Software Lines of Code – It has been proven that the larger the project, the higher the risks. Lines of code is used as a size measurement and is defined to be all code in the programs, including software that is reused and auto generated code.

Five categories of risk within each factor were identified based on input from software developers from all NASA centers. Values 1, 2, 4, 8, 16 were assigned to each category.

Factors contributing to probability of software failure	**Un-weighted probability of failure score**					**Weighting Factor**	**Likely-hood of failure rating**
	1	2	4	8	16		
Software team complexity	Up to 5 people at one location	Up to 10 people at one location	Up to 20 people at one location or 10 people with external support	Up to 50 people at one location or 20 people with external support	More than 50 people at one location or 20 people with external support	X2	
Contractor Support	None	Contractor with minor tasks		Contractor with major tasks	Contractor with major tasks critical to project success	X2	
Organization Complexity*	One location	Two locations but same reporting chain	Multiple locations but same reporting chain	Multiple providers with prime sub relationship	Multiple providers with associate relationship	X1	
Schedule Pressure**	No deadline		Deadline is negotiable		Non-negotiable deadline	X2	
Process Maturity of Software Provider	Independent assessment of Capability Maturity Model (CMM) Level 4, 5	Independent assessment of CMM Level 3	Independent assessment of CMM Level 2	CMM Level 1 with record of repeated mission success	CMM Level 1 or equivalent	X2	
Degree of Innovation	Proven and accepted		Proven but new to the development organization		Cutting edge	X1	
Level of Integration	Simple - Stand alone				Extensive Integration Required	X2	
Requirement Maturity	Well defined objectives - No unknowns	Well defined objectives - Few unknowns		Preliminary objectives	Changing, ambiguous, or untestable objectives	X2	
Software Lines of Code***	Less than 50K		Over 500K		Over 1000K	X2	
Total							

Table 1 Likelihood of Failures Based on Software Environment

Finally, a weighting factor of 1 or 2 was identified for each factor. This information is shown in Table 1.

To apply the criteria, the project manager identifies the category of risk for each factor and multiples the appropriate value (1, 2, 4, 8, or 16) times the weighting factor of 1 or 2. The sum for all factors yields an initial numerical representation of the project software development risk. For example, a project might rate the following:

- Software team complexity – "up to 20 people at one location" = 4 * 2 = 8
- Contractor support – "with minor tasks" = 2 * 2 = 4
- Organizational complexity – "two locations but with same reporting chain" = 2 * 1 = 2
- Schedule pressure – "non-negotiable" = 16 * 2 = 32
- Process maturity – " CMM Level 1 but with a successful history" = 8 * 2 = 16
- Innovation – between proven but new and cutting edge = 8 * 1 = 8
- Integration – almost stand alone = 2 * 2 = 4
- Requirement maturity – "preliminary objectives" = 8 * 2 = 16
- Lines of code = ~ 300K = 2 * 2 = 4

$\therefore$ TOTAL = 8+4+2+32+16+8+4 +16+4= 94

This risk information must now be combined with the consequence evaluation.

Consequence Evaluation

In general, the consequences of a software failure can be derived from the purpose of the software: i.e., what does the software control; what do we depend on it to do. NASA has many types of software, including flight software which is launched and contains mission critical functional, ground system that sends commands, scientific software for the experiments, and just about all other types of software imaginable. Factors which can be used to categorize software based on its intended function as well as the level of effort expended to produce the software are defined as follows:

a. Potential for loss of life - Is the software the primary means of controlling or monitoring systems that have the potential to cause the death of an operator, crewmember, support personnel, or bystander? The presence of manual overrides and failsafe devices are not to be considered. This is considered a binary rating: responses must be either yes or no.
b. Potential for serious injury - Serious injury is defined as loss of digit, limb, or sight in one or both eyes, sudden loss of hearing, or exposure to substance or radiation that could result in long term illness. This rating is also binary. This rating considers only those cases where the software is the primary mechanism for controlling or monitoring the system. The presence of manual overrides and failsafe devices are not to be considered.
c. Potential for catastrophic mission failure - Can a problem in the software result in a catastrophic failure of the mission? This is a binary rating.
d. Potential for partial mission failure - Can a problem in the software result in a failure to meet some of the overall mission objectives? This is a binary rating.
e. Potential for loss of equipment - This is a measure of the cost (in dollars) of physical resources that are placed at risk due to a software failure. Potential collateral damage is to be included. This is exclusive of mission failure.

f. Potential for waste of software resource investment- -This is a measure or projection of the effort (in work-years, civil service, contractor, etc.) invested in the software. This shows the level of effort that could potentially be wasted if the software doesn't meet requirements.
g. Potential for adverse visibility - This is a measure of the potential for negative political and public image impacts stemming from a failure of the system as a result of software failure. The unit of measure is the geographical or political level at which the failure will be common knowledge—specifically: local (Center), Agency, national, international. The potential for adverse visibility is evaluated based on the history of similar efforts.
h. Potential effect on routine operations - This is a measure of the potential to interrupt business. There are two major components of this rating factor: scope and impact. Scope refers to who is affected. The choices are Center and Agency. The choices for impact are inconvenience and work stoppage.

Now the Potential for failure had to be quantified. Four ratings were chosen: Grave, Substantial, Marginal and Insignificant. Each of the factors above were quantified for each rating. If *any* of the conditions are met, the software is considered to reside in that category. The categories are defined as follows:

- *Grave*
  Potential for loss of life - Yes
  Potential for loss of equipment – Greater than $100,000,000
  Potential for waste of resource investment – Greater than 200 work-years on software
  Potential for adverse visibility - International
- *Substantial*
  Potential for serious injury – Yes
  Potential for catastrophic mission failure – Yes
  Potential for loss of equipment – Greater than $20,000,000
  Potential for waste of resource investment – Greater than 100 work-years on software
  Potential for adverse visibility - National
  Potential effect on routine operations – Agency work stoppage
- *Marginal*
  Potential for partial mission failure - Yes
  Potential for loss of equipment – Greater than $2,000,000
  Potential for waste of resource investment – Greater than 20 work-years on software
  Potential for adverse visibility - Agency
  Potential effect on routine operations – Center work stoppage or Agency inconvenience
- *Insignificant*
  Potential for loss of life - No
  Potential for serious injury – No
  Potential for catastrophic mission failure – No
  Potential for partial mission failure – No
  Potential for loss of equipment – Less than $2,000,000
  Potential for waste of resource investment – Less than 20 work-years on software
  Potential for adverse visibility – No more than local visibility
  Potential effect on routine operations – No more than a Center inconvenience

For example, the project described below would determine the consequence as demonstrated.

- Software controls life-support systems (potential loss of life)
- There is no potential for loss of equipment
- There is potential for adverse visibility at the Agency level
- There would be no effect on routine operations

∴ This software would rate the consequence of failure as "GRAVE" because it met one of the conditions – loss of life support. Software consequence is determined by meeting any of the conditions within that category.

Combining the software the probability of failure rating and the consequences of failure yields a risk assessment, which can be used to identify the need for IV&V Application of these criteria only determines that a project is a candidate for IV&V – not the level of IV&V nor the resources associated with the IV&V effort. These must be determined as a result of discussions between the project and the IV&V Facility.

Figure 1 shows a dark region of high risk where software consequences, likelihood of failure, or both are high. Projects having software that falls into this high-risk area shall undergo IV&V. The exception is those projects which have already done hardware/software integration. An IV&V would not be productive that late in the development cycle. The gray regions of intermediate risk. Projects having software that falls into these areas shall undergo an evaluation to determine if IV&V is warranted.

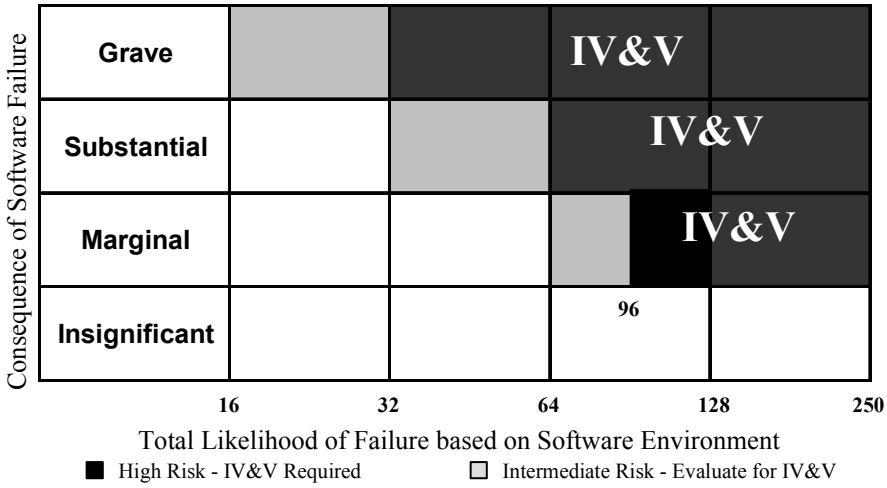


Figure 1: Probability & Consequence = Risk

Using the previous project example, a probability of 94 score means that they must perform IV&V if the consequence of failure is Substantial or Grave. Since the project's consequence was determined to be Grave, they must perform IV&V under the NASA IV&V Facility.

Step 3 - Project implementation
Prior to implementing IV&V, a company has to know about the software it develops. At NASA, software development is done at all 10 Centers spread from California, to Florida, Texas to Ohio, and Alabama to Maryland, and with Universities and industries. NASA projects are initially funded through multiple Enterprises at Headquarters. Each Enterprise was requested to compile a list of their projects. Since all projects have a software component, some very small and some mission critical, this list was used as a starting point for investigating the application of IV&V.

Approximately 100 projects were identified across all NASA Centers. Project managers were sent a request from NASA's Chief Engineer to apply the criteria to determine if their project was a candidate for IV&V. If they fit the category of potentially requiring IV&V, they were then ask to apply the criteria, estimating the probability of failure and the consequence. This information was sent to the NASA IV&V Facility where personnel evaluated the results and looked for inconsistencies in the data. For example, a project that indicated they were at a CMM Level 5, the contractor was identified, since very few companies are at Level 5. If a project marked their deadline was negotiable, the innovation was proven and accepted and the requirement maturity was well defined, with no unknowns, indicating a project doing nothing new and in no hurry to get there. This is not usually the case for NASA software. In many cases, the person completing the information did not understand a factor, such as what CMM is. In a few cases, the project chose the lowest possible risk to prevent the need for IV&V. (All data was scanned to identify those projects who checked the least risk category for each factor.)

Projects were given one week to complete the criteria information. Projects not completing the data were sent a second letter from NASA Headquarters with notification to appropriate personnel at the Center. Center Directors were eventually notified of the projects that did not respond (only two), and the data was promptly received. In total, data was received on approximately 100 projects. After discussions with projects to clarify the information and correct erroneous data, approximately seventy projects were identified as candidates for IV&V.

Applying IV&V to seventy projects immediately however, is an impossibility. The Facility currently does not have the resources to accommodate this many projects and this much work at this time. Using the data from the criteria (consequence and probability) and discussions with the project managers, the following guidelines to focus the IV&V efforts were implemented.

1. All project currently receiving IV&V will continue.
2. All projects classified as "GRAVE" should be addressed first for IV&V with the highest priority applied to those closest to operational date as a general rule, with some attention applied to why it is classified as "GRAVE".
3. All projects classified as "SUBSTANTIAL" and needing IV&V based on the risk probability value greater than 32 that are in the requirements or design phase.
4. All remaining projects classified as "SUBSTANTIAL" and needing IV&V based on the risk probability value greater than 32.

5. All remaining projects classified as “MARGINAL” and needing IV&V based on the risk probability value greater than 96, prioritized based upon how close to starting they are.

4 RESULTS

The application of IV&V on NASA projects has shown some very positive results and prevented costly errors. In one project, the IV&V activity identified design flaws in the Command and Control system which had it not been corrected, would have resulted in a catastrophic hazard. This critical piece of software sends commands to hardware elements, and if not working properly, could fail to send emergency response commands leading to the loss of attitude control, rapid depressurization, and other hazardous conditions. Using a code analysis, IV&V identified an error which would eliminate the vital command link between the ground control system and the satellite. A special software patch was generated for the on-orbit software to correct the problem. IV&V developed policy criteria used by one Program for non-flight software in integrated tests. This policy was key for insuring testing integrity while making it possible to keep tight development schedules.

IV&V activities have benefited many different domains of NASA’s software. In the Manned Space flight domain, over 4,000 problems were identified, 10 of the highest criticality, those that could result in loss of mission or loss of life. For Experimental Flight Vehicles, IV&V identified over 300 requirements and design problems. For Ground systems, over 250 legacy system requirements and mitigation problems were identified.

5 CONCLUSION

The results of NASA’s investigation have shown that in this environment, IV&V has been cost effective. Companies large and small, in today’s competitive world cannot afford software that is unreliable. To be effective however, IV&V must be applied effectively and the independence must not be lost. Project managers must understand the benefits above the cost, looking at the whole development, not just the current state.

NASA has recognized the value of IV&V and has taken steps to implement IV&V on all software projects where warranted. The decision to implement IV&V is no longer solely the decision of the project manager, but through an independent evaluation, the risk of the project is evaluated, and the need for IV&V is determined. Although the policy and criteria in this paper were written for NASA projects, they are applicable with minor modification to any software development.

ACKNOWLEDGEMENTS

Much of the work presented in this paper was a combined effort of NASA IV&V Facility personnel (Judy Bruner, John Hinkle, and Ken McGill), Goddard Space Flight Center personnel (Charles Vanek, John Dalton, Linda Rosenberg), and the members of the Software Working Group under Pat Schuller, Langley Research Center.

REFERENCES

[1] IEEE 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology , Institute of Electrical and Electronics Engineers, Inc.

[2] *NASA IV&V Business Plan*, Office of System Safety and Mission Assurance, NASA Goddard Space Flight Center, MD , June 2000, http:// ivvplan.gsfc.nasa.gov.

[3] *NASA Policy Guideline (NPG) 7120.5 "NASA Program and Project Management Processes and Requirements Highlight Code",* Office of Chief Engineer, NASA Headquarters, Washington, DC.

QWE2002 Paper K1-2

Mr. Robert A. Sabourin
(Amibug.com)

Creating Quality Web Systems

Key Points

- Developing Quality Web Systems depends on people.
- People issues can make or break a Web System.
- Requirement turbulence is part of life so be prepared!
- We are in the change business!

Presentation Abstract

This keynote reviews some important challenges of building a quality web system.

"It's all about people - and the occasional bug!". Technology churn and high frequency requirement turbulence combine with "aggressive" time to market demands cause us to challenge traditional methods and technique. Sometimes we pay a heavy price in staff turn-over, burn-out and ultimately Quality. It is important to ensure that team-members have access to the information and technologies they need - when they need it! It is important to ensure that basic testing fundamentals are well understood by all involved in the process! What ever happened to the course Software Testing 101?

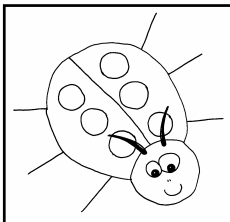
The author draws on his experiences in testing Web and e-Commerce sites as well as on Canada's massive "Government On-Line" initiative.

About the Author

Robert Sabourin is President of AmiBug.Com, Inc, a training and management consulting firm specialized in implementing "light effective process" to achieve excellence in delivering on-time, on-quality, on-budget commercial software solutions.

Robert Sabourin has been involved in all aspects of development, testing and management of software engineering projects. (He wrote his first program in 1972 & released his first shrink wrapped product in 1982) Robert is an accomplished software engineering and SQA management expert and evangelist (don't tell me it can't de done!). AmiBug.Com services include training, management consulting and professional development.

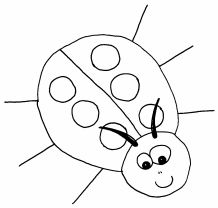
In additional to his professional activities Robert is an Adjunct Professor of Software Engineering at McGill University in Montreal Canada. Robert is the author of the popular "I am a Bug!" - a book designed to teach children of SQA professional what their parents really do a work!



Creating Quality Web Systems

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

 © Robert Sabourin, 2001

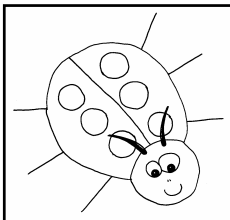


Creating Quality Web Systems

Overview:

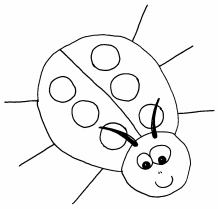
- Introduction
- Fundamental Question
- Defintions and some Wisdom from the ages
- Reality – Like a Big Ship
- Government On-Line
- People, training getting back to basics

 © Robert Sabourin, 2001



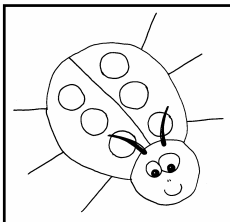
Just In Time Testing
Testing Turbulent Web Based Applications

- 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

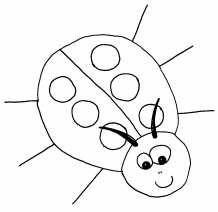
Robert & Catherine Sabourin

ISBN: 0-9685774-0-7

www.amazon.com
www.fatbrain.com

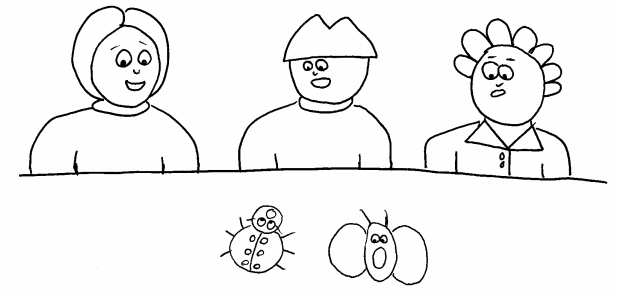
In the style of a children's book. Explains elements of software development process in a fun easy to read format.

© Robert Sabourin, 2001

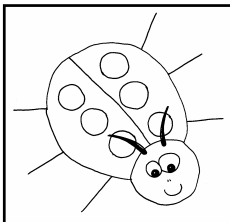


Finished?

- How do you know you are finished?

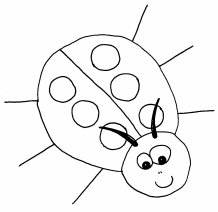


© Robert Sabourin, 2001

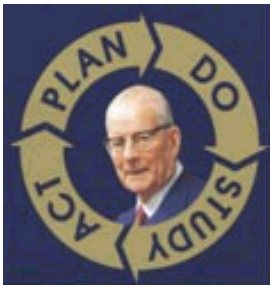


Crosby on Quality

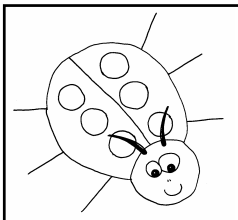
- "Quality is defined as conformance to requirements"
- "Quality is not a measure of GOODNESS"
  - Phil B. Crosby, *Quality is Free*



Deming Quality approach (PDCA)

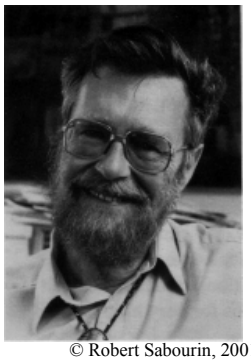


- Plan, Do Check, and Act:
  - ☑Plan what you want to implement.
  - ☑Do the pilot implementation.
  - ☑Check the results of the pilot.
  - ☑Act on the results by tweaking the process before the next project.



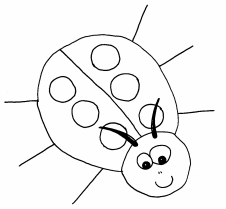
Edsger W. Dijkstra

- "Program testing can be used to show the presence of bugs, but never to show their absence"



© Robert Sabourin, 2001

AmiBug.Com, Inc.

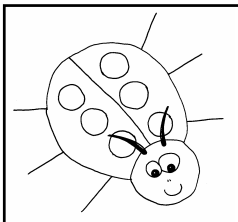


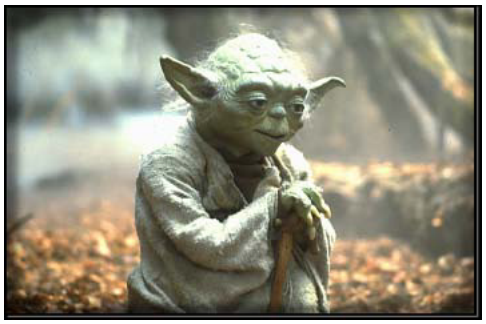
Ken Blanchard

- "Feedback is the breakfast of champions!"

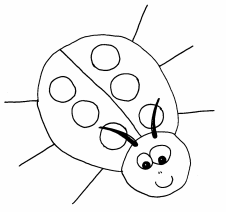
© Robert Sabourin, 2001

AmiBug.Com, Inc.





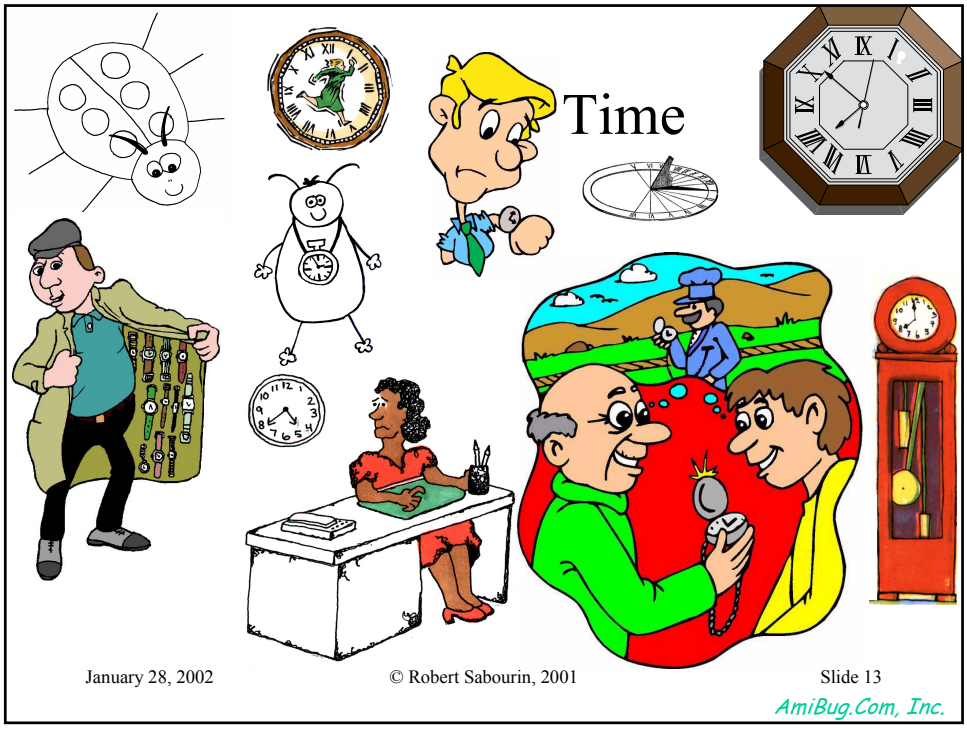
"No! Try not, Do. Or do not. There is no try."

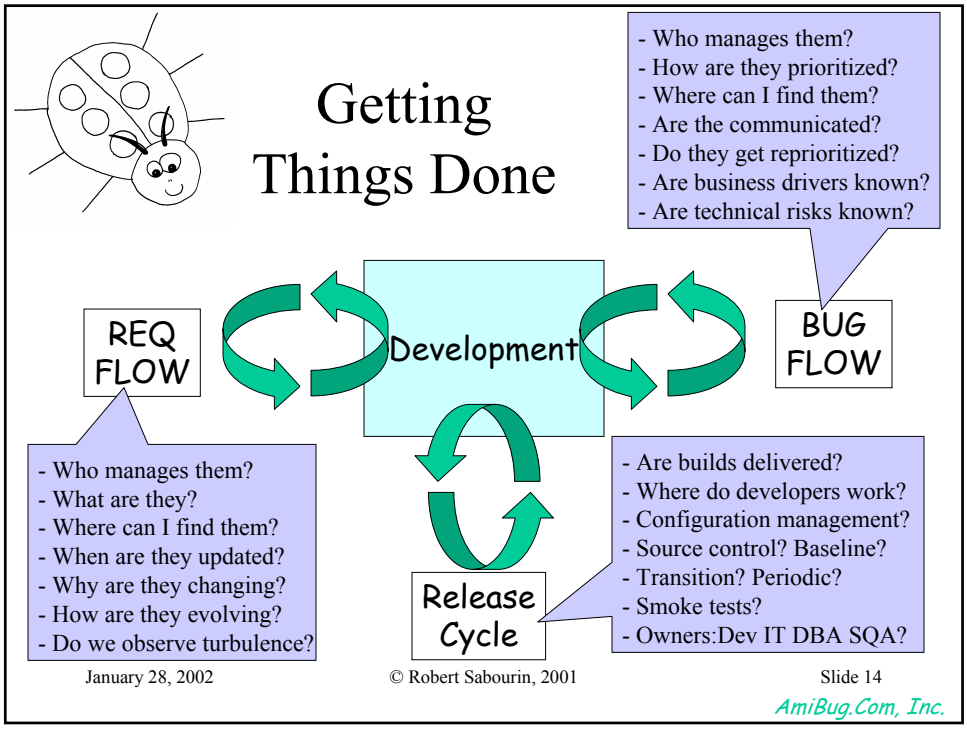


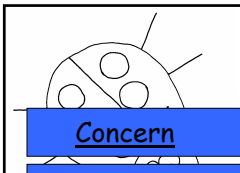
Philosophy

We have precious little time to run tests!

We must always be prepared!

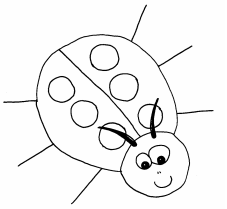

Time
January 28, 2002
© Robert Sabourin, 2001
Slide 13
AmiBug.Com, Inc.


Getting
Things Done
- Who manages them?
- How are they prioritized?
- Where can I find them?
- Are the communicated?
- Do they get reprioritized?
- Are business drivers known?
- Are technical risks known?
REQ FLOW
Development
BUG FLOW
- Who manages them?
- What are they?
- Where can I find them?
- When are they updated?
- Why are they changing?
- How are they evolving?
- Do we observe turbulence?
Release Cycle
- Are builds delivered?
- Where do developers work?
- Configuration management?
- Source control? Baseline?
- Transition? Periodic?
- Smoke tests?
- Owners:Dev IT DBA SQA?
January 28, 2002
© Robert Sabourin, 2001
Slide 14
AmiBug.Com, Inc.



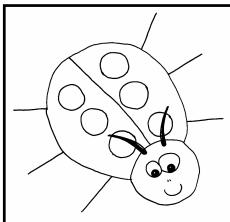
Getting Things Done

Concern	Being Prepared!
- Information Flow	Corporate information Key business drivers Sales Market Finance
- Technology Flow	Architecture Technology churn Tools Techniques Training
- Requirement Flow	Defined Understood Interrupt Poll Prioritize Turbulence Status *Truffle*
- Bug Flow	Defined Understood Business Technical Efficient Expedient Reassess
- Test Objectives	Quality Factors Technical Risk Failure Modes Importance
- Test Strategy	Plan Analytic Exploratory Checklists Parallel Chunking Scenarios Data
- Test Organization	Scheduling Staffing Outsourcing Contractors Students
- Testing Lab	Multi-tier Server Client Platforms Swap Pattern Synchronized
- Test Status	Bug charts Test Plan Elaboration Status Pass Fail Execution Status

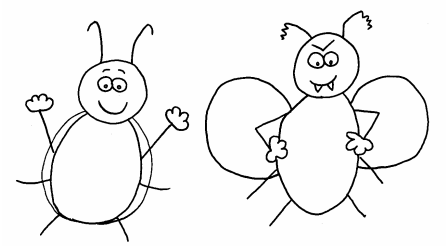


Spreading Effort Across Test Objectives

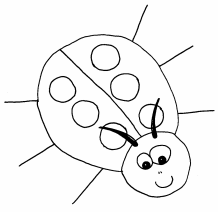
- *Reality of all testing activities*
- *Amplified on Web Applications*
- *How do you cover risks?*



About Bugs

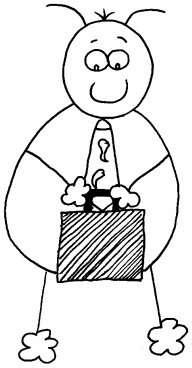


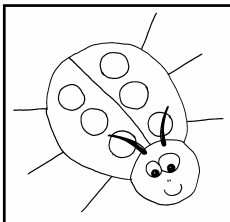
Bugs are not Good or Bad



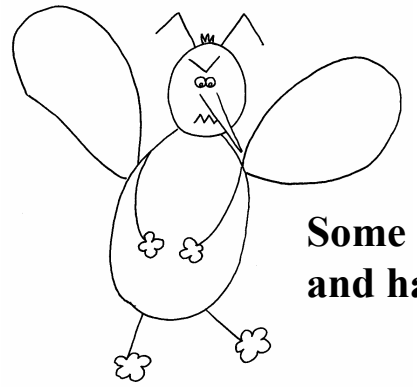
About Bugs

Some bugs are important and have a high priority!

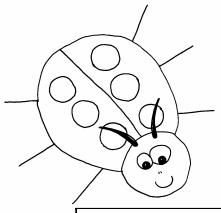




About Bugs

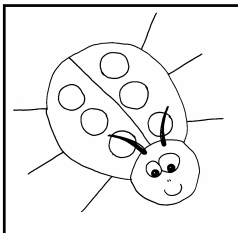


Some bugs are dangerous and have a high severity!



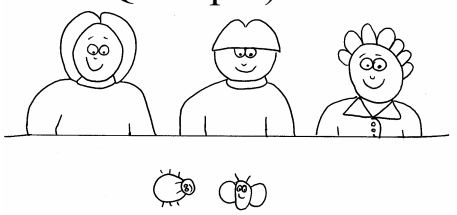
Bug Quadrants

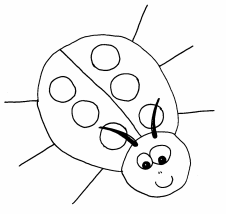
Urgent Severe	Urgent Not Severe
Not Urgent Severe	Not Urgent Not Severe



You know you are finished when *(a perspective)…*

- … the only bugs left are the ones that Project Stakeholders agree are acceptable (based on objective SQA input) ...

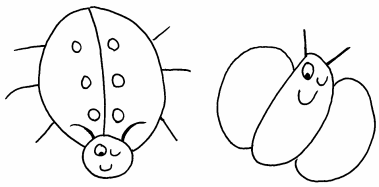


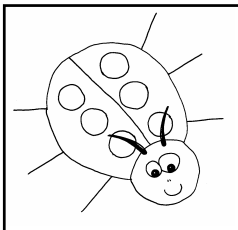


You know you are finished when …

- … the only bugs left are the ones that Project Stakeholders agree are acceptable (based on objective SQA input) …

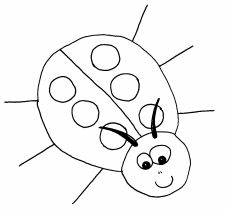
At least for now!





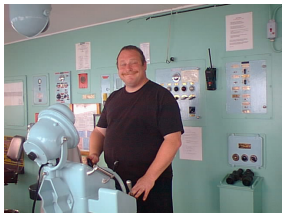
The Ferbec
Allegoric Web Quality

AmiBug.Com, Inc.

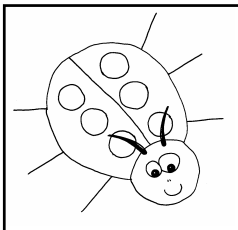


Does it scale?

…

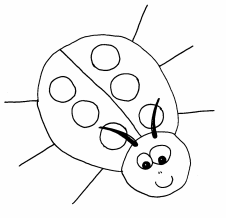


AmiBug.Com, Inc.



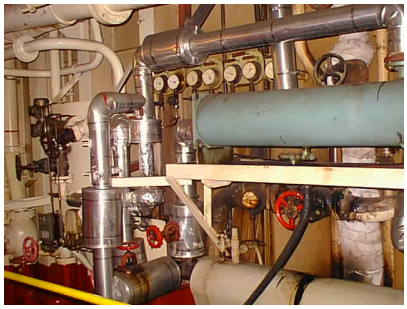
Power – Scale!

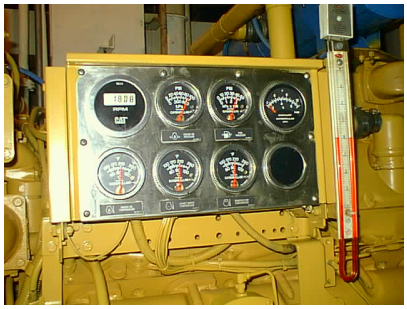
…

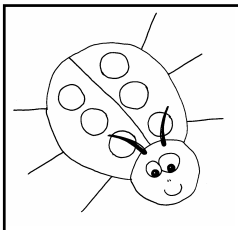


Measures Metrics Innovation

…

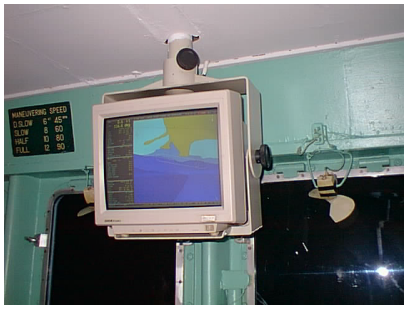


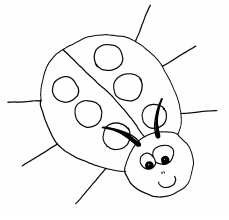




Blend of technologies

…

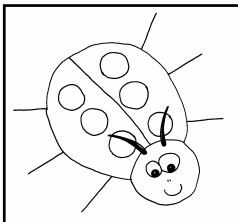




Government On-Line

Canada

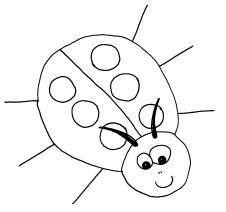
- *"By 2004, Canadians will be able to request and receive all key federal services through secure, interactive and timely on-line transactions."*
- `www.gol-ged.gc.cap`



Government On-Line

Training

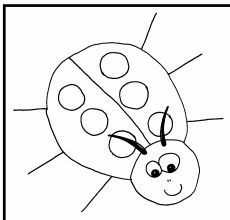
- Observed Q3+Q4-2001 and Q1-2002
  - High percentage of students are experienced testers moving to GOL applications from Client Server or Mainframe applications
  - High percentage have never used basic testing techniques such as:
    - Equivalence Partitioning
    - Boundary Analysis
    - State Modeling
    - 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

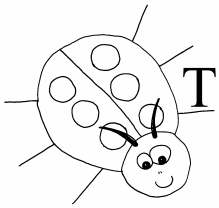
- Get back to basics!
  - Testing 101 courses and books all over the place
  - Practical books
  - Awareness of the many wonderful methods and techniques available in an objective and open minded approach
  - Encourage a foundation in fundamentals before overly skill based approaches



Creating Quality Web Systems

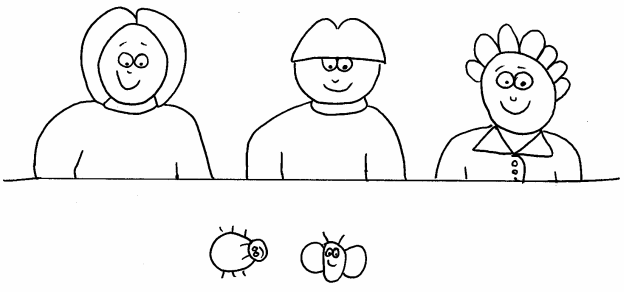
Conclusion:

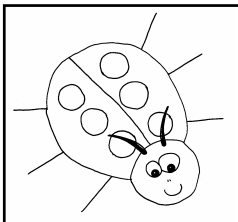
- Empower the team with relevant knowledge of business drivers and technical risks
- Focus on Fundamental Questions
- We are in the change business, things will change, it's a fact!
- Tools and technologies evolve
  - Ensure teams are well grounded in fundamentals
  - Ensure teams have a Swiss army knife of methods and techniques
  - Encourage flexibility, adapt, be *context sensitive*
  - Different strokes for different folks apply situational techniques



Testing Web and E-Commerce Applications

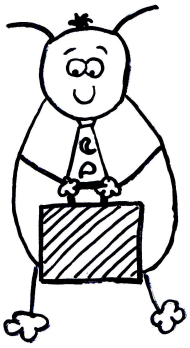
- It's all about people! (and the occasional bug too)

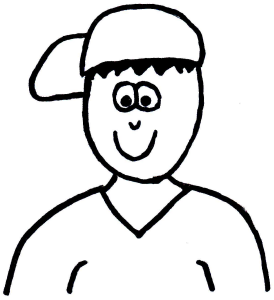


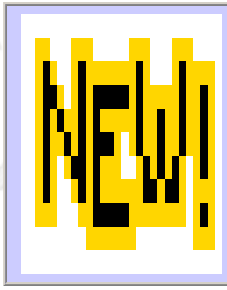


Thank You

- Questions?







QWE2002 Paper K2-1

Prof. Koenraad Debackere
(KUL)

Organizing for High Tech Innovation

Key Points

- To be supplied.

Presentation Abstract

To be supplied.

About the Author

To be supplied.

QWE2002 Paper K2-2

Mr. Bob Bartlett
(SIM Group)

Power Testing

Key Points

- Testing is the most powerful stage of the development process.
- Most development & implementation risks can be managed through defined & scheduled activities.
- The strategic use of testing tools can double the power to testing.

Presentation Abstract

Not only does testing have an important part to play throughout the entire development and implementation process, the testing activities can be used to manage most of the project and business risks that exist with software development and implementation. This paper will describe and illustrate how overall risk management and project control is managed with the strategic planning of testing activities.

A compelling business case will be put forward for the early involvement of testers in projects as well as the integration of the testing process. This will be accomplished by extensive use of examples from projects where this was done; many of these will come from e-commerce and e-business implementation projects. All types and phases of testing will be used in the examples from the many projects the author has worked on.

About the Author

Mr. Bob Bartlett has over 30 years in software, and using automated testing techniques throughout. He is an Executive Director and a Chairman of Software testing specialist company today. He is a member of the CSSA executive council. Also, he has designed, developed and sold automated testing tools.

PowerTesting

Bob Bartlett
bob@simgroup.com

© 2001, SIM Group Ltd.

Introducing

Power Testing

Using the Power of Testing to make a difference

© 2001, SIM Group Ltd.

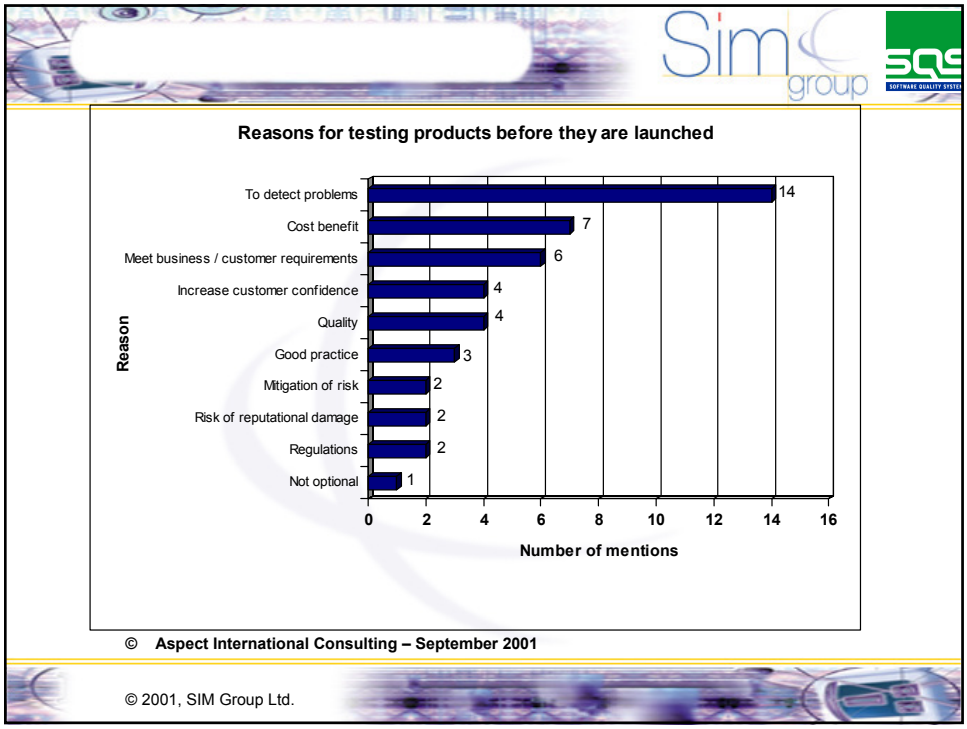
Agenda

- Some Interesting Information
- The business case for Strategically applying testing practices (Power Testing!)
- Cost Reduction
- Risk Management
- Control (Test Management)
- Conclusion

© 2001, SIM Group Ltd.

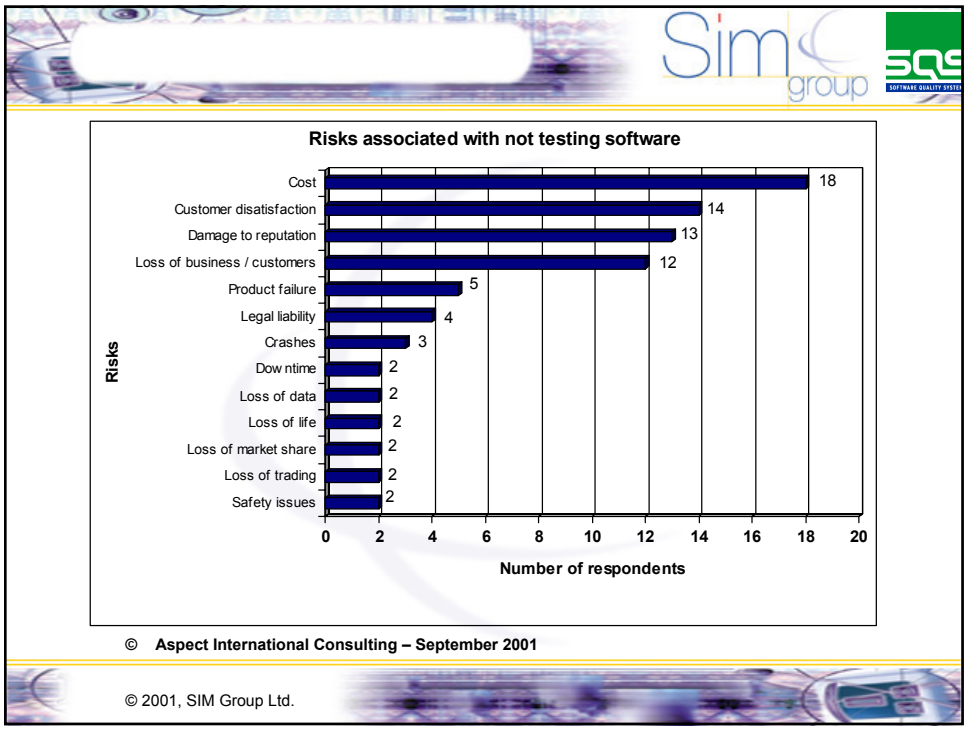
Research commissioned by the SIM Group and conducted by independent research company Aspect Consulting.

© 2001, SIM Group Ltd.



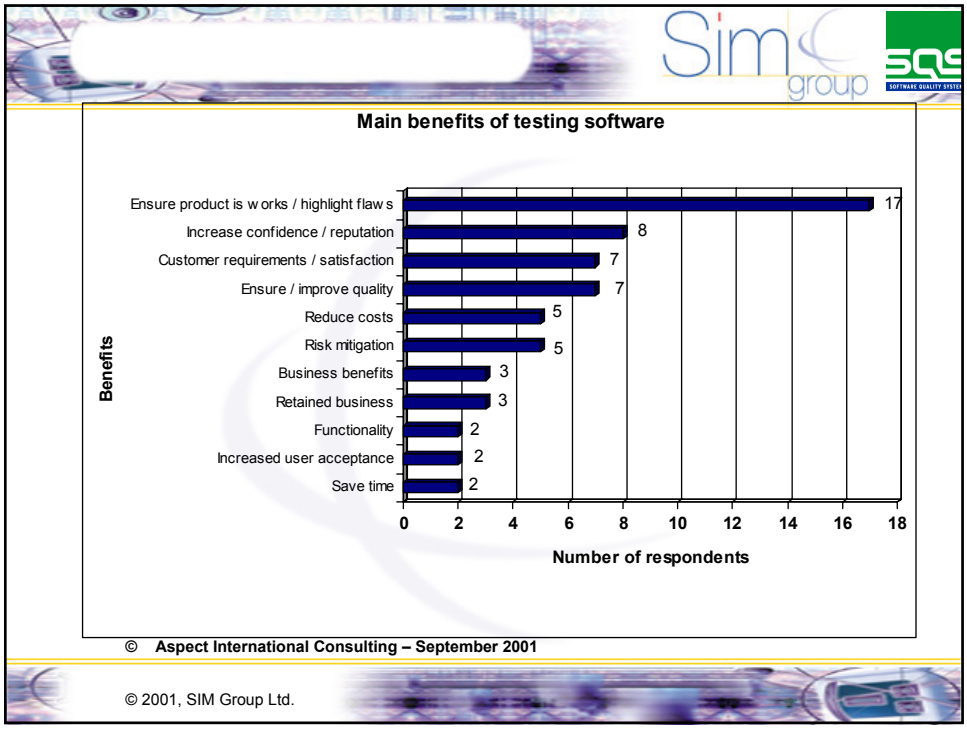
© Aspect International Consulting – September 2001

© 2001, SIM Group Ltd.



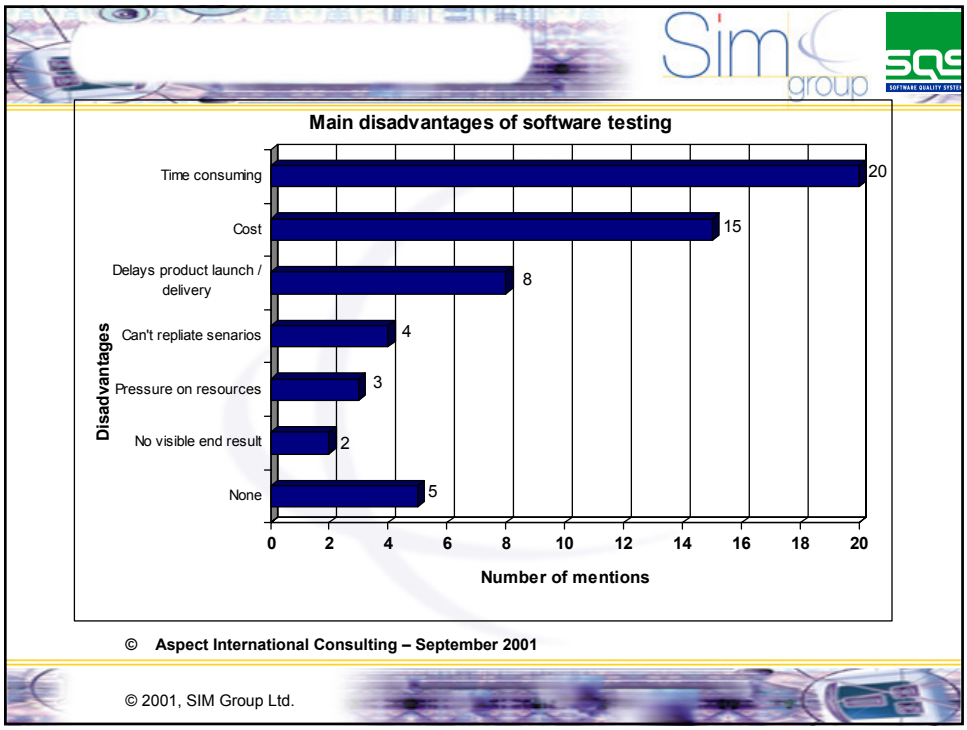
© Aspect International Consulting – September 2001

© 2001, SIM Group Ltd.



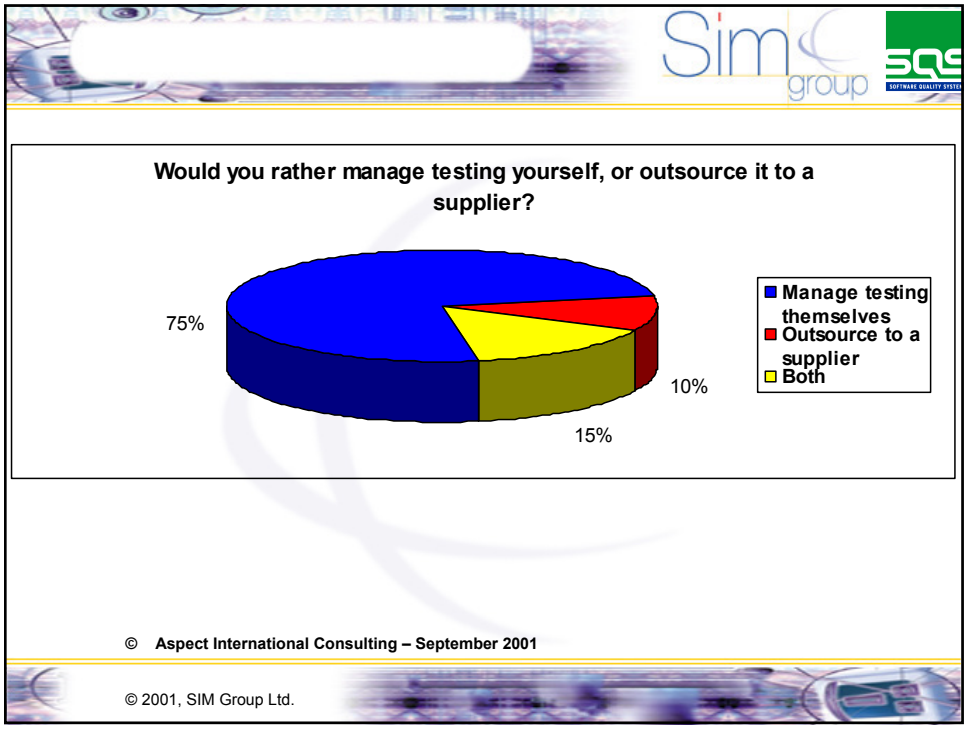
© Aspect International Consulting – September 2001

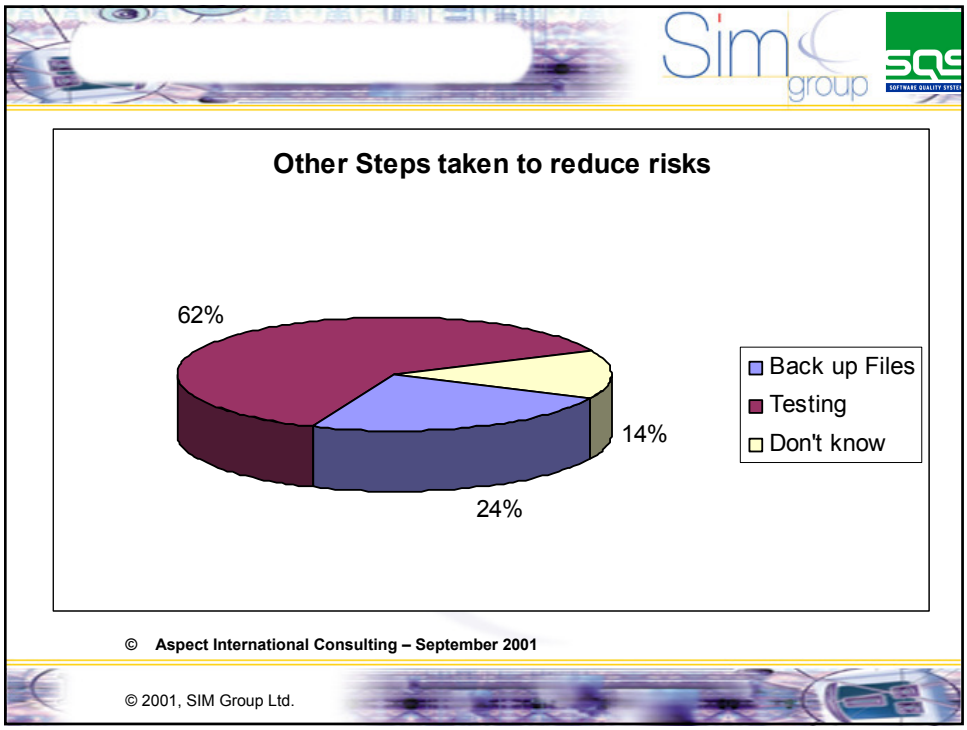
© 2001, SIM Group Ltd.



© Aspect International Consulting – September 2001

© 2001, SIM Group Ltd.


Would you rather manage testing yourself, or outsource it to a supplier?
75%
10%
15%
Manage testing themselves
Outsource to a supplier
Both
© Aspect International Consulting – September 2001
© 2001, SIM Group Ltd.


Other Steps taken to reduce risks
62%
14%
24%
Back up Files
Testing
Don't know
© Aspect International Consulting – September 2001
© 2001, SIM Group Ltd.

What does this mean?

- People think testing is a defensive activity that does not add value, but can make things more expensive and longer.
- People do no think of testing as a powerful way to positively contribute to success.
- We have failed to make the “business case” for good testing practice.
- We are loosing the argument for why good practices in testing are important, desirable and powerful.

© 2001, SIM Group Ltd.

The business case for PowerTesting

- Cost
  - Avoidance
  - Reduction
- Time to market or release
- Risk Reduction
  - Customers
  - Business and reputation
  - Quality
- Maintain control over testing activities

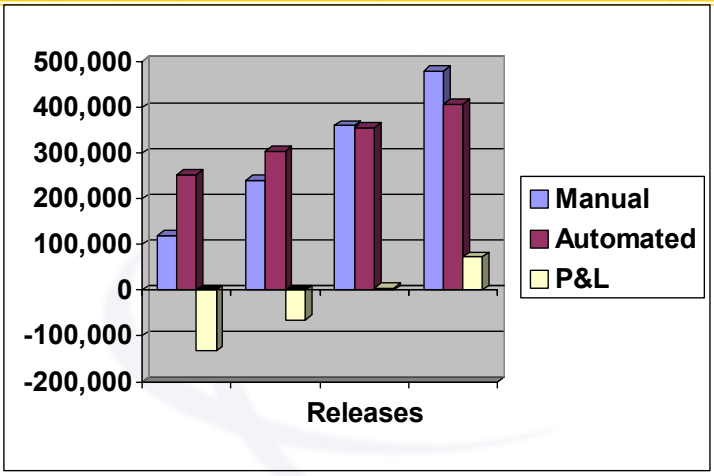
© 2001, SIM Group Ltd.

Cost Reduction

- Research shows 75% think testing costs are too high.
- Other research shows a vast disappointment in the cost reduction from test tool usage.
- Why have we failed to achieve benefits from automated testing?
  - Poor choice of tests to automate
  - Under-engineering (Record / Playback)
  - Over-engineering (parallel software development of test system)

© 2001, SIM Group Ltd.

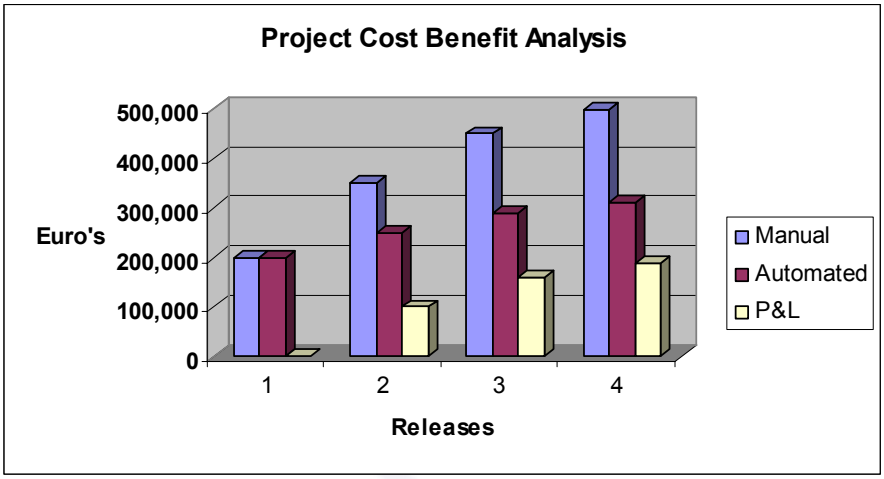
Industry Average



Average cost benefit of automated testing over 42 customers – Who achieved any benefit at all.

© 2001, SIM Group Ltd.

Power Testing



Average cost benefit of power testing on 12 projects.

© 2001, SIM Group Ltd.

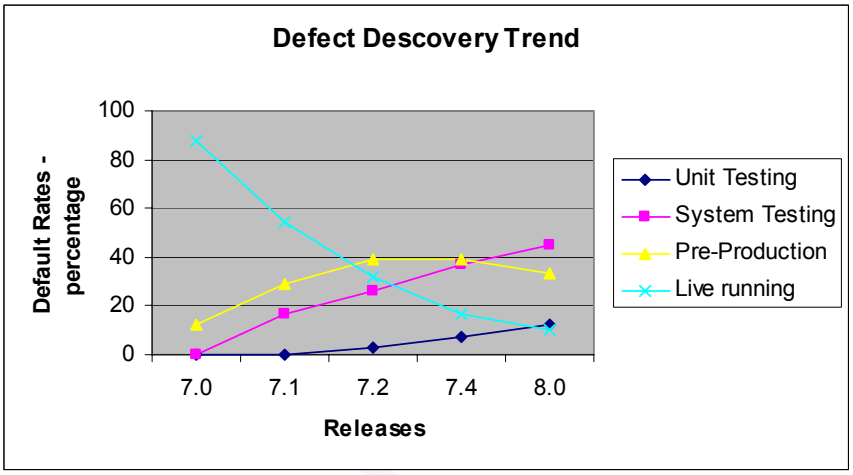
Cost Avoidance

- Research say that 90% of people know poor quality can cost more.
  - Fix releases, support, and damages
- One major bank states every production fault costs € 150,000.[1]

[1] Source – private & confidential

© 2001, SIM Group Ltd.

Power Testing pays off



Power testing used starting with release 7.1

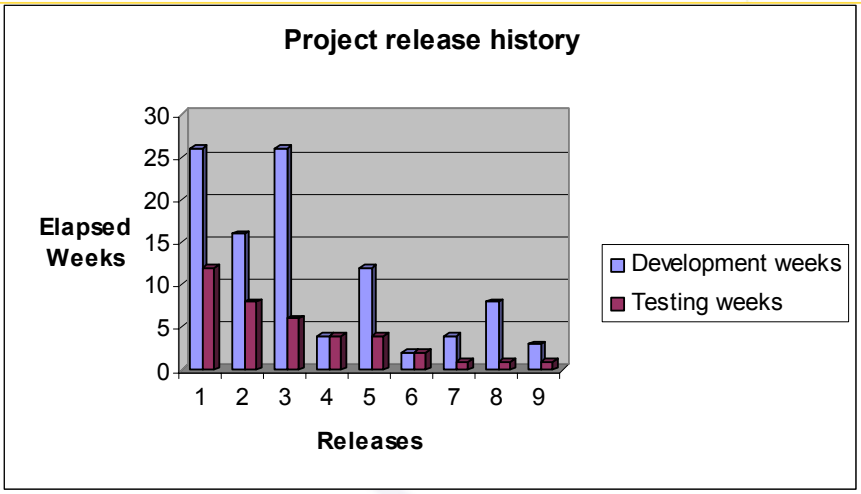
© 2001, SIM Group Ltd.

Time to Market

- 40% states that testing actually delays product delivery / launch.
- NO ONE states testing improves time to product delivery / launch!
- What about:
  - Requirements testing, reviews & inspections?
  - Finding problems earlier
  - Testing faster
  - Test Asset re-use

© 2001, SIM Group Ltd.

Improving time to market



Power Testing applied gradually.

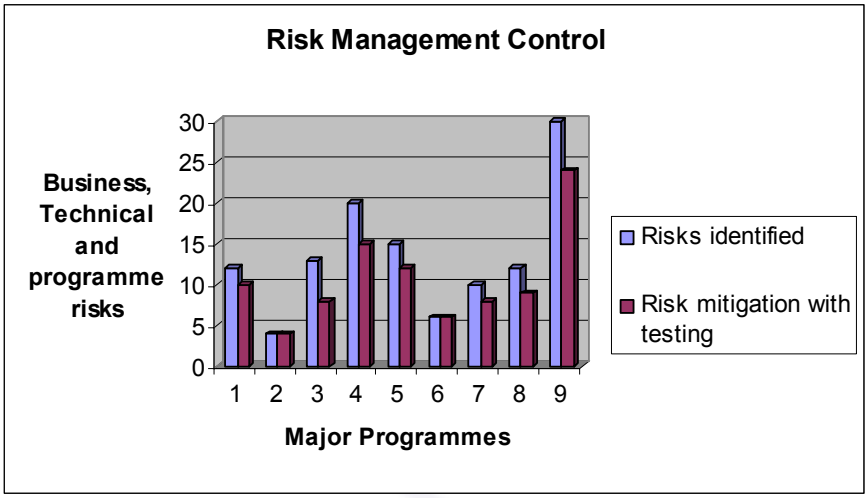
© 2001, SIM Group Ltd.

Risk Reduction

- When asked “Why Test?” – 10% said to reduce risk.
- When asked “How do you reduce risk when making system changes?” – 62% said **Testing.**
- Risk Management is a well defined discipline used in engineering, finance and management.
- It is not a “trick” of testing – (Risk Based Testing)
- Testing activities feature heavily in defining risk mitigation & control.

© 2001, SIM Group Ltd.

Quantifying Risk



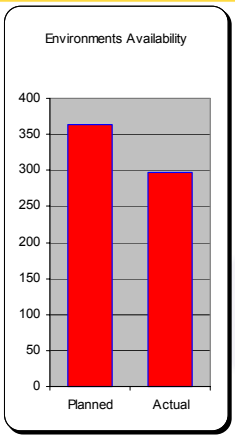
© 2001, SIM Group Ltd.

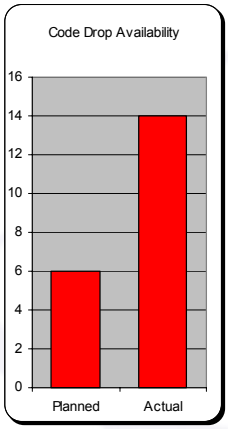
Maintain Control

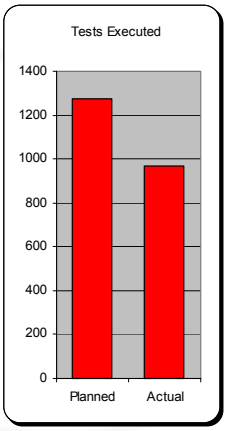
- 75% state they want to manage testing themselves.
  - Over half point to loss of control as to why they must manage testing themselves.
  - 25% admit they want to “adjust & negotiate” testing requirements.

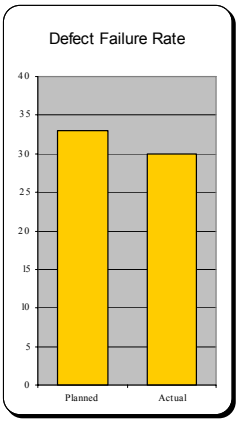
© 2001, SIM Group Ltd.

Test management Dashboard









Power Testing maintains objective control of all activities.

© 2001, SIM Group Ltd.

Conclusion

- What is Power Testing?

© 2001, SIM Group Ltd.

Power Testing

- Manage the use of tools to achieve early, significant benefits by integrating automation into a formal testing methodology.
- Introduce a rigorous approach to test phases that finds problems with great effectiveness.
- Testing teams must start working early on by preparation that includes critical testing of requirements, specifications and designs.
- Informed test management that quantifies status and decisions for controlled decision making.

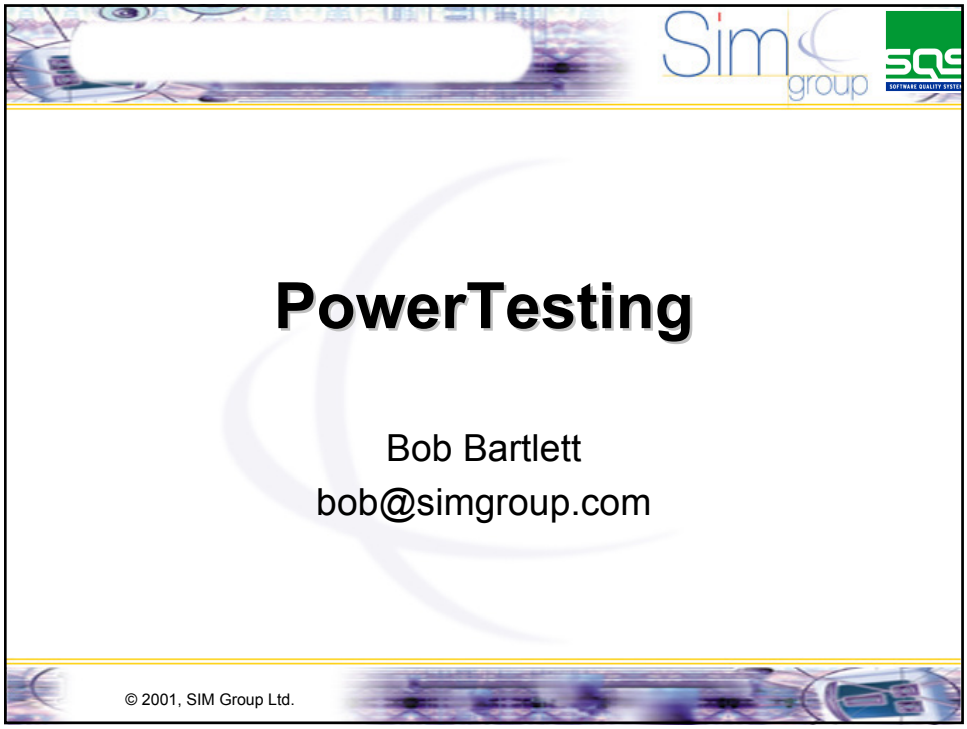
© 2001, SIM Group Ltd.

Power Testing is ……

Good Practice learnt at this conference

- System Validation
- Testing web systems
- Embedded testing
- Component testing
- Automation
- Metrics & process improvement
- Performance testing
- Test Environments
- Requirements testing

© 2001, SIM Group Ltd.


Sim group
PowerTesting
Bob Bartlett
bob@simgroup.com
© 2001, SIM Group Ltd.

QWE2002 Paper K3-1

Mr. Erik Simmons
(Intel Corporation)

From Requirements to Release Criteria

Key Points

- Product Quality.
- Quality Requirements.
- Quantified Quality.
- Release Criteria.

Presentation Abstract

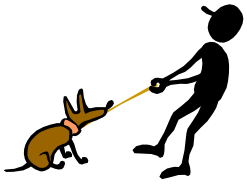
Without good planning and monitoring, product quality is often a matter of chance. Specifying quality requirements is an excellent first step to predictable, managed product quality, but it is just the beginning. Quality specification, quality planning, data generation, quality monitoring, and quality reporting all work together to ensure that quality levels are known throughout the project. Quality release criteria are a good way to ensure that product quality drives interim milestone releases and the final product release. The breadth of quality dimensions can force teams to learn new concepts and skills in order to demonstrate the qualities associated with the release criteria. Despite this challenge, quality specification and quality monitoring are excellent practices that help monitor end product quality throughout the lifecycle.

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.

From Requirements to Release Criteria:
Specifying, Demonstrating, and Monitoring Software Quality

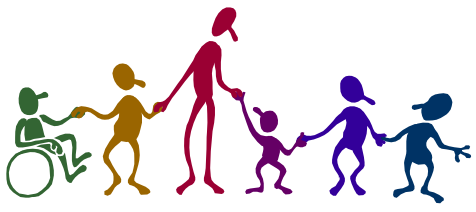
Erik Simmons, Intel Corporation



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

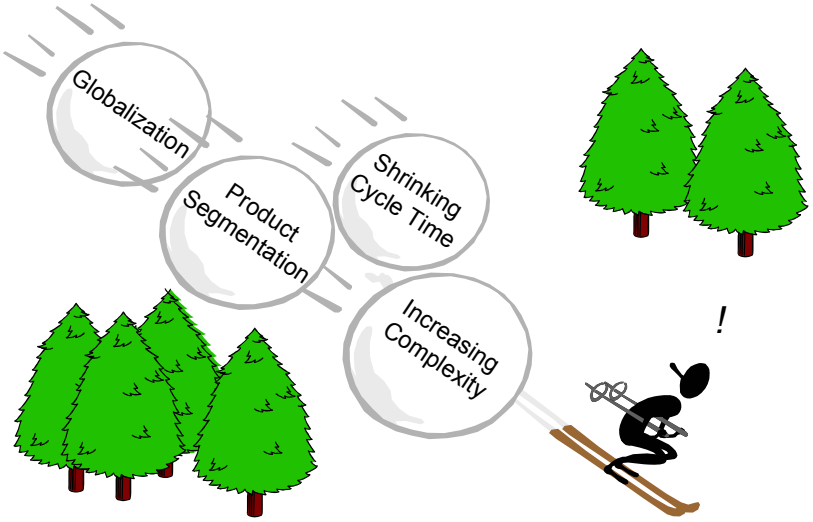
The Setting

- 80,000+ employees worldwide
- Diverse environment, team oriented
- Fast paced, market driven product development
- Strong R&D component
- Hundreds of development teams, varying skill levels
- Frequent reorganizations to attack new opportunities
- *Software? We write software??*



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

Product Development Challenges



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

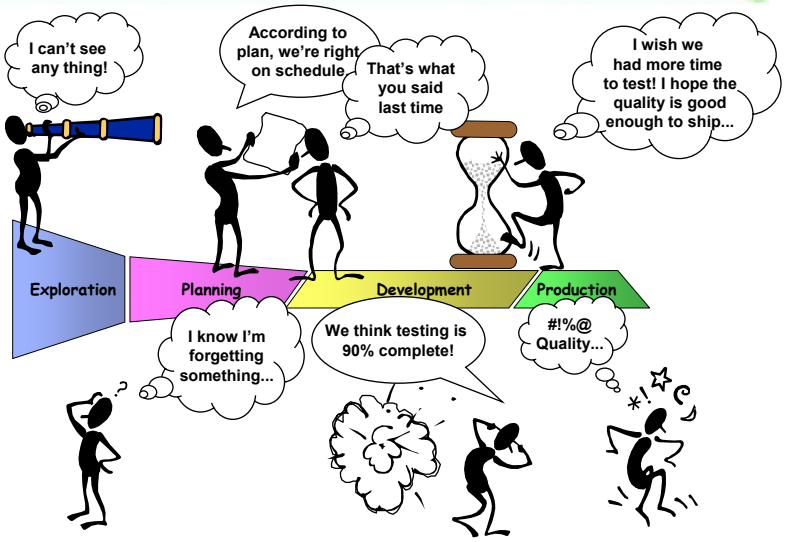
Big Q or Little q?

"Little q": A narrower view of quality, based mainly on defect removal and containment.

"Big Q": A holistic view of quality, including the processes by which products are created, that focuses on defect prevention activities as well as more traditional ones.

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

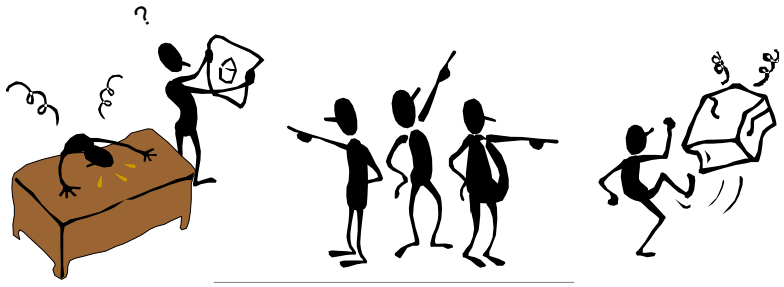
Does This Look Familiar?



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

Common Questions

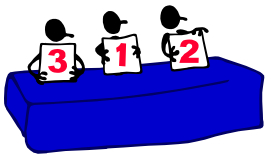
- How can we make quality more predictable?
- What should we measure? How?
- When should we start planning for product quality?
- How will we know if we're meeting our quality goals or not?



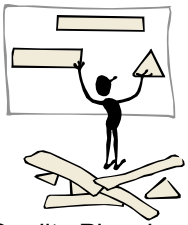
Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Pieces of the Puzzle

Quality Specification



Quality Reporting



Quality Planning

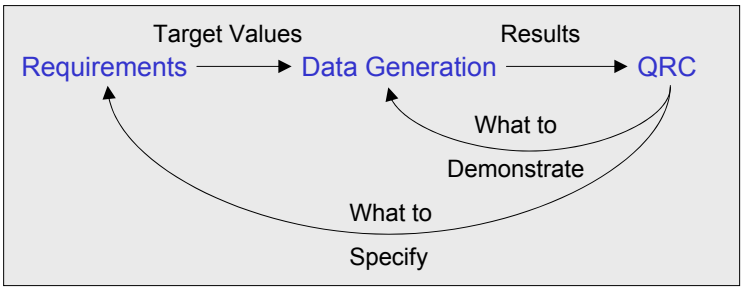
Data Generation

Quality Monitoring

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

Pieces of the Puzzle

Specification and Planning *Demonstration* *Monitoring and Reporting*

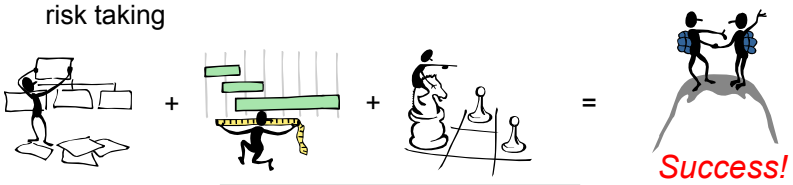


Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Quality and Requirements

- Poorly-defined requirements and uncontrolled changes to requirements often cause:
  - Productivity loss, increased rework, and increased TTM within the product development organization
  - Poor end-product quality, missed deadlines, and dissatisfied customers
  - Project cancellations that occur late in the development lifecycle

What we want: Good definition drives planning and informed risk taking



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

Complete Requirements Specification

A complete specification includes many types of requirements:

- Functional Requirements
- Product/Engineering Constraints
- Quality Requirements
- User Interface Requirements
- Product Documentation Requirements
- Internationalization and Localization Requirements
- Marketing and Sales Requirements
- Legal Requirements
- Other specialized requirement types that your product needs…

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

Quality Requirements

Quality requirements describe qualities that the product or system must possess.

Quality requirements can be organized in general categories:

- Performance
- Reliability
- Usability
- Supportability
- Security
- Maintainability
- Manufacturability
- Testability
- Extensibility
- Scalability

Many other schemes exist – pick one based on your specific environment and needs

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

Software Quality Release Criteria

Assessment Categories

- Functional Testing
- Product Reproducibility
- Legal Compliance
- Continuous Operation
- Defect Data
- Ease of Use
- Install Testing
- Customer Documentation
- Customer Acceptance
- Standards Compliance
- Compatibility Testing
- Performance Validation
- Regression Testing

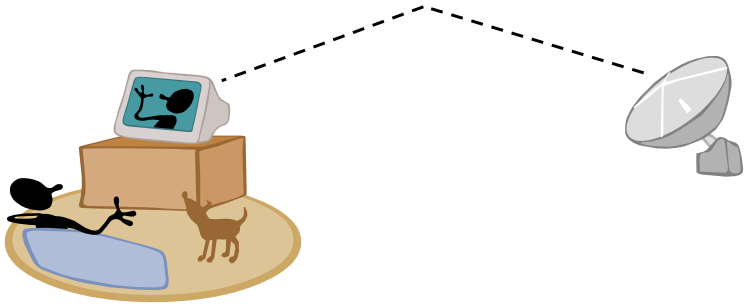
Adapt and customize the list based on product characteristics

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

Example

Suppose you have just been hired as Director of Quality for a new product. Some of the details:

- The product is a TV set-top box that stores broadcast content for later viewing based on customer likes & dislikes
- Prototypes have been built, but many details remain undecided



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

Example

Market snapshot:

- Large customer base
- Generally unskilled with technology
- Low tolerance for failure – TV & appliance mentality
- Competition and average selling price imply low margins

Risk Assessment:

Low margins, large install base, low failure tolerance add up to a major risk:

Customer Support Costs

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

Example

Your role in keeping support costs low as Director of Quality:

Ensuring the product meets goals for:
- Ease of Installation
- Ease of Use
- Reliability
- Fault Tolerance & Recovery

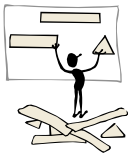
Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Choosing Release Criteria

Your Quality Concerns:

Ease of Use
Ease of Installation
Reliability
Fault Tolerance & Recovery

- Functional Testing
- Product Reproducibility
- Legal Compliance
- Continuous Operation
- Defect Data
- Ease of Use
- Install Testing
- Customer Documentation



Quality Planning

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

Quality Specification – Reliability

"The system must be reliable."

Reliable?
What does that mean?
How can I measure it?

Quality Specification

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

Quality Specification - Reliability

What is a failure? *There are many classes…*

What is the customer's tolerance for failure?

How long is a typical usage session?

Target = Failure Tolerance * Usage Duration

For example:

- Defects causing media playback failure but no damage to equipment
- Failure Tolerance = 1 in 100 uses
- Usage Duration averages 2.5 hrs.
- Target = 250 hours of playback (or 100 uses) for this failure class

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

Quality Specification – Reliability

Reliability.Playback: The reliability of media playback.

SCALE: Mean Time to Failure for playback-related failures only, using a Representative Operational Profile with randomization.

METER: A Probability Ratio Sequential Test with α=10%, β=10%, and a discrimination ratio of 2.0

PLAN [Gold]: At least 350 hours

MUST [Gold]: At least 250 hours

MUST [Beta]: At least 75 hours

Representative Operational Profile: DEFINED: An operational profile that is likely to occur during use of the system after deployment.

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

Related Quality Release Criterion

SW Quality Release Criteria - Alpha Checklist

Product:
Owner:
Last Update:

ID	QUALITY ASSESSMENT	CRITERIA	ALPHA TARGET	Date Done	OWNER	STATUS/COMMENTS
SWCT8	Compatibility Testing	HW/OS/Environment Matrix tests planned.				
SWCT10	Compatibility Testing	Install and Uninstall				
SWLC1	Legal Compliance	by Legal.				
SWLC4	Legal Compliance	Product name approved by Legal.				
SWCO1	Continuous Operation	The product is able to achieve the specified continuous failure free operation target defined in the PRD or SQP.				
SWRT1	Regression Testing	Strategy for performing regression testing, particularly for release candidates, is defined and documented in an approved test plan.				
	Regression	Regression test suite has been updated with tests for all				

SWCO1	Continuous Operation	The product is able to achieve the specified continuous failure-free operation target defined in the PRD or SQP

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

Issues For the SQA & Test Organizations

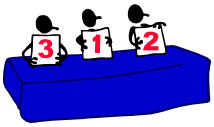
How to demonstrate the targets? The method is prescribed (PRST), but:

- Is the test organization skilled in this type of test?

The demonstration will require failure-free operation for about 3X the target values:

- Can the test be accelerated?
- If multiple systems are used, how many systems?
- What is the boundary for the test?
- How will the quality level be reported? To whom?

Data Generation



Quality Reporting

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

Quality Specification – Setup Time

"The system must be easy to set up."

Why is this taking so long, and what are all these extra parts for?

Quality Specification

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

Quality Specification – Setup Time

What is setup time?

What is “easy”? What other devices compare, and how long is their setup time?

“Easy” also means users don’t need installation support via telephone – the installation guide must suffice

For example:

- Setup time: Time from opening the box to completing the installation instructions included
- Benchmark: 30 minutes (value PC, VCR, DVD)
- Target: Under 30 minutes 80% of the time, 90% completing installation without a support call

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

Quality Specification – Setup Time

Setup.Time: The time needed for a user to set up the system.

SCALE: Mean Time from opening the box to completion of the enclosed instruction set.

METER: Usability testing with at least 25 focus group subjects in each of the top 3 product demographic groups.

PLAN [Gold]: No more than 20 minutes at least 80% of the time

MUST [Gold]: No more than 30 minutes at least 80% of the time

MUST [Beta]: No more than 45 minutes at least 80% of the time

WISH: No more than 10 minutes 100% of the time

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

Quality Specification – Setup Calls

Setup.Calls: The likelihood that technical support is required to complete a setup.

SCALE: Proportion of users having to ask for help at least once during setup.

METER: Usability testing with at least 25 focus group subjects in each of the top 3 product demographic groups.

PLAN [Gold]: No more than 5%

MUST [Gold]: No more than 10%

MUST [Beta]: No more than 20%

NOTE: Relates to Setup.DocQuality

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

Quality Specification – Setup Calls

Setup.DocQuality: The quality of the installation guide.

SCALE: Proportion of test subjects able to locate the solution to Common Problems in the installation guide.

METER: Common Problems tested on at least 20 focus group subjects in the primary product demographic group.

PLAN[Gold]: At least 95%

MUST[Gold]: At least 80%

MUST[Beta]: At least 60%

Common Problems: DEFINED: The top 3 problems located in installation testing for Setup.Calls

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

Related Quality Release Criteria

SW Quality Release Criteria - Gold Checklist
Product:
Owner:
Last Update:

SWIT1	Install Testing	End-user installation documentation or configuration guide exists, includes install and uninstall procedures, and has been verified for accuracy by testing.
SWCD1	Customer Documentation	Customer documentation is checked for technical accuracy and consistency against the product.

27	SWIT2	Install Testing	Installation process is repeatable. If installation over an existing install is required, the end result of the installation must be identical to the result of an install performed on a system that has not had a previous installation of the product.
28	SWIT3	Install Testing	Uninstall process is repeatable and returns the system to its original configuration.
29	SWIT4	Install Testing	Test plan reviewed and approved.
30	SWIT5	Install Testing	Test cases written.
31	SWIT6	Install Testing	All planned Install / Uninstall testing is complete, e.g., as indicated by progress against schedule charts.

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

Monitoring Product Quality

A scorecard of the release criteria form a quality dashboard:

SW Quality Release Criteria Scorecard
Product: Widgit 2.1
Owner: B. Lee Gured
Last Update: WW 30 2000
Current Software Milestone Objective: Beta - PC 2.2

Risk Legend:
1: Showstopper
2: High Risk
3: Medium Risk
4: Low Risk
5: Negligible Risk

ID	QUALITY ASSESSMENT	DESCRIPTION	RISK CURRENT	RISK PREVIOUS	OWNER	NOTES/COMMENTS
	Summary	Overall product health.				
SWFT	Functional Testing	Evaluation of the testing of features and functions to determine whether the testing is adequate for ensuring quality.				
SWDD	Defect Data	Evaluation of product defect information/verification of correct product operation.	3	4	Smith	Medium risk is due to spike in Open trend
SWPR	Product Reproducibility	Evaluation of the ability to archive, accurately and consistently reproduce/rebuild the software product, and ensure the consistency of the product package with the content expectations for the released product package.				
SWIT	Install Testing	Evaluation of install/uninstall testing to verify correct install and uninstall of product.				
SWCD	Customer Documentation	Evaluation of product user documentation for accuracy and consistency with product operation.				
SWCT	Compatibility Testing	Evaluation of the product under test to perform its required functions while sharing the same environment with one or more systems or components. This includes the categories of binary sharing, data sharing, HW/OS/Environments, and backward compatibility.				

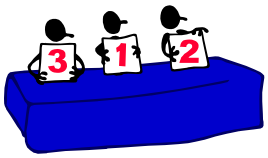
These reports can provide a succinct picture of progress towards quality goals, with details available behind the summary

Quality Monitoring

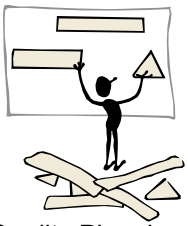
Copyright © 2002 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Summary

Quality Specification



Quality Reporting



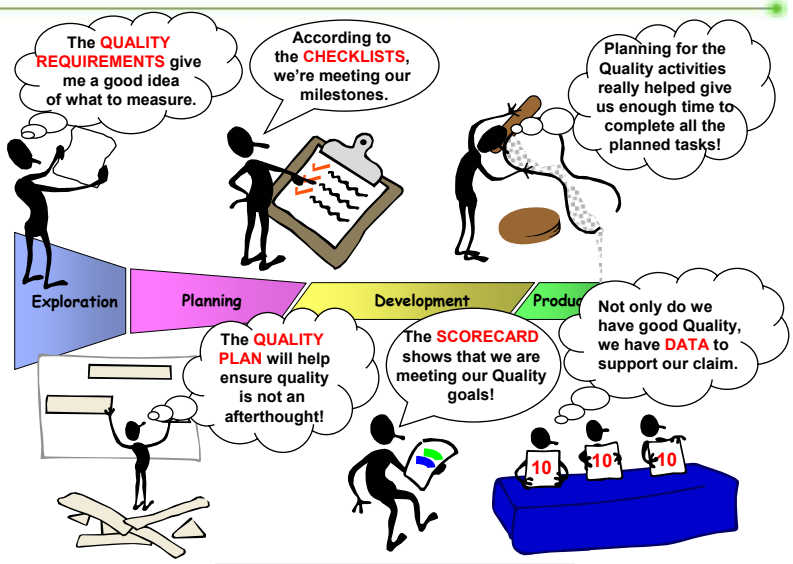
Quality Planning

Data Generation

Quality Monitoring

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

A Different Experience…



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

From Requirements to Release Criteria:

Specifying, Demonstrating, and Monitoring Product Quality

Erik Simmons
Intel Corporation
JF1-46
2111 NE 25th Ave.
Hillsboro, OR 97214-5961
erik.simmons@intel.com

Version 1.1, 08/17/01

Prepared for the Pacific Northwest Software Quality Conference, 2001

Key Words: Product Quality; Quality Requirements; Quantified Quality; Release Criteria

Author Biography

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.

Abstract

Without good planning and monitoring, product quality is often a matter of chance. Specifying quality requirements is an excellent first step to predictable, managed product quality, but it is just the beginning. Quality specification, quality planning, data generation, quality monitoring, and quality reporting all work together to ensure that quality levels are known throughout the project. Quality release criteria are a good way to ensure that product quality drives interim milestone releases and the final product release. The breadth of quality dimensions can force teams to learn new concepts and skills in order to demonstrate the qualities associated with the release criteria. Despite this challenge, quality specification and quality monitoring are excellent practices that help monitor end product quality throughout the lifecycle.

Copyright © 2001 Intel Corporation. All rights reserved.

Introduction

If you ask a typical product or project manager what his biggest challenge is, you will often get answers involving cost and schedule. Only occasionally in the commercial product development arena will you hear product managers mention product quality as a significant concern. There are several reasons for this:

- Recent market trends have placed tremendous pressure on companies to be the first to market with a product
- Venture capital funds expect to see a product quickly as a sign of progress
- Products are more complex than ever before, lengthening development schedules

There is some rationale behind a manager's focus on attributes other than quality at first. When bringing a new product to market, the innovators that make up the first wave of customers are much more concerned with features and the newness of the product than with price or quality. Later, when the early majority begins to buy, quality and price become greater concerns [Rogers95].

Despite this reasoning, without good planning and monitoring, product quality is a matter of chance. Numerous teams have launched products only to find out that required qualities were missing or inadequate, leading to expensive and embarrassing retrofits, product recall, or outright product failure in the marketplace. In order to prevent such mishaps, a product development team must perform several activities:

- Quality specification
- Quality planning
- Data generation
- Quality monitoring
- Quality reporting

These activities are not strictly sequential, but can overlap or occur in parallel. Each activity is introduced below and is then illustrated in the example that follows.

Quality Specification

Quality specification is the process of establishing and documenting the product's quality requirements[1].

Quality requirements are often overlooked when specifying the requirements for a product. This could be because the requirements team does not know what qualities should be included, does not know how to write quality requirements well, or is pressured by existing culture and practices to focus solely on functional requirements.

Quality has many dimensions. A holistic view of quality must consider all aspects of quality for all stakeholders, rather than focusing narrowly on defect detection and removal. By taking this approach, known in some circles as "big Q" rather than "little q", the resulting quality specification will thoroughly cover product quality.

[1] Quality requirements are called "non-functional requirements" in many sources, but this seems to be a poor term for what are essentially qualities representing how well (fast, reliable, etc.) the product's functions must deliver what they provide. In fact, many students laugh out loud when the term is introduced during requirements classes, joking that it must refer to the features in the product that don't work. The same term is also applied to some design constraints, further clouding its meaning.

Copyright © 2001 Intel Corporation. All rights reserved.

Although there is no standard categorization of quality requirements, the set of categories shown in Figure 1 is typical. Organizations typically customize such lists and add details based on the type of product they build and areas in which they have faced problems in the past. For example, software performance can be segmented into 'time' (speed) and 'space' (resource usage). Security contains many facets, including integrity, availability, confidentiality, and accuracy [Chung00]. Availability can be separated into reliability, maintainability, and integrity [Gilb88]. One exhaustive list of quality requirement categories has161 elements [Chung00]. Overlaps within definitions are extremely common within the different sources. The best advice is to establish a well-defined list of categories and maintain it over time, based on project experience and need.

- Performance
- Availability
- Usability
- Supportability
- Security
- Maintainability
- Manufacturability
- Testability
- Extensibility
- Scalability

Figure 1. Example Quality Requirement Categories.

Quantifying Quality Requirements

To be meaningful, quality requirements must be verifiable and measurable. Stating quality requirements precisely in natural language is difficult. Authors often use weak words such as *fast, easy, simple, minimal, as needed, adequate, normal,* and a host of others [Wilson96]. These qualitative terms leave the meaning of a requirement open to different interpretation by each reader. Such ambiguity leads to missed expectations in the final product, and prevents accurate testing.

Planguage is a technique to quantify qualitative statements that works very well on quality requirements. Planguage is a keyword-driven specification language that improves communication about complex ideas. Easy to learn and use, Planguage also provides many benefits [Simmons01, Gilb01, Gilb97a, Gilb97b]. A basic description of Planguage is given in Appendix A. Planguage is used to document the quality requirements in the example later in this paper.

Setting Target Values

Before we can quantify quality requirements, the levels of achievement for each quality requirement must be defined through target setting. Target setting is most often based on analysis, benchmarking, or standards. Each type of quality requirement (performance, reliability, scalability, etc.) can require a specific approach to target setting in order to obtain accurate, realistic, and meaningful targets. Setting quantitative targets for these attributes is challenging at first, forcing a team to think in new ways about the product it is building. However, most of the methods are not difficult and the results more than justify the effort. The toughest part of the process is often finding the necessary data or applicable standards.

Quality Planning

If quality only made it as far as a requirements specification, it would be of little value. The additional activities of quality planning, data generation, quality monitoring, and quality reporting are the keys to obtaining real value from quality requirements.

Many existing sources discuss the need to use quality requirements to guide the subsequent product design, and for traceabililty between requirements and design as a form of verification. But this is not enough. Quality must permeate beyond the specification and design to become one of the guiding principles for interim milestone reviews and the final release of the product.

Capturing the important product qualities from the requirements as quality release criteria helps ensure this happens.

Copyright © 2001 Intel Corporation. All rights reserved.

Quality Release Criteria
Quality release criteria (QRC) are used to assess and monitor the risk to interim milestone releases and the final product release. Quality release criteria cover a diverse set of quality attributes for a product. Some of the criteria are related to product quality requirements. Others are related to SQA, test, software configuration management, or project management data.

The QRC are grouped within assessments, which makes quality reporting and quality monitoring simpler. Assessments similar to those used at Intel are shown in Table 1 [Gatlin00]. Some assessments only apply to certain situations or categories of products. The assessments are always used as a point of departure on a project. Each project must customize and adapt the assessments and underlying quality release criteria to its needs.

Table 1. Assessments Containing Quality Release Criteria.

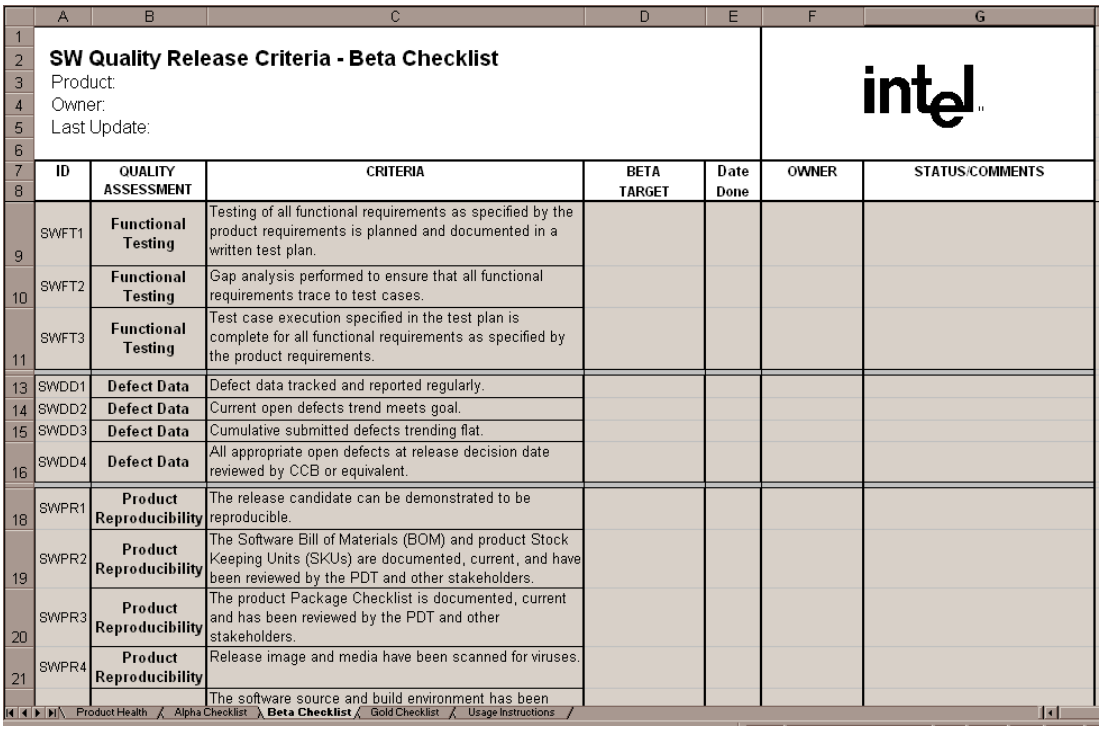
Assessment	Description
Functional Testing	Evaluation of the testing of features and functions to determine whether the testing is adequate to ensure quality.
Product Reproducibility	Evaluation of the ability to archive, reproduce, and rebuild the software product, and that the consistency of the product package with the contents of the released product has been verified.
Legal Compliance	Evaluation of product compliance with legal requirements.
Continuous Operation	A method for detecting time-dependant product defects such as memory leaks and race conditions. It is also one indicator of product reliability over time.
Defect Data	Evaluation of product defect information and verification of correct product operation.
Ease of Use	Evaluation of the ease of use testing to ensure it covers at least the product's user interface and all features related to ease of product use.
Install Testing	Evaluation of install and uninstall testing to verify correct, reliable install and uninstall of product is possible.
Customer Documentation	Evaluation of product user documentation for accuracy and consistency with product operation.
Customer Acceptance	Evaluation of the testing performed to determine whether the product meets its customer acceptance criteria.
Standards Compliance	Evaluation of product compliance with product standards including Intel engineering or other standards. Process standards are specifically excluded.
Compatibility Testing	Evaluation of the product under test to perform its required functions while sharing the same environment with one or more systems or components.
Performance Measurement	Evaluation of the performance attributes of the software under test to determine whether the validation effort is adequate for ensuring that performance goals are met.
Regression Testing	Evaluation of the testing of the product that determines whether changes have caused unintended effects and whether the changed product still meets its requirements.

Assessments can contain different quality release criteria and associated targets for the alpha, beta, and gold milestones. For example, at Intel the Defect Data assessment (named SWDD) is defined to contain the following four criteria at the beta and gold milestones, but only SWDD1 at the alpha milestone:

SWDD1. Defect data tracked and reported regularly.
SWDD2. Current open defects trend meets goal.
SWDD3. All open defects at release decision date reviewed by the Change Authority.
SWDD4. Defects waived by the Change Authority for release are documented in Release Notes.

Copyright © 2001 Intel Corporation. All rights reserved.

This system of definition is quite natural to implement in the form of a spreadsheet, with separate worksheets for the alpha, beta, and gold milestones. An example subset of the release criteria for the beta milestone is given in Figure 2.



SW Quality Release Criteria - Beta Checklist

Product:
Owner:
Last Update:

ID	QUALITY ASSESSMENT	CRITERIA	BETA TARGET	Date Done	OWNER	STATUS/COMMENTS
SWFT1	Functional Testing	Testing of all functional requirements as specified by the product requirements is planned and documented in a written test plan.				
SWFT2	Functional Testing	Gap analysis performed to ensure that all functional requirements trace to test cases.				
SWFT3	Functional Testing	Test case execution specified in the test plan is complete for all functional requirements as specified by the product requirements.				
SWDD1	Defect Data	Defect data tracked and reported regularly.				
SWDD2	Defect Data	Current open defects trend meets goal.				
SWDD3	Defect Data	Cumulative submitted defects trending flat.				
SWDD4	Defect Data	All appropriate open defects at release decision date reviewed by CCB or equivalent.				
SWPR1	Product Reproducibility	The release candidate can be demonstrated to be reproducible.				
SWPR2	Product Reproducibility	The Software Bill of Materials (BOM) and product Stock Keeping Units (SKUs) are documented, current, and have been reviewed by the PDT and other stakeholders.				
SWPR3	Product Reproducibility	The product Package Checklist is documented, current and has been reviewed by the PDT and other stakeholders.				
SWPR4	Product Reproducibility	Release image and media have been scanned for viruses.				
		The software source and build environment has been				

Product Health | Alpha Checklist | Beta Checklist | Gold Checklist | Usage Instructions

Figure 2. Example QRC Spreadsheet, Beta Milestone.

Data Generation

Because the quality release criteria cover an array of topics, the raw data to evaluate the QRC come from many sources. Functional testing, inspections, audits, and project metrics are all contributors to at least some of the criteria. Initially, the breadth of the quality release criteria places pressure on the data generation capabilities of most organizations, since evaluating all the necessary criteria requires many new skills, techniques, and tools. For example, relatively few software test teams have experience in demonstrating software reliability through a continuous operation demonstration.

When improved requirements specification techniques are adopted along with a broad set of quality release criteria, an organization must also consider how to train and equip personnel to support the required demonstrations. Without this, the investment in new practices within product requirements and quality release criteria will be frustrating, unsuccessful, and a waste of project resources.

Quality Reporting

Organizations must consider how, how often, and to whom to report the QRC data. The data should be reported to a group chartered with managing the product development effort and monitoring risks to the alpha, beta, and gold milestones. This might be a project office, a project manager, risk manager, product planning council, or another similar organization. Typically, a cross-section of stakeholder groups (such as product quality, program/product management, corporate finance, and product marketing) should receive the release criteria risk data to make sound project-related decisions.

Copyright © 2001 Intel Corporation. All rights reserved.

Quality reporting is often a new practice for an organization. Because of this newness and the nature of quality data, quality reporting can be a sensitive topic at first, so be sure that all parties understand the data reported. Quality data is prone to misinterpretation by those unfamiliar with data gathering practices and limitations. Misinterpretation or misuse of the quality data[2] can have lasting negative consequences for an organization. Educating the participants about the definition, collection practices, reporting formats, and allowed uses of the data can help overcome initial reluctance. Sometimes, placing allowed and prohibited uses of quality data into a company policy is also helpful, so long as compliance is monitored and there are consequences for violating the policy.

Quality Monitoring

The quality release criteria enable quality monitoring to begin as soon as the data can be generated. Exactly when this is depends on the nature of each criterion. For example, the product reproducibility criterion cannot be effectively measured until the development team generates a complete system build.

Because of the large number of quality release criteria, a quality dashboard made up of the assessment categories is useful during project review meetings to get a quick but complete picture of quality assessment status (Figure 3). Program managers and others with responsibilities related to product or risk management can monitor the progress towards alpha, beta, and gold milestone releases at a glance, while the details behind the dashboard are available on the sheets for each milestone if needed.

SW Quality Release Criteria Scorecard

Product:
Owner:
Last Update:
Current Software Milestone Objective:

Risk Legend:
1: Showstopper
2: High Risk
3: Medium Risk
4: Low Risk
5: Negligible Risk

ID	QUALITY ASSESSMENT	DESCRIPTION	RISK		OWNER	NOTES/COMMENTS
			CURRENT	PREVIOUS		
	Summary	Overall product health.				
SWFT	Functional Testing	Evaluation of the testing of features and functions to determine whether the testing is adequate for ensuring quality.				
SWDD	Defect Data	Evaluation of product defect information/verification of correct product operation.				
SWPR	Product Reproducibility	Evaluation of the ability to archive, accurately and consistently reproduce/rebuild the software product, and ensure the consistency of the product package with the content expectations for the released product package.				
SWIT	Install Testing	Evaluation of install/uninstall testing to verify correct install and uninstall of product.				
SWCD	Customer Documentation	Evaluation of product user documentation for accuracy and consistency with product operation.				
SWCT	Compatibility Testing	Evaluation of the product under test to perform its required functions while sharing the same environment with one or more systems or components. This includes the categories of binary sharing, data sharing, HW/OS/Environments, and backward compatibility.				
SWLC	Legal Compliance	Evaluation of product compliance with legal requirements.				
SWCO	Continuous Operation	A method for detecting time-dependant product defects such as memory leaks and race conditions. It is also one indicator of product reliability over				
SWRT	Regression Testing	Evaluation of the testing of the product to determine that changes have not caused unintended effects and the product still complies with requirements.				
	Customer	Evaluation of the testing performed to determine whether the product meets				

Product Health | Alpha Checklist | Beta Checklist | Gold Checklist | Usage Instructions

Figure 3. Example Quality Dashboard.

[2] Improper use includes employee performance appraisals, compensation reviews, or promotion evaluations. Misinterpretation includes incorrectly reading graphs, misinterpreting statistical results such as confidence intervals or significance levels, overlooking sampling problems, confusing association with causality, or violating distributional assumptions.

Copyright © 2001 Intel Corporation. All rights reserved.

Putting It Together

Near the beginning of a project, a team starts with the standard set of QRC assessments and the quality requirement categories found in the organization's requirements document template. From these, the team can form a complete framework of product quality based on the particular characteristics of the product they must produce.

The quality requirements both influence and are influenced by the quality release criteria (Figure 4). Quality requirements provide target values used to demonstrate the status of the QRC, but the QRC also influence the categories of quality requirements chosen by a team for specification. For example, the continuous operation release criterion influences the product's requirements specification, forcing a team to specify product reliability as one of the requirements for the product. At the same time, the reliability requirement establishes the target values for the continuous operation demonstration that feeds data to the quality release criterion.

The concepts and practices presented above all work in concert to help ensure that product quality is not a matter of chance. By taking a holistic view, product quality will not be a matter of chance, but rather of conscious decisions based on data. This risk-driven, data-driven approach will get problems out in the open early, while there is still time to correct them. The result will be a reduction in rework, improved schedule predictability, shortened time to market, and improved product quality.

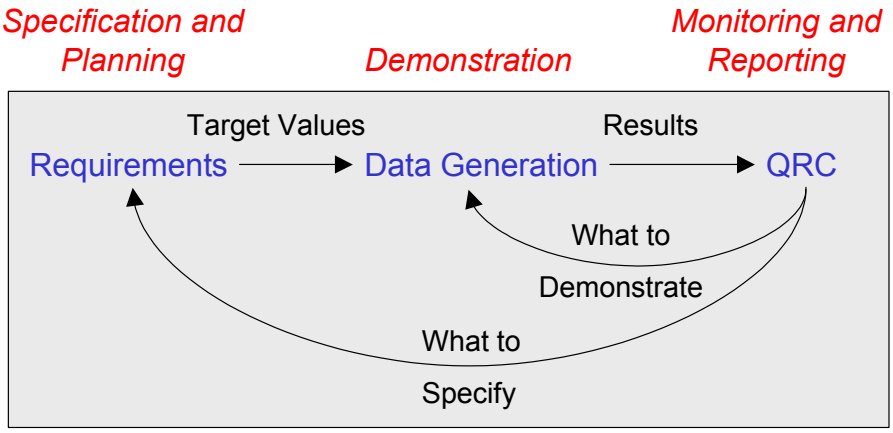


Figure 4. Relationship Between Requirements and Quality Release Criteria.

Example

Suppose that you are the Quality Manager for a new product. The product is a device similar to a television set-top box in appearance, and its primary function is to capture movies broadcast on sidebands of standard TV broadcasts for possible later viewing. The device chooses to capture a movie for the user based on a proprietary algorithm that combines several dimensions, including end user preferences for movie type, the viewer's age, and actors. When a user decides to watch a captured movie, a charge is incurred via a back-channel telephone link to a billing service.

The potential market for the product is large, since it does not rely on proprietary networks or equipment for transmission or billing. The user base has a low tolerance for failure based on experience with television and household appliances. Competition exists, so prices for viewing a

Copyright © 2001 Intel Corporation. All rights reserved.

movie will lead to small margins. The product will use the “shaver and blades” model, in which the product itself will be inexpensive and most of the revenue will be derived from movie viewing fees.

The project team’s risk analysis shows that the large potential user base, low margins, and user intolerance for failure lead to a significant risk to profitability: support calls. The team estimates that recouping the cost of a single user support call will require a full year of average product usage. A defect in any portion of the product, its documentation, or its usability could have disastrous financial effects.

As quality manager, you want to help reduce this risk during design and development of the product. Several factors might lead to a support call, including installation and setup, daily use, product reliability, and fault tolerance and recovery. By ensuring that the product is simple to install, quick to set up, easy to use, reliable, and self-correcting for as many errors as possible, support calls will be minimized. These issues trace into the QRC assessments as shown in Table 2.

Table 2. Support Call Issues Mapped to QRC Assessments.

Issue Leading to Support Call	QRC Assessments
Ease of Use	Ease of Use
Ease of Installation	Install Testing Customer Documentation
Reliability	Continuous Operation
Fault Tolerance and Recoverability	Functional Testing Defect Data

Let’s examine two of the concerns listed above in more detail: reliability and ease of installation.

Reliability

Several factors govern reliability target for the example product. First, there are many types of failure, each possibly having a different target value. The reason for this is that a user base has differing tolerances for different types of failure, based on characteristics such as the effects of the failure, the user’s innovativeness, and his experience with technology [Rogers95]. For the class of failures defined as *software defects leading to media playback failure* but no damage to hardware, suppose that market research shows the user base to have a tolerance of one failure per 100 uses. Given a projected average usage duration of 2.5 hours, this translates to a target of 250 hours of failure-free operation (or, alternatively, 100 consecutive uses). Building in an engineering margin of 100 hours to the plan, a traditional reliability requirement might state that the system Mean Time to Failure (MTTF) be at least 350 hours, measured at 95% statistical confidence. However, a good reliability requirement is made up of much more than just the target value, and this is where Planguage is so effective. The Planguage requirement looks like this:

TAG: Reliability.Playback
GIST: The reliability of media playback.
SCALE: Mean Time to Failure for playback-related failures only, using a Representative Operational Profile with randomization.
METER: A Probability Ratio Sequential Test with α=10%, β=10%, and a discrimination ratio of 2.0
PLAN [Gold]: At least 350 hours
MUST [Gold]: At least 250 hours
MUST [Beta]: At least 75 hours
Representative Operational Profile: DEFINED: An operational profile that is likely to occur during use of the system after deployment.

Copyright © 2001 Intel Corporation. All rights reserved.

This requirement is verifiable and measurable: the measurement method and targets are both specified. Planguage allows the specification of goals for different milestones (beta and gold). It also allows separation of planned levels of success from “must have” minimums needed to avoid financial, political, or other failure. In this case, a minimum reliability of 75 hours is required before the beta milestone is reached and the product is tested by a broad cross-section of the user base. This level is the minimum judged to protect brand equity, allowing earliest possible product use without damaging the reputation of the company. So, if you can’t demonstrate greater than a 75 hour MTTF, you do not pass the beta quality gate.

The continuous operation quality release criterion captures the reliability requirement and its associated targets:

CO1: The product is able to achieve the specified continuous failure free operation target defined in the Product Requirements Document or Software Quality Plan.

Once product development progresses to the point that we can measure reliability, this criterion will help determine whether the product meets the reliability requirement at all three milestones.

Ease of Installation
Targets for ease of installation must be based on some benchmark value that represents what the typical user would expect (and tolerate) as setup time without calling a technical support line for help. By examining devices such as CD or DVD players, VCRs, or value segment PCs, a reasonable target would be less than 30 minutes, with fewer than 10% of users requiring phone support during the process. Building in some margin, and recognizing that an ideal value might be closer to 10 minutes or less in all cases, we can write the requirement for installation time in Planguage as:

> TAG: Setup.Time
> GIST: The time needed for a user to set up the system.
> SCALE: Mean Time from opening the box to completion of the enclosed instruction set.
> METER: Usability testing with at least 25 focus group subjects in each of the top 3 product demographic groups.
> PLAN [Gold]: No more than 20 minutes at least 80% of the time
> MUST [Gold]: No more than 30 minutes at least 80% of the time
> MUST [Beta]: No more than 45 minutes at least 80% of the time
> WISH: No more than 10 minutes 100% of the time

The targets in this requirement allow progress towards increasingly stringent quality goals.

We can write a separate requirement to cover the proportion of the users requiring a support call to complete the installation:

> TAG: Setup.Calls
> GIST: The likelihood that technical support is required to complete a setup.
> SCALE: Proportion of users having to ask for help at least once during setup.
> METER: Usability testing with at least 25 focus group subjects in each of the top 3 product demographic groups.
> PLAN [Gold]: No more than 5%
> MUST [Gold]: No more than 10%
> MUST [Beta]: No more than 20%
> NOTE: Relates to Setup.DocQuality

The need for a call to technical support during installation relates to the quality of the user documentation, so we need an additional requirement for documentation quality:

> TAG: Setup.DocQuality

Copyright © 2001 Intel Corporation. All rights reserved.

GIST: The quality of the installation guide.
SCALE: Proportion of test subjects able to locate the solution to Common Problems in the installation guide.
METER: Common Problems tested on at least 20 focus group subjects in the primary product demographic group.
PLAN[Gold]: At least 95%
MUST[Gold]: At least 80%
MUST[Beta]: At least 60%
Common Problems: DEFINED: The top 3 problems located in installation testing for Setup.Calls

The related quality release criteria for these requirements are:

SWIT1: End-user installation documentation or configuration guide exists, includes install and uninstall procedures, and has been verified for accuracy by testing.
SWCD1: Customer documentation is checked for technical accuracy and consistency against the product.

Monitoring the release criteria associated with reliability, install time, support calls, and end user installation documentation as the product development effort progresses provides a risk-based assessment of product health in the areas needed to minimize support costs. More generally, the summarized release criteria constitute a product health dashboard that can be updated and reviewed at regular intervals throughout the project.

References

Chung00	Chung, Lawrence, Nixon, Brian, et al, *Non-Functional Requirements in Software Engineering*, Klewer Academic Publishers, 2000
Gatlin00	Gatlin, Manny, and Hannon, Gregory, *Managing Software Product Quality*, presented at the Pacific Northwest Software Quality Conference, 2000
Gilb01	Gilb, Tom, *Competitive Engineering*, Addison Wesley 2001 (forthcoming)
Gilb97b	Gilb, Tom, *Quantifying the Qualitative*, available at http://www.result-planning.com
Gilb97a	Gilb, Tom, *Requirements-Driven Management: A Planning Language*, Crosstalk, June 1997
Gilb88	Gilb, Tom, *Principles of Software Engineering Management*, Addison Wesley 1988
Rogers95	Rogers, Everett M., *Diffusion of Innovations 4th Ed.*, The Free Press 1995
Simmons01	Simmons, Erik, *Quantifying Quality Requirements Using Planguage*, presented at Quality Week 2001
Wilson96	Wilson, William, Rosenberg, Linda, and Hyatt, Lawrence, *Automated Quality Analysis of Natural Language Requirements Specifications*, available at http://satc.gsfc.nasa.gov/support.

Copyright © 2001 Intel Corporation. All rights reserved.

Appendix A: A Brief Description of Planguage

Planguage (PLANning lanGUAGE) was created by Tom Gilb [Gilb01, Gilb97a, Gilb97b]. Planguage is a keyword-driven language that can be used in requirements specifications, design documents, plans, and other places where qualitative statements are common[3]. Its primary benefits are that it quantifies qualitative statements, improves communication about complex ideas, prevents omissions within quality requirements, and is easy to learn.

Planguage has a rich set of keywords. The commonly used keywords are given in Table 3.

Table 3. Planguage Keywords.

TAG	A unique, persistent identifier
GIST	A short, simple description of the concept contained in the Planguage statement
STAKEHOLDER	A party materially affected by the requirement
SCALE	The scale of measure used to quantify the statement
METER	The process or device used to establish location on a SCALE
MUST	The minimum level required to avoid failure
PLAN	The level at which good success can be claimed
STRETCH	A stretch goal if everything goes perfectly
WISH	A desirable level of achievement that may not be attainable through available means
PAST	An expression of previous results for comparison
TREND	An historical range or extrapolation of data
RECORD	The best-known achievement
DEFINED	The official definition of a term
AUTHORITY	The person, group, or level of authorization

Besides keywords, Planguage also offers several convenient and useful sets of symbols:

- Fuzzy concepts requiring more details are marked using angle brackets: <fuzzy concept>
- Qualifiers, which are used to modify other keywords, are contained within square brackets: [when, which, …]
- A collection of objects is indicated by placing the items in braces: {item1, item2, …}
- The source for a statement is indicated by an arrow: Statement ← source

[3] The term Planguage is also used as the name of some programming languages for parallel processors, but that use is not related to its use in this paper.

Copyright © 2001 Intel Corporation. All rights reserved.

QWE2002 Paper K3-2

Mr. Rik Nuytten
(Cisco Systems Belgium)

Building the Infrastructure for The Future

Key Points

- Built on open standards.
- Flexability and intelligence must be incorporated.
- It must be able to handle voice, data and video simultaneously.

Presentation Abstract

To be supplied.

About the Author

In November 1998, Rik Nuytten joined Cisco Systems as Market Development Manager for the Public Sector for Europe, Middle East and Africa (EMEA). He covered both the defence and the civil sector. In October 1999 he was promoted to Channels Marketing Manager for EMEA. He currently runs all channel accreditation and quality programs across the theatre.

Rik Nuytten has a degree in Computer Sciences and started his career in the Utility sector as a software developer, project manager and consultant, prior to joining Digital Equipment Corporation in 1983. At Digital, Rik Nuytten held management positions in Software Services, Sales, Field Service and Marketing. He managed teams on a country basis, as well as Europe-wide. In 1997, at the purchase of the Digital's Networks Division by Cabletron Systems, Rik Nuytten became Vice-President Marketing for Cabletron Europe.

For many years, Rik Nuytten has been the Managing Director of the Belgium Cystic Fibrosis Association.

Note: This presentation will be available from the Conference web site.

QWE2002 Paper 1T

Mr. Jamal Said & Mr. Eric Steegmans
(Department of Computer Science, K.U. Leuven)

Validating Quality Requirements of Object Oriented Design

Key Points

- Transformation of the conceptual model into design model.
- Evaluating the quality of the design model.
- Iterating through the design model to fulfil the quality requirements.

Presentation Abstract

There is much about object-oriented design that is subjective- an experience designer 'knows' how to characterize an OO suystem so that it will effectively implements customer requirements. But as an OO design model grows in size and complexity, a more objective view of the characteristics of the design can benefit both the experienced designer (who gains additional insight) and the novice (who obtainsan idication of quality that would otherwise be unavailable. This viewpoint is achieved by considering both the functional and nonfunctional requirements at high level of abstraction.

About the Author

Jamal Said is a Ph.D. student at the university of Leuven- Belgium. He is a memeber of the software development methodology group headed by Prof. Eric Steegmans His research activities includes object oriented software design, and software quality. He has been working on this field for almost two years in this field.

Validating quality Requirements for OOD

Jamal Said & Eric Steegmans

Katholieke Universiteit Leuven

Department of Computer Science

Leuven-Belgium

Overview

- Introduction
- Transformational Approach
- Software Quality
- Software Metrics
- Quality Assessment Procedure
- Sample Metrics
- Conclusion

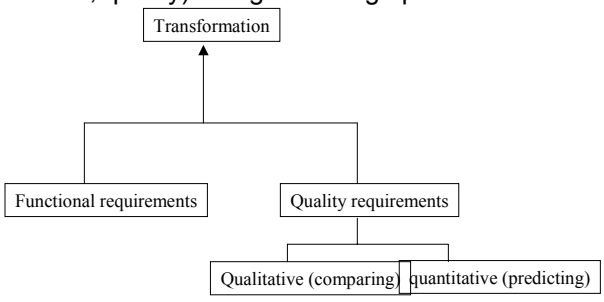
Introduction

- 80 % of the total system cost goes to maintenance.
- Conventional OOD methods lacking support for early evaluation of quality requirements.
- The need to provide accurate information and explicit guidelines to improve software development process.
  - Balance and address the quality attribute of the system
  - Support evolution.
  - Narrow the gap between the user and the developer.
  - cope with the dependencies between the different phases of the software life cycle (traceability).

Transformational Approach

- Design Transformation : is a typical multiple objective design activity where the software engineer has to balance the various requirements (Functional, quality)during the design process.

- Model's structure changes
- Affect quality factors, but not application domain functionality



Software Quality

Software Quality is conformance to

- explicitly stated *functional and performance requirements* ,
- explicitly documented *development standards* ,
- *implicit characteristics* that are expected of all professionally developed software.

Problems:

- Software specifications are usually *incomplete* and often *inconsistent*
- There is tension between:
  - *customer* quality requirements (efficiency,reliability,etc.)
  - *developer* quality requirements (maintainability,reusability,etc.)
- Some quality requirements are hard to specify in an unambiguous way
  - *directly* measurable qualities (e.g.,errors/KLOC),
  - *indirectly* measurable qualities (e.g.,usability).

Quality management is not just about reducing defects!

Software Metrics

Why Metrics?

- *When you can measure what you are speaking about and express it in numbers,you know something about it;but when you cannot measure,when you cannot express it in numbers,your knowledge is of a meager and unsatisfactory kind:it may be the beginning of knowledge,but you have scarcely,in your thoughts,advanced to the stage of science.— Lord Kelvin*

Measurement quantifies concepts

- understand,control and improve

Why Software Metrics?

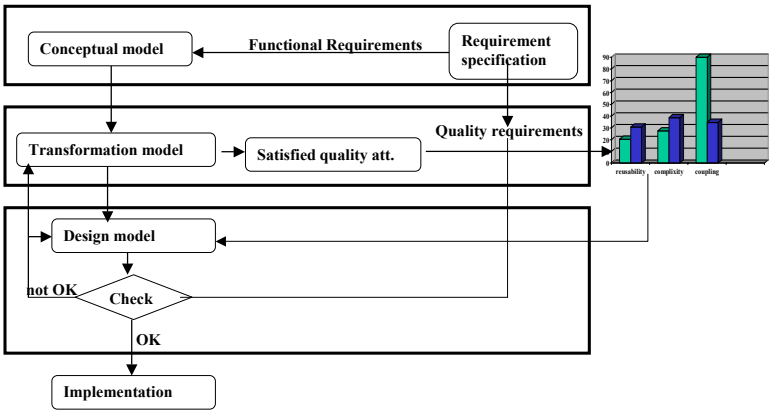
Effort (and Cost)Estimation

- Measure early in the life-cycle to deduce later production efforts

Quality Assessment and Improvement

- Control software quality attributes during development
- Compare (and improve)software production processes
- Remember *Quality Assumptions*

Quality Assessment Process



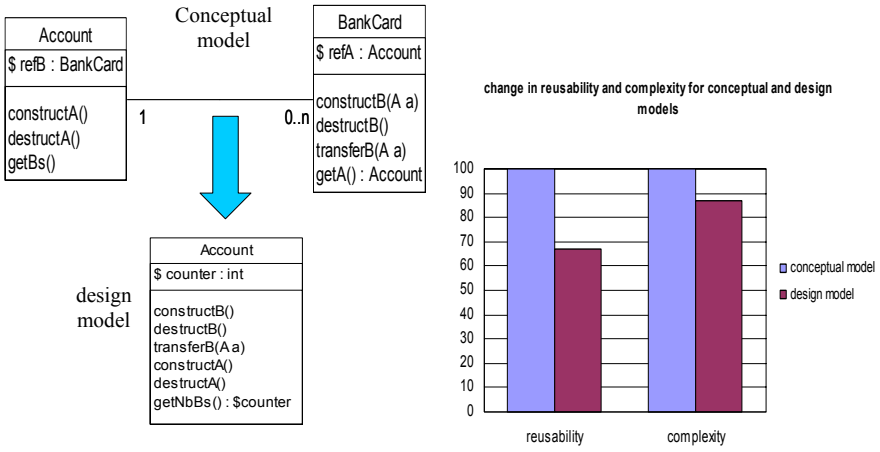
Sample Metrics

- Quality Metrics
  - Coupling , Cohesion & Size Metrics
    - LOR (Lack of Reuse)
      - LOR = total number of method pairs shared by sets of instance variables divided by the maximum possible pairs of methods.
    - CF (Coupling Factor)
      - CF = number of other class to which given class is coupled Interpret as “number of other classes a class requires to compile
    - WMC (Weighted Methods per Class)
      - WMC = #Σ of method size
- Beware

Researches disagree whether coupling/cohesion/Size methods are valid

- classes that are observed to be cohesive may have a high LOCM value
  - due to accessor method
- Classes that are not much coupled may have high CF value
  - no distinction between data, method or inheritance coupling

Example



Summary

- Transformation-based approach offers the following
  - Line between analysis & design is more accurate
  - Less manual optimization is required
  - Traceability : Visible correspondence between the two phases
  - Considerable maintenance
  - Explicitly address and balance the quality attributes of a system
  - Early prediction of the quality of the software system

Validating Quality Requirements of Object Oriented Design

Jamal Said, Eric Steegmans
Department of Computer Science, K.U.Leuven
Celestijnenlaan 200A
B-3001 Leuven
Belgium
jamal.said@cs.kuleuven.ac.be

Abstract

Conventional object-oriented design methods [1,2,3] tend to focus on achieving the required system functionality and pay only limited attention to quality requirements. So far the quality factors were measured after the system is put in operation. This requires software developer to redesign and re-implement the entire software system if the system fails to meet its quality requirements. The design methodology [4] developed in our group that transforms conceptual models to design models addresses implicitly the nonfunctional requirements. In the transformational approach we claim that each transformation performed automatically satisfies particular quality factors. In this paper we will formally validate those implicit satisfied-quality factors by measuring them with techniques available in the literature. By measuring the *relative effect* of the quality factors on each other we can get an overview of the whole system's quality which of course can give the user an early prediction about how the system supposed to operate after implementation. Monitoring the quality of the software system will be also supported by visual statistical graphs and figures that make the user easily grasp the software quality variation.

1. Introduction

Conventional object-oriented design methods [1,2,3] tend to focus on achieving the required system functionality and pay only limited attention to quality requirements. Implicitly, these methods assume that using an object-oriented modeling approach will automatically leads to reusable and flexible systems, thus one could state that the maintainability and reusability requirements are incorporated to some extent. However, only these quality factors are considered rather implicitly.
So far the quality factors were measured after the system is put in operation. This requires software developers to redesign and re-implement the entire software system if the system fails to meet its quality requirements. It requires changes to working code that can introduce subtle bugs, it also set you back days, even weeks if not done properly, or practiced informally or ad hoc and increase the cost of the software systems due to maintenance. Testing the quality of the software system after being built will not consider the fact that the system might evolve and consequently needs to be maintained. Thus, two major factors are lacking in traditional design methodologies, satisfying the quality factors and validating them at the design level. The design methodology we developed [4] strives to deal with this two factors which in turn minimize the risk of building

systems that fails to meet its quality requirements and consequently decrease the cost of system development.
Our methodology [4] basically transforms conceptual models into design models. It addresses both the functional requirements as well as the nonfunctional requirements simultaneously. This approach of object oriented software design transforms fragments (patterns) of the conceptual model (i.e. classes and relations) to produce a good quality design model. Each one of those transformations satisfies implicitly particular software quality factors. The transformational approach is meant to achieve a good design model with high quality. Consequently after the transformation process is performed, the obtained design model should satisfy to a certain extent the quality requirements.
In this paper we will validate the implicit quality factors explicitly by implementing formal methods that are used to validate and measure quality factors at the level of design. Implementing such methods, however, requires a precise, formal definition of the quality metrics. The definition of metrics shall be precise and unambiguous, the metrics should be based on design artifacts that are typically constructed in the design phase. Furthermore the model should be suited for automatic measurement as far as possible. This measurement should also be possible if the design is not complete. Such a formal model is presented in section 2.1. In section 2.2 an example is given to illustrate its use during design.
However, typical designs are so big that quality assurance becomes a time consuming activity. Therefore, it is more efficient to use an automated tool for design evaluation. Even though such a tool can never replace a human expert, it can help him/her to identify components of a design that are potentially troublesome. The tool can also help to compare design alternatives by evaluating each alternative and compare design alternatives by evaluating each alternative and compare them.

2. Assessing quality factors

At the level of the conceptual model some software quality factors such as flexibility and reusability are more feasible than performance, robustness, or maintenance. However, as the software development process evolve the latter quality factors starts to gain considerable attention by the developer. Therefore, we strive in our approach to offer transformations that support most of quality factors to end up with a good quality design model. However, unlike functional requirements, *quality requirements can generally not be pinpointed to particular part of the application but are property of the application as a whole*. Therefore, each transformation we perform fulfills a particular quality factor while affecting other quality factors of the whole system. For example one of the transformations that we performed was integrating one class inside the other which helps performance while weakening flexibility. Therefore, after we finish the transformation process we need to formulate the design model that satisfies as a whole the nonfunctional requirements of the software system in hand. Our aim here is to give a quantitative prediction of the overall software quality at the early stages of the software development process by measuring the *relative effect* of each transformation on the whole system's quality. Thus, giving the user an indication about how the system supposed to operate after implementation. Monitoring the quality of the software system will be also

supported by visual statistical graphs and figures that make the user easily grasp the software quality variation.

One of the core features of our approach is that the quality factors of the system or the application are explicitly evaluated during transformation. Traditional software engineers implement the system and then measure the actual values for the quality system properties. The obvious disadvantage is that a large amount of resources has been put into building a system that doesn't fulfill its quality requirement and numerous examples exist to illustrate this point. Being able to estimate the quality factors of the system during early development stages is important to avoid such misshape [5]. However, for some reasons it is not possible to measure the quality of the final software system based on design transformation. Instead the goal is to evaluate the potential of the design transformations to reach the required level for its quality requirement. For example, some transformations, e.g. association, are less suitable for systems where performance is a major issue, even though the flexibility of this style is relatively high. Consequently, each transformation we perform satisfies some quality factors while effecting the others. In our approach we strive to optimize and balance the software quality by assessing those effects and compare them with the quality requirements specified by the user. In the next section we will address our approach for measuring and assessing quality requirements. The approach uses some empirical formulas that evaluate the quality of patterns before and after transformation.

2.1 Method

To assess quality factors we have to navigate through three processes mainly analysis, transformation and design. The approach is illustrated in Figure 1 below. In the figure, we can see three important phases. The first phase represents the conceptual model, which needs to be transformed (analysis phase). We assume that the conceptual model has a certain degree of quality, which will be improved after transformation. The second phase presents the transformation process. During this process we use the most widely referenced sets of OO software metrics which has been proposed by Chidamber and Kenerer [6] and was validated by Basili[12]. This metrics is a class-oriented metrics in which the authors have proposed six class-based design metrics for OO systems. Among those are the weighted methods per class (WMC), coupling between object classes (CBO), and lack of cohesion in methods (LOCM). After each transformation we perform, particular criteria are achieved while others not. Therefore, after each transformation we should measure the whole quality of the model to see the relative impact of that transformation on that model. This measurement is done in a dynamic way since as we have mentioned before that quality factors *are property of the application as a whole*. Therefore, measuring the quality of the classes being transformed goes together with measuring the relative effect of those transformed classes on the whole design model. To measure the whole change on the quality of the design model we use design model based metrics. In a detailed treatment of software metrics for OO systems, Whitmire [7] described nine distinct and measurable characteristics of OO design (e.g. size, complexity, coupling, sufficiency, completeness, cohesion, primitiveness, similarity, and volatility). Some of these characteristics will be used to compare the quality requirements proposed by the customer with the quality of the resulted design obtained by the

developer. If there still a big difference then the developer has two alternatives to narrow that difference. He/she could either change or iterate through the design model or could apply new transformation until the quality requirements are satisfied to an acceptable level.

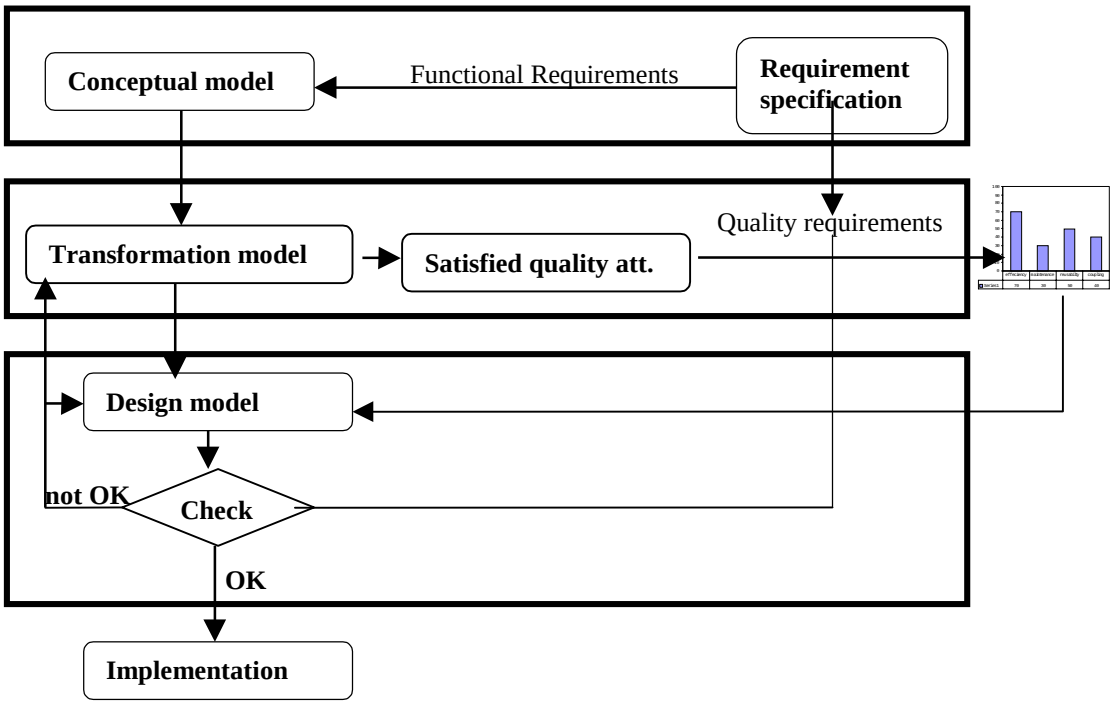


Figure 1: Outline of the software quality assessment methodology.

In section 2.2 we will explain the procedure of both class-based metrics and model based metrics. We will measure LOCM, WMC, and CBO for both the class and the model as well. In fact some of the empirical relations used in the class-based metrics will differ from the ones used in the model-based metrics. In section 2.3 we will workout an example -which represents a simple conceptual model consisting of two interrelated classes - to illustrate the validity of those empirical formulas.
This approach offers the developer an early prediction about the software system's quality which decrease the risk of building a system that fails to meet its quality requirements and consequently decrease the cost of system development. Furthermore, it evaluates the transformations we perform and it gives us an indication about the significance and the importance of the classes at the level of the analysis. Moreover, it gives the developers and idea of whether they are going on the right direction or not.

2.2 Procedure

An object oriented design can be conceptualized as a relational system, which is defined by Roberts as an ordered tuple consisting of a set of elements, a set of relations and a set of binary operations [8]. More specifically Chidamber and Kemerer (C& K) stated that,

an object oriented design, D, is conceptualized as a relational system consisting of object-elements (classes and objects), empirical relations and binary operations that can be performed on the object-elements. By starting with these definitions the mathematical role of metrics as a mapping (or transformation) can be formally outlined:

D (A, R1….Rn, O1…Om) Where
A is a set of object-elements
R1..Rn are empirical relations on object-elements.
O1 …Om are binary operations (transformations) on elements of A.

A useful way to understand empirical relations on a set of object-elements is to consider the measurement of their quality factors after transformation. A designer generally has some intuitive ideas about the complexity, flexibility, and other quality factors of different object-elements, as to which element is more complex than another or which ones are equally complex (C & K). For example we intuitively understand that a class that has many methods is generally more complex than one that has few methods. This intuitive idea is defined as a viewpoint. Fenton states that viewpoints characterize intuitive understanding and that viewpoints must be the logical starting point for the definition of metrics [9].

C & K in [6] start off giving empirical relation systems called viewpoints for the metrics proposed. Their viewpoints deal with the effects of each proposed metric on different (often external) factors, e.g. for weighed methods per class (WMC): maintainability (viewpoint 1) and reuse potential (viewpoint 3); for coupling between objects classes (CBO): reuse (viewpoint 1), maintainability (viewpoint 2), and testability (viewpoint 3); and for Lack of Cohesion in Methods (LCOM): encapsulation (viewpoint 1) and correctness (viewpoint 4). Discussing the effects of a metric on the other quality factors is certainly valuable in measuring the quality of the design model. In this paper we will use this technique to measure the quality factors of the transformed patterns of the conceptual model described in figure 2. In the next section we will measure LOR, CF and WMC for both the classes and the model before transformation (i.e. at the conceptual level) and after transformation (at the design level).

Class based metrics:

1- Lack of Cohesion in Methods (LOCM)

LOCM as defined by Chidamber and Kemerer [6]

LOCM = (total number of method pairs shared by sets of instance variables) subtracted by the number of method pairs that doesn't share instance variables.

However, we will not follow this definition precisely instead we will modify it and call it Lack of Reuse (LOR) instead of LOCM. The reason for this modification is to obtain a value out of one to show easily the relative change on the quality factors. Therefore, (LOR) = total number of method pairs shared by sets of instance variables divided by the maximum possible pairs of methods.

2- Coupling Factor (CF)

Another quality factor that might change due to integration is coupling. The Coupling Factor (CF) metric was proposed as a measure of coupling between classes [10], excluding coupling due to inheritance. CF has been defined formally [11] as:

$$\frac{\sum_{i=1}^{TC} [\sum_{j=1}^{TC} is_client(C_i, C_j)]}{TC^2 - TC}$$

where

$$is_client(C_i, C_j) = \begin{cases} 1 & iff\ C_c \Rightarrow C_s \wedge C_c \neq C_s \\ 0 & otherwise \end{cases}$$

and $C_c \Rightarrow Cs$ represents the relationship between a client class, C_c, and a supplier class, C, and *TC* is the total number of classes .

CF is calculated by considering all possible pairwise sets of classes, and asking whether the classes in the pair are related, either by message passing or by semantic association links (reference by one class to an attribute or by method of another class). Thus CF, is a direct measure of the size of a relationship between classes in a system. However, CF differs if we consider the association between the involved classes to unidirectional and bidirectional, but for simplicity we will assume the association between the classes at the conceptual level to be bidirectional. At the level of design we will not have the problem of defining the the type of navigation for the since at that point in time we would have already decided which type of association we will take. Note also the formula shown above is valid for both the model and the class base metrics.

3 - Weighted Methods per Class (WMC)

The third criterion that we will measure here in this example is complexity. The metrics used to measure. WMC based on [6] measures the complexity of an individual class, if we consider all methods of a class to be equally complex, then WMC is simply the number of methods defined in each class [12]. This metrics could be used to show the improvement on the design model before or after transformation. To calculate the relative change that occurred on the class itself or on the model as whole we measure it by subtracting the WMC after transformation from WMC before transformation and divide the whole result by total number of methods.

Method based matrices

1- Lack of Reuse (LOR)

As we mentioned before the empirical relation of the class based metrics would differ from those for the model based ones. For example to measure the LOR for the entire model we need to modify the relation of the class-based metrics. To measure the LOR we need to look for a weighted factor that plays a major role in the model. Our studies show that number of methods in each class plays that major role in that aspect. Therefore, to calculate LOR for the entire model we sum the multiplication of LOR of each class with its number of methods and divide it by the total number of methods for the whole model.

$$As\ a\ result\ LOR\ model = \frac{\sum_{1}^{N} LOR_N \times M_N}{Total\ number\quad of\ methods\quad in\ model}$$

Where

N = number of classes.

M = number of methods

2- WMC

For WMC the empirical formula for model based will be the same one used for the class based

3- CF

As we mentioned before the formula shown above is valid for both class based and method-based metrics.

2.3 Example

This section presents an example that uses the approach described above to measure the change on the quality of a simple conceptual model consisting of two interrelated classes illustrated in figure 2. These two classes have a relation between each other. This relation expresses *existential dependency* expressing that objects cannot exist without being related to objects of the classes participating in the given relation [5]

Figure 2: Structure of two classes involving unary relations.

In our example a relation will be introduced refining the objects of the class of bankcard in order to express that no bankcard can exist without being related to an account. Such kind of relations is called unary relations. They involve one participating class and one refined class.

In this example we will measure the quality of integration transformation performed on the model described in figure 2. Figure 3, shows the two classes after integration (combination) where class BankCard is integrated in the class Account.

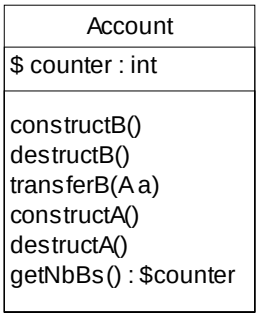


Figure 3: Implementing integration transformation

Metrics of conceptual model:

1- LOR:

1- For the class Account shown in Figure 2 the total number of method pairs shared by sets of instance variables = 3, and maximum possible pairs = n (n-1)/2 which is equal to 3. Therefore LOR of class Account before transformation = 3/3 = 1.
2- For class BankCard the total number of methods pairs shared by set of instance variables = 6, and the maximum possible pairs = 6. Therefore, LOR of class BanckCard = 1.
3- For the conceptual model LOR = 1 as well.

2- CF:

For bi-directional association we have CF of 1

3- WMC:

1- For the class Account WMC = 3
2- For class BankCard WMC = 4
3- For the model = 7

Metrics of the design model:

1- LOR:

1- For the class account LOR of Account class after transformation = (10/15 = 0.67), hence, LOR is decreased by 33 %. This 33 % drop in reuse is because of the increase in number of methods in class Account, which in turn make the class Account application specific and more complex.
2- For the model which is represented by the class Account in this example = 0.67 as well.

2- CF

1. It easily seen that integration will make CF = 0 for both the class and the model. This decrease in coupling will limit the number of message passing between classes, which in turn increases efficiency.

3- WMC

1- MWC for the class of Account will be 6. However, we have said that this criterion could be used to measure the relative change on the model before and after transformation. For example for Class Account it will be (6-3)/7= 0.42. This shows an increase of 40 % in complexity for the class of Account.

For the whole model WMC = 6. Therefore, if we compare WMC for conceptual model where WMC was 7, and WMC for the design model where WMC is 6 we can see a slight drop in the complexity of approximately 16 %.

2.4 Results

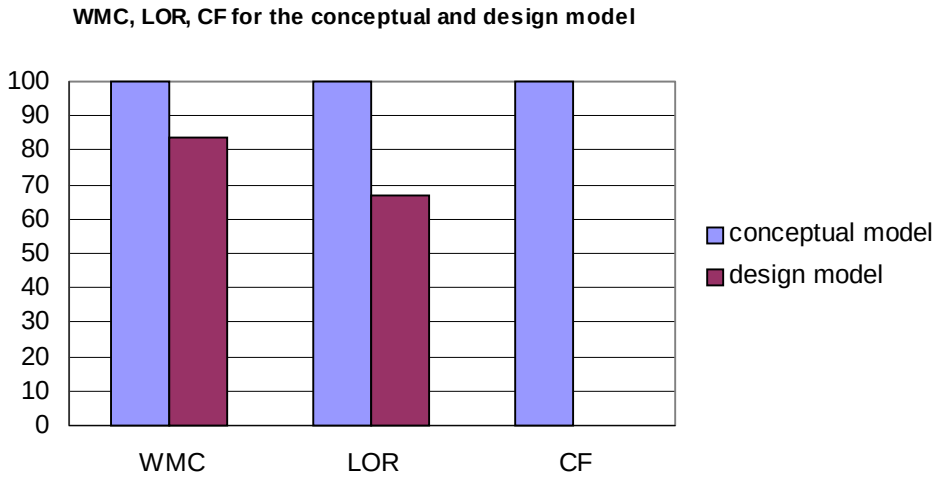


Figure 4a. Distribution of the analyzed OO metrics. The X-axes represents the type of the metrics. The Y-axes represents the value of the metrics.

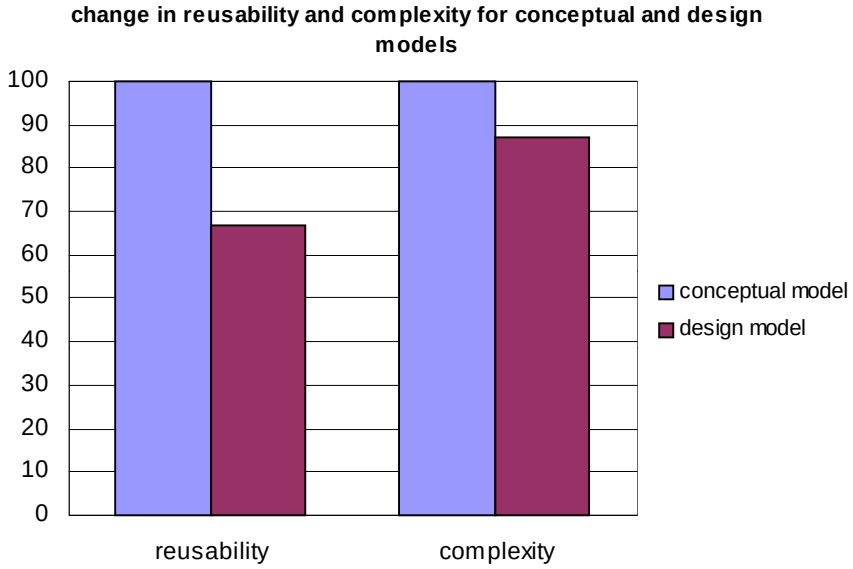


Figure 4b. The relative change on quality factors. The X-axes represents the type of quality factor. The Y-axes represents value of quality factors.

3. Conclusion

In this paper we have shown that the transformational approach offers the ability of measuring the quality factors of the design model obtained during development. This approach offers significant aid for monitoring, controlling, and improving software development processes. For example, the explicit change in some of the quality factors (e.g. reusability and complicity) was easily seen by calculating them and presenting them with visual statistical graphs and figures. Furthermore, our approach certifies that dealing with functional and non-functional requirements at the early stages of the software lifecycle will supports evolution and decreases the development cost which will lead to more optimized and maintained design model.

4. References

[1] Booch, G. '*Object-Oriented Analysis and Design with Application*, 1994.

[2] .Jacobson, I.,Christerson, M., Jonsson, P. and Overgaard, G., *Object-Oriented software engineering. A use case approach*. Addison-Wesley, 1992.

[3] Rumbaugh, J., Balaha, M., Premerlani, W., Eddy, F.and Lorensen, W., *Object-Oriented modleing and design*. Prentice hall, 1991.

[4] *J. Said, E. Steegmans, "Transformation of Unary Relations", Proceeding ICSSEA 2000, 13th International Conference on Software & System Engineering and their Applications, Paris – France 2000, volume 4*

[5] Jan Bosch '*Design and Use of Software Architectures ', Addison Wesley'*, 2000.

[6] Chidamber, S.R. and C.F. Kemerer, "A Metrics Suit for Object- Oriented Design", *IEEE Trans. Software Engineering*, vol.20, no. 6, June 1994, pp. 476-493.

[7] Whitmire, S., "Object Oriented Design Measurement, Wiley, 1997.

[8] F. Roberts, Encyclopedia of mathmetics and its applications. Reading, MA: Addison-Wesley, 1979.

[9] N. E. Feton, Software Metrics, A Rigourous Approach. New York: Chapman and Hall, 1991.

[10] F. Brito eAbreu, M. Goulao, and R. Estevers, "Toward the design Quality Evaluation of OO Software Systems, "*Pro. Fifth Int'l conf. Software Quality*, 1995.

[11] F. Brito eAbreu, and W. Melo, "Evaluating the impact of OO Design on software Quality, "*Proc. Third Int'l Software Metrics Symp.*, Berlin 1996.

[12] R. Basili, C. Briand, L. Melo, A validation of OO Design Metrics as Quality Indicators. *IEEE Trans. Software Engineering*, vol.22, no. 10, October 1996, pp. 751-761.

QWE2002 Paper 2T

Mr. Allan Beumer
(CMG)

Performance Testing in Internet Time

Key Points

- Performance.
- Testing.
- Internet.

Presentation Abstract

Testing the architecture of an Internet solution is different compared to a client/server solution. Off course, the main goal of testing stays the same: to retrieve information about the quality of the system. However, testing an Internet architecture definitely requires a different approach. This paper informs about the followed approach for testing an Internet portal architecture from a large international bank.

About the Author

Allan Beumer has been working for CMG since 1997 as a technical test consultant for the division Test Management and Consultancy. He is an expert in the field of automated test tools both for client/server as Internet applications. As the main tutor of the TestFrame Navigation Course, he develops and gives courses about using test tools like WinRunner, QARun and Rational Robot for clients and CMG consultants. Besides this, he is also involved in CMG's quality assurance as an auditor and team quality agent.

Performance Testing in Internet Time

Best practices for testing Internet architectures

17 January 2002

allan.beumer@cmg.nl - www.testframe.com

Agenda

- Introduction
- Importance performance testing
- Objectives
- Approach
- Tools
- Organization
- Test Environment
- Summary

TESTFRAME

Importance Performance Testing (1)

- Bad performance will drive off users
- User experience: based on response times and availability
- Client configuration unknown
- No influence on ISP performance
- High visibility of failures

TESTFRAME

Importance Performance Testing (2)

- Complex, multi-tier infrastructure
- Various hard- and software solutions
- Every infrastructure component (combination) possible bottleneck

TESTFRAME

Objectives (1)

- Performance information from end-user and infrastructure perspective
- Pinpoint infrastructure bottlenecks
- Infrastructure scalability information
- Improve performance based on test results
- Structured and repeatable test set

TESTFRAME

Objectives (2)

- Performance testing should lead to optimal component configuration
  - drastically improves performance
  - makes buying more hardware unnecessary

TESTFRAME

Approach

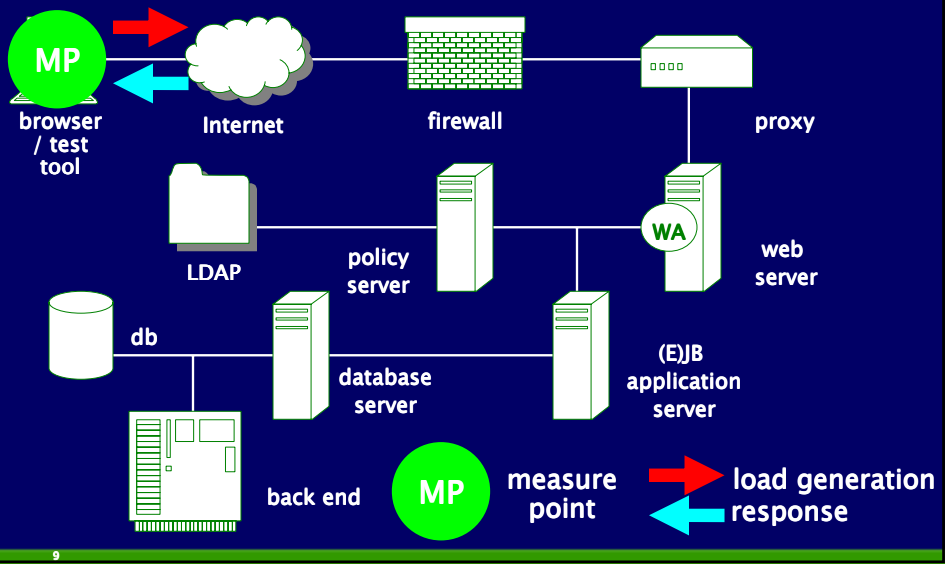
- Phase 1 (3 weeks):
  - Quickly gather information about the general performance of the infrastructure (end-2-end)
  - Quickly gather information about the maximum number of users the infrastructure can serve (scalability)
  - Pinpoint infrastructure bottlenecks
  - Create a baseline for future tests
- Phase 2 (3 months):
  - Detailed and complete performance test for every component

TESTFRAME

Approach: Phase 1 step-by-step

- Create and discuss test plan including performance criteria
- Describe a subset of user transactions
- Create transaction-mix
- Set-up test environment
- Develop stress generation scripts
- Set-up measure-points
- Run tests
- Analyse results
- Modify components

Approach: Phase 1 – End-2-End



Approach: Phase 1 – Overview

- Only end-user experience (response time) is measured
- Infrastructure is a 'black box'
- Useful to create a baseline
- Do not spend too much time on it
- Use results to convince stake-holders of the importance of performance testing

TESTFRAME

Approach: Phase 1 – Pitfalls (1)

- Quick end-2-end test as described is not sufficient
- Only response times are acquired
- No information about how these response times are build up

TESTFRAME

Approach: Phase 1 – Pitfalls (2)

- Every component has to be stressed & measured individually to pinpoint the bottleneck
- Some components require the development of specific performance test programs to
  - generate load
  - measure results
- Component's maximum performance?

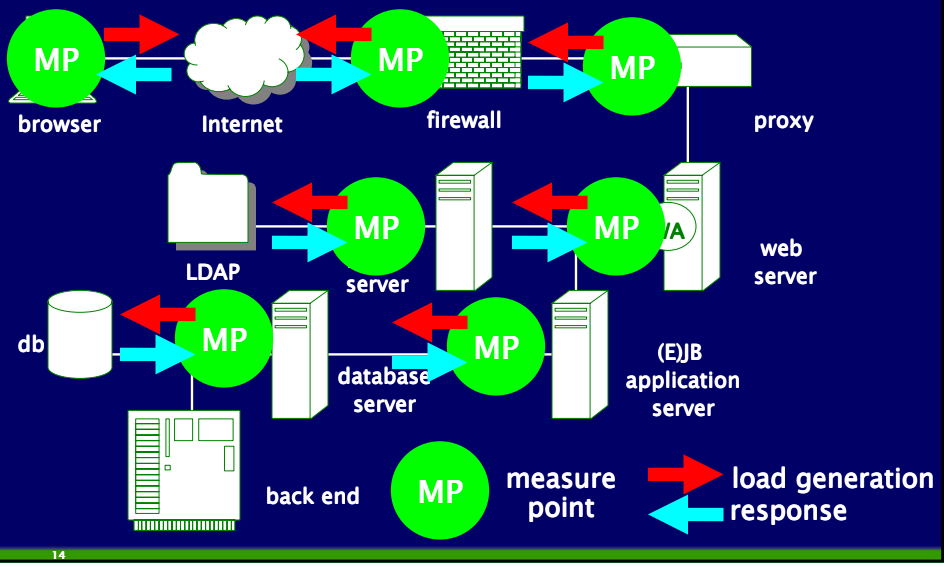
TESTFRAME

Approach: Phase 2 step-by-step

- Describe all user transactions
- Create transaction-mix
- Define number of concurrent users increase
- Set-up / modify test environment
- Develop stress generation scripts
- Develop test / monitoring programs
- Set-up measure-points
- Run tests
- Analyse results
- Modify components

TESTFRAME

Approach: Phase 2 – separate components



TESTFRAME

Approach: Phase 2 – Overview

- Stress & monitor every infrastructure component from back to front
- Compare response time of every component with complete end-2-end response time
- This way the bottleneck can be pinpointed
- Logging increases CPU/memory usage: repeat tests with logging off

TESTFRAME

Tools

- Client
  - Load / stress tools (LoadRunner, QALoad, SiteLoad,…)
- Server
  - Log file (analyzers)
  - Reporting facilities (resource usage)
  - Test tool agent information
  - Self-developed stress generators + monitors
- Network
  - Sniffers
  - Performance Monitor (NT), ps/top (UNIX),…
- Database
  - SQL queries
  - Log files
  - Profilers

Organization

- Performance Testing is not just a matter of the Test Department. Co-operation with the complete IT organization is a necessity
- Make sure experts are available for every component / tier to:
  - Prepare the test environment
  - Create test programs
  - Analyze test results / log files
  - Answer generic questions
- Solution: a “Task Force”…

Organization – Task Force Members

- System Developer (JB / EJB / JavaScript)
- DBA’s (database)
- Network operations (network)
- Infrastructure / architect (servers / proxies / firewalls)
- Security (SSL / authorization)
- Testers

- Arranging the availability of these experts can take more time than creating the performance test

TESTFRAME

Summary

- Performance testing is difficult
- Stop buying more hardware…
- End-2-end testing is not enough
- Test on both sides of the firewall
- Setup and maintain an isolated test environment
- Know your infrastructure
- Use tools to generate load and measure response times
- Developing a specific test program for every component is worth the effort
- Establish a Task Force

TESTFRAME

Some final tips about Performance Testing…

- Tip 1: Do it
- Tip 2 (for experts only): Do it now

QWE2002 Paper 3T

Mr. Oliver Niese, Tiziana Margaria & Prof. Bernhard Steffen
(METAFrame Technologies GmbH)

Automated Functional Testing of Web-Based Applications

Key Points

- Automated Functional Testing
- Web-based Testing.
- Test Tools.

Presentation Abstract

Web-based applications are very complex applications, as they are client/server based and run on heterogeneous platforms. It is increasingly unrealistic to restrict the consideration to single units, e.g. the client side only, since complex subsystems affect each other. Therefore scalable, integrated test methodologies are needed.

About the Author

Oliver Niese Born 31.8.70, received the Diplom in Computer Science from the University of Dortmund (Germany) in February 1999. He is currently a Ph.D. candidate at the University of Dortmund and is working at METAFrame Technologies GmbH as a consultant. He has experiences in large OO-projects with UML and Java as well as in formal verification methods. Within the ITE project, he is the technical leader at METAFrame.

Tiziana Margaria Born 14.10.64, she holds a Laurea in Ingegneria Elettronica from the Politecnico di Torino (Italy) (1988) and a Ph.D. in Computer and Systems Engineering from the same institution (1993). She held assistant positions in the Computer Science at the Technical University if Aachen and at the University of Passau. 1999 she was Visiting Professor at the Department of Computer Systems in Uppsala (S). Since June 1998 she is Senior Researcher at the University of Dortmund (D). She co-founded METAFrame Technologies GmbH in 1997, and serves since then as the company's CEO. Within the ITE project, Dr. Margaria in particular consults for formal verification methods.

Bernhard Steffen Born 31.5.58, he holds a Diplom in Mathematics (1983) and a Ph.D. in Computer Science (1987) from the University of Kiel (Germany). After holding research positions at the University of Edinburgh, (UK) (1988-89) and at the University of Aarhus, DK (1990) he was tenured Associate Professor at the Technical University of Aachen, D (1990-93) and held the Chair of Programming Systems at the University of Passau, D (1993-97). Since August 1997 he holds the Chair Programming Systems at the University of Dortmund, D, where he heads the METAFrame Project, concerning the development of a flexible environment for reliable definition and custom configuration of complex workflows. He co-founded METAFrame Technologies GmbH in 1997.

METAFRAME
The Personalized Online Service House

Automated Functional Testing of Web-based Applications

O. Niese[1], T. Margaria[1], B. Steffen[2]

[1] METAFrame Technologies GmbH, Dortmund, Germany
[2] University of Dortmund, Dortmund, Germany

Overview

- Problem
  - Automated functional testing on system level
  - Application area: Web-based Applications
- Concept
  - Component-based test design and test execution accompanied by formal methods
- Implementation
  - *Integrated Test Environment*
- Conclusion

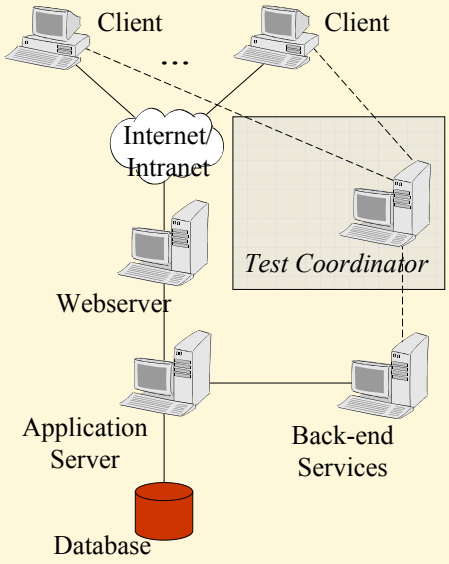
Problem

- System-level testing
  - ***Objective***
    - testing the aggregation of different sub-systems
  - ***Characteristics***
    - high complexity
    - employment of several different test tools
    - higher costs of error detection compared to previous test phases

Web-based applications

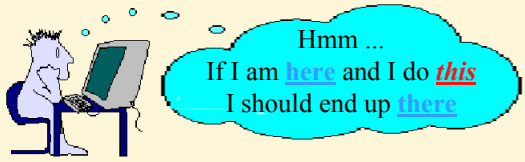
- Highly interactive
- Dynamic content
- Distributed architecture
- Concurrent access
- ...

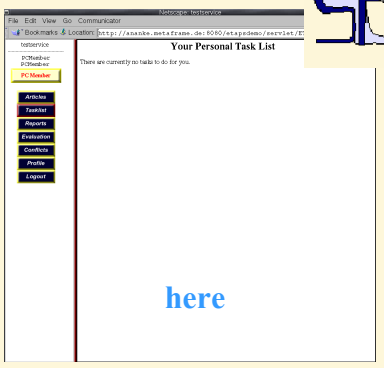
Concrete Test Setting



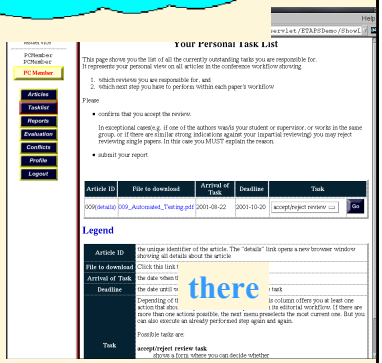
Concept - Coarse Granular Level

- Establishment of a coarse-granular level





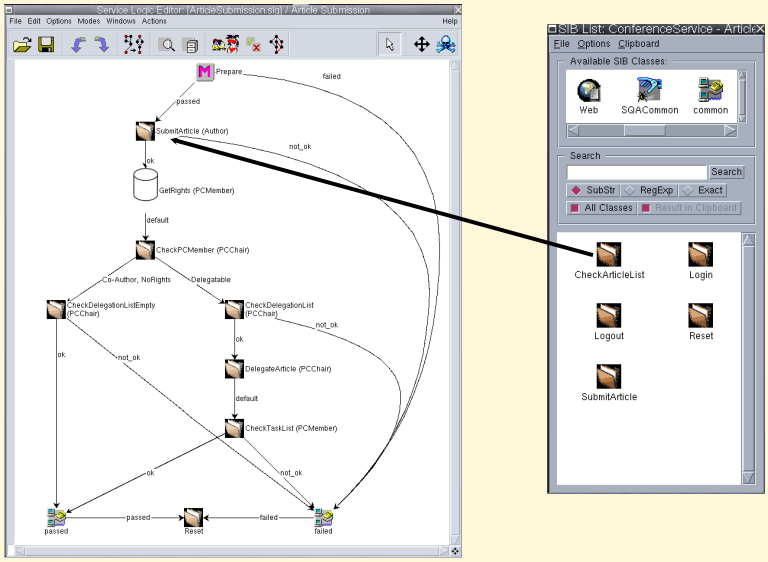
this
(“submit article")



Concept - Coarse Granular Level

- The coarse-granular level
  - Provide basic functionalities
    - generic input stimuli (e.g. "press button")
    - output checkpoints (e.g. "is button enabled?")
  - Design of test cases
    combine basic functionalities
    - by means of a graphical notation
    - according to the control flow

Concept - Coarse Granular Level

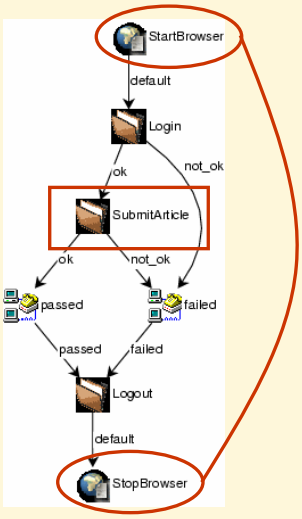


Concept - Aspect Oriented Level

- Establishment of an aspect-oriented level

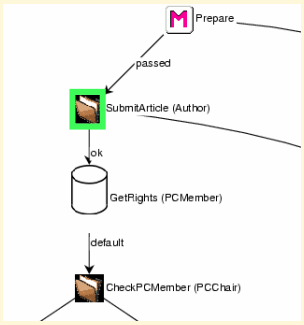
Concept - Aspect Oriented Level

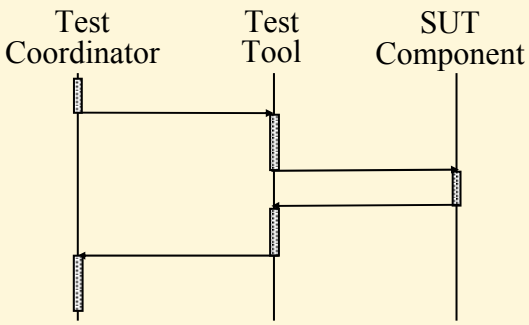
- The aspect-oriented level
  - Local constraints
  - Global constraints



Concept

- Test Execution
  - Control flow driven execution of test blocks

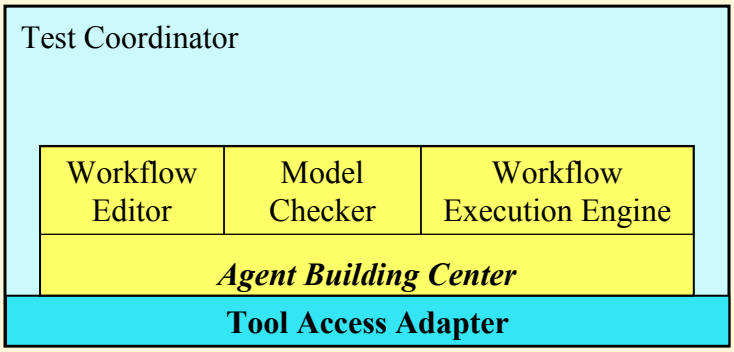




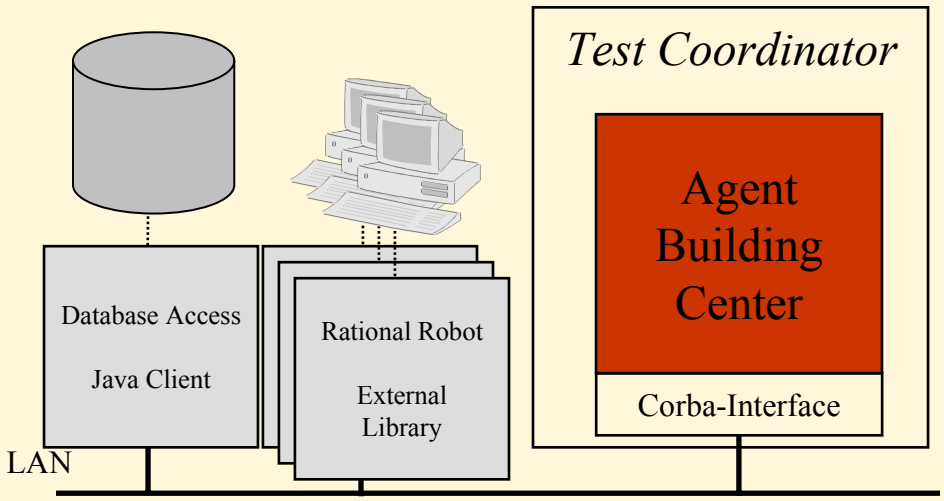
  - Preparation of an execution protocol

Implementation

- Implementation fundamentals
  Agent Building Center (ABC)



Implementation



Conclusion

- Web service can be developed with ABC
- Benefits:
  - Constraints of the web-service design can be reused
  - Automatic generation of test cases is under development

Conclusion

- Presentation of an integrated test environment for web-based applications
- Approach is feasible
- Open, flexible approach

Automated Functional Testing of Web-based Applications

Oliver Niese[1], Tiziana Margaria [1,2], Bernhard Steffen[2]

[1] METAFrame Technologies GmbH, Borussiastr. 112, 44149 Dortmund, Germany, {oniese,tmargaria}@metaframe.de

[2]Chair of Programming Systems, University of Dortmund, Baroper Str. 301, 44221 Dortmund (Germany), {tiziana,steffen}@guiness.cs.uni-dortmund.de

Keywords:
Automated Functional Testing, Testing of Web-based Applications, Test-Tools, System-Level Test Environments

Abstract

Modern *web-based applications* are multitiered, distributed applications that typically run on heterogeneous platforms. Their correct operation depends increasingly on the interoperability of single software modules, rather than on their intrinsic algorithmic correctness. We present a scalable, integrated test methodology capable of granting adequate coverage of functional testing, yet still easy to use. It bases on our previous work on system level test of Computer Telephony Integrated applications, where our coordination-based coarse grain approach has been applied very successfully. The practicability of the approach is illustrated by testing complex role based functional features of our *Online Conference Service*, an advanced online management system for the organization of scientific conferences.

1 Dealing with Web-Based Applications

Modern *web-based applications* are very complex multitiered, distributed applications that typically run on heterogeneous platforms. Their correct operation depends in steadily growing measure on the interoperability of single software modules, rather than on their intrinsic algorithmic correctness. This has a huge impact on the test and validation requirements: it is increasingly unrealistic to restrict the consideration to single units,

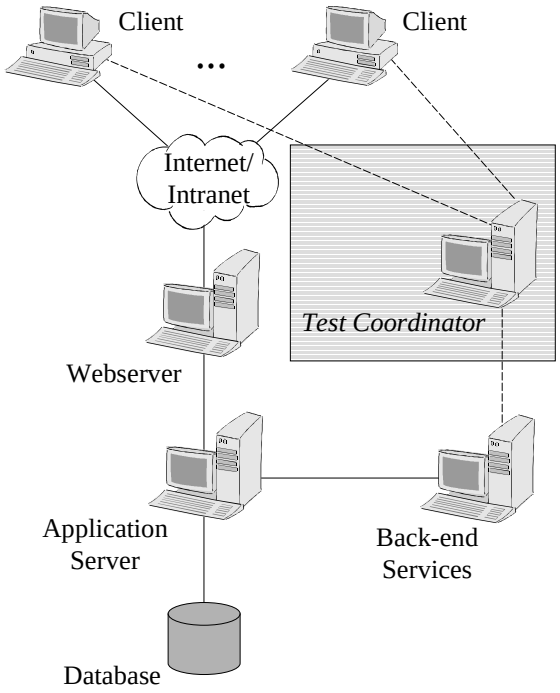


Figure 1: Overview of the Test Setting

e.g. the client side only, since complex subsystems affect each other. Instead, scalable, integrated test methodologies are needed that can handle the application in a whole.

Figure 1 sketches a typical architecture of a web-based application. The browser, located on a client, plays the role of the "classical" application GUI and interacts with a *Webserver*. The Webserver itself communicates with an *Application Server* which builds dynamically the requested web pages through an interaction with a *database* and (several) *Back-end services*. Often staging servers, firewall solutions, load balancing and redundant architectures increase the speed and availability of the offered application, and increase the complexity of the considered test setting. So test methods are needed that can deal with this kind of complex distributed system. Due to the architecture of such a system it is obviously not sufficient to test the user interface only but it is of central importance to consider the server side as well (i.e. the web-server, application-server and the back-end services). This task exceeds today's commercial internet-testing tools, as marketed e.g. by Compuware [1] or Radview [9]. An adequate approach should on the one hand allow an easy, graphical design of test cases and on the other hand the execution of these complex test cases through a coordination of different, heterogeneous test tools. These test tools locally monitor and steer the behaviour of the software on the different clients/servers, which are seen as units under test.

Depending on the considered application domain, different aspects of an application's functionality dominate the test purposes:

E-Commerce When testing E-Commerce applications (e.g. Web-shops) it is important to consider simple functional requirements like *a user can check out the shopping cart after buying an item* or security/administrative requirements like *a user can be logged in only once*. But also more complex tests must be performed, which take the whole distributed configuration of the service into account e.g. *check the validity of a credit card*. Here it is not sufficient anymore to perform tests that consider the client side only, since e.g. a faulty implementation of a back-end service may accept all credit cards without checking them correctly.

(Safety critical) Web-services A good example for this special kind of Web-services is e.g. *Online/Internet-Banking*. It is of crucial importance that on the one hand the user-interface reacts as prescribed and that on the other hand the back-end services are interacting correctly with the clients, as in this scenario there are normally well established back-end services or legacy systems in use. In this special field of applications every fault (either functional errors or a system crash) can be very expensive in terms of indemnification, and, more importantly, of loss of trust.

We propose an integrated test environment for web-based applications capable of addressing the distributed application in a whole and covering mostly the global functional aspects. To test these kind of complex systems, we add a *Test Coordination* layer driving the generation, execution, evaluation, and management of the system-level tests in the highly heterogeneous landscape, cf. figure 1. The coordination layer introduces the required flexibilization of the overall architecture of the test environment: it is a modular and open environment, so that diverse tools and units under test can be added at need.

In the following we first give a rough overview of the test environment in Section 2. Afterwards we present in Section 3 our approach along a real life example: the test of a complex online conference service. Finally in Section 4 we draw some conclusions.

2 An Integrated Test Environment for the Web

Complex subsystems affect each other in a variety of complex ways, so mastering today's testing scenarios for complex systems demands for an integrated, open and flexible approach to support the management of the overall test process, i.e. specification of tests, execution of tests and analysis of test results.

To handle the structural complexity, our approach offers a coarse grained testing environment where test cases can be easily composed, test suites can be configured and initialized, critical consistency requirements including version compatibility and frame conditions for executability are easily *formulated*. The consistency of test cases is fully automatically enforced via lightweight formal methods.

The heart of the environment is the Test Coordinator tool, built on top of METAFrame's *Agent Building Center* [10], which is a generic and flexible workflow management system.

In this application, we view system-level test cases of web-based application scenarios as executable workflows within the integrated test environment (which plays the role of an extended runtime environment). Building on the ABC's capabilities, test sequences can be graphically designed from palettes of subsystem-specific, generic test components, cf. figure 1(right). While the test components are developed by experts of the relative system, the test sequences (the actual workflows) can be designed graphically, at an intuitive level by application experts. A reporting tool concentrates the information coming from the observation points during execution of the test into reports.

Through a CORBA/RMI-based implementation of the communication layer we are able to address and encapsulate a wide range of commercial test tools: this increases over time the reach and the capabilities of the resulting environment. This way the *Test Coordinator* executes integrated test cases by controlling several test tools, each managing its own subsystem. The extensibility of the environment by additional test tools is the key of the approach.

So far the ITE has been successfully applied to the system level testing of complex system solutions like *Web-based applications* or *CTI-Solutions* [5, 6].

3 Case Study: Testing the Online Conference Service

We demonstrate the key features of the *ITE* on a real life example: the test of the Online Conference Service [2], a complex web-service that supports online the management of the scientific program of professional conferences. It proactively helps authors, Program Committee chairs, Program Committee members, and reviewers to cooperate efficiently during their collaborative handling of the composition of a conference program. The application provides a timely, transparent, and secure handling of the papers and of the related submission, review, report and decision management process. The online service is a powerful application, including in particular a role based access and rights management feature that is reconfigurable online, which accounts for a high flexibility and administration comfort, but which is responsible for potentially disruptive behaviour in connection with sensitive information. Because of the complexity of the underlying workflows (and of course therefore the complexity of the application itself) intensive testing is clearly unavoidable.

The test process supported by the *ITE* is organized in the following main steps:

1. Identification of generic and reusable test blocks,
2. Graphical design of test cases,
3. Verification of the test cases against predefined properties, and finally
4. Execution of the test cases and reporting.

3.1 Identification of Generic and Reusable Test Blocks

The first task when testing the conference service is to identify generic test actions which allow on the one hand to control the service via a browser and on the other hand to validate certain properties of the service state (e.g. via checking the current state of the GUI or via checking the entries of an underlying database). For each action a test block is prepared: a name and a class characterizing the block are specified and a set of formal parameters is defined to enable a more general usage of the block. In this way, for the *Web-based application* to be tested a library of test blocks has been incrementally defined. It includes test blocks representing and implementing, e.g.

Common actions for the initialization of test tools, system components, test cases and general reporting functions,

Common internet related actions e.g. starting/initialization of a browser, following a link,

Application-specific actions: these are miscellaneous actions for operating a specific application via its graphical user interface, e.g., log-on/log-off of a user, or checking labels of GUI-elements.

The library of test blocks grows dynamically whenever new actions are made available.

3.2 Graphical Design of Test Cases

The design of test cases consists in the behaviour-oriented combination of test blocks. This combination is done graphically, i.e., icons representing test blocks are graphically stuck together to yield test graph structures that embody the test behaviour in terms of control.

The test case shown in figure 2 tests wether an article, submitted by an author, is delegated correctly to a member of the program committee. From the submission to the start of the review process of an article at least three different roles are involved: the *Author* himself, the Conference Chair (*PCChair*), who must delegate the article to a member of the program committee, and finally the members of the program committee (*PCMember*), who are responsible for the reviews. So we need at least three concrete users to run this test, whereby each user is associated with a distinct role. At the beginning of the test case for every user a browser must be started to perform the required tasks. The single steps to be executed are here encapsulated in a *macro*, cf. Figure 2(left). Macros can be used as reusable atomic test actions in the design of new test cases. The submission workflow proceeds then as follows:

1. The *Author* submits an article.

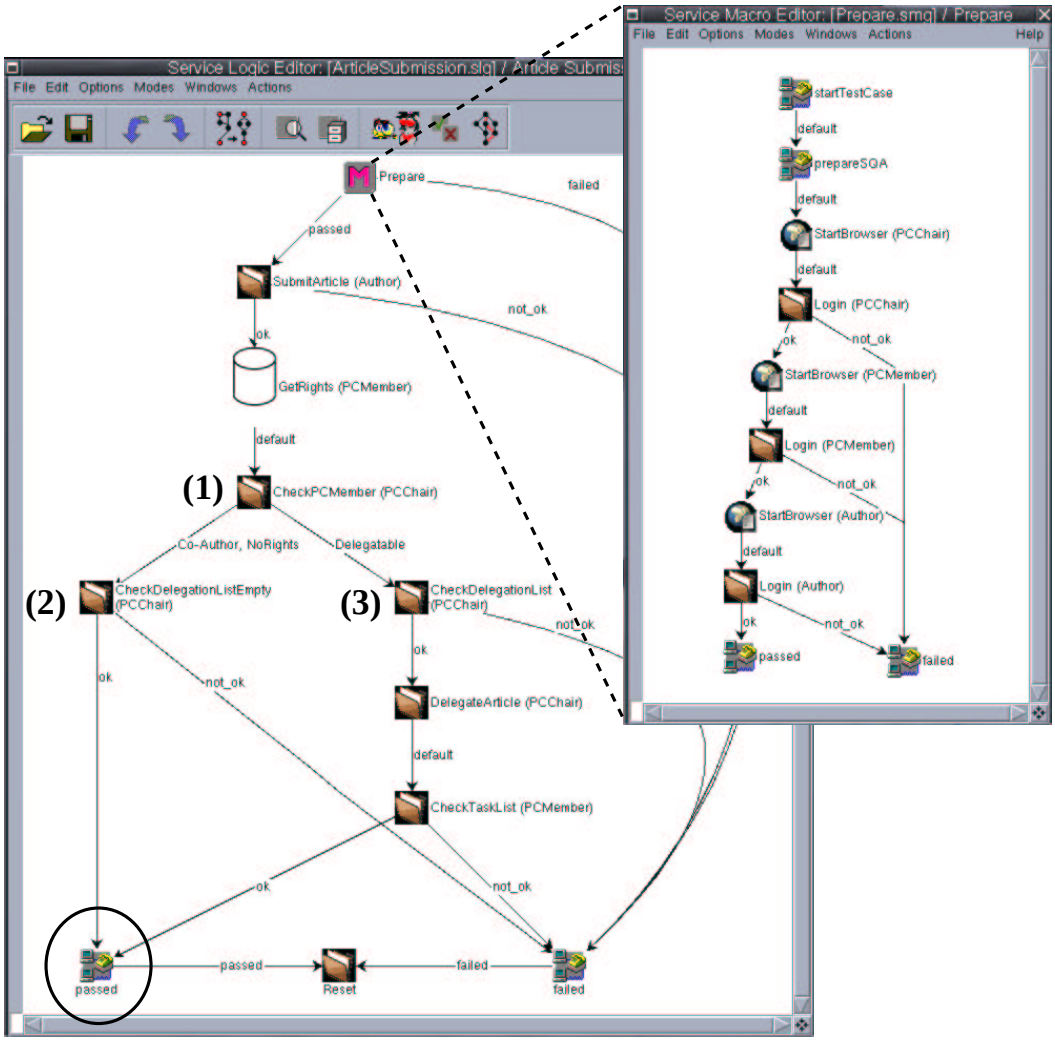


Figure 2: Example of a Test Case Definition, with Macros

2. If this operation has been successfully completed, the *PCChair* should delegate this newly submitted article to one of the program committee members. The test case continues in dependence of special conditions (which are checked in the two following test actions):

   (a) If the *PCMember* is a co-author of this article or has no access rights[1] to review this category of articles, the test case continues with step **(2)**, where it will be checked that the *PCMember* has not yet been assigned the reviw of this paper (technically, the member is not in the current *delegation list*),

   (b) otherwise this *PCMember* is a legal potential reviewer for the paper, thus he must be in the *delegation list* of the *PCChair*. This will be checked in **(3)**.

[1]This information will be requested directly from the underlying database.

3. If the article can be delegated, in this case the *PCChair* delegates the review task for this article to the *PCMember*.

4. Finally after validating the corresponding result, i.e. that the *PCMember* has a new review task regarding this article, the test can be evaluated.
   Note that here it is e.g. insufficient to check just the number of articles in the task list in order to conclude a correct delegation: instead, the service must ensure that exactly the article submitted in step 1 is now in the task list.

5. The last action of every test is a special reset operation, which makes sure that the test environment is correctly reset (e.g. clean up the databases, log out every user), and that all the browser sessions on the clients are properly terminated.

This test case is distributed, in the sense that it covers several subsystems involved in the application. In particular, test actions are executed on the clients marked with *Author*, *PCChair* and *PCMember*, as well as on the database and on the webserver in our test environment. Nevertheless the design of such a complex test is still easy and intuitive, since the tester (or test engineer) is not concerned with instantiation details like e.g. the concrete *IP-Addresses* of test clients.

3.3 Verification of the Test Cases

In our environment, the design of test cases is constantly accompanied by online verification of the global correctness and consistency of the test cases' control flow logic [7]. During the design phase, vital properties concerning the usage of parameters (local properties) and the interplay between the stimuli and verification actions of a test case (global properties) can be verified. Design decisions that conflict with the constraints and consistency conditions of the intended system are thus immediately detected.

Systems developed with the Agent Building Center, like the Online Conference Service, enjoy a double bonus: the same constraints enforced at design time (detailed for this service e.g. in [11, 2]) can now be reused to validate the test cases, ensuring consistency of the test purposes with the original system specification.

Local properties specified imperatively are used to check that the parameters are set correctly. Global properties concerning the interplay between arbitrarily distant test blocks of a test graph are expressed in a user-friendly specification language based on SLTL, the Semantic Linear-time Temporal Logic of [4], and are gathered in a constraint library accessed by the environment's model checker during verification.

Typical additional constraints for the test of *Web-based applications* refer e.g. to the resource management: one can ensure that all used resources are correctly released after the execution of the test case[2], and this independently of the tests outcome.

[2]E.g. every log-on for an user must be followed by a log-off for this user.

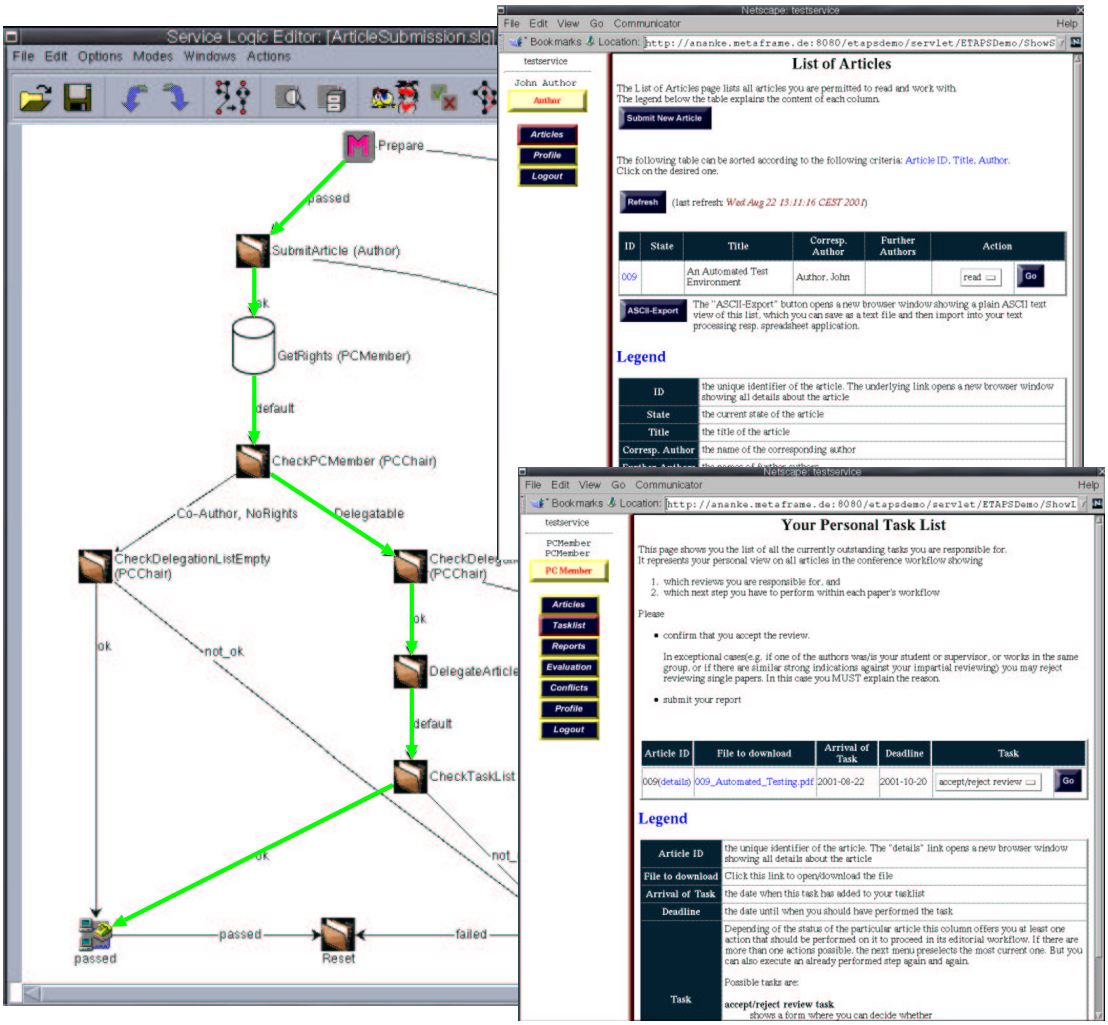


Figure 3: Test Execution: A Successful Execution Path and the Users Views

If the model checker detects an inconsistency, a plain text explanation of the violated constraint appears. In addition, test blocks violating a local property as well as paths violating a global property are marked.

3.4 Execution of the Test Cases and Reporting

Test cases can be executed immediately in the *Test Coordinator* by means of ABC's tracer. Starting at a dedicated test block of a test graph the tracer proceeds from test block to test block. The actions represented by a test block are performed, i.e., stimuli and inspection requests are sent to the corresponding system's component, responses are received, evaluated, and the evaluation result is used to select the relevant next test block.

Figure 3 illustrates these features on a concrete snapshot of a test session. Here, an *Author*

submits an article that should be reviewed by a member of the program committee. Even this relatively simple scenario shows the potential complexity of such tests.

In general, the implementation of this execution scheme requires two activities during set-up of the *ITE*:

1. The actions referenced via test blocks have to be implemented by means of test tools. This task is performed by testers which are familiar with test tools, their handling and programming. For each action, the tester has to specify instructions to be executed by the test tool determined to support the specific action, e.g., via recording GUI-activities. In case of the conference service the *Rational Robot* is used on the clients to control an internet browser.

2. Specific tracer code has to be developed, that is assigned to the action's test block and that will be executed by the tracer. Experience with the system level testing of CTI systems shows that this code can be generated automatically for most actions.

Finally, when executing a test graph, a detailed protocol is prepared. For each test block executed by the tracer, all relevant data (its execution time, its name, the version of the files associated with the test block, the block's parameter values, and the processed data) are written to the protocol.

4 Conclusion

The state of the art of web application testing tools is still dominated by static approaches, focussing on a posteriori structure reconstruction and interpretation [8, 3] rather than on functional aspects directly related to the applications design. As it has happened in the areas of telecommunications and CTI, this evolution is necessary in order to test applications also for their intention, rather than just for environmental accidents.

We have presented a methodology capable of granting adequate coverage of functional testing, yet still easy to use. It bases on the experience we gained in the last two years in the area of system level test of Computer Telephony Integrated applications, where our coordination-based coarse grain approach has been applied very successfully ([6]). We have shown so far the practicability of the approach by successfully testing complex role based functional features of the Online Conference Service, an advanced online management system for the organization of scientific conferences.

References

[1] Compuware Inc. http://www.compuware.com.

[2] B. Lindner, T. Margaria, and B. Steffen. Ein personalisierter Internetdienst für wissenschaftliche Begutachtungsprozesse. In *GI-VOI-BITKOM-OCG-TeleTrusT Konferenz on "Elektronische Geschäftsprozesse"*(eBusiness Processes), Universität Klagenfurt, September 2001.

[3] C.H. Liu, D.C. Kung, P. Hsia, and C.T. Hsu. Structural testing of web applications. In *Proc. of Int. Symposium on software reliability engineering (ISSRE 2000)*, pp. 84–96, 2000.

[4] T. Margaria and B. Steffen. Backtracking-free design planning by automatic synthesis in METAFrame. In *Proc. of the 1st Int. Conf. on Fundamental Approaches to Software Engineering (FASE 1998)*, volume 1382 of *Lecture Notes in Computer Science*, pp. 188–204. Springer Verlag, 1998.

[5] O. Niese, T. Margaria, A. Hagerer, M. Nagelmann, B. Steffen, G. Brune, and H. Ide. An automated testing environment for CTI systems using concepts for specification and verification of workflows. *Annual Review of Communication*, 54, 2000.

[6] O. Niese, T. Margaria, A. Hagerer, B. Steffen, G. Brune, and H. Ide. Automated regression testing of CTI-systems. In *Proc. of IEEE European Test Workshop (ETW 2001)*, 2001.

[7] O. Niese, B. Steffen, T. Margaria, A. Hagerer, G. Brune, and H. Ide. Library-based design and consistency checks of system-level industrial test cases. In H. Hußmann, editor, *Proc. of the 4th Int. Conf. on Fundamental Approaches to Software Engineering (FASE 2001)*, volume 2029 of *Lecture Notes in Computer Science*, pp. 233–248. Springer Verlag, 2001.

[8] F. Ricca and P. Tonella. Building a tool for the analysis and testing of web applications: Problems and solutions. In T.Margaria and W.Yi, editors, *Proc. of the 7th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001)*, volume 2031 of *Lecture Notes in Computer Science*, pp. 372–388. Springer Verlag, 2001.

[9] Radview Software. http://www.radview.com.

[10] B. Steffen and T. Margaria. *METAFrame in Practice: Design of Intelligent Network Services*, volume 1710 of *Lecture Notes in Computer Science*, pp. 390–415. Springer Verlag, 1999.

[11] B. Steffen, T. Margaria, and V. Braun. Coarse-granular model checking in practice. In M.B. Dwyer, editor, *Model Checking Software - 8th International SPIN Workshop*, volume 2057 of *Lecture Notes in Computer Science*, pp. 304–312. Springer Verlag, 2001.

QWE2002 Paper 4T

Dr. Nancy Eickelmann & Mr. Allan Willey
(Motorola Labs)

Traceability and Reproducability in Integrated System Test Environments

Key Points

- Automated Test Environments
- COTS tool integration
- System test technologies

Presentation Abstract

This paper examines software architectural constraints in relation to software and system test automation environments. The MATE architecture requires a mapping of multiple views of the environment including structure, functionality, process and data. An overview of the environment is represented by mapping the test functionality to hardware and software structures of the environment. An architectural representation provides a foundation for evaluating the impact of architectural and COTS choices on system test engineers and test managers. This paper discusses the integration of COTS tools into MATE including DOORS, Primavera, ClearQuest and ClearCase; the development of standardized test management support TMS; and the automation of the system test process. The MATE architecture is described and a detailed discussion of test automation and tool integration issues is undertaken. Software architectural analysis is used to determine if a specific structural decomposition and the functional allo cation to system structures supports or impedes certain qualities. Changes to an ATE such as enhancements to system functionality, improvements to performance (space and time), and reuse of components, data representation and changes to processing algorithm are all sensitive to system architectural constraints.

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, her research focused on integrating the Balanced Scorecard into the NASA context to provide a measurement framework for software test technology improvements. 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.

Allan Willey is a Memb er of Technical Staff at Motorola Labs in the Software and System Engineering Lab (SSERL). Allan leads the "Motorola Automated Test Environments" (MATE) Team. These applied researchers are developing techniques to improve the capabilities of software development groups to test new products. Projects using various advanced statistical analysis techniques, formal modeling, and simulation techniques are being carried out to assess their value for improving delivered product quality, as well as their impact on productivity and testing time. The MATE Team collaborates closely with development organizations in various Motorola product groups to transition successful results. Allan holds an AB in philosophy from the College of William and Mary, and both a BS in mathematics and an MBA in management sciences from George Washington University.

MOTOROLA LABS

MATE

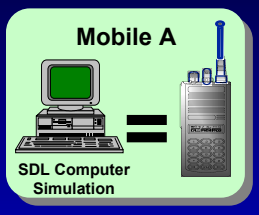
Traceability and Reproducability in Integrated System Test Environments

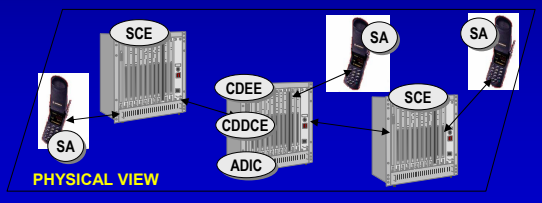
Dr. Nancy Eickelmann
Motorola Labs
1303 E. Algonquin Rd.
Schaumburg, IL 60196
USA
(847)538-0745
(847)576-3280
Nancy.Eickelmann@motorola.com

MOTOROLA LABS

Agenda

- Why do we need MATE ?
- How do we decompose the problem space ?
- How do we analyze our solution sets ?
- What did we learn ?

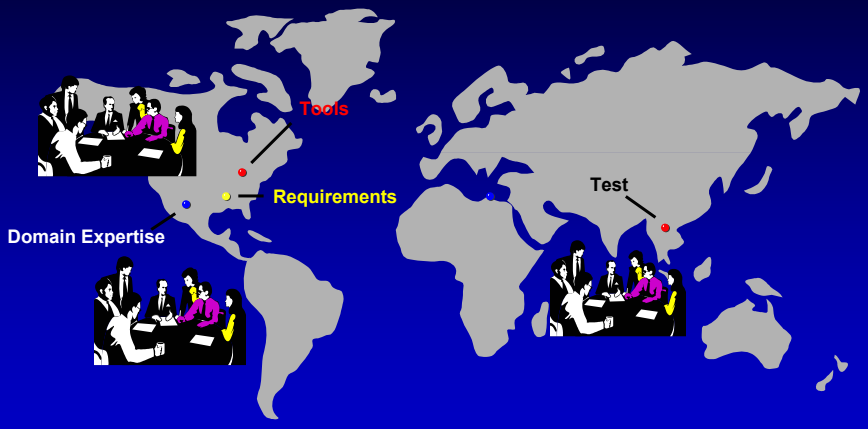




Dr. Nancy Eickelmann Quality Week Europe - March 2002

MOTOROLA LABS

Distributed Teams



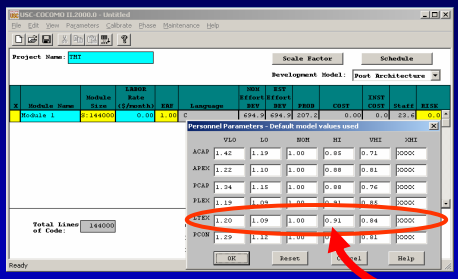
MOTOROLA LABS

Test Management

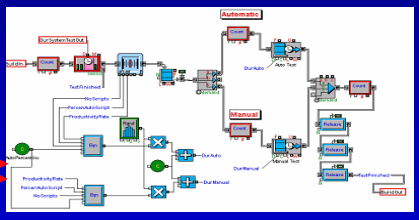
- Resource Allocations
  - Measurement
  - Planning
  - Control
  - Understanding
  - Training and Learning

MOTOROLA LABS

Test... Cost Estimating, Planning and Tracking



SDL Models



$$(NoAutoScripts \times x) + (NoManScripts \times 3 \cdot x) = Duration$$

$$\text{Productivity} = x = \frac{Duration}{NoAutoScripts + 3 \cdot NoManScripts}$$

Quality Week Europe - March 2002

MOTOROLA LABS

Issues

- Technology
  - » Test Quality
  - » Lifecycle traceability
  - » Test reproducability
  - » COTS interoperability, variability, stability
- Process Maturity
  - » Organization(s) maturity
  - » Multi-site organization maturity variance
  - » Communication and collaboration
- People
  - » Distributed teams of people
  - » Specialization of skill sets required

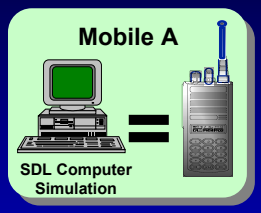
Dr. Nancy Eickelmann

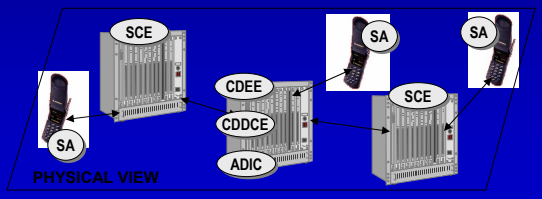
Quality Week Europe - March 2002

MOTOROLA LABS

Agenda

- Why do we need MATE ?
- How do we decompose the problem space ?
- How do we analyze our solution sets ?
- What did we learn ?





MOTOROLA LABS

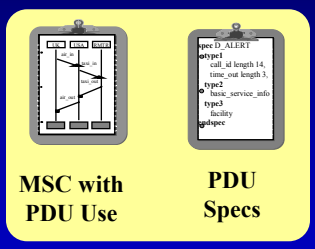
Traceability and Reproducability

Automation	Standardization	Integration
– Process	– Test Technologies	– COTS Solutions
– Lifecycle Tools	– Lifecycle Tools	– People
– Test Technologies	– Process	– Resources

MOTOROLA LABS

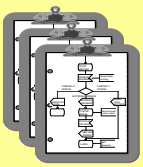
Automation
Test Case Generation

Functional Requirements Specification



ptk also performs:
- validation of behaviour
- verification of data use

Conformance Test Suite



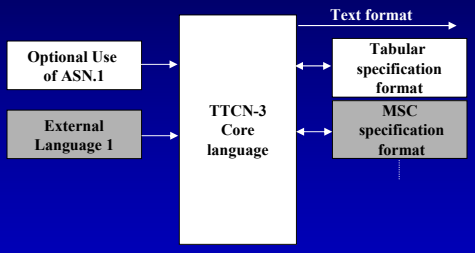
Test Scripts
- SDL
- TTCN

Executable Test Suite

MOTOROLA LABS

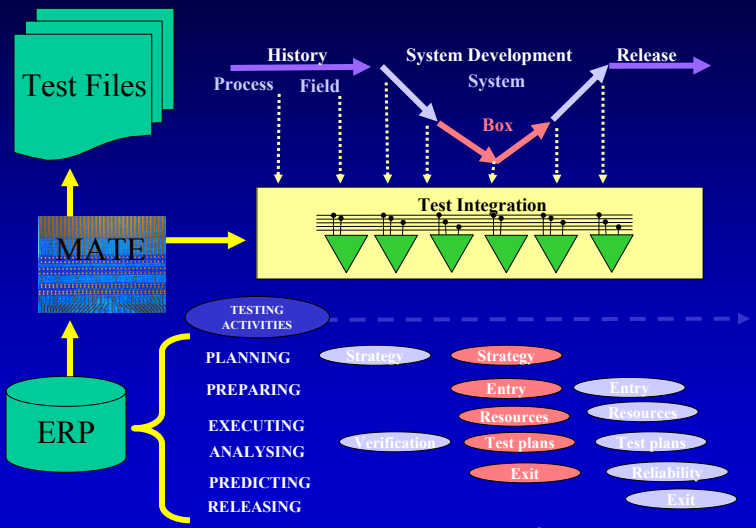
Standardization

- TTCN – 3: Tree and Tabular Combined Notation
- ASN.1: Abstract Syntax Notation
- SDL/SDT- 2000: Specification Description Language
- MSC – 2000: Message Sequence Charts
- UML – 2000: Unified Modeling Language



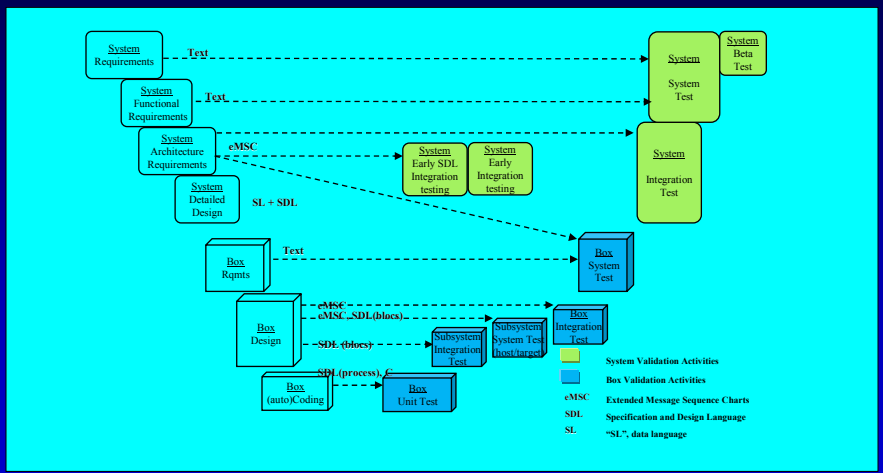
MOTOROLA LABS

Integration



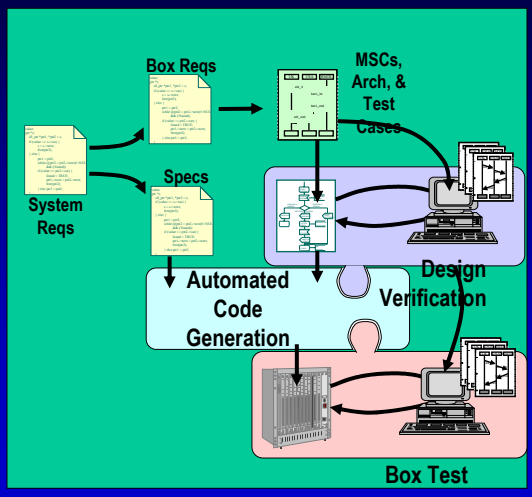
MOTOROLA LABS

Lifecycle View



MOTOROLA LABS

Process View

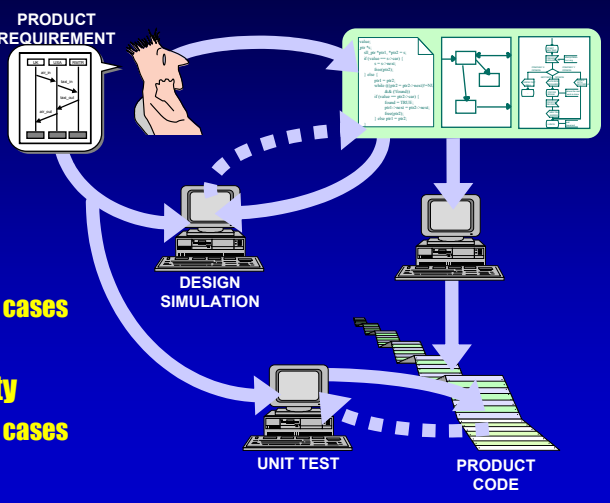


- Task order
- Work product flow
- Skill sets
- Scheduling
- Task completion criteria
- CM
- RM

MOTOROLA LABS

Data View

- **Traceability**
  - Text
  - Graphics
  - Scripts & Test cases
  - Code
- **Reproducability**
  - Scripts & Test cases



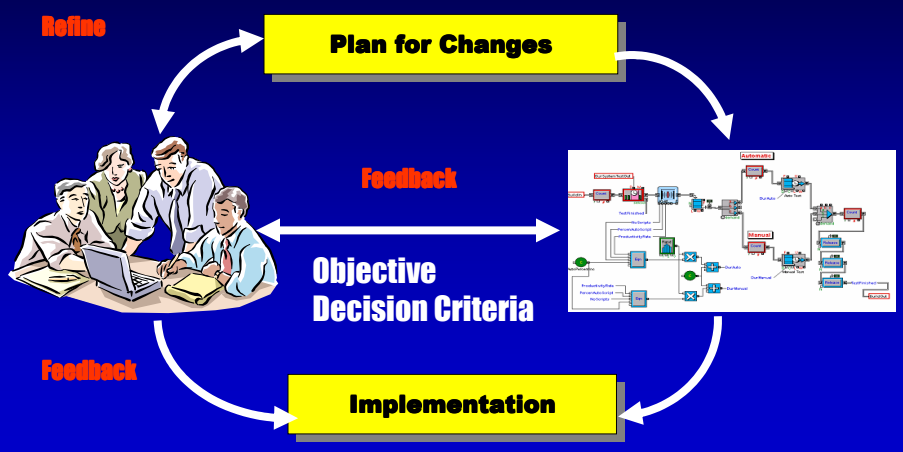
MOTOROLA LABS

Tools and COTS Views

- COTS Integration
  - Analysis and Design tools
  - Test tools
  - CM and PM
  - Resource Management

MOTOROLA LABS

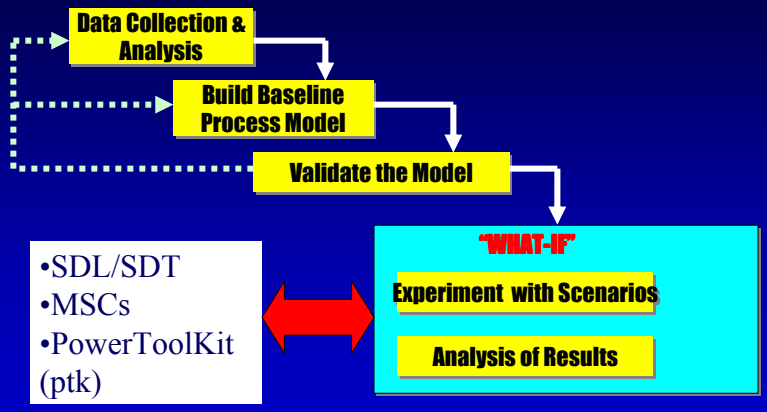
Tool Evaluation



Dr. Nancy Eickelmann Quality Week Europe - March 2002

MOTOROLA LABS

Tool Insertion Impact Analysis

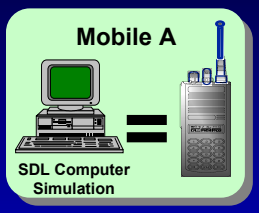


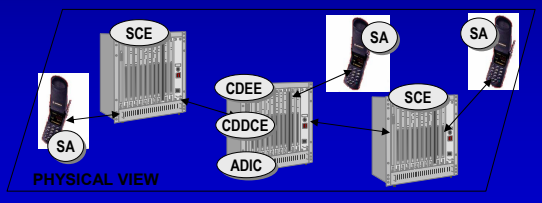
- SDL/SDT
- MSCs
- PowerToolKit (ptk)

MOTOROLA LABS

Agenda

- Why do we need MATE ?
- How do we decompose the problem space ?
- How do we analyze our solution sets ?
- What did we learn ?



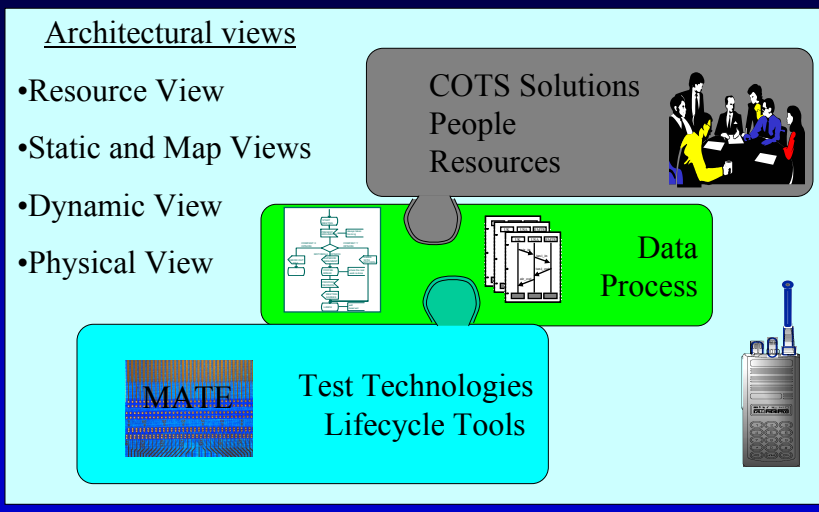


MOTOROLA LABS

Architecture Puzzle

Architectural views

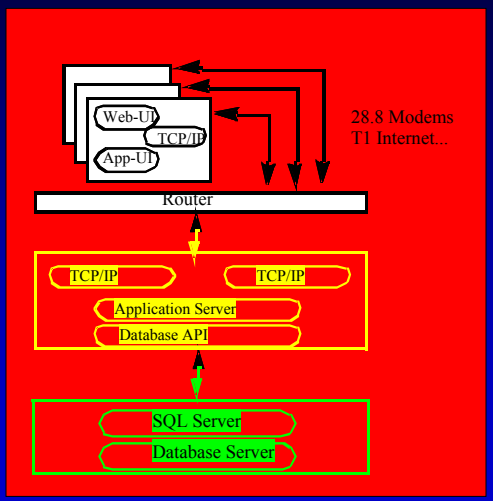
- Resource View
- Static and Map Views
- Dynamic View
- Physical View



MOTOROLA LABS

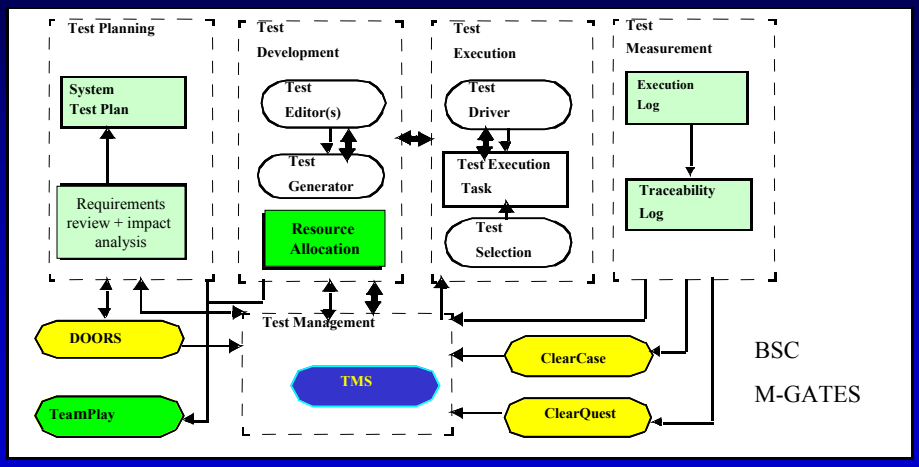
Resource View

- Resource constraints by project, business unit, functional area, country
- Topology requirements
- Security layer requirements



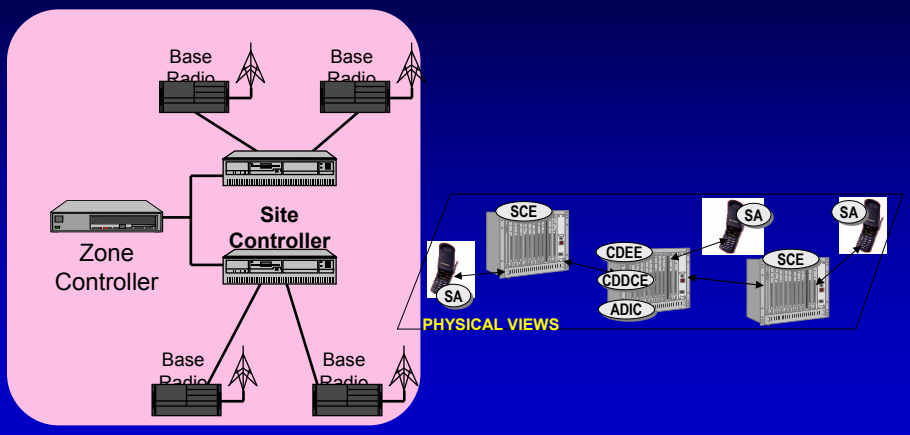
MOTOROLA LABS

Static and Map Views



MOTOROLA LABS

Physical Views



MOTOROLA LABS

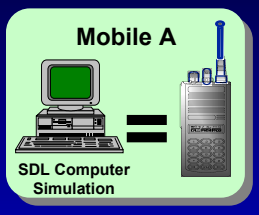
Combined Views

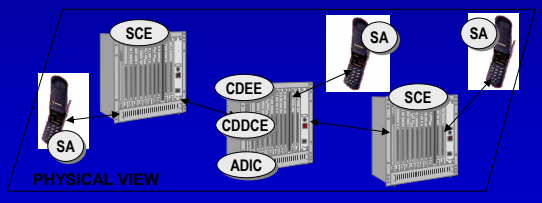
- System Test Functions
- Hardware/Software Structures
- Components/Interfaces
- Object Management
- External Systems

MOTOROLA LABS

Agenda

- Why do we need MATE ?
- How do we decompose the problem space ?
- How do we analyze our solution sets ?
- What did we learn ?





MOTOROLA LABS

Lesson Learned

- Need....
  - understanding of process imposed architecture constraints
  - measures of IT cost/benefit impact at project, business unit and enterprise level
  - better architectural analysis tools

MOTOROLA LABS

Questions? Answers? Comments?

Traceability and Reproducability in Integrated System Test Environments

Nancy S. Eickelmann
Motorola Labs
1303 East Algonquin Road
Schaumburg, IL 60196 USA
+1 847 538 0745
Nancy.Eickelmann@motorola.com

Allan L. Willey
Motorola Labs
1303 East Algonquin Road
Schaumburg, IL 60196 USA
+1 847 576 6343
Allan.Willey@motorola.com

ABSTRACT
The Motorola Automated Test Environment (MATE) provides a means of automating the test process and integrating tools to support required testing capabilities across the lifecycle. The MATE architecture is provided as a foundation to discuss the issues of test tool integration with COTS products that automate the full system test lifecycle. Specific tool integration issues include data, control, process, platforms and user-interface integration issues. The software architecture of MATE can facilitate or impede analysis of desired system properties and performance requirements. Architectural analysis is conducted to provide insight into the properties of the Motorola Automated Test Environment.

Keywords
Automated Test Environment (ATE), COTS Tool Integration, System Test.

1 INTRODUCTION

The Motorola Automated Test Environment initiative will enable Motorola to optimize the test process through standardization of tools; automation of manual processes; and integration of data, processes, and interfaces across a common test environment. Specific benefits of the MATE initiative relate to test management, automated test script generation and execution, UI platform independence, and improved quantification of release criteria. All of these factors provide the necessary infrastructure to support test traceability and reproducability.

- **Test management** - Provides a central controller for the test environment, test artifacts and test data.
- **Automated test script generation and maintenance** - Provides the automatic generation of test scripts from a formal requirement representation. Maintenance of the test scripts as the requirements change is facilitated.
- **Automated test execution** - Alleviates staffing shortages at critical process bottlenecks.
- **Web front end** - Provides for access to the test tools and the data from anywhere on any platform. The web was seen as the optimal solution.
- **Reliability prediction** - Provides objective criteria to determine when a product can ship based on predicting remaining defects in a product after a test cycle.

Providing for continuous improvements in core processes often requires the insertion of tools and technologies to automate and standardize industrial practice. Understanding the strengths of tools and applying them appropriately in the context of the organizational maturity is essential.

Despite the critical importance of automated test environments in the development of high quality products, full lifecycle integration of tools is rarely achieved in practice. To understand what prevents full lifecycle integration of tools with respect to data, process, platforms, user-interfaces and control, Motorola is conducting concurrent, focused R&D initiatives, Motorola Automated Test Environment (MATE) and Core Process Redesign (CPR).

The Motorola Automated Test Environment (MATE) initiative provides technology to design, develop, and test high quality software and systems. The MATE initiative addresses a subset of the product development issues focusing on testing, which remains a very costly, and time-consuming phase of the product lifecycle.

The Core Process Redesign (CPR) initiative is identifying the necessary integration factors to insert an automated test environment that seamlessly inter-operates in the context of the overall product lifecycle. The CPR identifies decision criteria (decision gates), process input/output pairs, entry/exit criteria and common resource requirements and constraints.

The overriding objectives of MATE and CPR are to radically improve time-to-market and predictability in schedules, costs, and quality of products developed by:

- development of a common core process that enables consistent practices
- build a common platform that supports interoperability and reuse
- increase resource allocation flexibility to facilitate cost effectiveness

The MATE and CPR initiatives are complementary efforts that focus on standardization, automation, and integration of tools, data, and processes. A primary goal of the MATE initiative is to develop a common and automated test environment. With a common environment the entire corporation would be able to take advantage of optimization improvements made to the environment based on technology advancements external to Motorola and technology acquisition internal to Motorola. Therefore the MATE architecture must accommodate legacy tools, process changes and future needs met by technology insertion. As such the MATE architectural requirements are a negotiated construct among the identified stakeholders [8].

This paper examines software architectural constraints in relation to software and system test automation environments. The MATE architecture requires a mapping of multiple views of the environment including structure, functionality, process and data perspectives. An overview of the environment is represented by mapping the test functionality to hardware and software structures of the environment. An architectural representation provides a foundation for evaluating the impact of architectural and COTS choices on system test engineers and test managers. MATE integrates COTS tools such as DOORS, ClearQuest and ClearCase; internally developed test management support, TMS; and automation of the system test process.

2 BACKGROUND

Motorola Labs is the research department of Motorola Inc. There are groups around the world conducting research on the leading edge technologies and on advancements to current product offerings. A majority of the research conducted by the Labs is initiated by requests from the product groups within the Communications Enterprise (CE) of Motorola. The CE is comprised of several large business units and accounts for more than half of Motorola's sales. The major businesses in the CE include the cellular handset and cellular infrastructure businesses, the paging device and paging infrastructure businesses, and public and private radio products with their associated infrastructures. All of these products, whether handheld or infrastructure, are computer-based. For the last two decades these products have evolved from a hardware component base to a computer platform with functional layers comprised of software specifically written to address the product requirements. The R&D organizations in Motorola have evolved accordingly so that at this time far more than half of the engineers developing new products worldwide in Motorola are software engineers.

The size of software components in our products vary according to their function. A simple one-way handheld pager will have no more than a few thousand lines of code. A cellular infrastructure product has over twenty million lines of code in its various components. A small core of the infrastructure products are concerned with “call processing,” on the order of a few hundred thousand lines of code, and this core will be the most highly reliable part of the system. A similar core component of cellular handset and two-way radios will be key to their functionality, and will consist of tens of thousands of lines of highly optimized code. Numerous additional functions are a part of each product, for example billing tracking and customer authentication components of a cellular system. Some of these, such as authentication, are real-time and interact with the core call processing functions. Others, such as system maintenance functions, are less time-critical but often tend to require large functional blocks.

The rate of growth of the software component of all of these products reflects general telecommunication industry experience. Further, we expect this evolution to continue as future product directions seem to show an unabated demand for new products. For example, while the popularity of simple one-way paging devices is declining, two-way short-message-service paging

offerings are a growing market. These two-way devices are an order of magnitude more complex, and contain that much more embedded code.

Numerous other examples can be cited of increased product complexity and size, but underlying these significant changes is the fact that testing has remained a very costly, time-consuming phase of the product development life cycle [2,9,11,12]. As the market pressures increase to deliver more, better, and faster products, testing is being viewed increasingly as a bottleneck to success.

To bypass the testing bottleneck, Motorola has launched focused R&D initiatives to provide technology to design, develop, and test high quality software and systems. The MATE initiative addresses a subset of the issues. A concurrent corporate initiative of Core Process Redesign (CPR) addresses another view of the issues. The CPR initiative is instrumental in identifying the necessary integration factors to insert an automated test environment that seamlessly interoperates with the rest of the product lifecycle. The overriding objective of CPR is to radically improve time-to-market and predictability in schedules, costs, and quality of product development by:

- Development of a common process across the divisions.
- Identify points of synchronicity across divisions.
- Improve efficiency of processes.
- Identify best practice tools to create a common and global R&D environment
- Build a foundation to propel common platform design and reuse
- Increase resource allocation flexibility across the corporation.

The establishment of the Motorola automated test environment contributes to all of the above goals as well. The testing activities have tremendous effect on product time-to-market, cost and quality [3,4]. The above goals can also be applied to internal product deliveries from tool development groups. Establishment of common tools and processes eliminates the redundancy of the internal tool groups and allows for reallocation of personnel to focus on product delivery and not internal support. It also allows for innovative ideas to be implemented in a common platform and taken advantage of by a multitude of users instead of restricting technological benefits to local groups. The MATE initiative will bring Motorola closer to achieving its goals. MATE focuses on standardization, automation and integration of tools, data and processes. To facilitate our discussion and provide common definitions for terms we introduce the automated test environment reference architecture used in our analysis.

3 AUTOMATED TEST ENVIRONMENT REFERENCE ARCHITECTURE

A reference architecture, the STEP model [5] provides a basis for the representation and formalization of the MATE architecture. The reference architecture segments the required functionality for the environment and captures the relationships among the functionalities in a diagrammatic format.

The domain is partitioned into six canonical functions: test execution, test development, test failure analysis, test measurement, test management, and test planning. Each of these functions is defined to provide consistency in the use of terms.

- **Test Execution** includes the execution of the instrumented source code and recording of execution traces. Test artifacts recorded include test output results, test execution traces, and test status.
- **Test Development** includes the specification and implementation of a test configuration. This results in a test suite, the input related test artifacts, and documentation. Specific artifacts developed include test oracles, test cases and scripts, and test adequacy criteria.
- **Test Failure Analysis** includes behavior verification and documentation and analysis of test execution pass/fail statistics. Specific artifacts include pass/fail state and test failure reports.
- **Test Measurement** includes test coverage measurement and analysis. Source code is typically instrumented to collect execution traces. Resulting test artifacts include test coverage measures and test failure measures.
- **Test Management** includes support for test artifact persistence, artifact relations persistence, and test execution state preservation. Test process automation requires a repository for test artifacts. A passive repository such as a file serves the basic need of storage. However, an active repository is needed to support relations among test artifacts and provide for their persistence.

- **Test Planning** includes the development of a master test plan, the features of the system to be tested, and detailed test plans. Included in this function are risk assessment issues, organizational training needs, required and available resources, comprehensive test strategy, resource and staffing requirements, roles and responsibility allocations, and overall schedule."

The STEP model, shown in Figure 1, stratifies test functionalities from the apex of the pyramid to its base in a corresponding progression of the test process lifecycle as described in [9]. The test process evolution is aligned with the arrow to the right of the pyramid and segments test functionality according to the test lifecycle focus. Each segment represented in the pyramid includes the functionalities of previous periods as you descend from the apex to the base.

The top section of the pyramid represents the function of test execution. Test execution is clearly required by any test process. The test process focus of debugging includes only test execution.

The second segment of the pyramid, from the top, is divided into two scalene triangles.

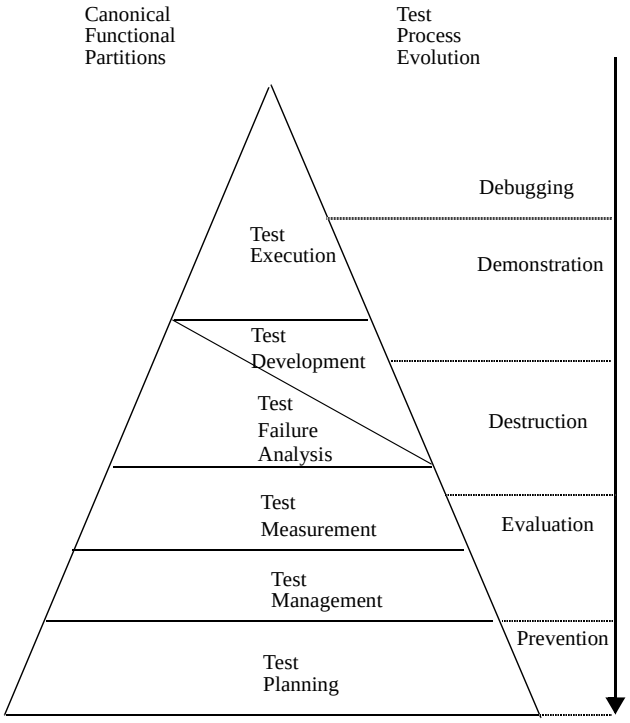


Figure 1. STEP model reference architecture [5].

The smaller scalene triangle represents test development. The larger scalene triangle represents test failure analysis. The relative positions and sizes have semantic significance

Test development played a more significant role to the overall test process when focused on demonstration and destruction due to the manual intensive nature of test development. Test development methods have not significantly changed, although they have improved in reliability and reproducibility with automation. Thus, their role in test process diminishes in significance as you automate the test development process.

Test failure analysis is less important when performed manually, as interactive checking by humans adds little benefit for test behavior verification. The methods that can be applied to test failure analysis have increased in their level of sophistication, making test failure analysis more significant to the overall test process. One of the most significant advances is specification-based test oracles [5]. This is a key difference in test process focus.

Test measurement is represented by the third segment in the pyramid. Test measurement is required to support an evaluation focus for test, which represents a full lifecycle approach. A significant change in the test process focus is that testing is applied in parallel to development, not merely at the end of development. Test measurement also enables evaluating and improving the test process.

Approaching the base of the pyramid, the fourth segment represents test management, which is essential to the test process due to the sheer volume of information that is created and must be stored, retrieved, and reused. Test management is critical for test process reproducibility.

The base, or foundation, of the pyramid is test planning. Test planning is the essential component of prevention focused test efforts. Test planning introduces the test process before requirements, so that rather than being an afterthought, testing is pre planned and occurs concurrently with development. The STEP model provides the core functionality and process focus to describe the MATE architecture. The architectural description requires integration of multiple views of the test environment including structure, functionality, process

and data.

In the next section, the MATE architecture is described and a detailed discussion of test automation and tool integration issues is undertaken. Software architectural analysis is used to determine if a specific structural decomposition and functional allocation to system structures supports or impedes achieving desired properties for MATE. Changes to an ATE such as enhancements to system functionality, improvements to performance (space and time), and reuse of components, data representation and changes to processing algorithms are all sensitive to system architectural constraints [8].

4 MATE ARCHITECTURAL VIEWS

There a several architectural views in the literature including: Map View (mapping of functions and components), Static View (structure diagram), Resource View (mapping of software onto hardware), Dynamic View (operational diagrams).

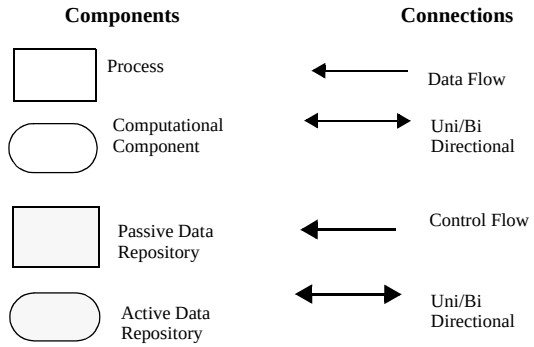


Figure 2. SAAM graphical architectural notation [10].

We provide a diagrammatic representation of the static, map, and resource architectural views for MATE. The static and map views are combined by using the SAAM notation. The resource view uses a generic representation.

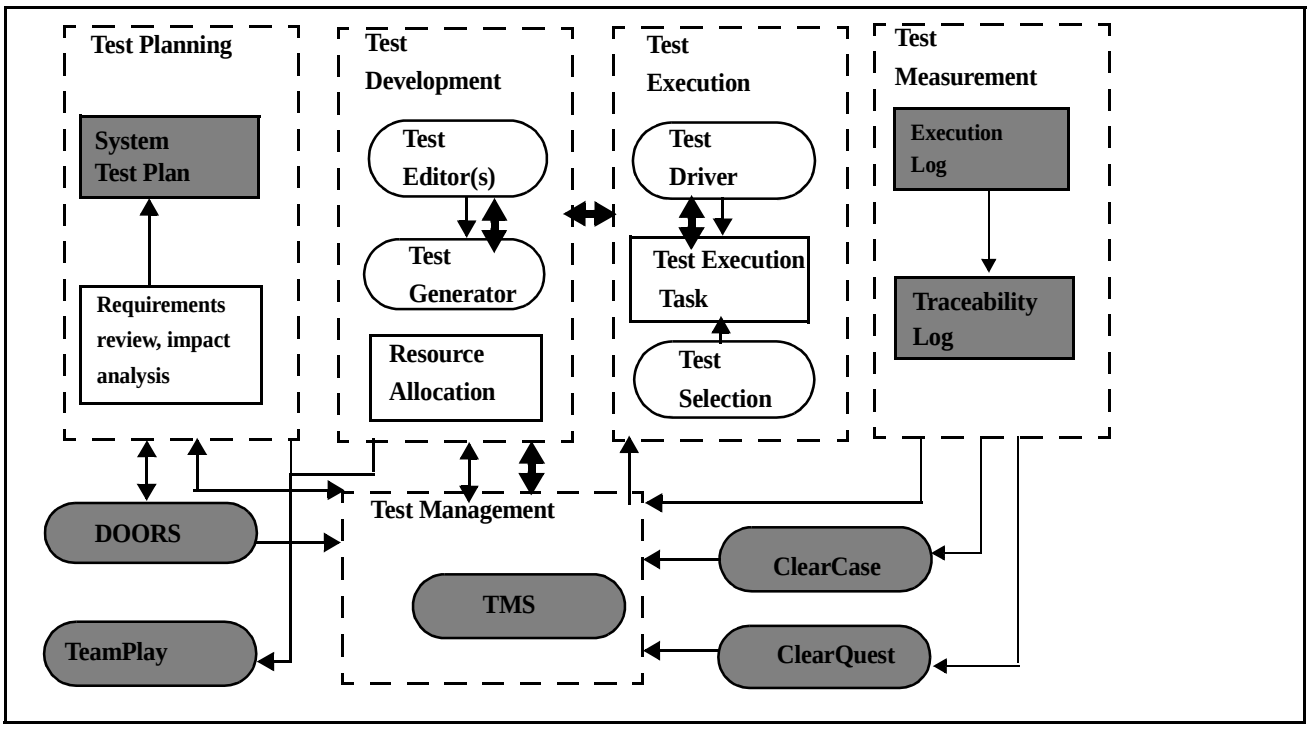


Figure 3. Combined Static and Map views.

4.1 MATE Architectural Static View and Map View

The Software Architectural Analysis Method (SAAM) provides a concise notation that includes a static view and a map view [10]. The static view represents the structure as a decomposition of the system components and their interconnections. The map view groups the components according to their high level functionality. The canonical functions for MATE were defined in the reference architecture.

The SAAM graphical notation is shown in Figure 2. In this notation there are four types of components: a process (unit with an independent thread of control); a computational component (a procedure or module); a passive repository (a file); and an active repository (database). There are two types of connectors: control flow and data flow, either of which may be unidirectional or bidirectional.

The allocation of domain functionality to the software completes the graphical representation of the ATE, see Figure 3. The allocation provides the mapping of a system's intended functionality to the concrete interpretation in the implementation. The diagram segments the canonical functions for MATE using a broken line. The COTS active repositories are not included in the domain functionality as they interact through the TMS management system. The five functionalities inclusive in MATE, test execution, test development, test measurement and test management are discussed below.

Test Execution includes the execution of the system in the target environment testbed, and recording of execution traces. Test artifacts recorded include test output results, test execution traces, and test status. The test execution environment must provide a bridge to interface with legacy solutions that must be migrated in an evolutionary versus revolutionary approach. Testing in a distributed context with heterogeneous platforms was considered desirable for flexibility and reusability. Another factor of significance was language independence and open interfaces for interoperability.

Test Development includes the specification and implementation of a test configuration. This results in a test suite, the input related test artifacts, and documentation. Specific artifacts developed include test oracles, test cases and scripts, and test adequacy criteria. Test development solutions should support multi-platform and multi-site access to data while providing adequate response time and usability in the user interface. Automated test case generation solutions focused on SDL and MSC model specifications that support auto test case generation. Significant process and data integration issues arose that were addressed by interfacing ClearCase test data with TMS and ClearQuest CM and defect data with TMS. Resource allocations for system test are documented in TeamPlay and provided on-line to the test developers.

Test Measurement includes documentation and analysis of test execution pass/fail statistics. Specific artifacts include pass/fail state, test failure reports and test traceability reports. MATE's support for test measurement must be addressed across all the functionality and interface with legacy data integration issues for test and development groups. A distributed collection and repository tool provides for heterogeneity with transparency of access issues through web-enabled tools.

Test Management includes support for test artifact persistence, artifact relations persistence, and test execution state preservation. Test process automation requires a repository for test artifacts. A passive repository such as a file serves the basic need of storage. However, an active repository is needed to support relations among test artifacts and provide for their persistence. Test management should provide a state-of-the-art solution with web-based access and interface. TMS was available on UNIX platforms and was being migrated to an NT platform as well. Since it was Java based, TMS would easily adopt to new platforms as Motorola test groups use them. Prototypes of TMS had been operational on SUN Solaris, HPUX, Windows NT, Windows 98, MacOS and Linux. TMS has been designed to take advantage of the multi-site features of Oracle and a software configuration management tool by using both as underlying technology to the tool. The TMS API will also allow for a quick

replacement of either technology as improved technologies become available. A significant finding through the MATE effort was that terminology was not used uniformly throughout the organization. A living glossary was implemented continues to track and resolve discrepancies in word usage and understanding.

Test Planning includes the development of a master test plan, the features of the system to be tested, and detailed test plans. Included in this function are risk assessment issues, organizational training needs, required and available resources, comprehensive test strategy, resource and staffing requirements, roles and responsibility allocations, and overall schedule. Test planning requires inputs from the DOORS repository and TMS test management function. Test planning also requires project management tool support which is provided by TeamPlay. Test planning is not well supported by most commercially available tools as it requires integration with data repositories, process issues, and tool interfaces.

4.2 Resource View

A common representation for architecture is the resource view. This view of the MATE architecture is diagrammed in generic terms such as provided in Figure 4.

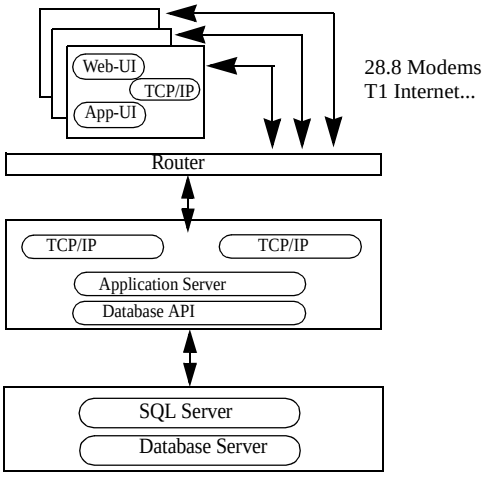


Figure 4. MATE Architecture Resource View.

The resource view focuses on essential communications links such as routers, modems, internet and intranet connections and the architectural topology, client/server. Analysis of this view lends itself to performance analysis, queueing models and throughput simulations, see Table 1. This view is also used to evaluate system availability constructs using a composite quantification of the system hardware failure/time interval and the corresponding software failure/time interval. Mean time to repair is estimated for hardware and corresponding software providing an incidence frequency with an estimate in hours to restore the system. The declaration of system hardware and software computational units also provides for an affordability analysis. The initial investment costs, updates, and maintenance costs can be obtained from the vendors of choice. Given the payback period, and the minimum time required for return on investment, a Net Present Value (NPV) comparison can provide an objective valuation of hardware and software configurations.

Table 1: Performance Analysis

Scenarios	Comments
Network latency from client to server and from server to client in ms.	Latency time provided by network specification
Server latency consists of dequeueing time (Cdq=ms) and task computation time (Cfnc=ms)	Latency time provided in server specification
Distribution time of periodic updates minimum	8@10sec;8@20sec;8@40sec;8@80sec
Distribution time of periodic updates maximum	48@10sec;24@20sec;12@40sec;8@80sec

5 CORE PROCESS DESCRIPTION

The CPR identifies decision criteria (decision gates), process input/output pairs, entry/exit criteria and common resource requirements and constraints. The CPR initiative provides a high level common process with fixed decision gates. However, it also provides for insertion of customized process segments for insertion into organizations with legacy tools and technologies that impose unique process constraints. The software development process and software and system test groups in Motorola follow various lifecycle models that comply with internal

standards. An external standard that provides a high level process description for test is the IEEE Std. 1012-1998, Standard for Software Verification and Validation. This current standard is an update of the IEEE Std. 1012-1986. The updated standard is comprehensive in its scope and details full lifecycle activities and cross-references according to compatibility with other process standards. The substantial benefit of a well-defined process with objective decision criteria is realized when it is instantiated in an automated test environment.

6 TECHNOLOGY TRANSITION PLAN

The MATE transition effort is analyzed in the context of the 3 vector space of the SEI Technology Transition Conceptual Framework [6], see Figure 5. The 3 vector space represents the relationships among an increasing magnitude of technological change; the required effort to adapt or learn new skills, procedures, structures, strategies, or culture; and time required to adapt

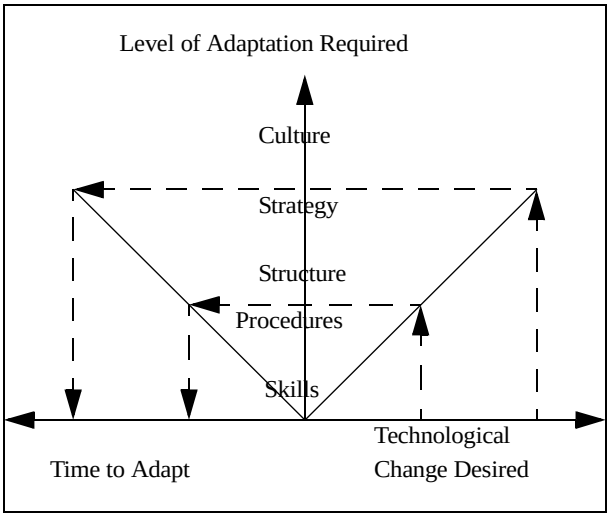


Figure 5. Dimensions of technology transition adapted from [CMU/SEI-93-TR-31]

and institutionalize the changes. This structure supports evaluating the “size” of a change based on the level of adaptation required. We will reference this figure in our subsequent discussion of MATE. The MATE effort as described using the SEI’s technology life cycle conceptual framework depicts an effort towards a common test environment that is comprised of technologies in various phases of maturity. The majority of MATE technologies will require new skills and procedures as a foundation for their introduction. These changes are seen as requiring typically a time horizon measured in months.

The time required to achieve a technology transition is partially dependent on an additional classification of the technology life cycle. The technology life cycle includes 3 distinct phases, R&D, new product development, and adoption and implementation. Tools and technologies of MATE span the full technology life cycle and therefore may be described as short term to long term change efforts. The insertion of MATE represents new technologies for the system test organizations, however the technologies include both recent R&D developments of Motorola Labs and mature commercial tools to be integrated into the common environment.

We outline the steps of each of the 3 technology life cycle phases as described in Figure 6. The focus of phase 1, R&D of the technology life cycle, is primarily the technology itself. Inclusive in the R&D phase of the life cycle;

- concept formulation
- development and extension
- enhancement and exploration (internal)
- enhancement and exploration (external)
- early popularization.

The aspects relevant to MATE in phase 2, the new product development phase includes;

- generating new product ideas
- screening those ideas
- testing product concepts

Adoption and implementation, phase 3 of the technology life cycle includes multiple steps relevant to the MATE transition effort;

- needs assessment
- selection of candidate products
- evaluation of candidate products
- introduction of product to user groups, management, stakeholders
- gathering feedback from target groups
- implementation planning and execution

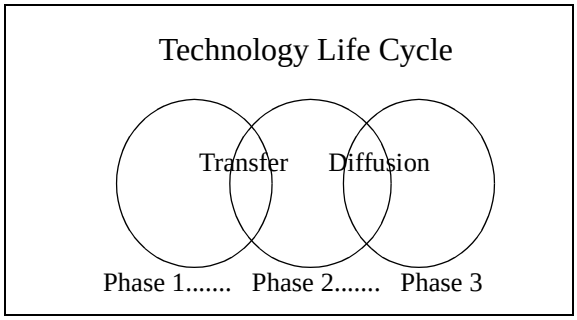


Figure 6. Technology life cycle phases adapted from [CMU/SEI-93-TR-31].

We will use the steps outlined for the 3 phases of the technology life cycle in describing the MATE effort and will reference the tools and technology maturity with regard to this structure as appropriate.

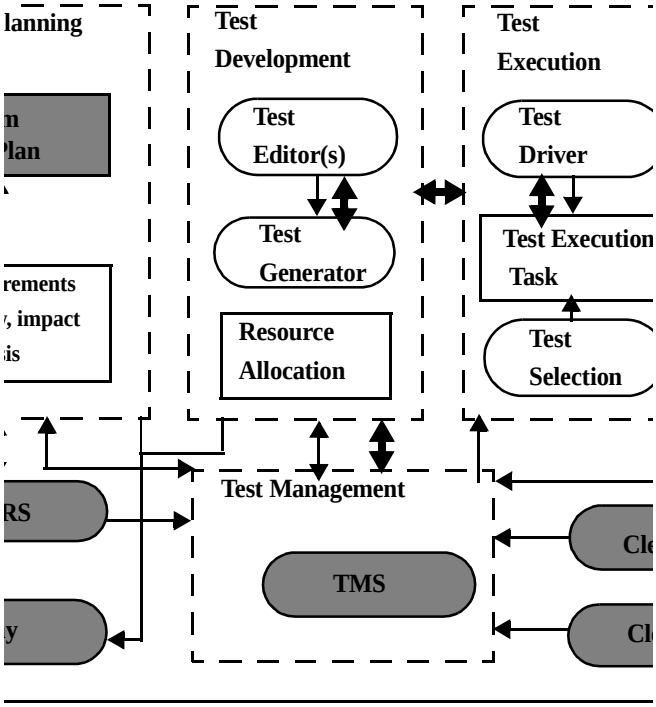
Test execution was comprised of primarily mature technologies in phase 3 of the technology lifecycle, yet aspects of the total solution for full automation and integration might represent less mature technologies. Test case generation technologies were seen as being in the transfer phase, or phase 2, of the technology lifecycle and experiencing the early stages of transfer. The technology lifecycle maturity of the test management technologies are typically in the third phase of the lifecycle. Test measurement technologies were primarily in phase 3 of the technology lifecycle and represented the diffusion of measurement technologies.

The MATE effort as described using the SEI's technology lifecycle conceptual framework and technology lifecycle maturity evaluation depicts an effort towards a common test environment that is comprised of technologies in various phases of maturity. The majority of MATE technologies are in phase 3 of the technology lifecycle and are well-supported for successful transition.

REFERENCES

1. Brown, Alan W., Earl, Anthony N. and McDermid, John A., Software Engineering Environments: Automated Support for Software Engineering, McGraw-Hill, 1992.
2. Eickelmann, Nancy S., "Emerging Test Technologies: Shifting Tester Requirements," In the Proceedings of the 17th International Conference on Testing Computer Software, June 12-16, 2000.
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. and Richardson, Debra J. "An Evaluation of Software Test Environment Architectures" in proceeding of the Eighteenth International Conference of Software Engineering, 1996.
6. Fowler, Priscilla and Levine, Linda "A Conceptual Framework for Software Technology Transition", Software Engineering Institute Technical Report CMU/SEI-93-TR-031, December 1993.
7. D. Garlan, G. Kaiser, and D. Notkin. "Using tool abstraction to compose systems." IEEE Computer, vol. 25, June 1992.
8. D. Garlan and M. Shaw. "An introduction to software architecture.". Advances in Software Engineering and Knowledge Engineering, Volume I, World Scientific Publishing Co., 1993.
9. D. Gelperin and B. Hetzel. "The growth of software testing." Communications of the ACM, 31(6):687-695, June 1988.
10. R. Kazman, L. Bass, G. Abowd, and M. Webb. "SAAM: A method for analyzing the properties of software architectures." In Proceedings of the Sixteenth International Conference on Software Engineering, 81-90, Sorrento, Italy, May 21, 1994.
11. G. J. Myers. The art of software testing. New York, John Wiley and Sons, 1978.
12. E. Miller. Mechanizing software testing. TOCG Meeting, Westlake Village, California, April 15, 1986.
13. Guideline for Lifecycle Validation, Verification, and Testing of Computer Software. National Bureau of Standards Report NBS FIPS 101. Washington, D.C., 1983.

14.


lanning
Test
Development
Test
Execution
Test
Editor(s)
Test
Driver
Test
Generator
Test Execution
Task
Resource
Allocation
Test
Selection
Test Management
TMS

QWE2002 Paper 5T

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 (Paper 5T)

Dr. Edward Miller

eValid, Inc.
901 Minnesota Street
San Francisco, CA 94107 USA
Email: miller@soft.com

Presentation Outline

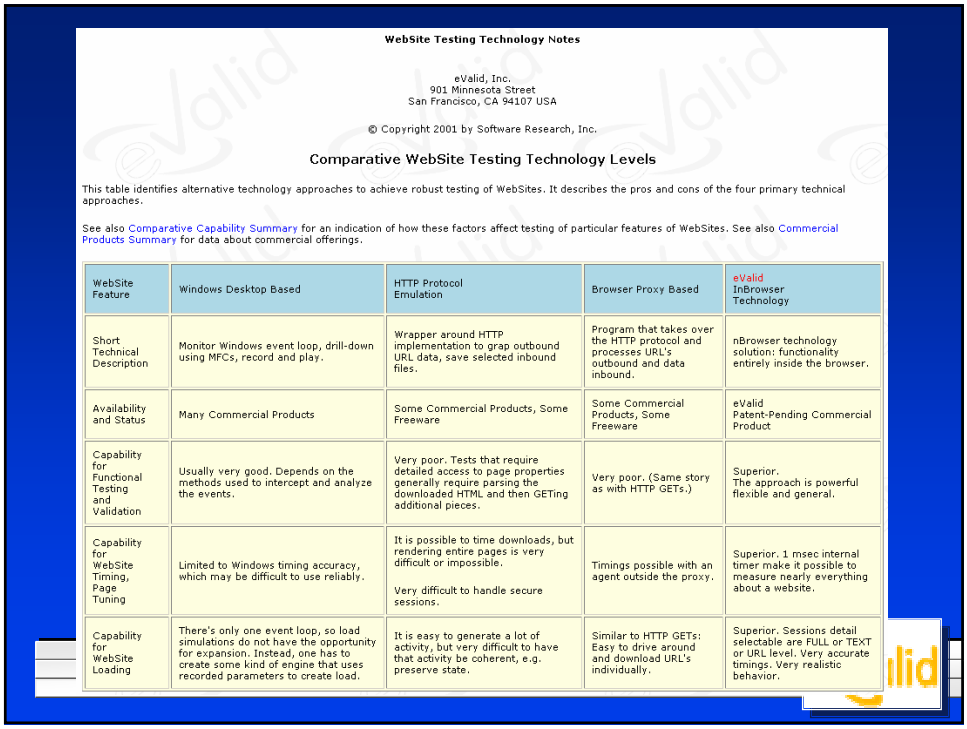
- Overview of Technology
- General Operation Description
- Validation & Verification Modes
- Timing and Tuning Process
- Load Imposition Mode
- SiteMap Mode
- Performance Considerations

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

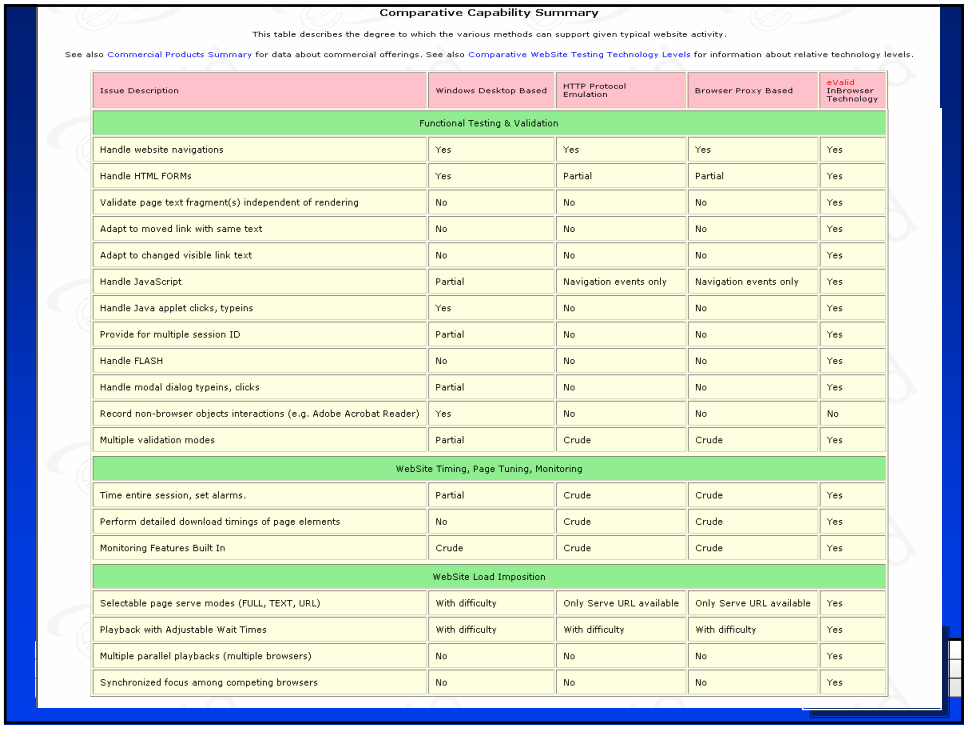
This table identifies alternative technology approaches to achieve robust testing of WebSites. It describes the pros and cons of the four primary technical approaches.

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

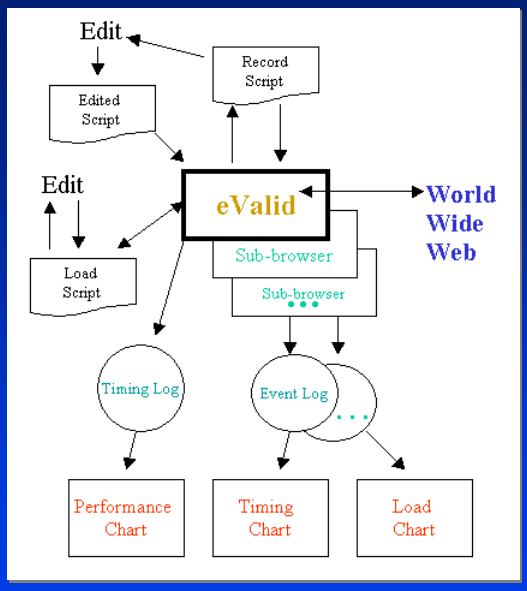
This table describes the degree to which the various methods can support given typical website activity.

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

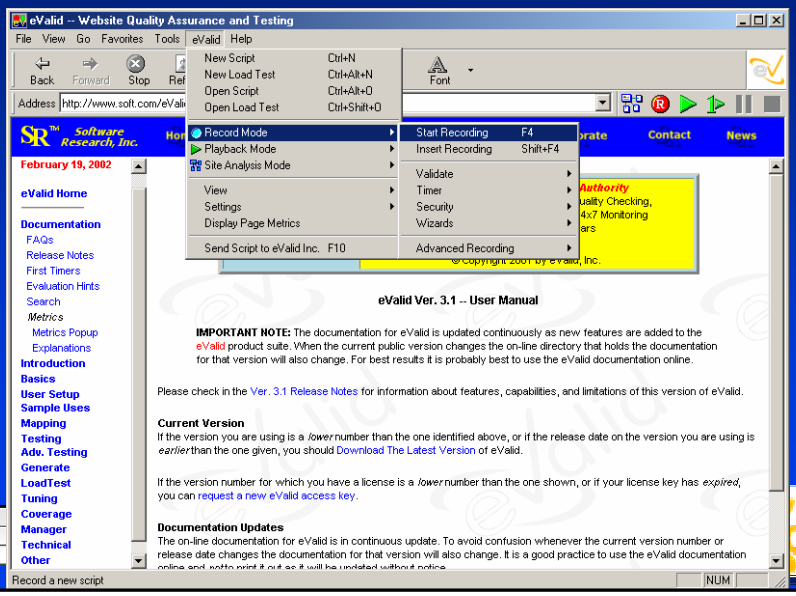
- Advantages
  - User View
  - Realistic
  - Natural operation
  - Accurate timings
- Disadvantages
  - Browser Variations
  - UNIX platform support



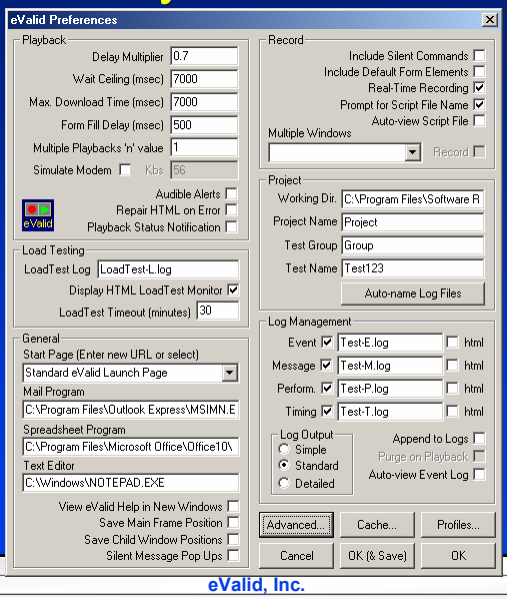
eValid Characteristics

- 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

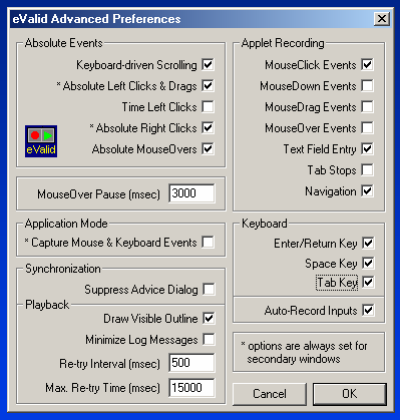


Record/Play Preferences Menu

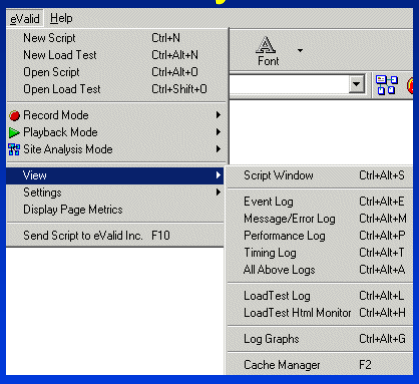


eValid, Inc.

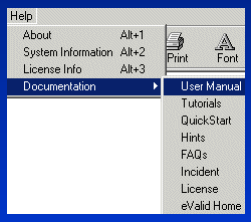
Advanced Preferences Meun



eValid Functional Testing Record/Play Pulldown



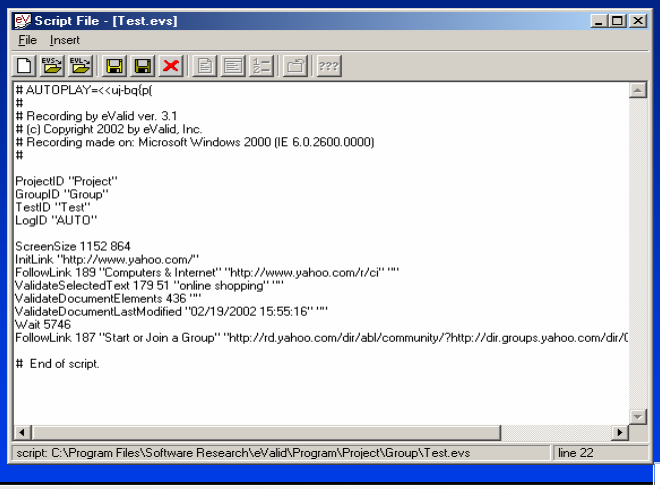
eValid Documentation/Support Pulldown



Recording Modes

- Navigation (including Frames)
- Modal Dialog
- Applet
- Application Mode
- Dependent Sub-Window
- Independent Sub-Window

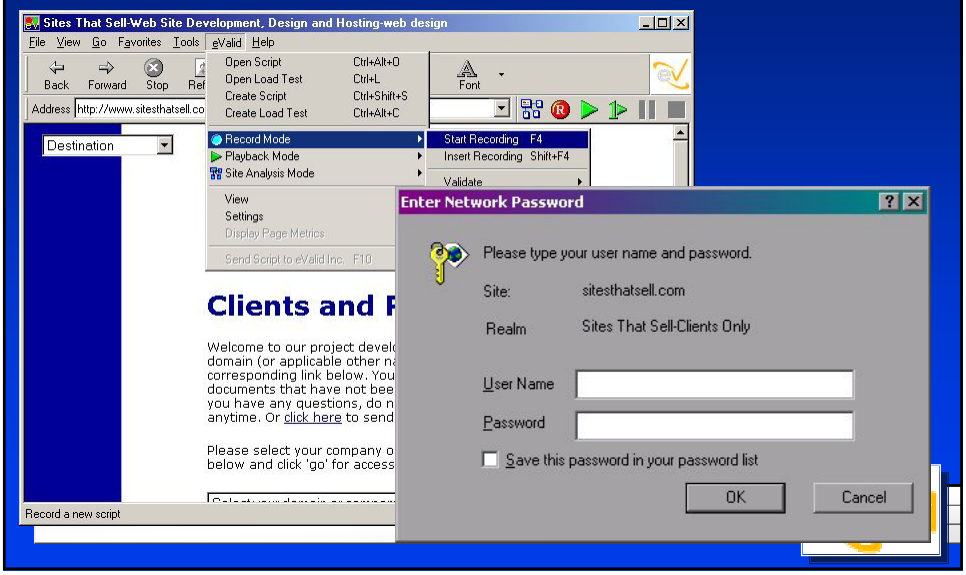
Example Script Window



Modal Dialog Recording

- Modal Dialog Has Screen Focus
- Access Recording via Script Window
- Pulldown for Allowable Input Sequences
- Extrinsic Commands on Same Window

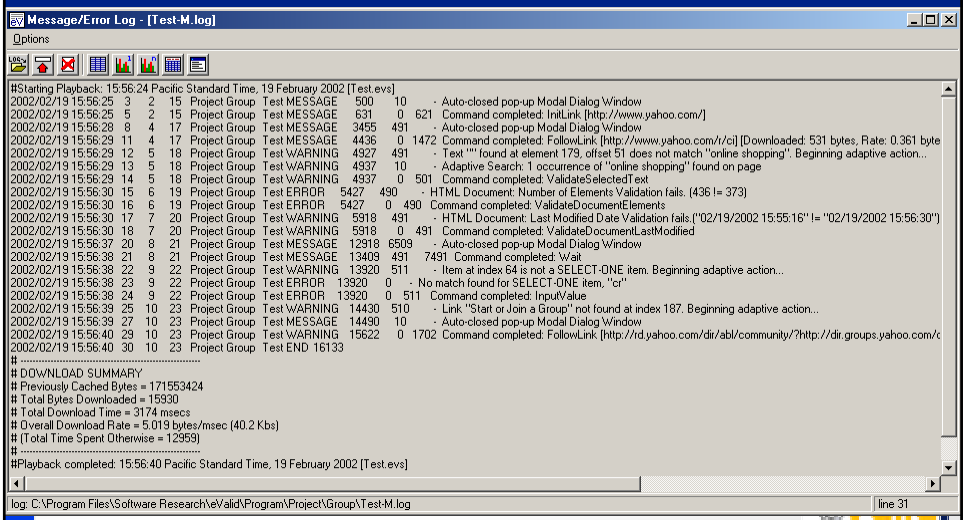
Sample of Modal Dialog Recording



Validation Modes

- Text Fragments
- Images, Links, Objects
- Image Part (with Synchronization)
- Document Properties
- Table Cells
- Element Ids (ID Tags)
- Applets (Size, Identification)

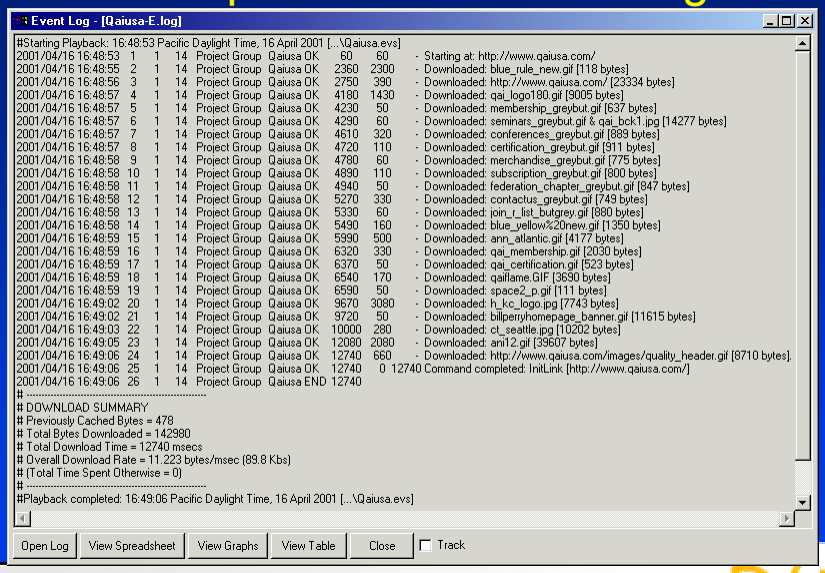
Example Message/Error Log



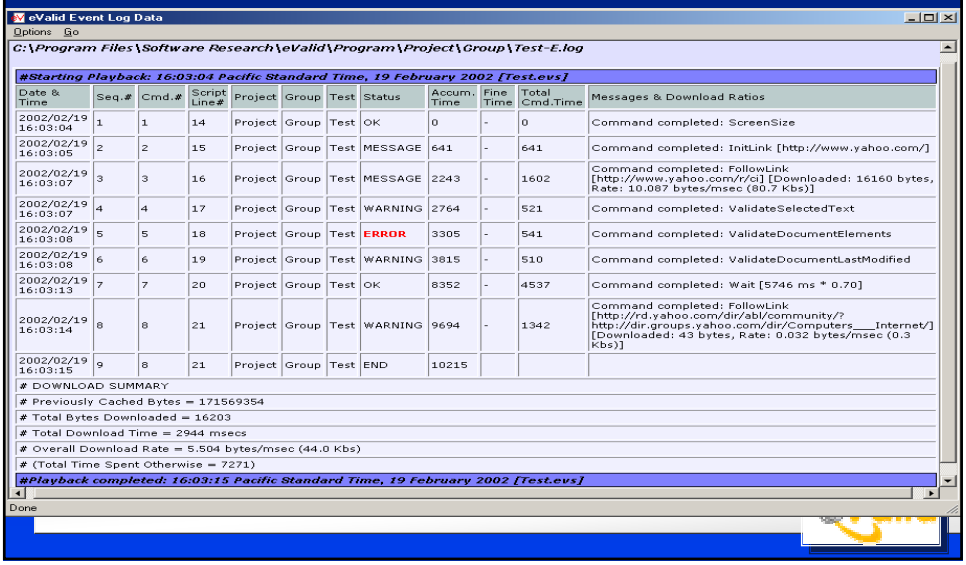
Performance Timing/Tuning

- Single and Multiple Download Timings
- Overall User-level Response Times
- Perceived User-level Response Times, Thresholds
- Web Effects
- Page Element Timings (Detailed Timing)

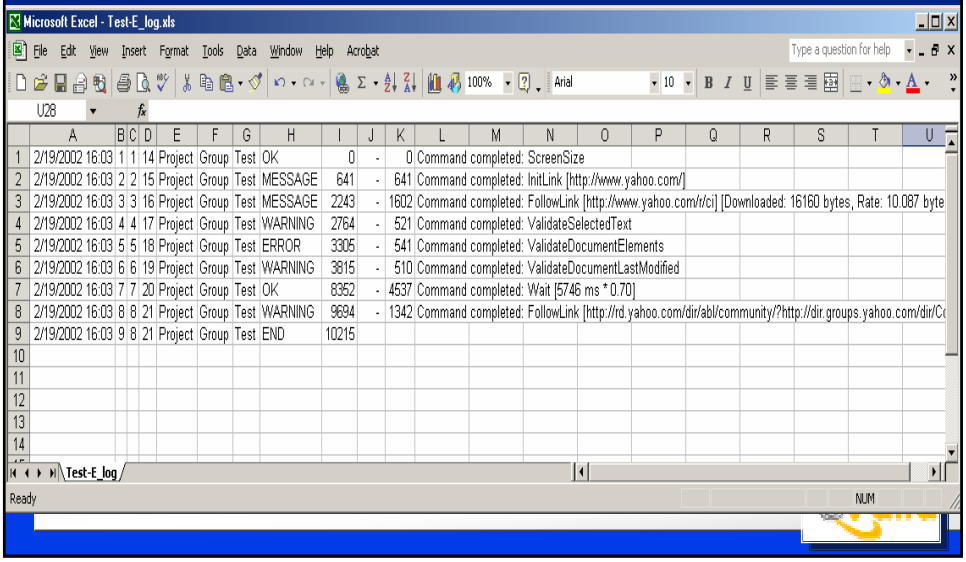
Example Detailed Event Log



Example Event Log in HTML

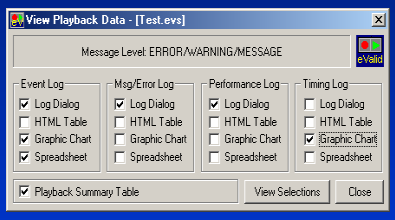


Example Event Log as Spread Sheet

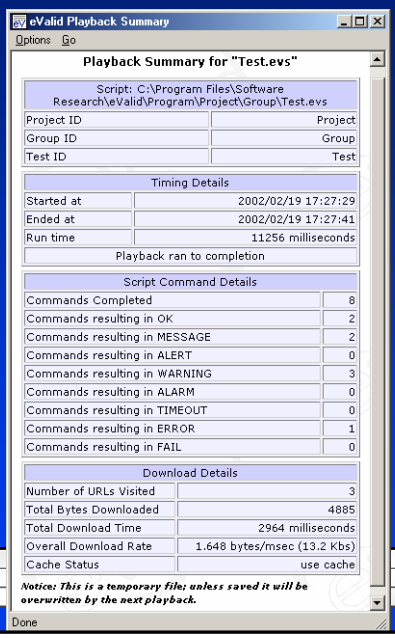


	A	B	C	D	E	F	G	H	I	J	K	L
1	2/19/2002 16:03	1	1	14	Project	Group	Test	OK	0	-	0	Command completed: ScreenSize
2	2/19/2002 16:03	2	2	15	Project	Group	Test	MESSAGE	641	-	641	Command completed: InitLink [http://www.yahoo.com/]
3	2/19/2002 16:03	3	3	16	Project	Group	Test	MESSAGE	2243	-	1602	Command completed: 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



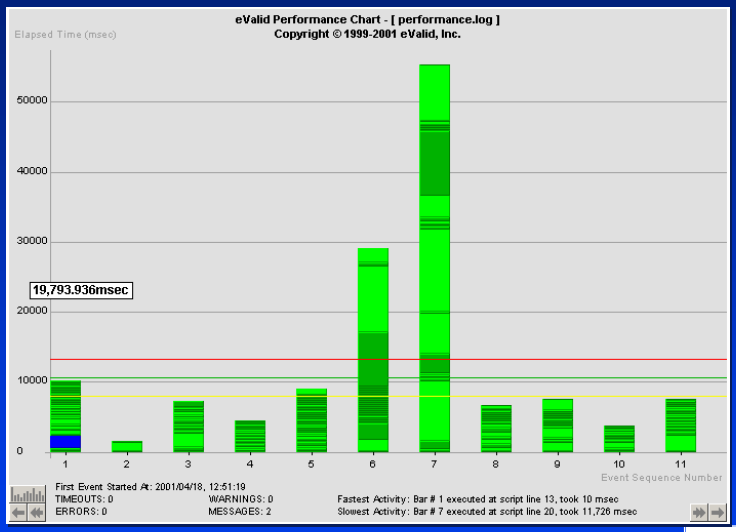
Alernative Playback Modes

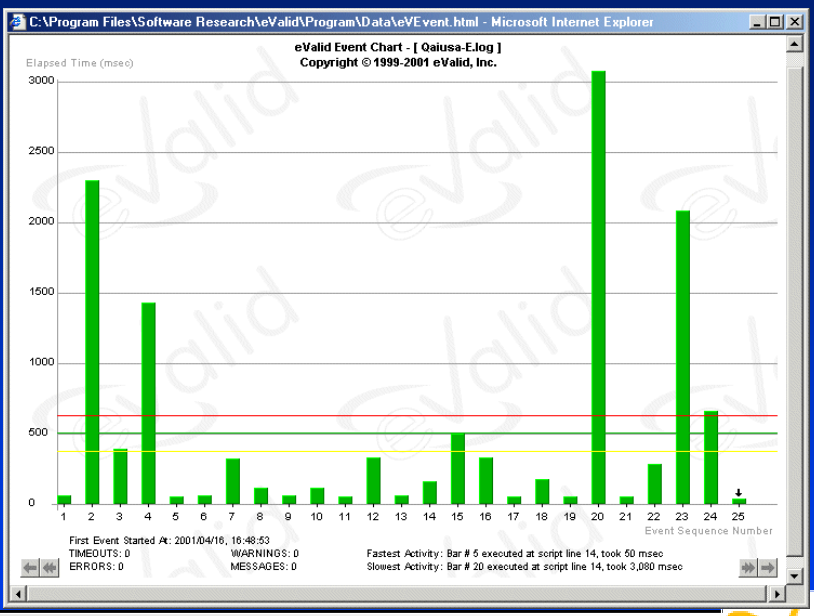
- Play Multiple
- Play Forever
- CallScript/GoScript
- OnErrorCallScript/OnErrorGoScript
- Parameter Passing
- Limitations
- Batch Mode Interface

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

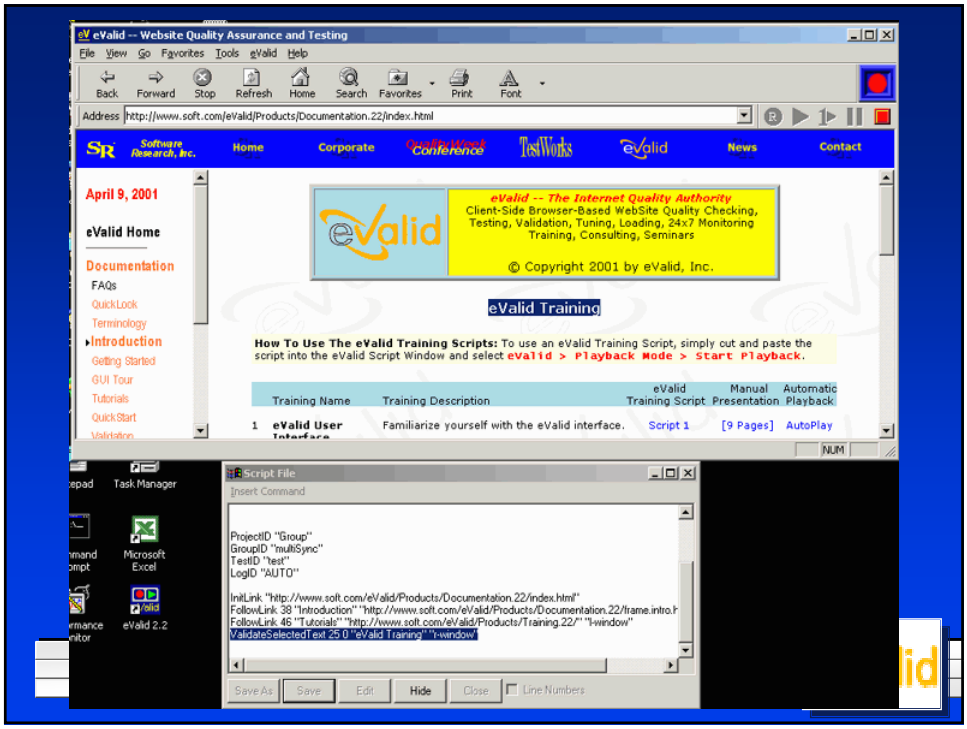
Sample "Stack Chart" for Timing/Tuning





Dynamic Testing: Unsolved Problems

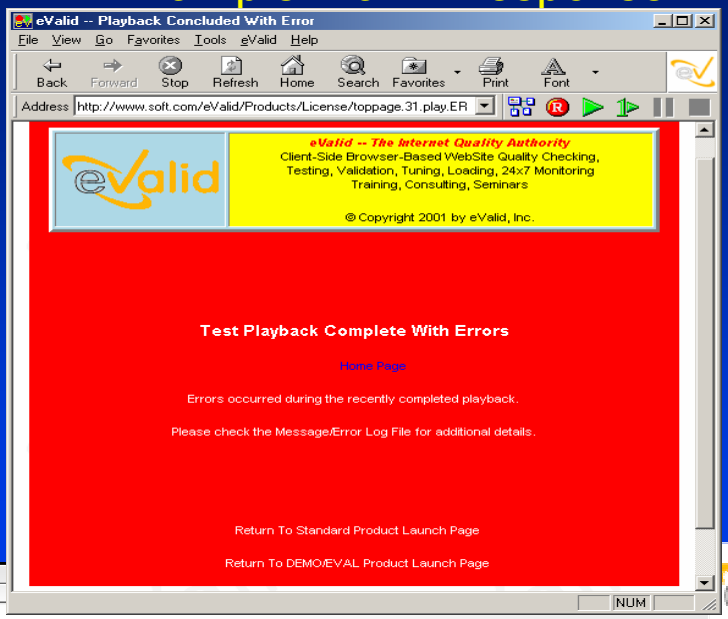
- Repeatability With Drags
- Databases that “Remember” (Initialization)
- Transient Page Effects
- Multi-Media Displays (Except Last Page)
- “Flash” Media Presentations
- Asynchronous Interactions



Alarm/Event Timing

- Overall Timer
  - Total Time
  - Total Byte Counts
- Page Timing
  - Base Page
  - LINKed Files
    - JavaScript (*js)
    - Cascading Style Sheets (*css)
  - Images

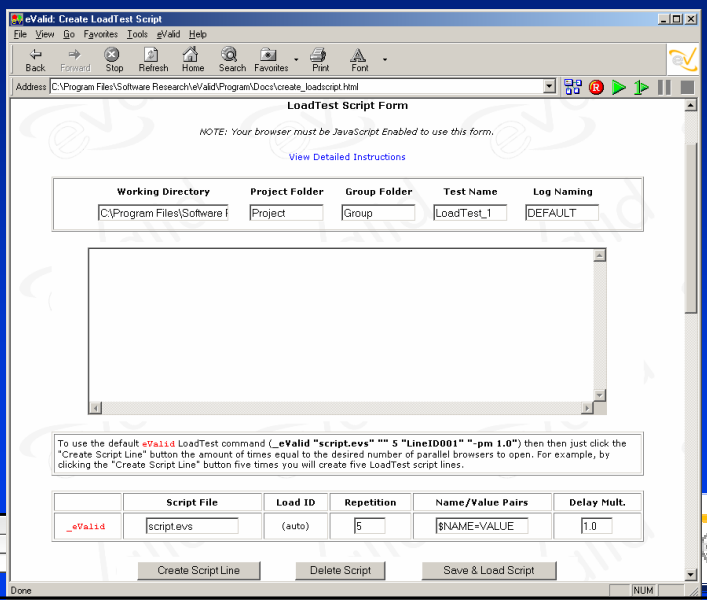
Example Alarm Response



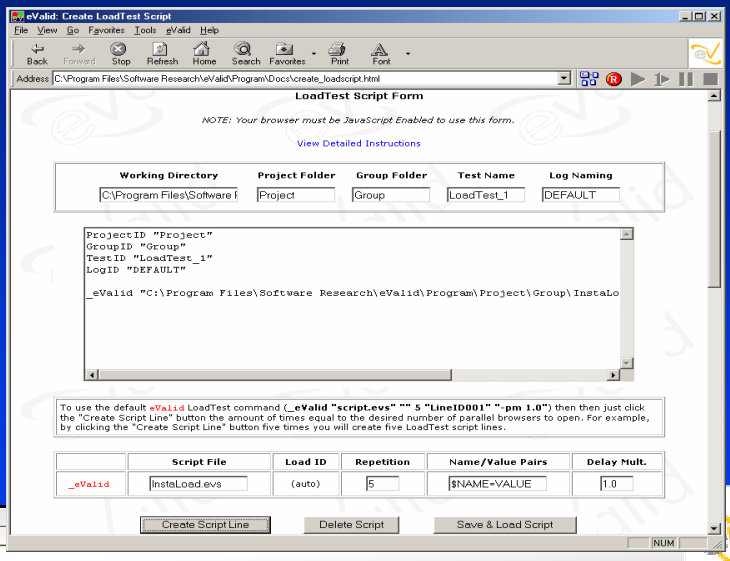
Load and Capacity Checking/Testing

- 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

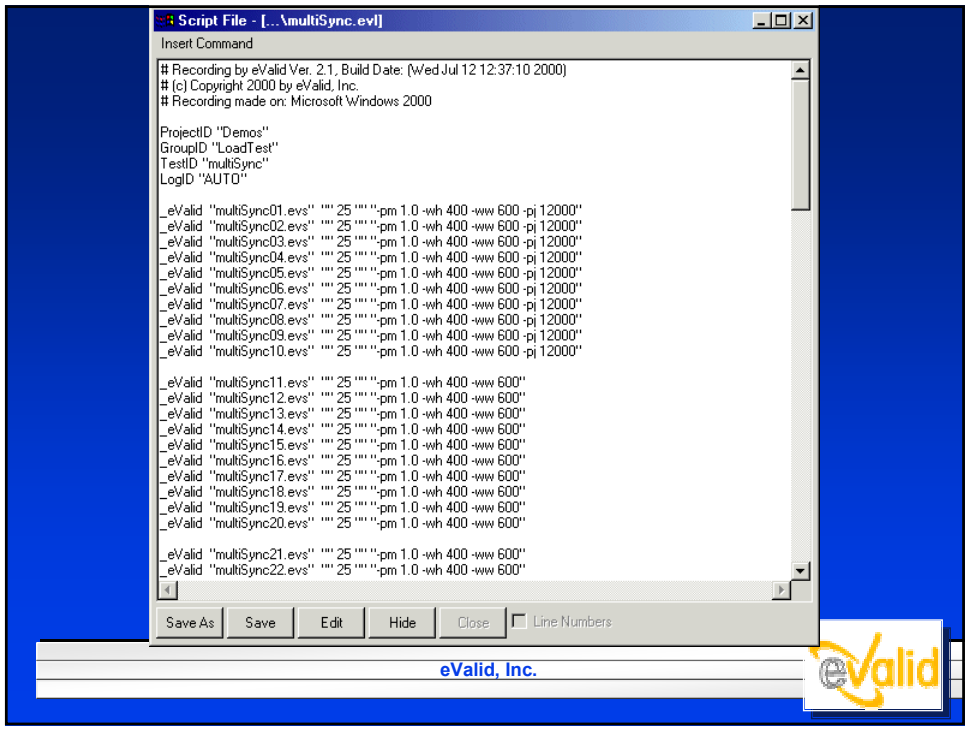
- Artificial “User Class”
- Single *evs file
- Variable Delay Multiplier
- Overall Test Result Timing Aggretation
- No Restrictions on Number of Classes
- No Restrictions on Composition
- Infinite Capability to Emulate Real Users

Machine Capacity Issues

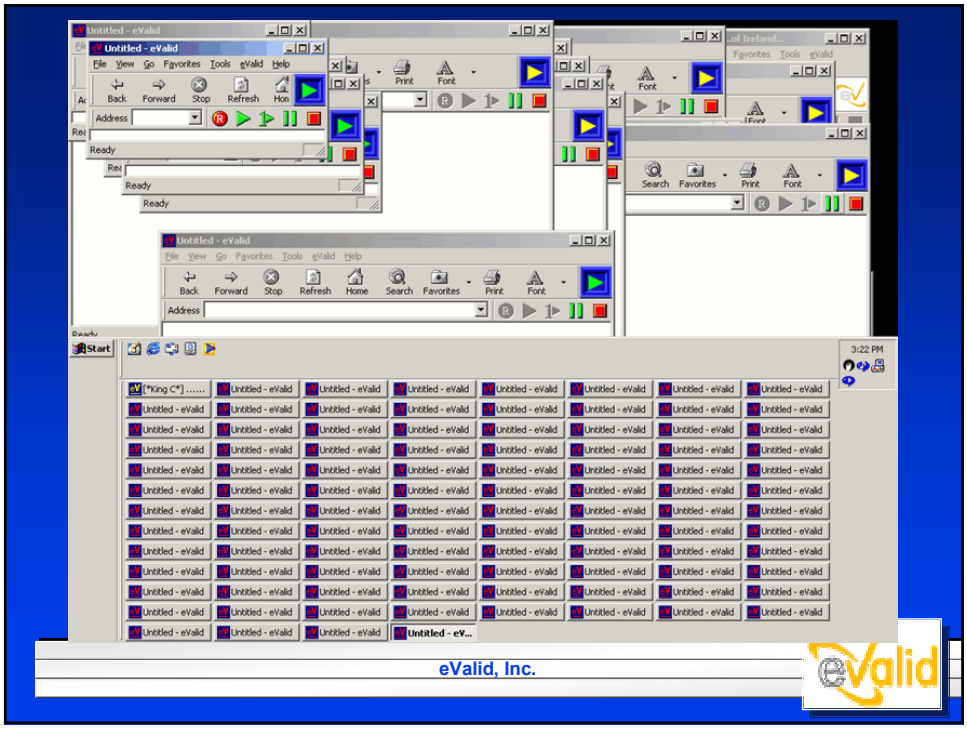
- RAM
- Virtual Memory
- Heap Space
- Machine Bus Speed
- LAN Speed
- Display Driver Memory
- Other

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				

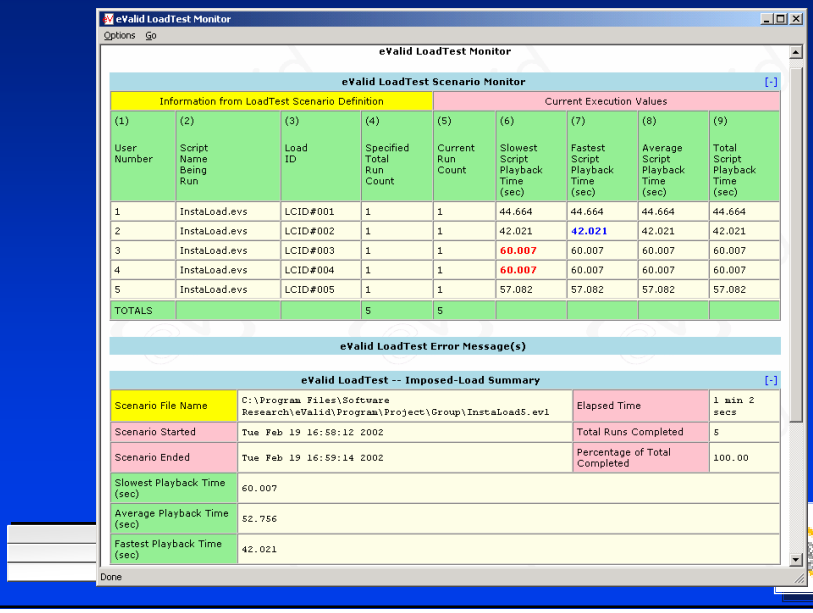
eValid LoadTest Error Message(s)

eValid LoadTest -- Imposed-Load Summary

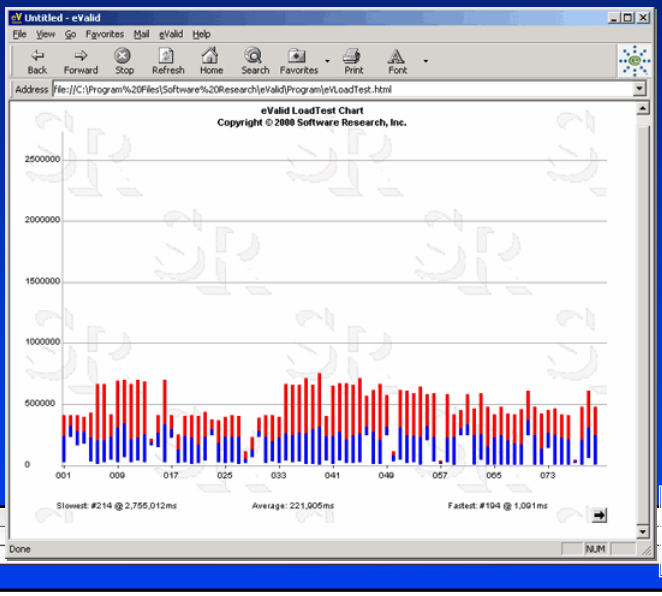
Scenario File Name	C:\Program Files\Software Research\eValid\Program\Project\Group\InstaLoad5.evl	Elapsed Time	47 secs
Scenario Started	Tue Feb 19 16:58:12 2002	Total Runs Completed	2
Scenario Ended	Scenario Still In Progress	Percentage of Total Completed	40.00
Slowest Playback Time (sec)	44.664		
Average Playback Time (sec)	43.342		
Fastest Playback Time (sec)	42.021		

Done

Example LoadTest Final Report



Sample Multi-User Load Scenario Chart



Conclusions & Recommendations

- Nature of WebSite Testing
- Complexity Required
- Realism Requirement

QWE2002 Paper 6T

Dr. Stacy Prowell
(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 results.

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.

Software Quality Research Laboratory

SQRL

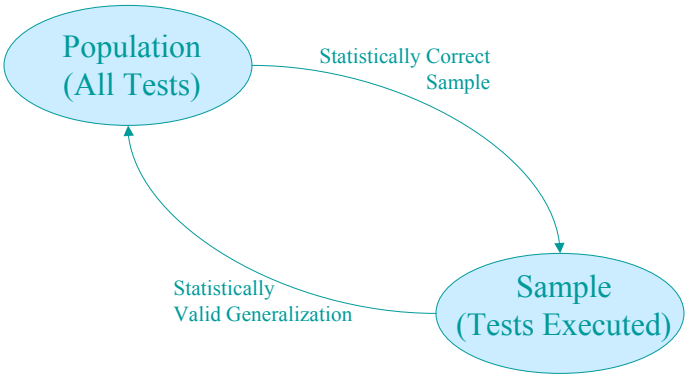
Tool Support for Model Based Statistical Testing

Stacy Prowell
http://www.cs.utk.edu/sqrl/
sprowell@cs.utk.edu

Jump to first page

SQRL

Software Testing



Jump to first page

SQRL

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

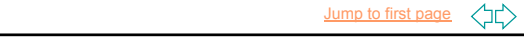
SQRL

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



SQRL

JUMBL

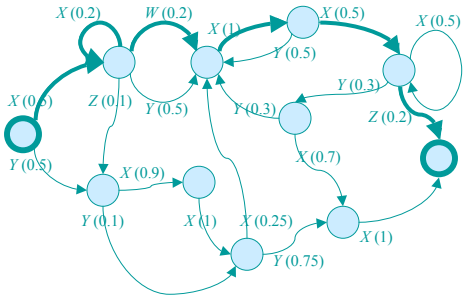
- Java Usage Model Builder Library (JUMBL)
- Java class library and toolkit for statistical testing
- Developed and maintained by the Software Quality Research Laboratory (SQRL)

Jump to first page

SQRL

Characterize Population

- Requires a model of population: Finite-State Markov Chain



$\Pr[X \;\; X \;\; W \;\; X \;\; X \;\; Z] = 0.0002$

- •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

SQRL

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

SQRL

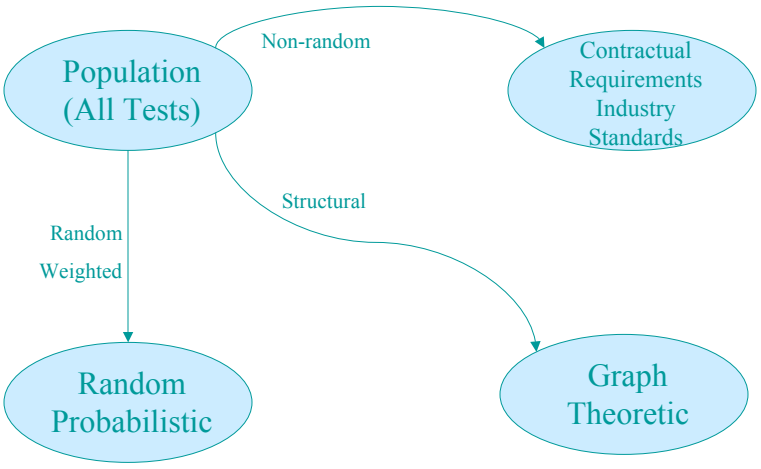
Population Analysis

- 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

SQRL

Test Selection

- **Importance sampling**:
  JUMBL: GenWeighted
- **Graph theoretic**:
  JUMBL: GenMinCover
- **Random walk**:
  JUMBL: GenTest

Jump to first page

SQRL

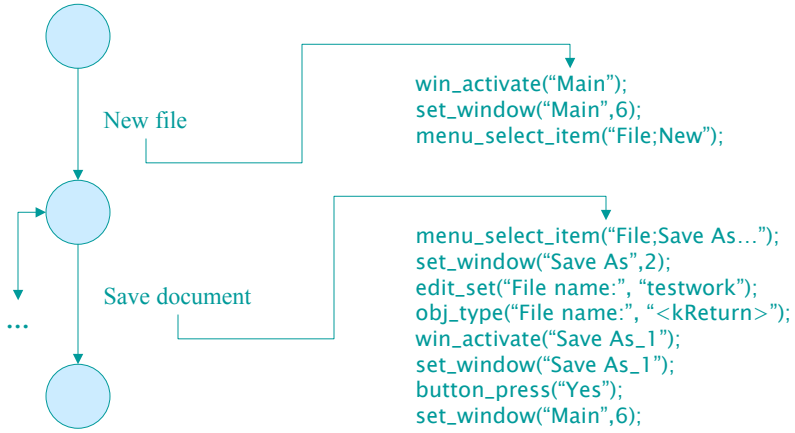
Test Execution

- 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

SQRL

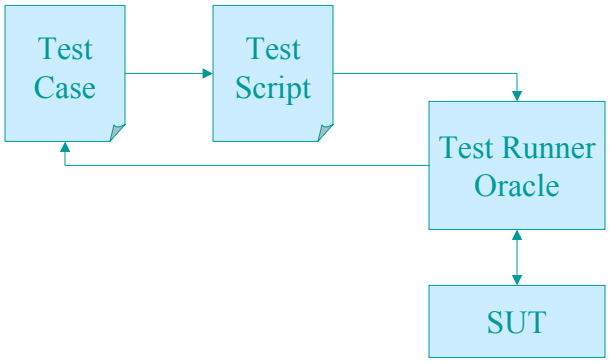
Automated Testing



Jump to first page

SQRL

Automated Testing



Jump to first page

SQRL

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
  - For some cases, let a human decide

Jump to first page

SQRL

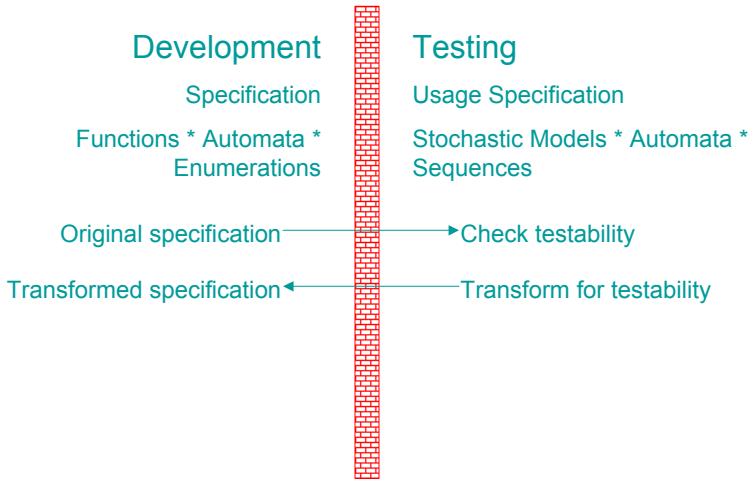
Test Analysis

- Record information from tests
  JUMBL: RecordResults
- Obtain reliability, MTTF, confidence, and test sufficiency
  JUMBL: WriteTestReport

Jump to first page

SQRL

Testability



Jump to first page

SQRL

Tool Availability

- JUMBL is available from U.T. SQRL under license agreement
- Contact sprowell@cs.utk.edu for more information

Jump to first page

QWE2002 Paper 7T

Dr. Mihaela Barreau, Dr. Jean-Yves Morel & Dr. Alexis Todoskoff
(University of Angers)

State-of-the Art Information on Petri Nets Applied to Software Quality

Key Points

- Advantages of Petri Netsin the embedded systems development
- Realibility of Real Time software/hardware systems (critical temporal specifications).
- Model of functional behavior and failure recovery .

Presentation Abstract

The purpose of this article is to highlight the advantages of Petri Nets (PN) in the embedded systems development (easy understanding, modeling, modularity, structural analysis, bench-marking, simulation) in particular in the design, the checking, the evaluation, and the validation of the reliability of these systems with software and hardware components. It is a bibliographical synthesis presenting the use of PN throughout the cycle of development, within the framework of reliability study.

About the Author

Dr. Mihaela Barreau received the PhD in automatic control and computer science from the University of Angers (France) in 1997. She is associate professor in automatic control and computer science in the Quality & Reliability Engineering Department of the University of Angers. Her research interests are in Petri Nets and complex software/hardware systems reliability.

Dr. Jean-Yves Morel received the PhD in automatic control and computer science from the University of Angers (France) in 1996. He is an associate professor in automatic control and electronics engineering in the Electrical Engineering Department of the University of Angers. His research interests are in the application of Petri Nets to hardware-software systems reliability.

Dr. Alexis Todoskoff received the PhD in automatic control and man-machine systems from the University of Valenciennes (France) in 1999. Since 1999, he is an associate professor in automatic control and software engineering in the Quality & Reliability Engineering Department of the University of Angers. His main research field is in the reliability of software systems.

Quality Week Europe 2002

State-of-the Art Information on Petri Nets Applied to Software Quality

Mihaela BARREAU
Jean-Yves MOREL
Alexis TODOSKOFF

University of Angers - LASQUO
Laboratory of Reliability, Quality and Organization

Contents

1. **Software Crisis**
2. **Software Quality**
3. **Petri Nets : a powerful modeling tool for complex systems**
4. **Applications of Petri Nets**
5. **Concluding remarks**

❶ Software crisis

Software development

User's requirements

Architectural design

Functional specification

Programmer's results

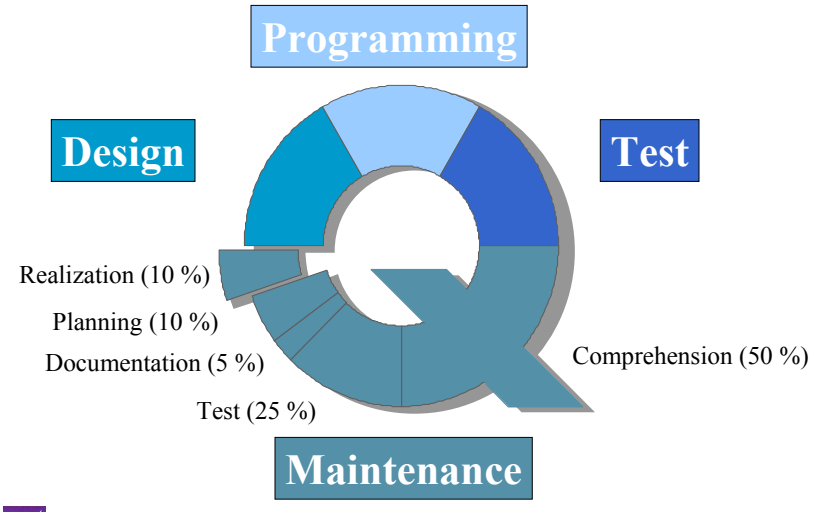
Installation and check-out

What was really needed ...

Some figures … /Laprie 01/, /Printz 95/

- **Development costs and delays**
  - Average costs (70%) & mean delay (50%) overflow
- **Software quality**
  - 80% of software doesn't satisfy user's needs
- **Software reliability**
  - low (patients deaths, ATT, Ariane 5, ...)
  - 60% of failures = design faults
- **Software maintenance**
  - difficult, expensive (60% of the global costs ; space shuttle some 100 M $ /an) and providing new errors

Effort distribution… /IBM 98/



State-of-the Art Information on Petri Nets Applied to Software Quality

❷a) Software quality

Failures and solutions

■ Failure causes	■ Some solutions
• over evaluation of know-how	• concept exploration, skills management
• too much confidence	• process control
• bad organization	• project management
• bad specifications	• prototyping, acceptance tests
• too many modifications	• configuration management
• inappropriate validation	• tests, inspections, reviews

⇨ **Software Quality Assurance**

2a

General Quality Concepts

Quality of Computer Systems

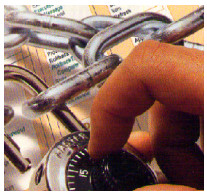
Process Quality
(CMM, ISO SPICE, TRILLIUM, …)

Product Quality (software quality)
(ISO 9126, Mac Call, …)

2a

Development Process Quality

- **C.M.M.**
  (Capability Maturity Model)



➔ **Five maturity levels :**

1. Initial (oblivious)	⇨ **Uncertainty**
2. Repeatable (variable)	⇨ **Awakening**
3. Defined (routine)	⇨ **Enlightenment**
4. Managed (steering)	⇨ **Wisdom**
5. Optimized (anticipating congruent)	⇨ **Certainty**

2a

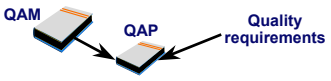
Software Quality (ISO 9126)

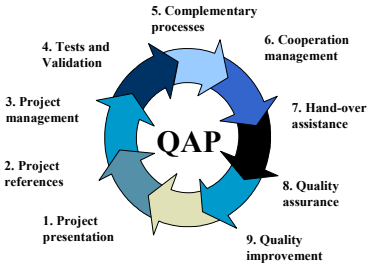
Functional capacity	Reliability	Usability	Efficiency	Maintainability	Portability
• correctness	• maturity	• easiness to comprehend	• temporal behavior	• easiness to analyze	• adaptability
• inter-operability	• fault tolerance	• easiness to teach	• resource consummation	• easiness to modify	• easiness to install
• conformity	• safety	• easiness to use		• stability	• reusability
• integrity				• testability	• inter-operability

2a

Quality Assurance and Control

- **P.S.Q.A. management**
  - *project management*
  - *configuration management*
  - *documentation management*
  - *software engineering*
  - *tests & validation*
  - *quality assurance and improvement*
- **Re-readings, check-lists**
  - *self-control*
  - *simple or crossed*
  - *programming rules*
- **Inspections**
  - *improvement possibilities*
  - *corrections*

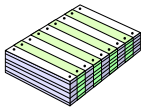




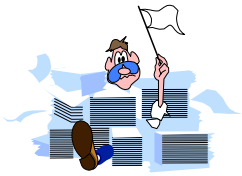
2a

Quality Assurance and Control

- **Reviews (phases & documents)**
  - *phase quality level verification*
  - *document contents validation*

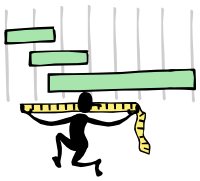


- **Project audit**
  - *project quality evaluation*
  - *efficiency evaluation*
  - *corrective actions planning and monitoring*
- **Estimators & quality measurement**
  - *corrective actions*
  - *improvement actions*
  - *estimators for the development quality*
  - *estimators for the software quality*



2a

Software metrology

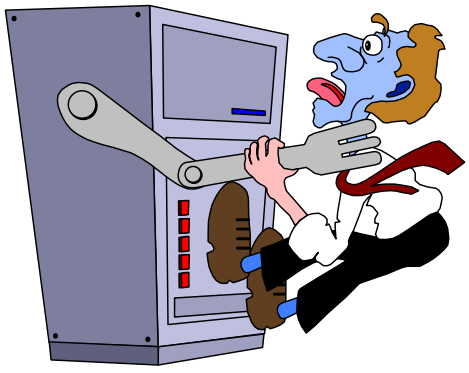


- **Intuitive measures**
- **Textual complexity measures** /Halstead 77/
- **Structural complexity measures** /McCabe 76/
- **Architectural complexity measures**

❷a Software testing

- **Finding non-conformities**
- **Testing activities**
  - module testing
  - integration testing
  - validation
- **Testing techniques**
  - structural
  - functional
  - stochastic

❷b) Computer systems reliability



2b

Computer systems reliability

Software reliability

Maintainability
analyzability, testability

Reliability
fault tolerant, safety

Security
confidentiality, integrity

Availability
responding to requests

2b

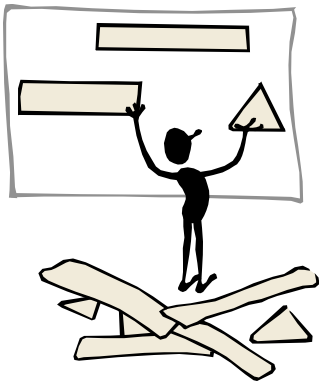
Reliability analysis

- **Risks identification**
- **Reliability construction**
  - fault tolerant software
  - defensive programming
  - diversification
- **Analysis methods**
  - Fault trees
  - AMDE (AEEL)
  - Behavioral analysis : **Petri Nets**
  - Specifications and architecture design
- **Reliability evaluation and proofs**

❷b Some standards

- **CEI 60880** (86) **:** *Software for safety systems used in nuclear energy production.*
- **CEI 61508-3** (97) **:** *Software engineering indications.*
- **DO-178 B** (93) **:** *Software analysis for system and equipment certification*
- **CEI 60300-3-X** (92) **:** *Software aspects of dependability*
- **NF F 71-011/012/013** (90) **:** *Software reliability - General purpose / Software constraints / Analysis methods for software safety*
- **EN 50126** (95) **:** *Railways applications. The specification and demonstration of dependability, reliability, availability, maintainability and safety (RAMS)*
- **ECSS-Q80A** (96) **:** *Space Product Assurance : Software Product Assurance*

❸ Petri Nets : a powerful modeling tool for complex systems

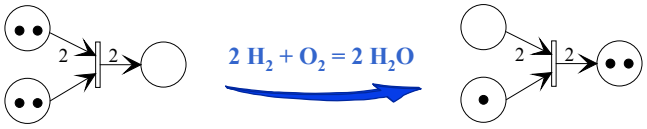


Why Petri Nets ?

- easily comprehensible
- huge modeling capacities
- flexibility and modularity
- structural analysis
- time representation
- performance evaluation
- discrete event simulation
- Markov chains, default trees
- available software

Petri Nets (PN) (1)

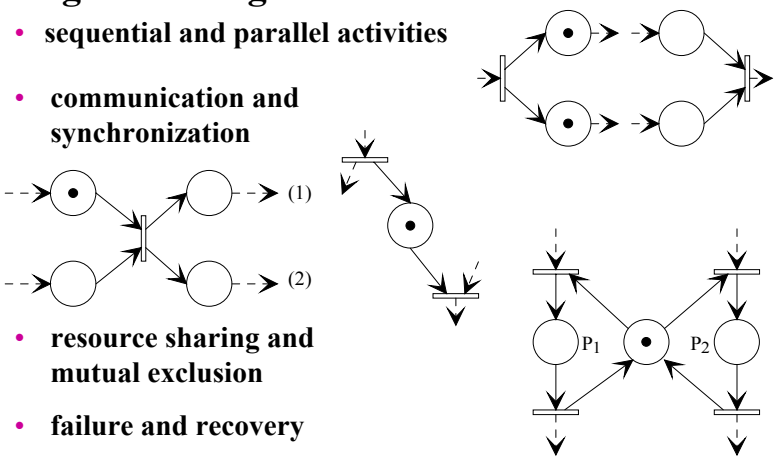
- **mathematical and graphical tool**
  - static structure : places, transitions, valued arcs
  - dynamic structure : marking, evolution rules



  - mathematical representation
  - structural and behavioral properties
    - linear algebra
      - marking evolution equation
      - marking and firing invariants

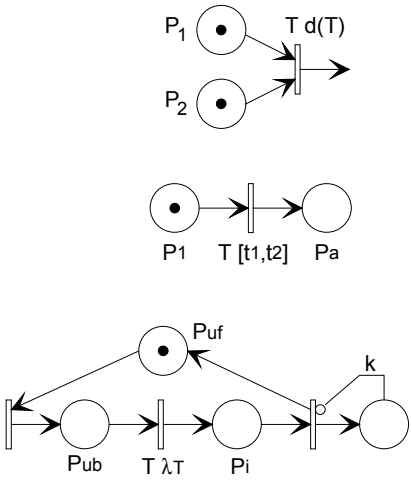
Petri Nets (PN) (2)

- **large modeling abilities**
  - **sequential and parallel activities**
  - **communication and synchronization**
  - **resource sharing and mutual exclusion**
  - **failure and recovery**



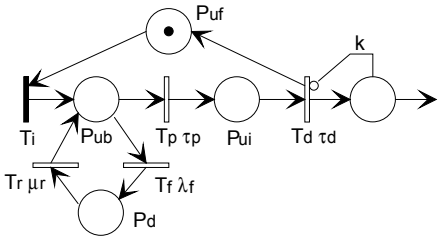
Non autonomous PN (1)

- **Timed PN**
  - timing (delay)
- **Temporal PN**
  - sensitivity interval
- **Stochastic PN**
  - firing rates
  - inhibitor arcs
  - messages



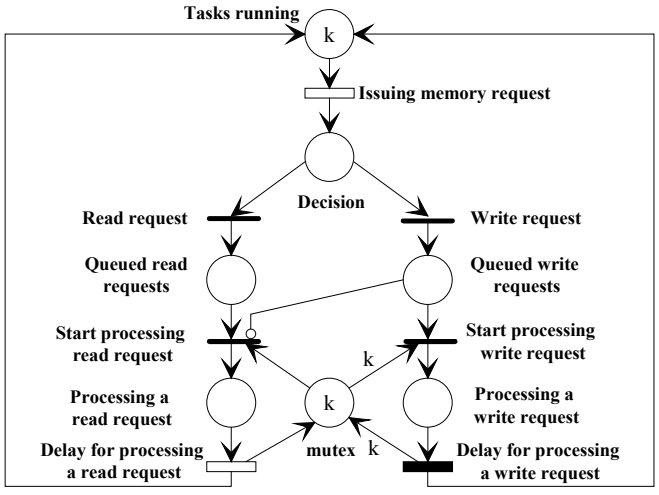
Non autonomous PN (2)

- **Generalized stochastic PN**
  - transitions
    - **stochastic**
    - **immediate**
  - marking
    - **tangibles**
    - **vanishing**



- **Deterministic and stochastic PN**
  - stochastic, immediate and timed transitions
  - *free choice deterministic transitions
  - *queuing models and coloration

Non autonomous PN (3)



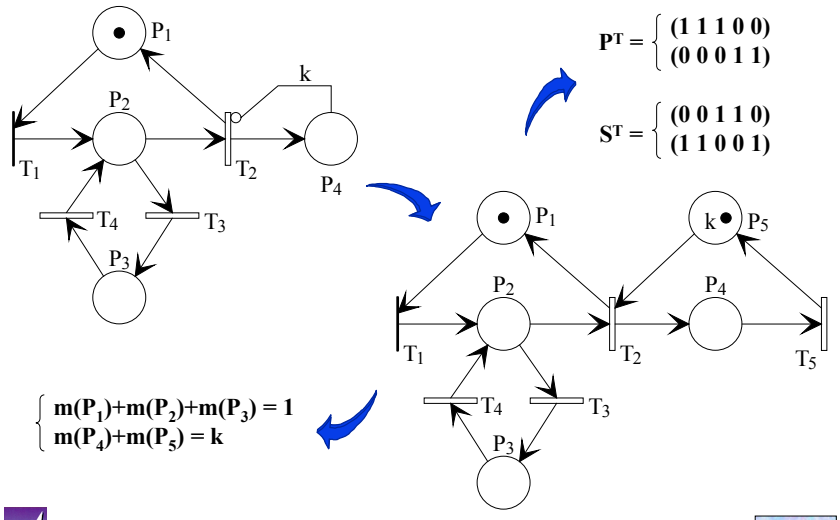
Non autonomous PN (4)

- **Structural analysis**
  - Ordinary underlying PN
    - marking and firing invariants
    - structural and behavioral properties
- **Behavioral analysis**
  - marking graph
    - embedded Markov chain
    - firing rate
    - steady state probabilities

Non autonomous PN (5)

- **Performance evaluation**
  - estimation
    - probability of a special condition
    - average marking value
    - mean firing rate
  - discrete event simulation
    - statistical (estimators)
    - functional (system state)
- **Evolution surveillance**
  - periodical tests

GSPN : example (1)

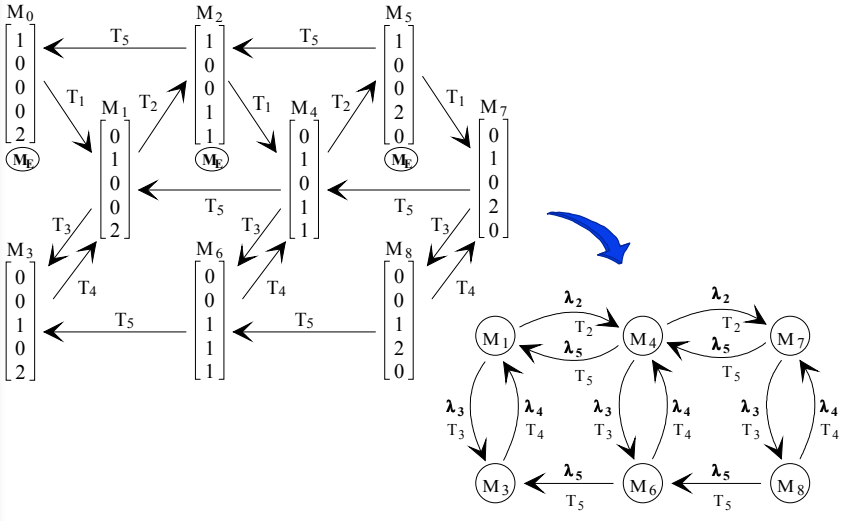


$$P^T = \begin{cases} (1\ 1\ 1\ 0\ 0) \\ (0\ 0\ 0\ 1\ 1) \end{cases}$$

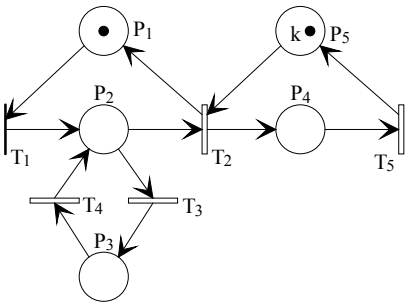
$$S^T = \begin{cases} (0\ 0\ 1\ 1\ 0) \\ (1\ 1\ 0\ 0\ 1) \end{cases}$$

$$\begin{cases} m(P_1)+m(P_2)+m(P_3) = 1 \\ m(P_4)+m(P_5) = k \end{cases}$$

GSPN : example (2)



❸ GSPN : example (3)



Mean running delay
$Pr\ \{m(P_2) = 1\}$

Full buffer probability
$Pr\ \{m(P_4) = 2\}$

Dead time for execution
$Pr\ \{m(P_3) = 1\} + Pr\ \{m(P_4) = 2\}$

Mean outing rate
$\lambda_5.Pr\ \{m(P_4) \neq 0\}$

Average inventory
$E[m(P_4)] = 1.Pr\{m(P4) = 1\} + 2.Pr\{m(P_4) = 2\ \}$

❹ Applications of Petri Nets

Applications of PN

- Chemical reactions
- Organizational process modeling
- Social & technical system modeling
- AI and ES : knowledge modeling and analysis
- OR : activities diagrams (MPM et PERT)
- Logical time, fuzzy systems, neural networks, programmable circuits (VLSI)
- Mathematics (accessibility)

Applications in engineering

- Control and supervision (FMS)
- Telecommunications (protocols, ARINC629)
  - radar for Rafale (avionics)
  - network systems for "Charles de Gaulle" ???
- Real time Software
  - aerial traffic control (CAUTRA)
  - transports : embedded RT (TGV)
  - airbags (Siemens)
  - multiprocessors and shared memory (Bull)
  - nuclear energy monitoring and control (EDF)

PN and Reliability (1)

- **Qualitative characteristics**
  - Ordinary PN
  - Fault tolerant techniques and diagnosis
  - Hardware and software specification and verification (PN-based formal methods)
    - UML and SDL (ITU) are included in PN
- **Quantitative characteristics**
  - Stochastic PN (G)
  - Distributed real-time systems evaluation
  - Periodical test (surveillance)

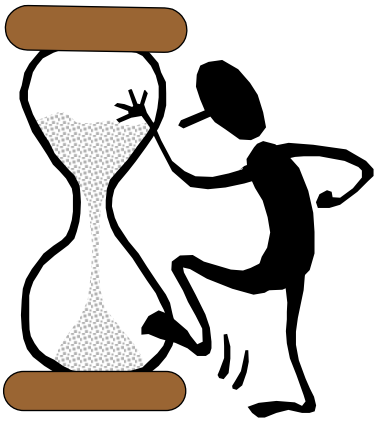
PN and Reliability (2)

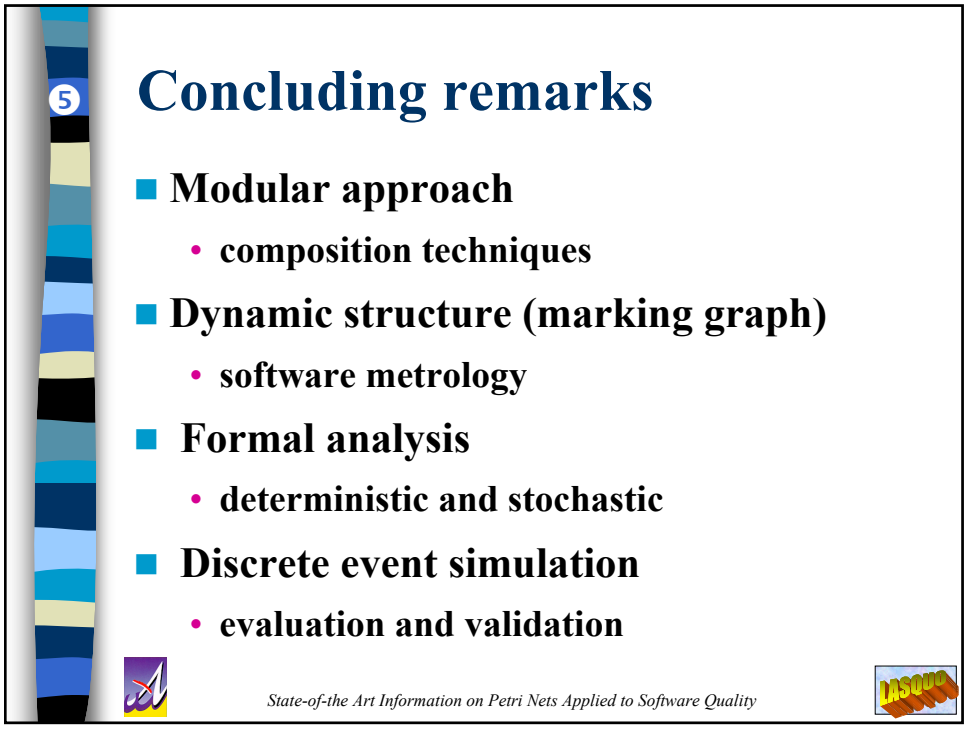
- **Real time systems**
  - **quality of service**
    - response delay, temporal conformity
    - behavioral validity (safety, integrity, blocking situations)
  - **tools must allow**
    - verification (temporal constraints : worst case)
    - evaluation
      - functional aspects : response delay
      - reliability aspects : MTTF
  - **limited analysis techniques**
    - critical systems validation
  - **deterministic and stochastic techniques**

PN and Reliability (3)

- **distributed systems**
  - modularity, abstraction ⇨ composition techniques
- **software**
  - control graph ⇨ marking graph
- **maintenance**
- **tools**
  - PN ⇨ fault trees and RBD

❺ Concluding remarks




5
Concluding remarks
Modular approach
composition techniques
Dynamic structure (marking graph)
software metrology
Formal analysis
deterministic and stochastic
Discrete event simulation
evaluation and validation
State-of-the Art Information on Petri Nets Applied to Software Quality

State-of-the Art Information on Petri Nets applied to Software Quality

Mihaela Barreau, Jean-Yves Morel, and Alexis Todoskoff

University of Angers
Reliability, Quality, and Organization Laboratory (LASQUO)
62, avenue ND du Lac - 49000 ANGERS – France
Tel.: (33) 2 41 22 65 20 - fax: (33) 2 41 22 65 21
Email: mihaela.barreau@univ-angers.fr

Index Terms - Petri Nets, Modelling, Simulation, Reliability, Computer System, Failure, Software, Validation, Checking.

I. Introduction

The increasing size of software and its integration in most industrial sectors make crucial the problem of its reliability, availability, and maintainability. Reliability is not an objective for a software alone, it is declined in a whole set of requirements concerning the software in its environment (hardware, procedures of exploitation and integrated software and hardware).

To eliminate every fault contained in systems with software components, during the design process or at the end of the development, is a Utopia: size and complexity prevent zero-defect software from existing.

These difficulties are still increased in the particular case of systems (possibly distributed) known as «Real Time», for which there are critical temporal specifications that must be respected. These systems have additional requirements in term of quality of service: their temporal accuracy and behavioural validity (security, safety, deadlock avoidance) are fundamental.

It is thus necessary to implement fault-tolerance mechanisms, while guaranteeing the absence of alarming events whose consequences would be considered to be too serious.

The specification of such systems requires models integrating both temporal constraints and stochastic aspects (temporal distributions). They must enable the checking of temporal constraints (in particular, worst cases), of evaluation of the functional aspects (like the response time) and of reliability estimators, e.g. M.T.T.F. In addition, the systems being distributed, their modelling requires concepts of abstraction and composition of models.

II. Why Should we use Petri Nets ?

Petri Nets (PN) are distinguished among the numerous models at the disposal of reliability engineers. Indeed, they can be employed throughout the process development cycle: one can thus preserve the same formalism to understand the architecture and the behaviour of a process (as well as functional analysis) but also to model its failing behaviour (like Markov graphs or failure trees).

This versatility is still increased by their graphical and mathematical representations.

The first one is of simple comprehension, thus supporting the dialogue between designers and reliability engineers, while the second has the power of expression of a formal method, while authorizing simulation (both functional and statistical).

To represent the dynamics of a system obliges to resort to a Markov approach or to Petri Nets. However, the Markov approach quickly reaches its limits. Beyond ten components, only PN associated with a Monte Carlo simulation remain usable. Moreover, PN modelling makes it possible to analyse a priori as well as a posteriori (at the critical phases of validation and specification) the specific properties of the model for nominal and out-of-order operations.

The behavioural or performance analysis enables to highlight possible problems of incorrect or incomplete definition of the operating modes or transitions between these modes (specifications). In the same way, one can highlight problems of dimensioning, scheduling, resource sharing, inputs/outputs synchronization or related to communication protocols.

«PN prove to be the best tool to establish a model of functional behaviour and failure recovery and they are often the only possible approach. They present moreover the major advantage to remain comprehensible» [2].

III. Basic Concepts of Petri Nets

Formally, a Petri net is a graph made from two types of nodes: *places,* which describe the states of the modelled system and *transitions,* which represent the changes of state. Furthermore, directed arcs connect some places and some transitions. A place can contain an integer of tokens (or marks); the set of the marks present at a given moment constitutes *the marking* of the network, i.e. a description of the state of the system.

To any marking accessible from the initial marking (by a mechanism described further) corresponds a state of the system. A place alone is not a state (as for a state graph) but it takes part, through its marking, with the description of the global state of the system. The whole set of accessible markings represents the set of all possible states.

A transition has a certain number of arcs upstream (or input arcs) and arcs downstream (or output arcs); a positive integer called *weight of the arc* is associated with each arc. If all the arcs are unit weighted, one speaks about *ordinary* PN; they are otherwise *generalised* PN.

A. Formal Definition

A PN is a 4-uple R = (P, T, Pre, Post) where:

P = (P1, P2, ..., Pm) is a finished set of places, with $m = |P|$,

T = (T1, T2, ..., Tn) is a finished set of transitions, with $n = |T|$ and $P \cap T = \varnothing$,

Pre: $P \times T \rightarrow N$ is the front incidence application,

Post: $P \times T \rightarrow N$ is the back incidence application.

The Pre application gives the weights of the arcs placed upstream of the transitions, while the Post application gives the weights of the arcs placed downstream from the transitions.

A *marked network* is a couple (R, *M).* The marking *M* of a PN is an application of P in N. *M(p)* represents the number of tokens contained in the place *p.*

B. Matrix Formalism

M can be represented by a vector, Pre and Post by matrices whose lines are subscripted by *the* number *m* of places and the columns by the number *N* of transitions.

One calls the *front incidence matrix* the matrix $W^- = [w_{ij}^-]$, where $w_{ij}^- = Pre(Pi, Tj)$ and *back incidence matrix,* the matrix $W^+ = [w_{ij}^+]$, where $w_{ij}^+ = Post(pi, Tj)$.

C. Transition Firing

PN enable the representation of the dynamics of a system thanks to the mechanism of transition firing. A transition *T* is said to be enabled (or fireable) for a marking *M* if each place located upstream (places of input) has a number of tokens higher or equal to the weight of the arc joining this place to *T,* i.e. if and only if (iff) $\forall p \in P, M(p) \geq Pre(p,t)$.

The firing (or shooting) of *T* leads to the new marking *Me* such as:

$\forall p \in P, Me(p) = M(p) - Pre(p,t) + Post(p,t)$.

The difference between markings *M* and *M'* *such as* $M \xrightarrow{t} M'$ *is* equal to *Post (p,t) - Pre (p,t).* One introduces *the incidence matrix* $W = W^+ - W^- = [w_{ij}]$ such as:

$\forall p \in P, \forall T \in T, W(p,t) = Post(p,t) - Pre(p,t)$.

A column of this matrix corresponds to the modification of marking brought by the firing of the corresponding transition. It should be noted that the terms from the front and back incidence matrices should not be compensated in order not to lose information at the time of the transcription of the model in matrix form. This is why one excludes PN such as a transition firing withdraws and adds marks in the same place. If this condition is observed, they are "pure" PN.

D. Analysis Method

The method of analysis we detail leads to the determination of structural properties of the model independently of the initial marking (i.e. invariant properties). Therefore, analysis by enumeration is no longer useful.

1. T-invariant

A marking *Me,* deduced from *M0* after a firing sequence *S,* is connected to *M0* by the state equation:

$Me = M0 + W.\ \underline{S}$

where the incidence matrix W represents the structure of the network and $\underline{S}$ is the characteristic vector associated with *S.*

Let us consider the particular case where *Me = M0;* the sequence of firings $\underline{S}$ thus left the system in its initial state. One deduces the definition of a *transition invariant*:

A vector $\underline{S}$ of dimension *N* (N is the number of transitions of the PN considered) and whose components are integer and positive or null is *a T-invariant* iff $W.\ \underline{S} = 0$. $\underline{S}$ thus belongs to the right kernel of *W*.

2. P-invariant

In the same way, a vector $\underline{X}$ of dimension *m* (m is the number of places of PN considered) and whose components are integer and positive or null is *a P-invariant iff* $\underline{X}^T . W = 0$.

Let us take again the fundamental equation, valid for any marking *Me* accessible by a firing sequence $\underline{S}$, and multiply on the left each of its terms by $\underline{X}^T$:

$\underline{X}^T.\ Me = \underline{X}^T.\ M0 + \underline{X}^T .\ W.\ \underline{S}$

If $\underline{X}$ is a P-invariant, then $\underline{X}^T . W = 0$. One then obtains: $\underline{X}^T.\ ME = \underline{X}^T.\ M0$

This equality is valid whatever $\underline{S}$, i.e. for any accessible *marking* Me starting from *M0.* Thus, $\underline{X}^T.\ M'$ *is* a marking invariant. In

other words, the number of marks in the whole set of the places which correspond to the non null components of $\underline{X}$ (i.e. its *support)* balanced by vector $\underline{X}$ is constant.

3. Global properties

The invariants of PN enable the deduction of certain global properties of the system such as reversibility, deadlock freeness or capacity overflow.

For example, a PN which all places belong to a P-invariant support *is bounded,* i.e. its number of marks is limited. The number of accessible states is thus finished, which is always the case of a physical system. The specification of a real system by PN must be bounded.

The majority of industrial processes have repetitive operations. A necessary condition for their PN model to be *reversible* is that they have a maximum support T-invariant.

Let us note finally that there are PN graphical analysis methods. They are based on the fact that PN can be obtained by composition of known sub-PN (i.e. basic circuits) in such way that the local properties remain valid globally.

Lastly, all types of PN can take temporal characteristics (deterministic or stochastic) into account, for performance analysis and optimisation.

IV. Applications of Petri Nets

Applications of PN can be found in many various fields such as chemical reactions, organisation and socio-technical systems, artificial intelligence and expert systems (for knowledge and reasoning modelling), operational research (for PERT or MPM activity diagrams), temporal and fuzzy logic, artificial neural networks, VLSI circuits and mathematical modelling (accessibility).

Despite these many application areas, we must admit that PN have mainly been used for the design and analysis of flexible manufacturing systems and computer systems, even though they are likely to interest the software quality community.

A. Reliability of Industrial Systems

Stochastic PN are used by reliability engineers to compute the reliability of industrial systems (we are talking about the system reliability in its full meaning, therefore including software or automata reliability). The association of interpreted stochastic PN as behaviour model with a Monte Carlo simulation is a very efficient way to deal with reliability, availability, production availability, maintainability, dependability, and problems. In fact, this is usually the only way tractable without too much approximations for actual industrial systems [22].

B. Protocol Engineering

PN are used for protocol engineering. This methodology provides various models for specification and many methods for verification and other software engineering tasks concerning protocols. In particular, many property-preserving transformations and compositional methods are available for reducing the impact of state explosion on the two well-known verification approaches – reachability analysis and invariant analysis -. By conversion, PN can be used for studying systems specified by many of the formal description techniques frequently used for protocol investigation such as MSC, SDL, ESTELLE, LOTOS, CCS, CSP and CCSP. For example, PN can be used for deriving

test sequences and cyclomatic complexity measure for LOTOS. Also, many equivalence relations concerning the theoretical foundation of protocol engineering have been formulated on the basis of PN. Other developments of PN related to protocol engineering include temporal logic, feature interaction, synthesis, complexity measure, timed or object-related PN. A review of the role PN play in protocol engineering can be found in [15].

C. Formal Methods and Performance Analysis

Habitually, models and methods for the analysis of the functional correctness of reactive systems and those for the analysis of their performance (and dependability) aspects, have been studied by different research communities. However, the difference between their functional features and their performance properties has become blurred in modern systems. During the last decade, this trend has motivated an increased interest in combining insights and results from the field of formal methods, traditionally focused on functionality, and techniques for performance modelling and analysis. Examples of this cross-fertilization are extensions of process algebra and PN allowing for the automatic generation of performance models, the use of formal proof techniques to assess the correctness of randomised algorithms, and extensions of model checking techniques to analyse performance requirements automatically. The Formal Methods and Tools Group of the University of Twente has just published a book presenting the state of the art of research and tool development for an integrated approach towards modelling and analysis of functional and performance aspects of reactive systems [16].

V. CONCLUSION

Even if one encounters some concrete cases of use of PN stochastic extensions (typical applications relate to reactive systems control and supervision, telecommunications and distributed software or real time systems), it seems nevertheless useful to diffuse this tool to the reliability community.

The advantages of PN in the embedded systems development (easy understanding, modelling, modularity, structural analysis, bench-marking, simulation) deserve to be highlighted. PN can be used throughout the cycle of development, for the design, the checking, the evaluation, and the validation of the reliability of systems with software and hardware components.

VII. REFERENCES

[1] G. Girault and R. Valk, *Petri Nets for Systems Engineering, A guide to Modelling, Verification, and Applications,* Springer, 2001.

[2] J-P Signoret, *Modelling the Reliability of Dynamic Industrial Systems,* joint meeting ISDF/ Methodology and Tools Group and French-speaking Group on Petri Nets, April 2001.

[3] M. Kaâniche, *Evaluation of data processing Reliability,* PhD Thesis, Toulouse (France), LAAS-CNRS, 1999.

[4] N. Lopez-Benitez, *Petri-Net Based Performance-evaluation of Distributed Homogeneous Systems,* IEEE Transactions on Reliability, vol. 49, June 2000, p. 188-198.

[5] M.M. Hosseini, R.M. Kerr, R.B. Randall, *An Inspection Model with Minimal and Ma-*

jor Maintenance for a System with Deterioration and Poisson Failures, IEEE Transactions on Reliability, Vol. 49, March 2000, p. 88-98.

[6] M. Malhotra, K.S. Trivedi, *Dependability Modeling using Petri Nets*, IEEE Transactions on Reliability, Vol. 44, September 1995.

[7] G. Juanole, *Petri Nets and Communications,* LAAS-CNRS, Toulouse, 1995.

[8] F.T. Sheldon, S. Greiner, M. Benzinger, *Specification, Safety and Reliability Analysis using Stochastic Petri Nets Models*, 10th Int. Workshop on Software Specification and Design 2000, p. 123-132.

[9] M.L. Yin, C.L. Hyde, L.E. James, *A Petri-net Approach for Early Stage System-level Software Reliability Estimation,* Annual Reliability and Maintainability Symposium 2000, p. 100-105.

[10] S. Palazzo, A. Puliafito, M. Scarpa, *Reliability Analysis of a Redundant Database for UMTS Systems,* 7th Int. Workshop on Petri Nets and Performance Models, 1997, p. 247-256.

[11] S. B. Hong, K. Kapsu, *A Reliability Analysis of Distributed Programs with Colored Petri Nets,* IEEE Int. Conf. on Systems, Man, and Cybernetics, Vol. 4, 1997, pp. 3975-3980.

[12] W.G. Schneeweiss, *Petri sub-nets for Minpath-Based Fault Trees*, Annual Reliability and Maintainability Symposium, 2001, p. 161-166, 2001.

[13] Y. Liang, M.A.J. Smith, K.S. Trivedi, *Uncertainty Analysis in Reliability Modeling,* Annual Reliability and Maintainability Symposium, 2001, p. 229-234, 2001.

[14] Noriega-Carranza, D.T. Wang, *RETPN : Reverse Engineering Methodology Based on Timed Petri Nets: a Software Reliability Approach*, IEEE Int. Conf. On Systems, Man, and Cybernetics, pp. 3075-3080, October 1995.

[15] T-Y Cheung, *Petri Nets for Protocol Engineering*, Computer Communications, vol. 19, Issue 14, December 1996, P. 1250-1257.

[16] E. Brinksma, H. Hermanns, J-P Katoen eds., *Lectures on Formal Methods and performance Analysis*, 1st EEF/Euro Summer School on Trends in Computer Science, Springer Verlag, 2001.

[17] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, P. Schnoebelen, P. McKenzie, *Systems and Software Verification, Model-Checking Techniques and Tools,* Springer, 2001.

[18] J. Arlat, *Composants logiciels et sûreté de fonctionnement, intégration de COTS,* Hermès, Paris, 2000.

[19] M. Roper, *Software Testing*, Mc Graw-Hill, 1994.

[20] F. Bause, P. Kritzinger, Stochastic Petri Nets, An introduction to the theory, Vieweg, Advanced Studies in Computer Science, 1996, Wiesbaden.

[21] C. Lindemann, *Performance Modelling with Deterministic and Stochastic Petri Nets*, Wiley, 1998.

[22] A. Leroy, J-P Signoret, *Le risque technologique*, Presses Universitaires de France, Paris, 1992.

QWE2002 Paper 8T

Dr. Christian Bunse & Dr. Oliver Laitenberger
(Fraunhofer Institute for Experimental Software Engineering)

Improving Component Quality Through the Systematic Combination of Construction and Analysis

Key Points

- Component-based development needs tailored and smoothly integrated quality assurance.
- SORT is a systematic technique for mapping UML models to code in high-quality.
- ACI is a systematic inspection technique for object-oriented projects.

Presentation Abstract

The reliable attainment of quality requirements is still a weakness in component -based development projects. A major reason for this situation is the isolated u se of construction and subsequent analysis activities. This paper describes a practical strategy for addressing this problem. The strategy relies on an effective combination of development guidelines as provided by the Systematic Object-Oriented Refinement and Translation (SORT) technique and rigorous quality checks of the resulting products by means of the architecture-centric inspection (ACI) approach. Both, SORT and ACI, have already been successfully applied to the KobrA development method. However, the synergy effects when used in the context of a combined strategy clearly outweighs the isolated benefits of the techniques. The integration of such a strategy into any object-oriented or component-based development method not only results in a high quality component implementation, but also significantly increases the confidence that the desired quality levels can be attained in a successful manner.

About the Author

Christian Bunse received the PHD degree in computer science from the University of Kaiserslautern Germany and his B.S. (Vordiplom) and M.S. degree (Diploma) in computer science with a minor in theoretical medicine from the University of Dortmund, Germany. From 1993 to 1995 he was a faculty research assistant of the Software-Technology-Transfer-Initiative at the University of Kaiserslautern, Germany. He is currently responsible for object-technology and component-based software development at the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern.

Oliver Laitenberger is currently a researcher and consultant at the the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern. His interests are centered on software quality assurance with software inspections, inspection measurement, and inspection improvement as well as experimentation. As a researcher, Oliver Laitenberger has been working for several years in the development and evaluation of inspection technology. As a consultant, he has worked with several national and international companies in introducing and improving inspections. Oliver Laitenberger received his Ph.D. and M.Sc. in computer science from the University of Kaiserslautern, in 1996 and 2000, respectively, .

Improving Component Quality through the Systematic Combination of Construction and Analysis Activities

IESE

Fraunhofer Institut für Experimentelles Software Engineering

Dr. Christian Bunse
Dr. Oliver Laitenberger

IESE

Component based Software Development

Component Quality
- Motivation
- Quality Strategy
- SORT
- ACI
- Example
- Summary

- COMPONENTS: Self-contained units of software that can be easily assembled into new units of software
- Expected benefits -
  - enhanced reuse
  - enhanced quality
  - rapid application development
  - COTS Marketplace
- BUT
  - Today, being a component means being a JavaBean, CORBA application or COM object
  - ⇒ How should components be represented in earlier phases of the life-cycle, especially in the UML?
  - ⇒ How could the quality of components be assured?

Copyright © Fraunhofer IESE 2001

The KobrA Method

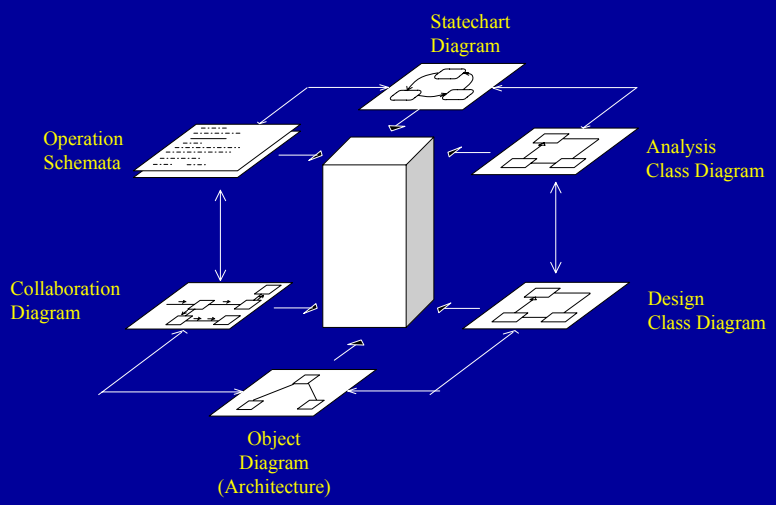
Component Quality
Motivation
Quality Strategy
SORT
ACI
Example
Summary

- Component based development of object-oriented systems
- Main Ingredients:
  - UML based Analysis and Design
  - Systematic Quality Assurance
  - Development of Productlines
- Compatible with other methods (RUP, …)
- 100% support of the UML
- Principles
  - „One man`s component is another man´s system“
  - Active support for component reuse
- Quality Strategies
  - Refinement & Translation Patterns
  - Architecture Centric Inspections

Copyright © Fraunhofer IESE 2001

Kobra Komponente

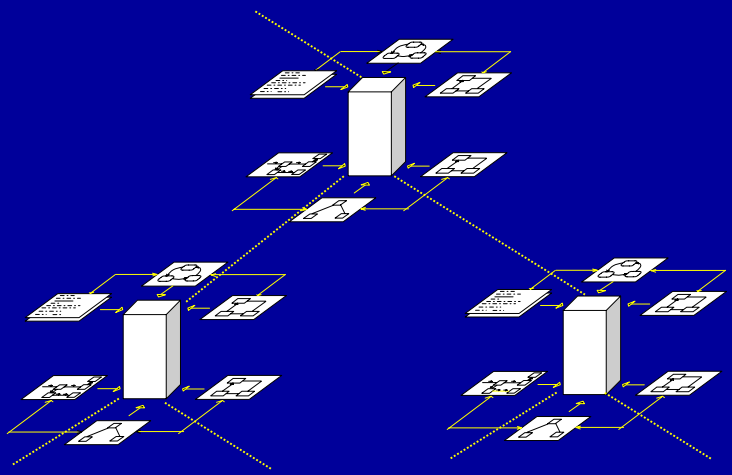
Component Quality
Motivation
Quality Strategy
SORT
ACI
Example
Summary



Copyright © Fraunhofer IESE 2001

The KobrA Method (continued)

Component Quality
Motivation
Quality Strategy
SORT
ACI
Example
Summary

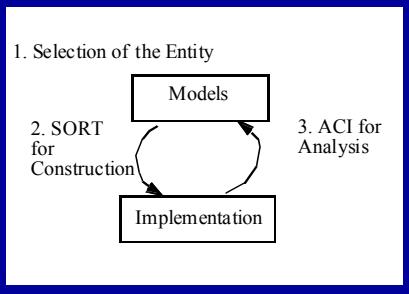


Copyright © Fraunhofer IESE 2001

The KobrA Quality Strategy

Component Quality
Motivation
Quality Strategy
SORT
ACI
Example
Summary

- Seamless integration of systematic construction&analysis techniques
  - Pattern-based refinement and translation of models
  - Architecture-centric inspections for checking quality



Copyright © Fraunhofer IESE 2001

Implementing Component Models

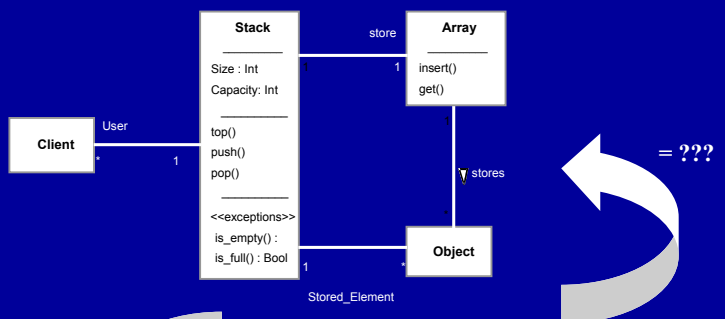
Component Quality
- Motivation
- Quality Strategy
- SORT
- ACI
- Example
- Summary

- The success of the UML reflects the general acceptance that "modeling" is a "good thing".
  - → Models created in the earlier phases are of limited value unless they can be readily mapped into executable forms.
  - → Problems in the transformation path from models to code have
    - a negative impact on the quality.
    - hinder future maintenance and/or reuse.
- The systematic development and mapping of graphical models into code is one of the most neglected links.

Copyright © Fraunhofer IESE 2001

Example

Component Quality
- Motivation
- Quality Strategy
- SORT
- ACI
- Example
- Summary




```
template<class Object-Type>
class Stack
public:
      Stack(int s,int c);      // constructor
      ~Stack();                // destructor

      int push (const T&);
       Object-Type* top();
      Object-Type* pop();
       bool is_empty();
       bool is_full();
private:
      int Size;
      int Capacity;
      Object-Type   *elements[Capacity];
};
```

Copyright © Fraunhofer IESE 2001

KobrA Implementation Strategy

Component Quality
Motivation
Quality Strategy
SORT
ACI
Example
Summary

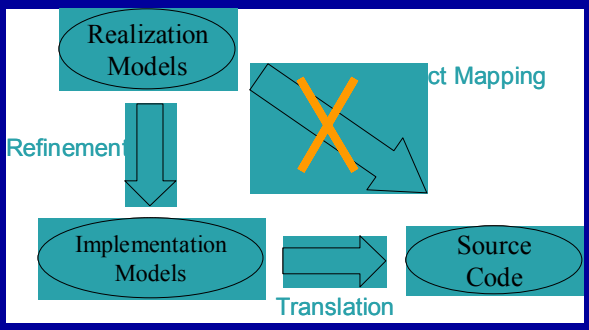
- Realization model must be mapped to an executable form before the component can be executed
- Implementing models involves 2 basic transformations
  - Refinement
  - Translation
- BUT, these are rarely distinguished by most methods -
  - Large semantic gap
  - Undocumented decisions
  - Loss of reuse opportunities
  - Replication of information

Copyright © Fraunhofer IESE 2001

KobrA Implementation Strategy (continued)

Component Quality
Motivation
Quality Strategy
SORT
ACI
Example
Summary

- Cleanly decouple Refinement & Translation -
  - Refine Component Realization Models
  - Then, translate Models to Source-Code

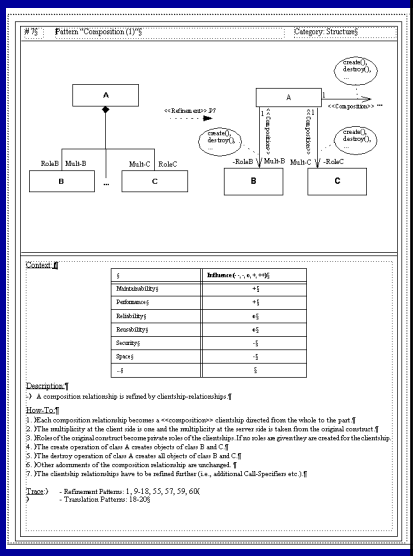


Copyright © Fraunhofer IESE 2001

KobrA Refinement Patterns

Component Quality
Motivation
Quality Strategy
SORT
ACI
Example
Summary

- Predefined Refinements of UML Concepts
- Grouped in Families
- Characterize Impact on nf-Requirements
- Form a Pattern Catalogue
- Are adaptable

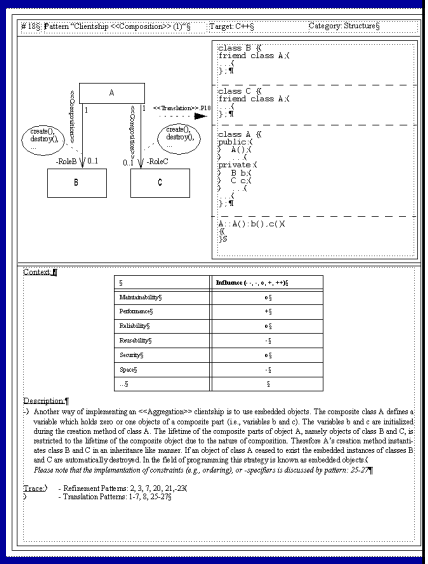


Copyright © Fraunhofer IESE 2001

KobrA Translation Patterns

Component Quality
Motivation
Quality Strategy
SORT
ACI
Example
Summary

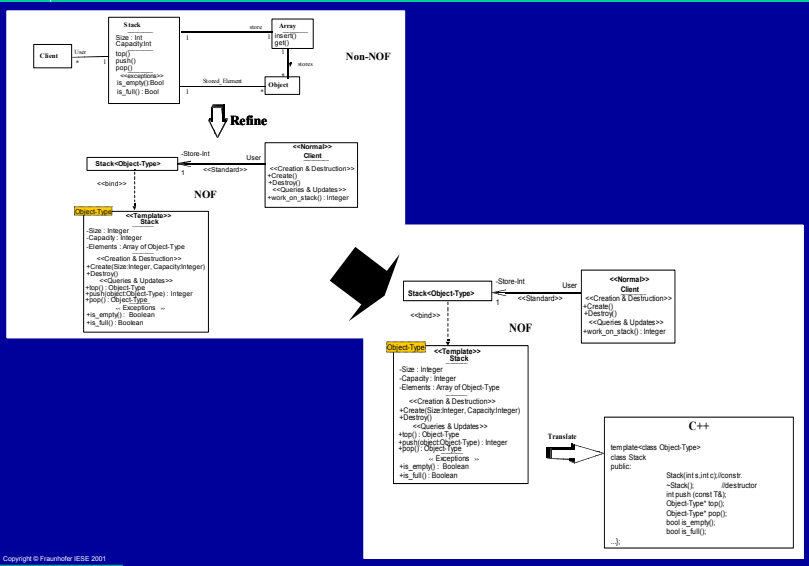
- Predefined Implementations of refined Constructs
- Grouped in Families
- Characterize Impact on nf-Requirements
- Form a Pattern Catalogue
- Are adaptable



Copyright © Fraunhofer IESE 2001

Example

Component Quality
- Motivation
- Quality Strategy
- SORT
- ACI
- Example
- Summary



Copyright © Fraunhofer IESE 2001

Checking Component Quality

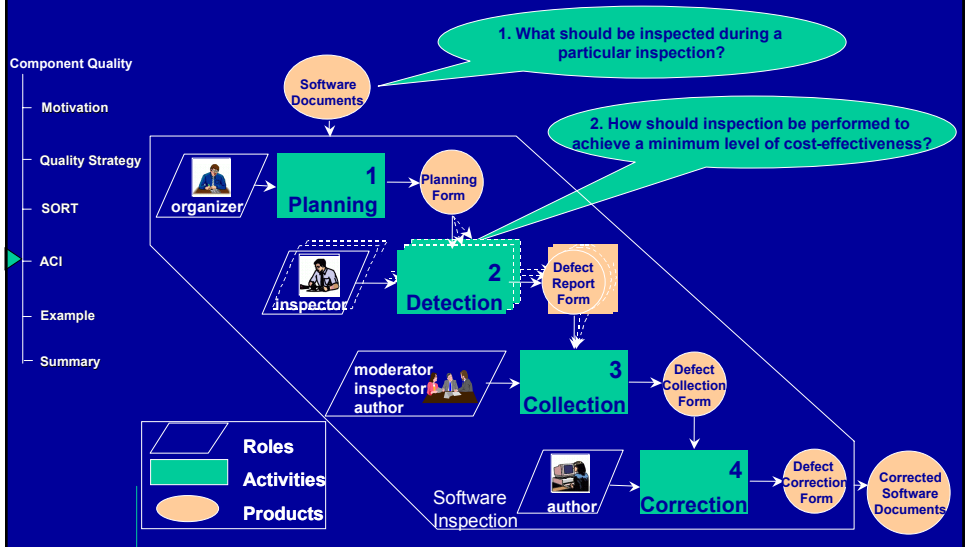
Component Quality
- Motivation
- Quality Strategy
- SORT
- ACI
- Example
- Summary

- Component-based development increases the importance of quality since poor-quality parts lead to a poor end product.
- Most of the leading development processes (Booch, OMT, Unified Process) primarily rely on testing for assuring component quality.
- Testing should be complemented with other quality-enhancing activities, such as inspection
  - One cannot test quality into a component!
  - Many empirical studies show that the costs per defect are much lower for inspection than for testing.
  - Testing does not help detect all defects.
- Even component code needs to be inspected. Because of the focus on reuse, non-functional quality criteria (such as understandability) become increasingly important.
- In KobrA, a specific form of inspection is used: Architecture-centric inspection

Copyright © Fraunhofer IESE 2001

Architecture-centric Inspection

Component Quality
- Motivation
- Quality Strategy
- SORT
- ACI
- Example
- Summary



Copyright © Fraunhofer IESE 2001

Architecture-centric Inspection

Component Quality
- Motivation
- Quality Strategy
- SORT
- ACI
- Example
- Summary

1 . What should be inspected during a particular inspection?

- What is the "right" *chunk* of information for an inspection?
- What is the "right" set of documents/diagrams that describes the chunk of information?

2. How should inspection be performed to achieve a minimum level of cost-effectiveness?

- What is a good reading techniques?

Copyright © Fraunhofer IESE 2001

Architecture-centric inspection organization

Component Quality
- Motivation
- Quality Strategy
- SORT
- ACI
- Example
- Summary

1 . What should be inspected during a particular inspection?

- **Determine the logical entities that warrant inspection (e.g., system, subsystem, component, class operation, or their relationship). This step is also important for refinement and translation**
- **Collect the relevant parts of the documentation of the logical entity (e.g., all relevant UML models and the code).**
  - If the volume of the documentation to be inspected is too large, perform further partitioning.
  - If the volume of the documentation to be inspected is appropriate, use the available documentation for organizing the inspection.
- **Organize and run an inspection.**

Copyright © Fraunhofer IESE 2001

Reading Technique

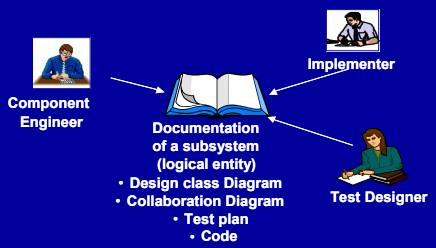
Component Quality
- Motivation
- Quality Strategy
- SORT
- ACI
- Example
- Summary

2. How should inspection be performed to achieve a minimum level of cost-effectiveness?

Two reading techniques are applicable:

- **Checklist-based Reading (is a technique that only describes what to check)**

- **Perspective-based Reading (is a technique that describes what to check and how to perform the checking)**



Copyright © Fraunhofer IESE 2001

Reading Technique: An Example Scenario

Component Quality
Motivation
Quality Strategy
SORT
ACI
Example
Summary

- Tester‘s Perspective

Assume you are inspecting the implementation from the perspective of a tester. The main goal of a tester is to ensure the testability of a component or parts of it. High quality thus corresponds to full testability. Assume that you have to develop some test cases for the implementation. A test case consists of a set of input values plus a set of output values and/or state changes expected for each combination of values. Follow the instructions below and answer the questions carefully.
Define equivalence classes for the input/output values and interfaces. Use these equivalence classes to define a minimal set of test cases to fully exercise the behavior including the public interfaces of the component.
While following the instructions answer the following questions:

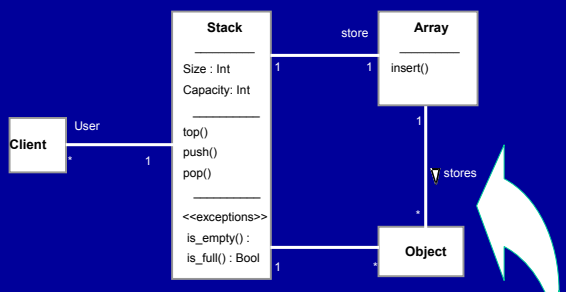
Are all possible sets of input values properly addressed by the provided diagrams?

Are operations preconditions indicated?

Copyright © Fraunhofer IESE 2001

Example

Component Quality
Motivation
Quality Strategy
SORT
ACI
Example
Summary




```
template<class Object-Type>
class Stack
public:
      Stack(int s,int c);      // constructor
      ~Stack();                // destructor

      int push (const T&);
       Object-Type* top();
      Object-Type* pop();
       bool is_empty();
       bool is_full();
private:
      int Size;
      int Capacity;
      Object-Type   *elements[Capacity];
};
```

Tester-scenario

ExampleTest-Cases:
top ();
pop ();
push (object)

Copyright © Fraunhofer IESE 2001

Summary and Conclusions

Component Quality
- Motivation
- Quality Strategy
- SORT
- ACI
- Example
- Summary

- Component quality becomes increasingly important in today's development projects.
- The combined strategy of using SORT and ACI helps improve design and code quality.
  - Sort focuses on systematic refinement and translation of UML design models.
  - ACI involves the rigorous scrutiny of the artifacts created.
- The benefits derive from the prescriptive guidance that is offered to developers.
- Although both technologies have been succesfully integrated in the KobrA-development method, they are also promising in the context of other approaches.
- Future work includes the empirical investigation of the strength/weaknesses of the presented strategy.

Copyright © Fraunhofer IESE 2001

Improving Component Quality through the Systematic Combination of Construction and Analysis Activities

Christian Bunse and Oliver Laitenberger

Fraunhofer Institute for Experimental Software Engineering

Technopark II

Sauerwiesen 6

D-67661 Kaiserslautern, Germany

{Christian.Bunse,Oliver.Laitenberger}@iese.fhg.de

ABSTRACT

The reliable attainment of quality requirements is still a weakness in component-based development projects. A major reason for this situation is the isolated use of construction and subsequent analysis activities.

This paper describes a practical strategy for addressing this problem. The strategy relies on an effective combination of development guidelines as provided by the Systematic Object-Oriented Refinement and Translation (SORT) technique and rigorous quality checks of the resulting products by means of the architecture-centric inspection (ACI) approach. Both, SORT and ACI, have already been successfully applied to the KobrA development method. However, the synergy effects when used in the context of a combined strategy clearly outweighs the isolated benefits of the techniques. The integration of such a strategy into any object-oriented or component-based development method not only results in a high quality component implementation, but also significantly increases the confidence that the desired quality levels can be attained in a successful manner.

Keywords: UML, Object-Oriented and Component-based Development, KobrA, Patterns, Inspection, Reading Techniques

1. INTRODUCTION

The component-based development of high quality systems has become a key issue for many software development organizations. Typically cited promises of components include higher reuse opportunities, increased development speed through code generation and off-the-shelf components, rapid and recursive development methods, improved software quality through lower failure rates, and lower costs associated with failure diagnosis and repair [10]. Because of these promises, component-based development approaches have become the approach of choice for many new development projects.

Unfortunately, existing component-based development methods provide little guidance on how to achieve the promised benefits. In particular, quality requirements are often discarded until component testing or it is taken for granted that the component-based method results, by definition, in high quality artifacts right from the beginning of a project [1]. Both practices are detrimental. Quality must therefore be “built in” the components in a systematic manner right from the very beginning. In doing so, existing techniques, methods, and tools need to be tailored and used for achieving this goal. However, quality enhancing technologies are often limited to conventionally structured development methods. Prominent examples are techniques for early quality assurance, such as software inspections [11] . Moreover, few techniques are actually tailored to the specificities of component-based development and vice versa and few component-based methods include an adequate adaptation of techniques.

An exception is the KobrA-development method [1, 2]. KobrA (**Ko**mponenten-**b**asier**t**e **A**nwendungsentwicklung) offers a prescriptive, systematic approach for component-based product-line engineering across the full software life-cycle. A major emphasis of the method is on quality because the very power of components makes it imperative that the reusable assets are of the highest possible quality. Clearly an organization that systematically produces or reuses poor-quality components will not fare well in the long run. Systematic techniques are therefore integrated into and tailored to the method activities to ensure the highest quality possible. The integration and tailoring is particularly performed for systematic techniques that can be used throughout the construction and subsequent analysis activities.

While the construction activity in KobrA uses the Systematic Object-Oriented Refinement and Translation (SORT) technique [4], the analysis activities involve the architecture-centric inspection (ACI) approach [16]. SORT supports the systematic implementation of components by strictly separating refinement and translation activities. This principle is supported by patterns that contain "pre-verified" refinements or translations, as well as a UML profile, known as the Normal Object Form (NOF) [5]. The NOF defines the appropriate level of abstraction at which refinement should end and translation into a tool-comprehensible form should start. ACI on the other hand supports the systematic analysis of the developed artifacts through a specific instantiation of a software inspection method. The major elements of the ACI approach consist of the optimal packaging of information and the systematic scrutiny of the packaged information using the perspective-based reading technique. Although both, SORT and ACI, are very effective when used in isolation in the KobrA-Development Method their benefits can be further increased through their combination.

This paper presents a strategy to combine SORT and ACI. The main focal point of the strategy is component implementation. In this way, the strategy can be integrated into many existing development methods without significant changes to the underlying method or process. Using the strategy in the context of a development project not only results in components of higher quality but also help to demonstrate that certain quality goals have been fulfilled.

The paper is structured as follows. Chapter 2 provides background information for how to achieve high quality in component-based development projects. This includes methodological support, patterns, tools, and a more detailed description of SORT and ACI in the context of the KobrA-Development Method. Chapter 3 elaborates upon the combination of the techniques in the form of a unified strategy. Chapter 4 presents an example from a component-based development project and shows how the strategy can be used in this context. Finally, chapter 5 concludes.

2. BACKGROUND

2.1 Methodological Support

2.1.1Development Methods

There is a large number of different object-oriented methods to choose from when considering the application of component technology. Despite their prima-facie differences, however, they all share the same underlying assumption that high-level "analysis" models will be developed during the early phases of development, and lower level executable code (i.e. the implementation) will result from the later phases. The terms "earlier" and "later" may no longer apply in a strict waterfall sense due to the prevalence of incremental processes, but the basic idea of progress depending on

abstract models being transformed to concrete code is more or less universal.

Obviously the developers of major OO development methods, such as OMT [17], Booch [3], Fusion [7], Catalysis [10], and the Unified Process [15] have faced the problem of systematically attaining a specific level of quality. Hence, each of these methods claims to provide a holistic approach to software development, supporting the seamless development of software systems from early analysis to executable systems in a qualitative manner. However, when examined in more detail these methods often concentrate on “pure modelling” leaving other issues, such as quality assurance or project management, to the developer or manager. Even recent method developments such as Catalysis [10] or the Unified Process [15] provide only little help in this regard.

2.1.2Patterns

The problems encountered in the development of large and complex software systems with object-oriented and component technology has motivated the development of patterns [6, 12]. The motivation stems from the fact that the development of software systems (e.g., the identification of objects and classes, their relations, behavior, etc.) even if supported by a methodology requires experience or best-practice know-how. In this way, experienced software developers do not always solve recurring problems from scratch, but reuse solutions proven to be good. This experience must be packaged and edited to make it widely available. Gamma et. al. [12] package such expert knowledge in the form of a systematically edited, schematized and structured catalogue of patterns.

In general, patterns describe best-practice software development knowledge. In the case of object technology, their use concentrates on architecture (i.e., high-level design) [6], design [12], and source-code [8]. Regardless of their specific point of application, all these patterns capture good (i.e. useful) object-oriented structures/behaviors (i.e., they focus on the product). Thus, patterns are a means of increasing the quality of a system by reusing known-to-be good object-structures.

The idea of patterns as predefined, ’correct’ object structures/behaviors which can be plugged into an object-oriented (design-) model seems to be very promising for the problem of systematically assuring the quality of component-based software systems. However, their use has to be guided through a systematic process. Furthermore, a reasonable number of patterns have to be identified which do not only concentrate on recurring design problems, but also take the specifics of component technology, ranging from technical (e.g., how to use CORBA orbs) to management and organizational issues.

2.1.3Tools

The benefits of the object-oriented paradigm, when applied correctly, and the definition of the UML as a standard modelling language have led to the widespread adoption of object technology over the past decade. Tool vendors reacted to this situation by developing CASE tools which claim to support the standard modelling language (UML), one or more programming languages, and an associated development methodology. Today there are more than 100 commercial tools available on the market. Most of them promise full support of the UML, automatic code-generation from models, round-trip engineering capabilities, and build-in quality assurance. These widely-advertised features have fuelled the widely held view that CASE tools have already solved the problem of developing high-quality systems which fulfil their prevailing non-functional requirements.

The first generation of CASE tools, such as

IDE-STP or early versions of *RationalRose,* provided only limited methodological support, whereas the latest generation of tools, such as *ParadigmPlus, Bridgepoint,* or *Rational Rose RT,* and *RationalRose 2000* claim to provide full support together with high-sophisticated quality assurance activities. These tools make use of the UML's in-built extension mechanisms to embed rules within the modelling language. Although it is possible to define rules which support the use of single constructs in a way that does not violate quality requirements, this mechanism is mostly used for code generation purposes.

2.2 The SORT-Approach

2.2.1Refinement and Translation

There are a large number of different methods to choose from when considering the use of component technology in a software development project, with an equally large number of different processes and modelling approaches. A fundamental problem with most of the suggested approaches is that they do not distinguish refinement and translation. The lacking distinction often confuses the developers, which in turn results in defects and poor quality. Both, refinement and translation take place while turning high-level abstract models into concrete executable code and can be further explained as follows:

1. Refinement - is a relationship between two descriptions of the same thing, with one, the *abstraction*, containing less information than the other, the *realization*.
2. Translation - describes a given phenomenon in two different ways at the same level of detail. In contrast to refinement, translation describes a relation between two descriptions at the same level of detail, but using different notations.

In software development the mixture of refinement and translation results in unclear relationships between models and code, which breaks two of the most fundamental and time-honoured principles for solving engineering problems: divide-and-conquer, and separation-of-concerns. Failing to apply these principles by trying to perform complex refinement and translation tasks in a combined step increases the size of the "intellectual gap" to be spanned in a single transformation step. This does not only increase the likelihood of errors, and thus quality problems, but it also leads to problems during maintenance.

2.2.2The Normal Object Form

Following the time-honoured principle of separation of concerns, the SORT approach is based on the premise that the only way to seriously address this problem is to cleanly decouple refinement and translation within the software development process [5]. Instead of bundling both transformations together into a single step they have to be performed independently. According to this scheme, analysis and design models, representing different views onto a system, should be developed in the same way as before, but before translation they have first to be refined to a lower level of detail. Only if models at the appropriate level have been obtained they will be translated in a direct and straightforward manner to source-code.

The idea of separating refinement from translation in the implementation of high-level modelling concepts seems to offer numerous advantages, but its practical realization begs one major question - to what level should "high-level" models be refined before they are ready for translation, and how can this level be identified? To address these questions the concept of the *Implementation Level* is introduced.

Most programming languages provide constructs at different levels of abstraction (e.g., abstract data-types, direct memory manipulation, etc.). Therefore it is impossible to

define a universally applicable level of abstraction to which a model should be refined. However, most of these constructs are located within a relatively small band of the abstraction hierarchy, which is known as the *implementation level* [4]. The set of modelling constructs located within this abstraction band (i.e., constructs at the implementation level), is termed the *implementation set*. These constructs are "semantically close" to object-oriented programming features and can therefore be mapped into elements of a program in a manner that approximates translation. One such set, which can be specialized for specific purposes is the *Normal Object Form* (NOF). The NOF defines to what level of detail refinement proceeds until translation can start. The mechanism behind this idea includes the definition of a set of UML modelling concepts (i.e., a UML profile) which correspond to the core constructs of object-oriented programming so that they can be mapped into elements of a program in a manner that approximates translation (i.e without a significant change in abstraction level). This set is coined NOF because in a sense it represents a "normal" form, akin to that used in relational databases, to which UML models must be reduced before translation can begin.

2.2.3Refinement and Translation Patterns

In view of the success of the pattern cataloguing approaches pioneered by Gamma [12] and Buschmann [6], SORT refinement and translation guidelines are packaged in a similar manner. However, there is a subtle differences between the patterns defined in *SORT* and those of Gamma and Buschmann. Whereas the latter essentially capture good (i.e. useful) object-oriented structures/behaviors, *SORT* patterns capture good (i.e. useful) mappings between object-oriented structures/behaviors.

Two forms of patterns are recognized in SORT: *refinement patterns,* which describe "good" refinements within the UML for reaching structures at the implementation level specified by the NOF (see Figure1), and *translation patterns* (see Figure2), which describe the "good" mapping of UML-NOF models to a specific object-oriented programming language (e.g., C++). The latter are similar to "idioms" [8] in that they are language specific. However, as mentioned above they represent more of a mapping guideline than a useful programming practice.

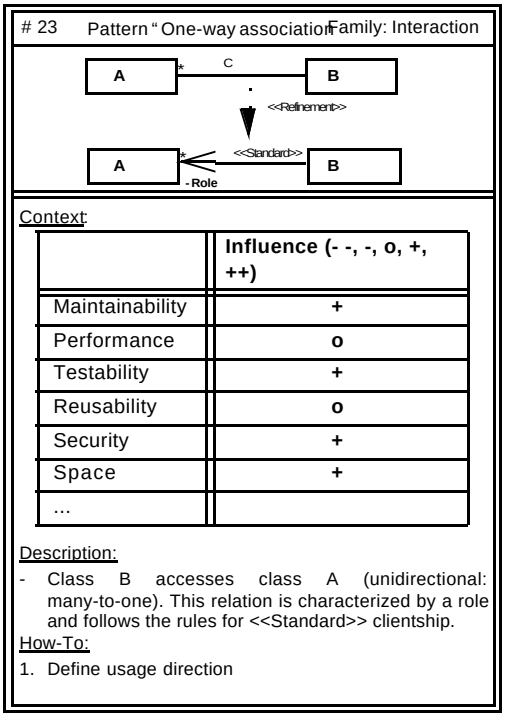


Figure 1: Refinement Pattern

Of course, there is rarely a single pattern which provides the best mapping (refinement or translation) of a given structure under each and every circumstances. There are often several of tgem, and the one which is most appropriate in a particular context depends on the associated non-functional requirements (e.g. performance needs, space limitations, reliability etc.). Therefore patterns have to provide a context description which allows

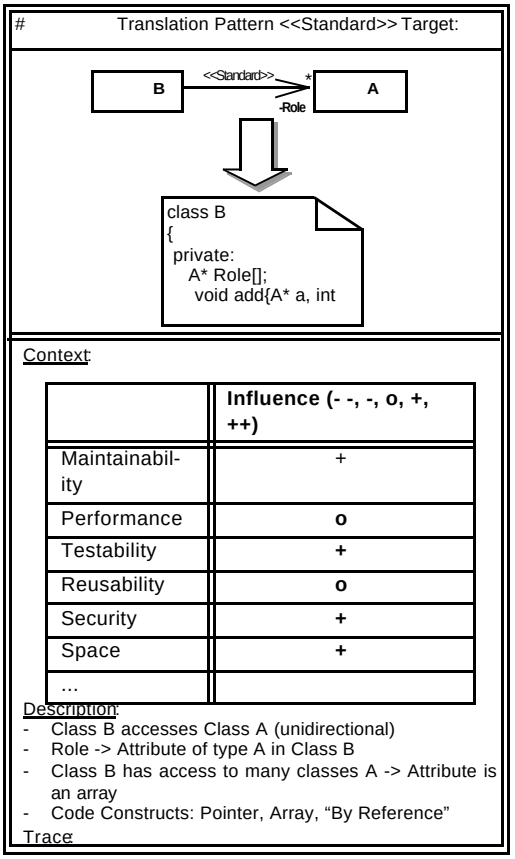


Figure 2: Translation Pattern

a developer to choose the most suitable one for his/her particular needs. Providing such information allows *SORT* to offer a level of context sensitivity which is impossible in automated mapping tools, while at the same time still being reasonably systematic. Developers are told precisely what to refine or translate, how to perform these refinements/translations, and when to perform the relevant activities. At the same time, however, they can tailor the particular refinements and translations in different ways that depend on the relevant context factors. More detailed information on SORT and its patterns can be found in [4].

2.2.4The SORT Technique

In summary, the SORT technique provides a practical way to address the mapping between design and source-code of a component or component-based system. It cleanly separates refinement and translation by systematically defining the relationship between models and code. This separation, together with its practical support in the form of refinement- and translation patterns, tightens the gap to be bridged between models and code by solving problems that developers typically face. Model implementation is therefore broken down into well-defined individual refinement and translation steps. In the context of component-based development every component of a system can therefore be transferred to source-code. Since a sequence of several smaller steps is easier to understand than a big and highly creative one, the resulting implementation is less error-prone. This is similar to theorem proving, in which the correctness of a theorem is demonstrated in terms of the correctness of "smaller" parts. SORT in the context of the development phases is summarized in Figure 3.

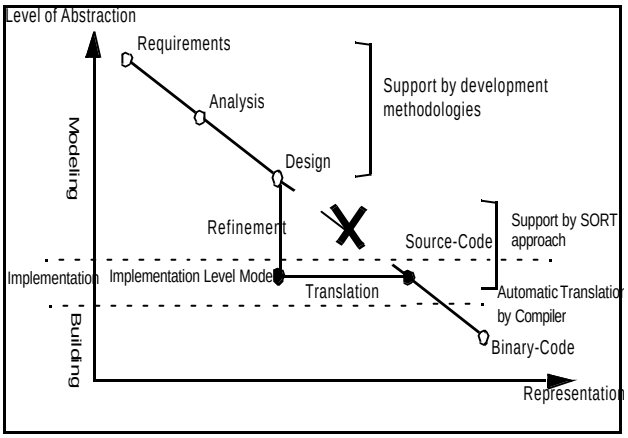


Figure 3: Application of SORT

2.3 Architecture-Centric Inspections

2.3.1Inspection Principles

The SORT approach allows for the systematic implementation of components or parts of it. This represents an important step towards high quality software. However, there is still the possibility for defect injection. Software inspection is an approach that allows the detection and removal of defects immediately after software documents are created. An inspection involves activities in

which qualified personnel determine whether software documents are of sufficient quality for subsequent development activities. Figure 4 illustrates a typical software inspection approach.

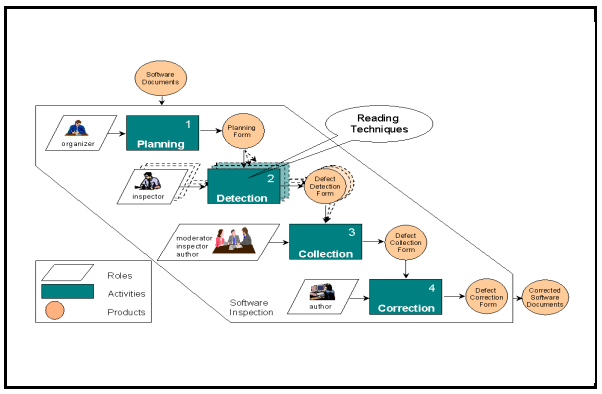


Figure 4: A Software Inspection

We model an inspection in terms of its main activities, that is, inspection planning, defect detection, defect collection, and defect correction. There are two reasons for using this terminology. First, we want to emphasize that the problems and solutions we present in this paper are independent of any particular inspection implementation, such as that of Fagan [11] or Gilb and Graham [13]. Second, by modelling the inspection process in terms of its main activities we avoid ambiguity in the terminology.

Inspection planning is performed by an organizer who schedules all subsequent inspection activities. The organizer is often the project manager, since he or she organizes all activities for the software development project. While the defect detection is performed individually, the collection activity is performed in a group meeting. An inspection usually ends with the correction of the documented defects by the author.

2.3.2Architecture-centric Inspection

Organization

The Unit of Inspection

Before performing an inspection, the inspection organizer must determine the subject of the inspection, that is, the unit to be inspected. The term "unit" refers to the set of information that inspection participants scrutinize for defects in one particular inspection. The selection of the "right" unit represents a problem in the software domain because large software development projects have special problems due to their scale. The volume of the developed documentation is often too large to be handled in a single inspection. Hence, several inspections must be organized on different document fragments. However, this requires a rationale for partitioning and grouping them. This problem is comparable to the situation in other engineering disciplines. The various plans for a bridge, for example, cannot be inspected in a single inspection. Hence, they need to be partitioned and grouped into smaller units for which an inspection can be organized and performed.

Most existing inspection variations follow a document-oriented strategy for inspection organization. This strategy means that a particular inspection is organized around a particular type of document, such as a requirements, design, or code document (or parts of it), rather than the structure or content of the information represented in the documents.

Although at first sight the document-oriented approach appears to be a good strategy, it leads to two difficulties. First, crucial information is often distributed across various parts of a document or even across different document types. Thus, if the inspection is limited to a particular (part of a) document, an inspector may miss crucial information for a sound inspection. Object-oriented development methods [3, 15, 17] and the recent appearance of the Unified Modelling Language (UML) accentuate this problem

because they usually use different types of diagrams to represent various sets of information. Hence, information about a given logical entity, such as a component, can be described in many different documents, and a specific document can contain information about many different logical entities, that is, there is a many-to-many relationship between logical entities and diagrams. An inspection whose goal is to check a particular (part of a) document may end up either having to analyze many logical entities, or may only partially cover a logical entity that it describes.

The second difficulty arises when the document or parts of it is still too large after decomposing it. Hence, some authors recommend the use of size information alone as a further splitting criterion. However, this recommendation causes some problems as the following example illustrates. Let us assume a code document (as part of a larger system) has 20000 LOC after following the document-oriented strategy for inspection organization. As suggested in the literature [8], this partition obviously is still too large to scrutinize for defects in a single inspection. Hence, the document needs to be partitioned further. One approach would be to split it into partitions of 500 LOC, which is recommended in the literature [13], and inspect the first 500 LOC in the first inspection, the next 500 LOC in the second inspection, and so on. However, the first 500 LOC provides information about the definition and declaration of variables, which are a necessary prerequisite for inspecting the other code fragments. This example demonstrates that the document-oriented approach and the use of size information alone to decide upon the unit of inspection do not solve the unit of inspection problem in each and every case.

Architecture-centric Principle

The architecture-centric principle distinguishes between a logical entity and the physical documentation of the entity. This situation is graphically depicted in Figure 5.

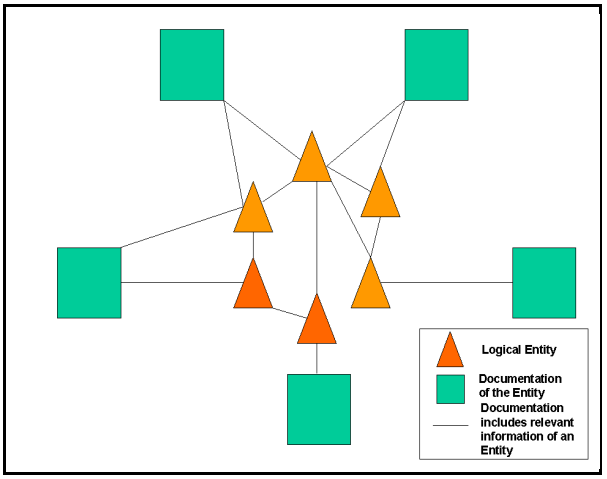


Figure 5: Logical Entities and their Documentation

A triangle, or a line between two triangles, represents a logical entity. A square, on the other hand, represents the documentation of one or more logical entities. The new strategy suggests that an inspection be organized around logical entities rather than the physical documentation of the logical entities. The choice of the logical entity or the logical entities determines the document fragments that contain relevant information to be scrutinized for defects in one particular inspection.

Although the majority of modern software engineering methods, such as the Unified Process [15] or the Object Modelling Technique [17], explicitly separate logical entities from their documentation, this has not been considered useful for inspection organization. Of course, each method uses its own terminology, which may be different from the one used in this paper. The Unified Process, for example, distinguishes between structural elements, such as subsystems or classes, and models that describe the structural elements, such as use-case or collaboration diagrams [15]. Hence, the Unified Process uses the term "structural element" instead of logical entity and "model" instead of documentation of the logical entity. We

recognize that various terms could have been used in this paper to describe this difference. Examples are "unit" or "concept" instead of "logical entity" and "model", "representation", or "description" instead of documentation. We decided to use the word "logical entity", since it best conveys the conceptual and invisible nature of software, and "documentation", since it best conveys the idea of something tangible that can be used for the purpose of inspection.

In a more general sense, the logical entities making up a system, and the relationship between them, are collectively viewed as the architecture of the system. The architecture encompasses the significant decisions about the organization of a software system, the major structural elements and their relationship that will comprise the system, and the composition of the elements into progressively larger subsystems. Due to the importance of the architecture, we call the new strategy the architecture-centric approach for organizing inspections.

The Benefits of the Architecture-centric Solution

The architecture-centric solution for inspection organization is beneficial for three reasons: First, the set of information for each logical entity by definition is logically self-contained and conceptually complete. It therefore provides an inspector with all crucial information for performing a sound inspection and, at the same time, represents the appropriate set of information that is intellectually manageable. The latter prevents inspectors from being swamped with a lot of unnecessary information. Second, the architecture-centric approach is scalable. If the documentation of a logical entity is still too large, an inspection organizer can look at the substructure of the logical entity and choose an appropriate logical entity of smaller granularity. This process can be repeated until the right scope for a single inspection is determined. Finally, the architecture-centric approach has been implicitly embedded in conventional structured development processes that use decomposition as a principle for structuring software systems. However, the use of the architecture-centric solution is not limited to conventional design principles. It can also be applied in the context of object-oriented development methods. This is important because over the past decade, object-oriented development methods have replaced conventional structured approaches as the embodiment of goodness in software development, and are now the approach of choice in most new software development projects. Software inspections must be tailored to this new situation.

2.3.3Perspective-based Reading

Although each of the presented inspection activities is important for a successful inspection, the most important one is the defect detection activity. In this phase of an inspection, inspectors read the software document(s) to determine whether quality requirements, such as correctness, consistency, testability, or maintainability, have been fulfilled. "Reading" implies the systematic examination of a document to extract, gain, and understand certain information about the inspected software. The ability to read, and to understand what has been read, are therefore critical skills for the participant of an inspection. Understanding itself is a necessary prerequisite for finding more crucial defects in the software documentation. These defects are often the expensive ones if detected in later development phases, and the most difficult to detect in an inspection, since they usually go well beyond more trivial defects, such as spelling mistakes.

The second important ingredient of ACI is therefore the application of a systematic reading technique, that is, perspective-based reading, throughout the defect detection phase of an inspection. Perspec-

tive-based reading was selected since it complements the architecture-centric organization of an inspection in an optimal manner.

Goal of Perspective-based Reading

The basic goal of PBR is to examine the documentation of a software entity from the perspectives of the entity's various stakeholders for the purpose of identifying defects. An inspector in a perspective-based inspection reads the documentation from the perspective of a particular stakeholder in such a way as to determine whether it satisfies the stakeholders' particular needs. A stakeholder perspective may be, for example, a future user of the system who wants to ensure the completeness of the inspected analysis documents. If the documentation of the software entity meets the stakeholders' quality requirements, the end product, that is the final software system will meet the specified quality goals.

Perspective-based Reading Scenarios

Throughout the reading process, an inspector follows the instructions of a perspective-based reading scenario (in short: scenario). A scenario tells the inspector how to go about reading the documentation from one particular perspective and what to look for.

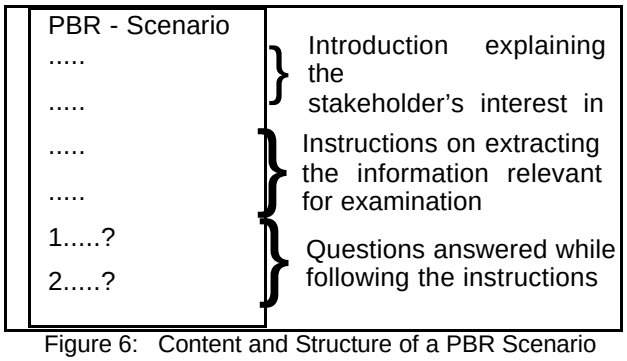


Figure 6: Content and Structure of a PBR Scenario

As depicted in Figure 6, a scenario consists of an introduction, instructions, and questions framed together in a procedural manner. The introductory part describes the stakeholder's interest in the logical entity and explains the quality factors most relevant for this perspective. The instruction part describes what kind of document an inspector is to use, how to read the document, and how to extract the appropriate information from them. While identifying, reading, and extracting information, inspectors may already detect some defects. However, the motivation for providing guidance for inspectors in the form of instructions on how to perform the reading activity is three-fold. First, instructions help an inspector gain a focused understanding of the entity. Understanding involves the assignment of meaning to information in a particular document and is a necessary prerequisite for detecting more subtle defects which are often the expensive ones if detected and removed in later development phases. Second, the instructions require an inspector to actively work with the documentation rather than passively scanning it. Third, the architecture-centric strategy ensures that the relevant information for all stakeholders is available for scrutiny. However, since the attention of an inspector is focused on the information most interesting for a particular stakeholder, the inspector is not swamped with details irrelevant for the stakeholder's perspective. A process for scenario development is described in [16].

Once an inspector has achieved an understanding of the documented information about an entity chosen by the architecture-centric approach, he or she can examine and judge whether it fulfils the required quality properties. For making this judgement an inspector is supported by a set of questions that are answered while following the instructions. The questions may also involve consistency checks as presented in [18].

3. THE STRATEGY FOR HIGH-QUALITY COMPONENT CONSTRUCTION AND

ANALYSIS

3.1 Combination Approach

Based on the construction and analysis strategies (i.e., SORT and ACI) defined in the previous section, the ultimate goal is to integrate them seamlessly into a combined approach for systematic high quality component implementation. Figure 7 displays an overview of this strategy.

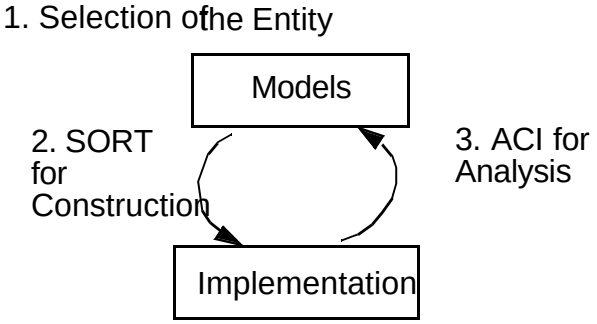


Figure 7: Overview of Strategy

The strategy can be described as follows:

1. In real projects it is unlikely that every component will be implemented by a single developer. Therefore a strategy has to be defined which allows to partition the component into several chunks which can then be handled by a single developer. This can either be done by simply reusing existing strategies, such as defined in the context of the Fusion method [9], or by applying a diagram-based, level-based, or concept-based strategy [4]. Diagram-based strategies use the principle of divide and conquer. When applying the strategy one concentrates only on a specific diagram within the design model. Level-based strategies are oriented towards the different levels of abstraction within a model. It starts with most abstract constructs (i.e., a high-level class) and proceeds to less abstract constructs in a top-down manner. Finally, concept-based strategies concentrates on the concepts of the model by using semantically coherent pieces within the model. Despite the actual strategy, the result is a collection of units (i.e., a component or parts of it) which can be prioritized and given to a single developer.
1. The developers use Translation/Refinement patterns for the implementation of a component or parts of it. The SORT approach tells developers precisely what to refine or translate, how to perform these refinements/translations, and when to perform the relevant activity. At the same time, however, they can tailor the particular refinements and translations in different ways that depend on the relevant context factors.
1. An ACI is organized and performed on a specific implementation or a whole set of implementations. The unit of inspection is already defined through the work assignments. The quality checks inspectors need to perform include
   - quality checks to ensure that the developers performed the tailoring in an appropriate manner.
   - quality checks to ensure that the interfaces between the different patterns and their implementation are correct.
   - quality checks that are stated in the context part of the description.

   For the various checks, the inspectors get perspective-based reading scenarios.

The quality characterization in the context part of the description are helpful to select the perspectives from which to scrutinize the implementation. If “testability” is important one inspector performs the quality checks from the perspective of a tester.

The concern of the tester is to ensure that the operation is defined in a way that the implementation is testable. The basic idea, therefore, is for the inspector to work through the creation of test cases, and to ensure that the implementation is correct with respect to these test cases. Traditional testing concepts are thus highly applicable

here, such as black box/white box testing, equivalence class partitioning, state-transition testing etc. Figure 8 depicts an example of a tester's scenario.

Assume you are inspecting the implementation from the perspective of a tester. The main goal of a tester is to ensure the testability of a component or parts of it. High quality thus corresponds to full testability. Assume that you have to develop some test cases for the implementation. A test case consists of a set of input values plus a set of output values and/or state changes expected for each combination of values. Follow the instructions below and answer the questions carefully.
Define equivalence classes for the input/output values and interfaces. Use these equivalence classes to define a minimal set of test cases to fully exercise the behavior including the public interfaces of the component.
While following the instructions answer the following questions:

1. Are all possible sets of input values properly addressed by the provided diagrams?
2. Are operations preconditions indicated?

Figure 1: Scenario for Tester's Perspective

If "maintainability" is important one inspector performs the quality checks from the perspective of a maintainer. The task of a maintainer is to ensure the maintainability of the implementation. In practice this means that the complexity of the implementation resulting from the tailoring to the contextual factors needs to be kept to a minimum. Figure 9 depicts an example of a maintainer's scenario.

Assume you are inspecting the implementation from the perspective of a maintainer. The main goal of a maintainer is to ensure that the code is written in a way that can be easily changed and maintained. High quality, therefore, means the conformance to low coupling, high cohesion and the minimization of complexity.
Locate the refinement class diagram. Examine the diagram and the descriptions to identify points of converge from the design class diagram and from good design practice.
Determine whether translation/refinement-patterns were used appropriately to implement the code.
While following the instructions answer the following questions:

1. Are there any ways in which the number of objects, or the number of messages could be reduced?
2. Are there any cycles of messages in the diagrams?
3. Is there any way in which the control structure of the implementation could be simplified?
4. Do the messages entering an object indicate the possibility of low cohesion (are the messages totally unrelated)?
5. Is there a particularly high number of message between a pair of objects?
6. Are there elements in the refined class diagram that cannot

Figure 9: Scenario for Maintainer's Perspective

3.2 Integration of the Strategy in Existing Development Methods

An important criterion to determine the strengths of the strategy is the possibility to integrate it into existing development methods. We have selected two methods, that is, the Unified Development Process and the KobrA method to demonstrate how the integration may take place.

3.2.1The Unified Development Process

The Unified Process (UP) defines a "process framework" for different kinds of software systems, domains, and organisations. The main characteristics of this process are that it is "use-case driven" and "architecture-centric", while at the same time defining iterative and incremental process phases. The overall UP development life-cycle is divided into four main phases (inception, elaboration, construction and transition) and consists of a sequence of iterations. This is depicted in the following Figure 10.

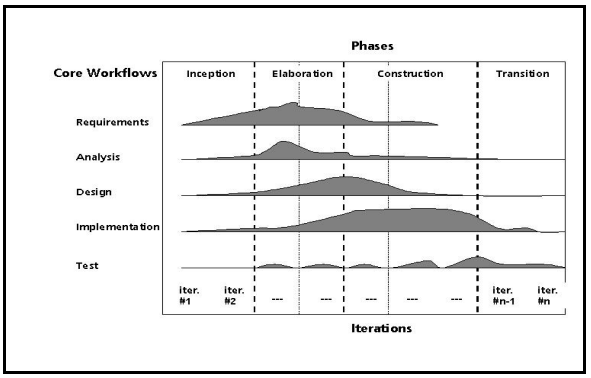


Figure 10: Core Workflows of the Unified Process and their Emphasis Shifts over Iterations

The definition of the phases reflects the main focus of the iterations each conducted at certain stages of product development.

- The goal of the inception phase to develop a "good idea" into a vision of the end product. The major deliverables of this phase are an outline of the candidate architecture, a list of identified risks and a "proof-of-concept" prototype.
- During the elaboration phase about 80% of the functional requirements are specified, a stable architecture is designed, and a project plan is established.

- The purpose of the construction phase is to develop the software product, which is ready to be exposed to the customer during a beta release. During this phase, the software components are implemented and the features of the software system are tested.
- During the transition phase beta tests are executed, users are trained and the software is given to the end users. At the end of this phase the product is fully released.

The focus of the presented strategy is the construction phase. Throughout this phase the developers can directly follow the guidelines provided by the SORT approach to implement the components or parts of it. This ensures systematicity throughout the implementation phase. Afterwards, the ACI approach can be used to perform the required quality checks.

3.2.2KobrA

The KobrA method [1], developed at Fraunhofer IESE, represents a synthesis of well-known software engineering technologies, including product line development, components-based development, frameworks, architecture-centric inspections, quality modelling and process modelling. Therefore, the method is aimed at providing an systematic approach to the development of high-quality, component-based application frameworks.

The central artifact of the KobrA method is a tree of components that describe either a generic framework or a particular application. Each component is described by a set of models (i.e., UML diagrams [14]). Thus, all KobrA artefacts are organized around, and oriented towards, the description of individual components. Although the different diagrams which belong to one component are well-defined, the method do not requires the creation of every diagram at all times. Consequently, the method provides guidelines on which diagrams to create and when (i.e., KobrA is scalable).

In addition to the central component tree the KobrA method provides a unique approach to ensure that the systems under development is of high quality. It provides a combined strategy for quality assurance on both, the construction and analytical side of quality assurance (i.e., construction guidelines and techniques for efficiently identifying and removing defects). The strategy is specifically adapted to the needs of component-based and object-oriented systems and intertwined with the single process steps. Thus, quality assurance is an integral part of the method and not an add-on as in more traditional approaches.

3.2.3Expected Benefits of the Strategy

The strategy provides detailed guidelines for the construction of the component or parts of it (by means of the SORT approach) and the analysis (by means of the ACI approach). SORT reduces the opportunities to inject defects throughout the component implementation activity. ACI, on the other hand, helps remove the defects that have not been avoided. In this way, the combination assures maximum component quality before the testing or the integration of the component in the target software system.

4. THE LIBRARY-CASE STUDY

A detailed example is used to illustrate the different aspects and benefits of the proposed strategy. The example is based on a larger case-study [1] which describes the development of a library information system.

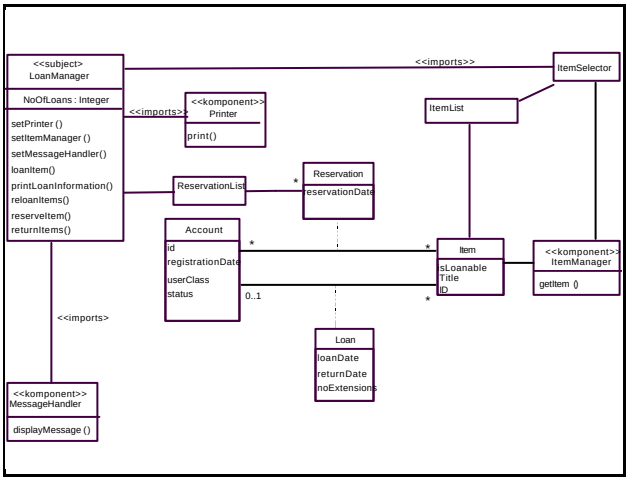


Figure 11: Analysis Class Diagram

The domain of library systems encompasses systems that help librarians and information professionals to perform their work. Library systems provide support for different aspects of library work, including assistance in customer interaction, stock management, and accounting. A basic library system, as used in the following, supports the management of registered users and available items and tracks all performed loans, returns, and reservations. Figure 11 shows an analysis class diagram for a loan manager component of that system.

The complete case study, including both the basic library system and a library system product line, is described in Appendix C of [1].

In the context of a development project the analysis class diagram, shown in Figure 11 has to be modified to incorporate design issues and technical properties. Finally, a design class diagram for the loan manager component will be created (see Figure 12). In more conventional development projects this diagram is given to the developers with the task to code it in the selected programming language. However, the information contained in this diagram may not be sufficient for a high-quality implementation.

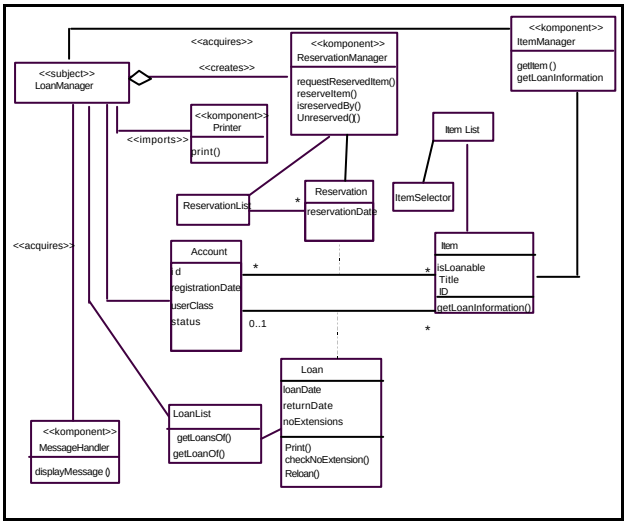


Figure 12: Design Class Diagram

Open questions that are related to the insufficient information are the following:

1. How can the association class 'Loan' and the corresponding association between the 'Account' and 'Item' Class respectively, be efficiently implemented? If the expected number of links is small it may be sufficient to implement the association class as a normal class and connect all three classes by individual associations. In contrast, if the number of links is expected to be high the association class may become a dictionary class, which can use hash tables to provide quick access.
2. Which parameters and datatypes are going to be used for attributes and operation parameters of the single classes? Without specifying the concrete interface of a class it is nearly impossible to correctly implement access to that class. Even languages which do not need the specification of types (e.g., Smalltalk) need a carefully specified interface to guarantee for correct responses at runtime.
3. Are there any regulations concerning visibility and is there a need for get/set operations? A typical example are attributes which should not be manipulated directly from the outside, to prohibit unwanted side-effects (i.e., a private attribute), but which can be accessed and updated in a clean way (i.e., access is granted by get/set operations).

4.1 Selection of the Entity

In most cases the selection of the entity represents a trivial step. In this example, it is the loan manager component that is to be implemented.

4.2 SORT for Construction

The stated questions could be resolved by applying the SORT development guidelines concerning refinement and translations throughout the second step of the presented strategy. Figure 13 shows an excerpt of the

refined class diagram (i.e., an implementation class diagram).

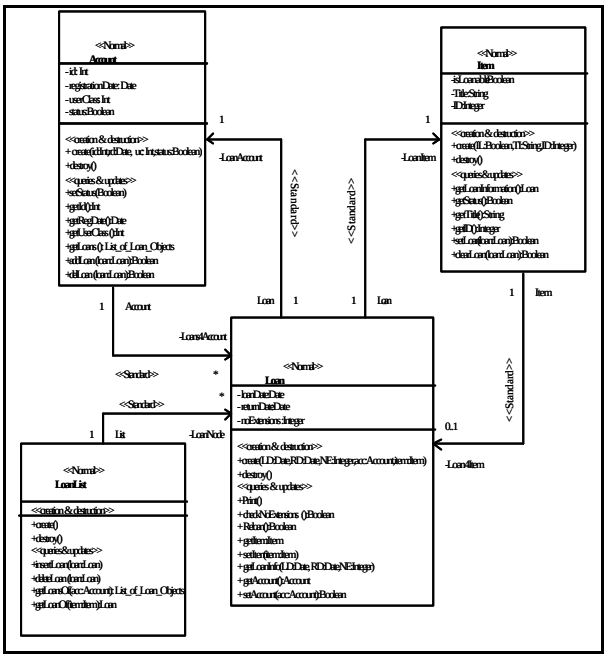


Figure 13: Refined Class Diagram

This diagram shows that the association class construct is refined to several unidirectional associations between the 'Loan' and 'Account' classes, as well as between the 'Loan' and 'Item' classes. Each association now has clearly defined roles, as well as a clientship stereotype which determines its implementation (i.e., the <<Standard>> stereotype, as defined by the NOF, depicts a relationship where the client holds a direct reference to the public parts of its server in its main datastructure). All attributes are declared to be private (information hiding), which required the addition of several get/set operations, and operations to be public. For each operation the parameters and types are defined to be close to their implementation. In this example the class 'LoanList' holds a list of Loans in its datastructure. This list can be either be implemented as a linked list or via hashtables. On basis of this diagram the next step, the translation to source-code, can be performed. Applying transformation patterns on the refined diagram in source-code.

Figure 14 visualizes how the LoanList class is implemented in C++. The example displays only an excerpt of the header file to

```
class LoanList {
  public:
// creation & destruction
LoanList();
~LoanList() {};
// queries & updates
Boolean getLoanOf(Item &item);
Boolean insertLoan(Loan loan);
LoanNode* getLoansOf(Account acc);
  private:
struct LoanNode {Loan Element;  LoanNode *Next, *previous;};
LoanNode *Header;
```

Figure 14: C++ Source-Code

indicate the implementation. Additionally the translation patterns or other pseudo-code artifacts (i.e., describing the behavior of single methods) have defined the prototypes of the methods which are then stored in a body-file.

4.3 ACI for Analysis

In the final step of the presented strategy, all the available documentation of the loan manager component is subject to analysis throughout an architecture-centric inspection. In the first of an inspection, the inspection participants scrutinize the refined class diagram and the implementation using the scenarios provided by the perspective-based reading technique. For example, one inspector performs quality checks from the perspective of a tester and, thus, follows the guidelines presented in Figure 8. He or she develops some test cases to ensure the testability of the implementation and the consistency with the refined class diagram. The careful application of the tester scenario would reveal, for example, the following inconsistency between the refined class diagram and the implementation. The implementation uses a return type Boolean for the getLoanof-method, but in the refined class diagram it is stated that it should return a value of type Loan.

Another inspector adopts the perspective of a maintainer and executes the scenario in Figure 9 starting from the refined class diagram. In this case, the inspector would realize that the translation pattern was not applied in a correct manner, since the developer has forgotten the implementation of the delete loan - method. Of course, the inspector also checks the consistency with the design class diagram.

In the subsequent team meeting the two inspectors together with the moderator and the author discuss and consolidate their findings. The author susbsequently corrects the defects detected prior to testing and component integration. This improves the quality of the component and at the same time minimizes defect costs.

5. SUMMARY AND CONCLUSIONS

With the rapid rate of innovation in component-based technology, one might have expected to have seen significant improvements on how to ensure the quality of object-oriented systems. In practice, however, this has rarely be happened. There is still a lack of techniques to improve the quality of components.

While many of the existing development methods focus on the early phases, this paper presents a combined strategy to support the construction and subsequent analysis of components. The strategy is built upon existing technologies, i.e., the SORT- and ACI-approach, that have been successfully used in the KobrA-development method. The combination of both techniques allow for the high quality implementation of components. An important benefit of the suggested strategy is the fact that it can be easily integrated into currently available development methods, such as the Unified Development Process. Moreover, the strategy represents another impetus for high quality component implementation since more prescriptive guidance can be given to developers on how to implement and analyse a component or parts of it. The fundamental principles of the suggested strategy are demonstrated with a detailed example. It is clear that an extensive presentation of all refinement/translation patterns and reading scenarios is too voluminous to be included in the paper. Hence, we refer to the literature for more detailed information.

We currently plan an empirical study to investigate the return on investment of the suggested strategy. Moreover, we are looking for tools to support both approaches and automate steps as best as possible. Both are necessary ingredients to drive the adoption of this approach in practical development situations.

REFERENCES

[1] Colin Atkinson, Joachim Bayer, Christian Bunse, Erik Kamsties, Oliver Laitenberger, Roland Laqua, Dirk Muthig, Barbara Paech, Jürgen Wüst, and Jörg Zettel. *Component-Based Product Line Engineering with the UML*. Pearson Publishing, 2001.

[2] Colin Atkinson, Joachim Bayer, Christian Bunse, Erik Kamsties, Oliver Laitenberger, Roland Laqua, Dirk Muthig, Peter Rösch, Tanya Widen, Jürgen Wüst, and Jörg Zettel. *Kobra Method Handbook*. Technical Report IESE-Report No. 058.99/E, Fraunhofer Institute for Experimental Software Engineering, September 1999.

[3] Grady Booch. *Object Oriented Analysis and Design with Applications*. Benjamin/Cummings, Redwood City, California, 2nd edition, 1994.

[4] Christian Bunse. *Pattern-Based Refinement and Translation of Object-Oriented Models to Code*. PhD thesis, University of Kaiserslautern, 2000. Published in 2001 by Fraunhofer IRB Verlag.

[5] Christian Bunse and Colin Atkinson. The Normal Object Form: Bridging the Gap from Models to Code. In Bernhard Rumpe and Robert France, editors, *Proceedings of the Second International Conference on the Unified Modeling Language*, Fort Collins, Colorado, USA, October 1999.

[6] F.Buschmann, R.Meunier, H.Rohnert, P.Sommerlad, and M.Stal. *Pattern-Oriented Software Architecture – A System of Patterns*. John Wiley and Sons, 1996.

[7] D.Coleman, P.Arnold, S.Bodoff, C.Dollin, H.Gilchrist, F.Hayes, and P.Jeremaes. *Object-Oriented Development: The Fusion Method*. Prentice Hall, 1993.

[8] James O. Coplien. *Advanced C++ Programming Styles and Idoms*. Addison–Wesley, 1992.

[9] Todd Cotton. *Object Oriented Development at Work:*

Fusion in the Real World, chapter 6 - Evolutionary Fusion: A Customer-oriented Incremental Life Cycle for Fusion, pages 179–202. Prentice Hall, 1996.

[10] DesmondF. D'Souza and AlanCameron Wills. *Objects, Components and Frameworks with UML: The Catalysis Approach*. Object Technology Series. Addison Wesley, October 1998.

[11] M.E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):182–211, 1976.

[12] E.Gamma, R.Helm, R.Johnson, and J.Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[13] Tom Gilb and Dorothy Graham. *Software Inspection*. Addison-Wesley Publishing Company, 1993.

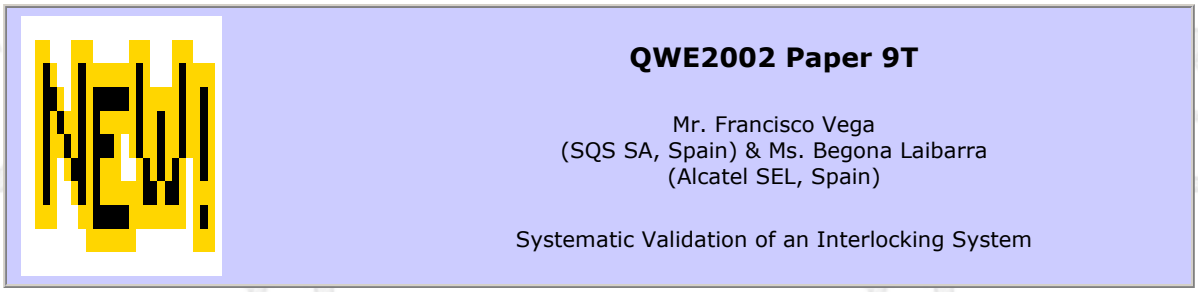
[14] ObjectManagement Group. *OMG Unified Modeling Language Specification*, Version 1.3. Technical report, OMG, June 1999.

[15] Philippe Kruchten. *The Rational Unified Process. An Introduction*. Object Technology Series. Addison-Wesley, 1998.

[16] Oliver Laitenberger. *Cost-effective Detection of Software Defects through Perspective-based Inspections*. PhD thesis, University of Kaiserslautern, 2000. Published by Fraunhofer IRB Verlag.

[17] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[18] G.Travassos, F.Shull, M.Fredericks, and V.R. Basili. Detecting Defects in Object Oriented Designs: Using Reading Techniques to Increase Software Quality. In *Proceedings of the Conference on Object-oriented Programming Systems, Languages & Applications (OOPSLA99)*, 1999.



QWE2002 Paper 9T

Mr. Francisco Vega
(SQS SA, Spain) & Ms. Begona Laibarra
(Alcatel SEL, Spain)

Systematic Validation of an Interlocking System

Key Points

- Use of a modular stepped validation approach to ensure a quick and reliable re-usability.
- Use of State Transition Models as the basic strategy to perform the validation of an int.
- How this validated process has been automated through a self-developed set of tools.

Presentation Abstract

The interlocking systems are railway devices responsible for the control and management of the signal and field equipment. They are safety critical real-time systems. The railway sector is a fast-change environment nowadays. New standards and new technologies are being introduced. Within this environment, Alcatel has developed a new interlocking system that complies with all the new demands. This conference will present a view over the validation process followed in this project: process features theoretical methodology and its deployment tools, commercial and self developed tools used in the life cycle results, including some figures for: effort time. number of test cases

About the Author

Mr. Francisco Vega: Industrial Engineer. Since 1989 working for ALCATEL, for its Transport Automation Division in Spain. He is responsible of the Software Process Improvement and Quality of the technical area, and Quality Manager of the electronic interlocking developments. This includes the overall responsibility and management over the validation area.

Presentation

A systematic approach for the validation of an electronic interlocking

Brussels, 11-15 March, 2002

Validation of an electronic interlocking

- Understanding of the problem
- Validation strategy
- Validation principles
  - Test specification
  - Traceability
  - Documentation
  - Automation process
  - A real example
- Conclusions

The Problem

- Validation of a new product
- Modular concept
- New markets: globalization
- New standards applicable: CENELEC

Starting point

- Validation:
  - Intensive black box functional testing
  - Functional experts are needed to execute the tests.
  - It is difficult to establish the traceability between requirements and tests.

Validation targets (I)

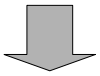
- Design a validation strategy:
  - Including systematic procedures
  - Test execution repeatable by functional non-experts.
  - Integrated in the development life cycle.
  - Assuring traceability.
  - Considering CENELEC requirements.

Validation targets (II)

- Prepare test cases specification to achieve the following coverage levels:
  - 100% Requirements.
  - 100% Code coverage
- Design and implement an automated test process.

Validation strategy

- Stepped validation:
  - Phase 1: Module validation
  - Phase 2: System Integration and Validation

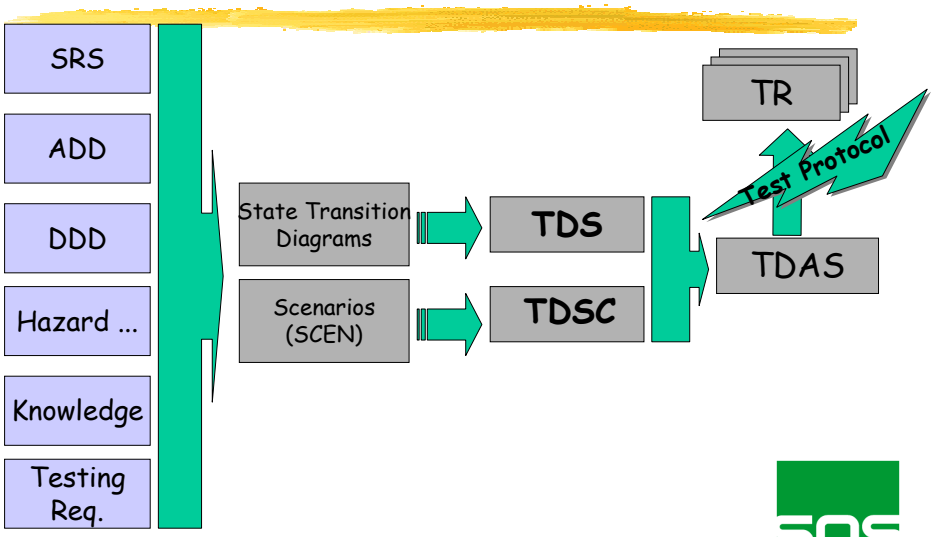


- ***Reusability of the independent modules***

Validation tasks

- Tasks:
  - Review of completeness and correctness of the design documents: SRS, ADD, DDD
  - Review of verification plan and verification results.
  - System (module) Test specification.
  - Test results asssessment.
  - Document the validation process.

Validation Process



State transition diagrams

STD

Model the behaviour of the elements.

Systematic design of test cases.

Principles for STD

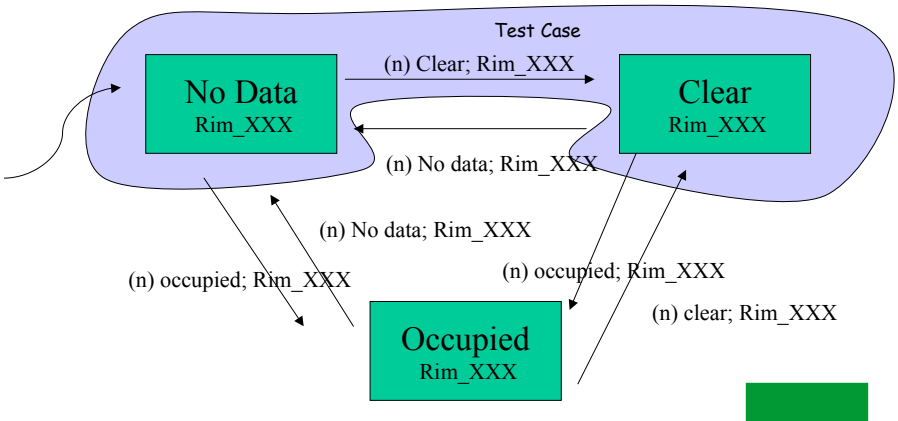
- Simplicity.
- Excluding/Homogeneous states.
- Reflecting the SRS: Traceability.

State transition diagrams

Examples

- STD - Basic behaviour.
- STD - receiving Commands
- STD - in a Route
- STD - in LO

State transition diagrams



Scenarios

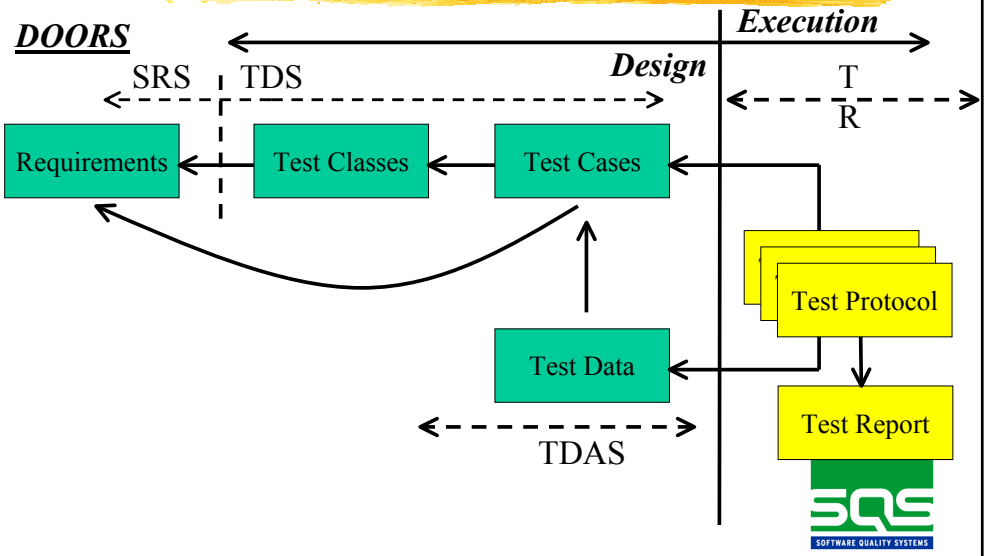
Problem

- Only with STD is not possible to validate the dynamic behaviour of the system
- For this part we'll need knowledge, expertisement and experience.

✓We need

SCENARIOS

Traceability

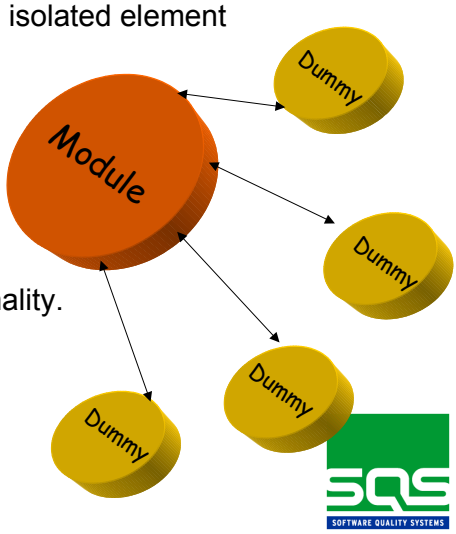


Test Automation

Problem: Module validation as an isolated element

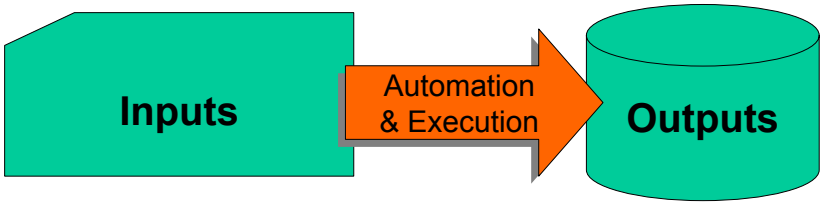
How? Using *Dummies** instead of the elements (modules) that interact with it

**Dummy*: Modules without functionality. It includes only basic functions: queues reading/writing, basic diagnostics, ...



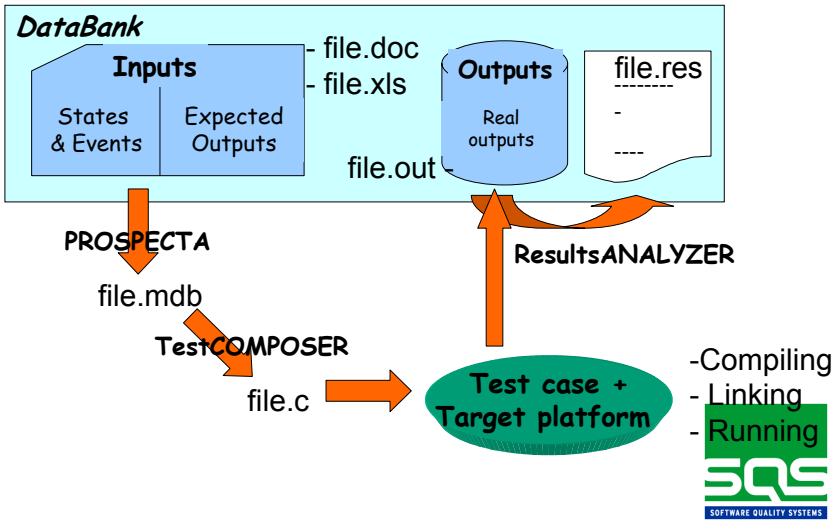
Test Automation

Process inputs:
1. Test cases
2. Test data



Process outputs:
1. Results
2. Message queues
3. Test coverage

Test Automation



Examples

Test case specification

3.1.1.2 TCL-1.1-TC-2

Test Case Identifier:	TCL-1.1-TC-2
Diagram:	Track section states.
Tittle:	Failed to clear
Test Case Description:	The track section state is "Failed". There is a notification that the track station state is "Clear" and the track section state becomes "Clear". The new state is indicated to the MMI.
Test Class Dependency:	TCL-1
Requirements Identifiers:	rim_58; rim_92; rim_56; rim_5646; rim_5234
Input Specification:	The Track section state is Failed. (The other track section states are irrelevant).
Output Specifications:	The track station state is Clear. The rest of the states are not modified. The new state "Clear" is indicated to the MMI.
Event:	The notification that the track section state is "Clear" (There is no train in the section).
Inter-Case Dependencies:	
Test type:	Functional test.
Environmental Needs:	Every element in the station is "Clear" except this track section that its state is failed. There is no route, no block, no Local Operation established, no ARS activated and the station is under local control.

Examples

Test Data Specification file *.xls

Type	ID	Information	Description	TC1	TC2
S	1	OCC-PH	Clear	0	1
S	2		Occupied	0	0
S	3		Failed	1	0
S	4	OCC-LOG	Log.Occupied	1	0
S	5		FreeAsOccupied	M	M
S	6		NotSEQOccupOrFreed	M	M
S	7		UnexpOcc	M	M
S	8		SeqOcc	M	M
S	9		BSEQ	M	M
S	10	INH	R.Inhibition	0	0
NI:S	11		R.Inhibition by Destination	0	0
S	13		MRPmr	0	0
S	14		MOmr	0	0
NI:S	15		MAmr	0	0
S	16		MFPmr	0	0
S	17	BL	MB	0	0
NI:S	18	LO	MLO		
E	1	***Events***	Train_Occupies	2,3	1,2,4
E	2		Train_Leaves	1,3,4	SC
E	3		No_Data	SC	1,3,4
E	4	***Commands***	R._Inhibition		
NI:E	5	Commands	R._Inhibition_by_Destination		
E	6	Commands	Cancel_RI		
NI:E	7	Commands	Cancel_RID		
NI:E	8	**Set_Route_no_ARS_Where**	Approach		
E	9	**Set_Route_no_ARS_Where**	Running_Path_Same_Direction_no_last		
NI:E	10	**Set_Route_no_ARS_Where**	Running_Path_Different_Direction_no_last		
NI:E	11	**Set_Route_no_ARS_Where**	Running_Path_Same_Direction_last		
NI:E	12	**Set_Route_no_ARS_Where**	Running_Path_Different_Direction_last		
E	13	**Set_Route_no_ARS_Where**	Overlap_Same_Direction		
NI:E	14	**Set_Route_no_ARS_Where**	Overlap_Different_Direction		
E	15	**Set_Route_no_ARS_Where**	Flank Protection		

Examples

```
TEST FUNCTIONS

void testim1(unsigned short Cycle)
{
 unsigned char TestCase[15]="Case s1_0";

 switch (Cycle)
   {
  case 1:  CICLO 1 FILE PRINTING INIT
    TS_SetFree (21,false);
    TS_SetOcuppiedPhysically (21,false);
    TS_SetF...
    break;
  case 2:  CICLO 2 INIT CHECK
    if ( !TS_IsFree (21) && !TS_IsOcuppiedPhysically        (21) && TS_IsFailed(21)
                    && TS_IsBlocked(21) && TS_IsInMainRoute(21) &&
                     TS_I....
    else
                   {
                     Init NO OK
                   }
    break;
```

Test Scripts to be compiled, linked and executed together with the validated system (file.c)

Examples

Real outputs (file.out)

Test Case Case s1_0 TS_IsFree(21): 0 0 TS_IsOcuppiedPhysically(21): 0 1 TS_IsFailed(21): 1 0 TS_IsBlocked(21): 1 1 TS_IsInMainRoute(21): 1 1 TS_IsInOverlapOfMainRoute(21): 1 1 TS_IsInAproximationRoute(21): 1 1 TS_IsInFlankProtectionRoute(21): 1 1 TS_IsInBlockRoute(21): 1 1 C: TS_IsOcuppiedPhysically, TS_IsFailed	Test Case Case s1_2 TS_IsFree(21): 0 0 TS_IsOcuppiedPhysically(21): 0 0 TS_IsFailed(21): 1 1 TS_IsBlocked(21): 1 1 TS_IsInMainRoute(21): 1 1 TS_IsInOverlapOfMainRoute(21): 1 1 TS_IsInAproximationRoute(21): 1 1 TS_IsInFlankProtectionRoute(21): 1 1 TS_IsInBlockRoute(21): 1 1 C: --

Examples

Message queues (file.out)

```
Case s1_0 Init OK
Queue 6, n-mssg 0
T: 961085411,764 -- pos 6 s,1,0,1,0,15,0,0,0,3,0,0,0,
T: 961085411,865 -- pos 6 r,1,0,1,0,15,0,0,0,3,0,0,0,
...
Queue 0, n-mssg 168
T: 961085411,967 -- pos 0 s,4a,15,0,1,87,1,0,
Queue 4, n-mssg 0
T: 961085411,968 -- pos 4 s,3,0,1,0,23,0,1,0,0,0,0,0,0,0,
Queue 4, n-mssg 1
T: 961085411,968 -- pos 4 s,3,0,1,0,22,0,1,0,0,0,0,0,0,0,
T: 961085411,969 -- pos 4 r,3,0,1,0,23,0,1,0,0,0,0,0,0,0,
T: 961085411,970 -- pos 4 r,3,0,1,0,22,0,1,0,0,0,0,0,0,0,
...
Queue 0, n-mssg 169
T: 961085412,475 -- pos 0 s,4b,3,1,40,0,0,0,56,0,0,0,1,1,
Queue 0, n-mssg 170
T: 961085412,476 -- pos 0 s,4b,3,1,40,0,0,0,55,0,0,0,1,1,
...
Queue 0, n-mssg 171
T: 961085412,578 -- pos 0 s,4a,f,0,1,3,0,0,
Queue 0, n-mssg 172
T: 961085412,578 -- pos 0 s,4a,12,0,1,3,0,0,

Test Case Case s1_0
There are queues with messages
```

Examples

Results analyzer (file.res)

```
Test Case Case s1_0: OK
Test Case Case s1_1: OK
Test Case Case s1_2: OK
Test Case Case s1_3: NOK -->
TS_IsInMainRoute(21): 1 0
Test Case Case s1_4: OK
Test Case Case s1_5: OK
Test Case Case s1_6: OK
Test Case Case s1_7: OK
Test Case Case s1_8: OK
Test Case Case s1_9: OK
Test Case Case s1_10: OK
Test Case Case s1_11: OK
Test Case Case s1_12: OK
Test Case Case s1_13: OK
Test Case Case s1_15: OK
Test Case Case s1_15: OK
Test Case Case s1_16: OK
Test Case Case s1_17: OK
Test Case Case s1_19: OK
Test Case Case s1_20: OK
```

Conclusions and figures (I)

- Main Targets achieved:
  - Systematic validation strategy
    - benefit: repeatable
  - Test execution automated
    - benefit: decrease the execution time (2 weeks-2 days)
    - benefit: allows to be executed by everybody
  - Validation integrated in the development life cycle
    - benefit: sinergy effects
  - Traceability maintained

Conclusions and figures (II)

- Test cases automated: app. 1800
- Test cases semi-automated: app. 750
- Test cases non automated (manual): app. 250
- Overall effort (excluding test execution): 2,5 p/y
- Overall time spent: 1,5 year

QWE2002 Paper 10T

Mr. Anders Claesson
(Enea Realtime AB)

How to Use Scientific Methods In Software Testing

Key Points

- Enhance your software testing efficiency by the use of scientific methods.
- Learn the different steps in the scientific method.
- Learn how to apply the methods for different test activities.

Presentation Abstract

The scientific method is a process that describes how scientists think and what they do during scientific research. Creativity and critical thinking are important parts intimately integrated in the used problem-solving methods. The purpose of science is to gain knowledge about nature.
The purpose of software testing is to gain knowledge about the system you are testing.

About the Author

Anders have been working with testing of software and systems, mainly real-time and telecommunication systems Since 1982, but also large client server systems. Anders is currently working as a consultant at Enea Realtime AB. Anders is writing test processes and methods for system testing. He also works with test process improvements for different customers.

ENEA Realtime

How to use Scientific Methods in Software Testing

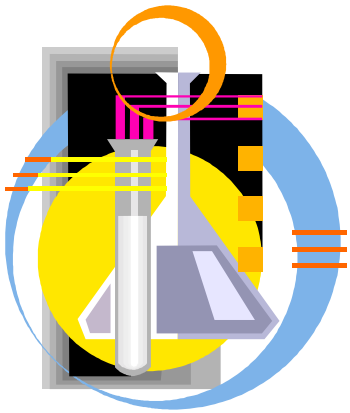
Anders Claesson
ancl@enea.se

ENEA Realtime

ENEA Realtime

Contents

1. Introduction
2. Overview
3. Detailed Steps
4. Application
5. Conclusions

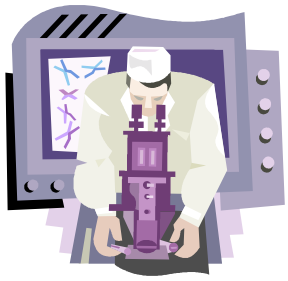


© Anders Claesson

1. Introduction

1.1 The Basic Steps in the Scientific Method

1 Observe.
2 State questions.
3 Form hypothesis.
4 Design experiment.
5 Test the hypothesis.
6 Draw conclusions.
7 State additional questions.



1. Introduction

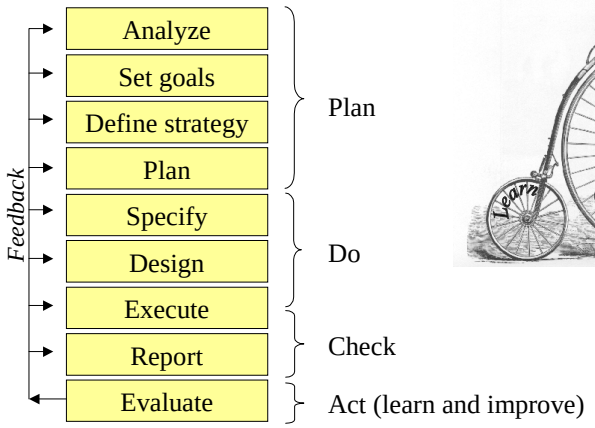
1.2 Why use Scientific Methods in Software Testing

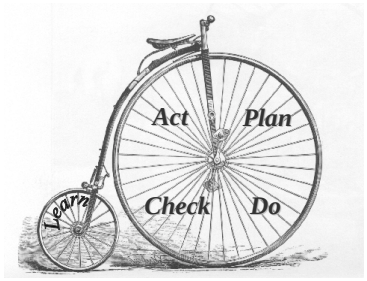
- Define testing goals.
- Select test strategies.
- Identify and select areas to test.
- Select test cases.
- Execute test cases.
- Find faults.
- Draw conclusions from the test results.
- Determine when to stop testing.

© Anders Claesson

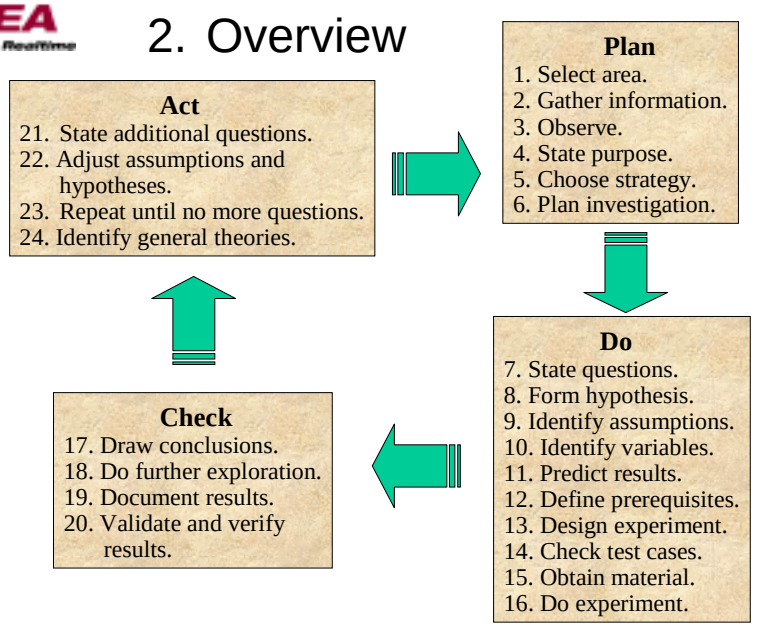
1. Introduction

1.3 The General Steps in the Testing Process





2. Overview



© Anders Claesson

ENEA Realtime

3. Detailed Steps

Plan

1. Select area
2. Gather information
3. Observe
4. State purpose
5. Choose strategy
6. Plan investigation

3.1 Select Areas to Investigate

- Study usage, functions and characteristics in the current system.
- Brainstorm for ideas, analyze risks.
- Divide areas in categories of the user, system and risks.
- Use the PMI method for the selection.

3.2 Gather Information

- Gather written information.
- Read between the lines.
- Interview people.

ENEA Realtime

3. Detailed Steps

Plan

1. Select area
2. Gather information
3. Observe
4. State purpose
5. Choose strategy
6. Plan investigation

3.3 Observe and Analyze the Area you want to Investigate

- Obstacles and uncertainties.
- Likely/unlikely to fail.
- Highest risks.
- Critical system parts.
- Problem areas.

3.4 State Purpose and Goal for the Investigation

- Why you are doing the investigation.
- Subjects to be investigated.
- Depth of investigation.
- What you expect to learn.

© Anders Claesson

ENEA Realtime

3. Detailed Steps

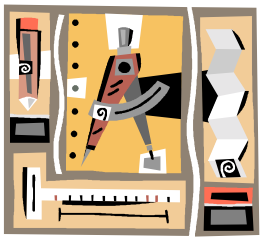
Plan
1. Select area
2. Gather information
3. Observe
4. State purpose
5. Choose strategy
6. Plan investigation

3.5 Choose Strategy for the Investigation

- Strategy for the inquiry.
- Methods, tools and processes.
- Selection of test cases.
- Risk management.
- Preventive activities.

3.6 Plan your Investigation

- Prepare a time schedule.
- Specify test environment.
- Identify test objects.
- Align the plan with the overall test plan.



ENEA Realtime

3. Detailed Steps

Do
7. State questions
8. Form hypothesis
9. Identify assumptions
10. Identify variables
11. Predict results
12. Define prerequisites
13. Design experiment
14. Check test cases
15. Obtain material
16. Do experiment

3.7 State your Questions for the Investigation

- Use the “six thinking hats” when questioning.
- Ask what, which, where, when, how, who.
- Ask “what if ….” many times.
- Ask why at least five times.
- Ask questions based on your observations.
- Possible answers must be measurable.

© Anders Claesson

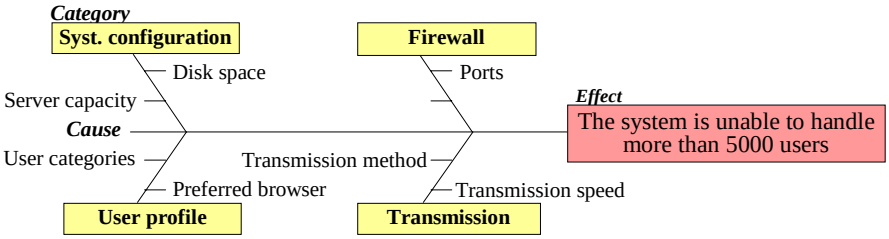
3. Detailed Steps

Do

7. State questions
8. Form hypothesis
9. Identify assumptions
10. Identify variables
11. Predict results
12. Define prerequisites
13. Design experiment
14. Check test cases
15. Obtain material
16. Do experiment

3.8 Form a Hypothesis

- A hypothesis is an assumption that you want to prove or disprove.
- Suggestions of cause and effect relationships of issues, problems or faults.



3. Detailed Steps

Do

7. State questions
8. Form hypothesis
9. Identify assumptions
10. Identify variables
11. Predict results
12. Define prerequisites
13. Design experiment
14. Check test cases
15. Obtain material
16. Do experiment

3.9 Identify general Assumptions

- Parts of the system assumed to be working.
- Assumed user behavior.
- Assumed system behavior when unexpected events occur.

3.10 Identify Variables

- Identify what factors may affect the subject under investigation.
- List variables in order of importance.
- Identify independent and dependent variables.
- Specify type and value range.

ENEA Realtime

3. Detailed Steps

Do
7. State questions
8. Form hypothesis
9. Identify assumptions
10. Identify variables
11. Predict results
12. Define prerequisites
13. Design experiment
14. Check test cases
15. Obtain material
16. Do experiment

3.11 Predict the Results of the Hypothesis

1. If the system is unable of handling 5000 users,
2. The the system will get blocked (i.e. too long response time),
3. When trying to send 5 transactions per second from 5000 simultaneous users.

3.12 Define Prerequisites for when the Hypothesis can be Considered True

- Which other hypotheses must be true?
- Which system configuration must be used?
- Which functionality must be delivered?

ENEA Realtime

3. Detailed Steps

Do
7. State questions
8. Form hypothesis
9. Identify assumptions
10. Identify variables
11. Predict results
12. Define prerequisites
13. Design experiment
14. Check test cases
15. Obtain material
16. Do experiment

3.13 Design Experiments to Test your Hypothesis

- Make a step-by-step list.
- Prepare test cases.
- Only change one variable at a time.
- Prepare both positive and negative test cases.

3.14 Check Test Cases

- Applicable, understandable, complete, consistent, correct, unambiguous, executable.

© Anders Claesson

ENEA Realtime

3. Detailed Steps

Do

7. State questions
8. Form hypothesis
9. Identify assumptions
10. Identify variables
11. Predict results
12. Define prerequisites
13. Design experiment
14. Check test cases
15. Obtain material
16. Do experiment

3.15 Obtain Materials, Equipment and Prepare for the Test

- Test equipment.
- Load test data.
- Start the log functions.

3.16 Do the Experiments and Record the Data

- Run different categories of test cases for each question and hypothesis.
- Measure how much the changed input variables affect the system.
- Run improvised test cases.
- Record all your observations.

ENEA Realtime

3. Detailed Steps

Check

17. Draw conclusions
18. Do further exploration
19. Document results
20. Validate/verify results

3.17 Draw Conclusions from the Experiments

- Were predictions confirmed?
- Did any failures occur?
- Are the results consistent?
- Explain what proves/disproves your hypothesis?

3.18 Do further Exploration of the Hypothesis if needed

- Consider other hypotheses making the same predictions.

Original hyp: The system is unable to handle more than 5000 users.

Additional hyp: The system gets blocked when 5 of the most common transactions are sent from 3000 users at 5 transactions/second.

© Anders Claesson

ENEA Realtime

3. Detailed Steps

Check

17. Draw conclusions
18. Do further exploration
19. Document results
20. Validate/verify results

3.19 Document Test Cases and Results

- System under investigation.
- Found faults.
- Risks.
- Executed test cases.
- Conclusions.

3.20 Validate and Verify the Results

- Did you investigate the right problem?
- Were the results reliable?
- Did the evaluated cause and effect relationships cover your questions and hypotheses?

ENEA Realtime

3. Detailed Steps

Act

21. Additional questions
22. Adjust hypotheses
23. Repeat until no more Q
24. Identify general theories

3.21 State New or Additional Questions

- Identify areas you want to obtain more knowledge.
- Add questions based on previous experiment results and conclusions.

3.22 Adjust Previous Assumptions and Hypothesis

- Are your hypotheses wrong or inaccurate?

3.23 Repeat the Preceding Steps

- Repeat steps to gain more knowledge.

3.24 Identify General Theories

- Are there strong evidence of a hypothesis being true?

4. Application

4.1 How to Apply the Scientific Method

- Start with a simple approach:
  1 State question.
  2 Form hypothesis.
  3 Test hypothesis.
  4 Draw conclusions.
  5 State new/additional questions.
- You don't have to write down everything. Just know what you are doing and why (in a structured way).
- Use the rigorous approach as a checklist.

5. Conclusions

5.1 Summary

The scientific method provides a framework and a structure for:

- Solving problems.
- Investigating different subject areas.
- Stating questions and hypotheses.
- Testing the hypotheses.
- Evaluating experiment results.
- Drawing conclusions and gaining new knowledge.

It is a efficient way to systematically research the different properties of a system under development, and a way to improve the software testing process.

How to use Scientific Methods in Software Testing

Anders Claesson, Enea Realtime - Test
Enea Realtime AB
www.enea.se
Box 232, S-183 23 Täby, Sweden
Tel: +46-8-50 714 000
Fax: +46-8-50 714 040
E-mail: ancl@enea.se

© 2001 Enea Realtime AB
All rights reserved

Abstract

The scientific method is a process that describes how scientists think and what they do during scientific research. By using scientific methods researchers can systematically investigate their observations, ideas or problems in their specific domain (e.g. physics, chemistry, biology). By conducting experiments the scientist's hypotheses can be tested and new knowledge obtained.

This paper describes how scientific methods can be used in software testing. In testing you want to obtain knowledge in many different areas, such as:

- Which parts of the system works?
- Which parts don't work?
- What kind of faults exists?
- Where are the faults located?
- Are all requirements implemented?
- What are the risks in the system?
- When can the system be released?

The basic steps in the scientific method are:

1. **State the question.**
   What happens if..? Is the system stable when..?
2. **Form a hypothesis.**
   A proposed explanation, or assumption, of an observed or predicted condition, behavior or characteristic of the system. Example: non-alphabetical characters given as input may cause the system to crash because the input data is not properly checked.
3. **Test the hypothesis.**
   Design test cases that show if the hypothesis was correct or not. Predict the result of the test based on what would be considered a correct behavior of the system (for example a received error message). Run the test.
4. **Draw conclusions from the test results.**
   Was the hypothesis correct? Or does it need to be changed based on the new knowledge obtained from the test? Analyze any unexpected events or behavior and write trouble reports if the unexpected event or behavior was considered to be a fault. Did the test results reveal any new issues that needs to be investigated?
5. **State a new question based on the results from the previous.**

Scientific methods can be used to:

- Choose the most efficient test strategies.
- Select test cases, run the test cases and evaluate the results.
- Explore the system (perform exploratory testing).
- Identify, and isolate faults (debugging).
- Evaluate risks (risk based testing).

Contents

1 Introduction 3
2 Overview of the steps in the scientific method 4
3 Description of the different steps 6
3.1 Select areas to investigate 6
3.2 Gather information 8
3.3 Observe and analyze the area you want to investigate 8
3.4 State the purpose of the investigation 9
3.5 Choose strategy 9
3.6 Plan your investigation 10
3.7 State your questions 10
3.8 Form a hypothesis 11
3.9 Identify general assumptions 11
3.10 Identify variables 12
3.11 Predict the results of the hypothesis 12
3.12 Define the prerequisites for when the hypothesis can be considered true 13
3.13 Design experiments to test your hypothesis 13
3.14 Check test cases and test scripts 14
3.15 Obtain materials, equipment and prepare for the experiment 14
3.16 Do the experiments and record the data 14
3.17 Draw conclusions from the experiments 15
3.18 Perform further exploration of the hypothesis if needed 15
3.19 Document your test cases and your observations, results and conclusions of the experiment 16
3.20 Validate and verify the results and the conclusions 17
3.21 State new or additional questions based on the experiment results and your conclusions 17
3.22 Adjust previous assumptions and hypotheses 18
3.23 Repeat the preceding steps until no further questions remain 18
3.24 Identify general theories based on the experiments performed 18
4 How to apply the scientific method 19
5 Conclusions 19
6 References 20

1 Introduction

Scientists study all kinds of natural phenomena, such as the origin of the universe, electricity carrying power lines, cell structures, medical treatments, etc., covering many different areas.

Common for all scientists is their way of obtaining knowledge. How they form new theories and explanations of what they see or experience. This is performed by using scientific methods, which is a structured way to systematically investigate observations, problems or ideas in the domain of the scientists' research [Edmund -00]. By conducting experiments the scientists' hypotheses (ideas for explanations) can be tested and evaluated in order to gain new knowledge. Whether your experiments succeed or fail, you will always gain knowledge (of what did or did not work).

To design a suitable experiment you must make an educated guess about the things that may affect the system you want to investigate. These are called variables.

The existence of "cause and effect relationships" in nature is what makes experimental science possible. Hypotheses can only be verified using the scientific method if there is a cause and effect relationship between the variables you have chosen and the system you are studying. Your hypothesis is the best guess at what the cause and effect relationship is. Your conclusions will allow you to predict the result of future cause and effect relationships.

Conclusions may be derived by using inductive reasoning, that is, to attempt to discover general principles by careful investigation of specific cases. To apply these general principles to the investigation of other specific cases is called deductive reasoning.

The scientific research method can also be used in software testing. There are many things that are interesting for you as a tester to find out, which could be to:

- Find test cases.
- Find the most critical faults in the system.
- Identify and evaluate the most critical risks.
- Determine if the system is ready for release.

Your knowledge, experience, strategies, processes, methods and tools will be crucial for your level of success. The scientific method can be used in all steps of a testing process [Claesson2 -01], [Claesson3 -01].

When you have been working as a tester for a couple of years your testing skills will in most cases naturally develop into a scientific way of reasoning and problem solving (since most of it is common sense). If you want to reach a higher level of efficiency faster you may want to start using the practices described in this paper.

Your testing awareness and maturity level will also have a major effect on which goals you choose for testing [Beizer-90] and how you want to apply the scientific method. You may want to focus your research on revealing the risks in using the system (risk based testing) [Claesson3 -01] as well as searching for faults you encounter during testing.

A number of different thinking techniques can be used together with the scientific method (e.g. critical thinking, logical reasoning, visualization, prediction, imagination, interpretation, perception) [De Bono-85], [Claesson3 -01].

The description of how to use scientific methods in software testing is divided in three parts.

- The first part gives an overview of the different steps.
- The second part consists of the detailed descriptions of each step in the process.
- The third part describes how you can start using the methods described in this paper.

2 Overview of the steps in the scientific method

Testing can be seen as a research project where you want to obtain knowledge in a number of different areas. When researching you will conduct investigations, solve puzzles and explore the outcome of the design. It is a creative, fun and rewarding task when successful. To increase your ability of reaching success the following steps in the scientific method can be used.

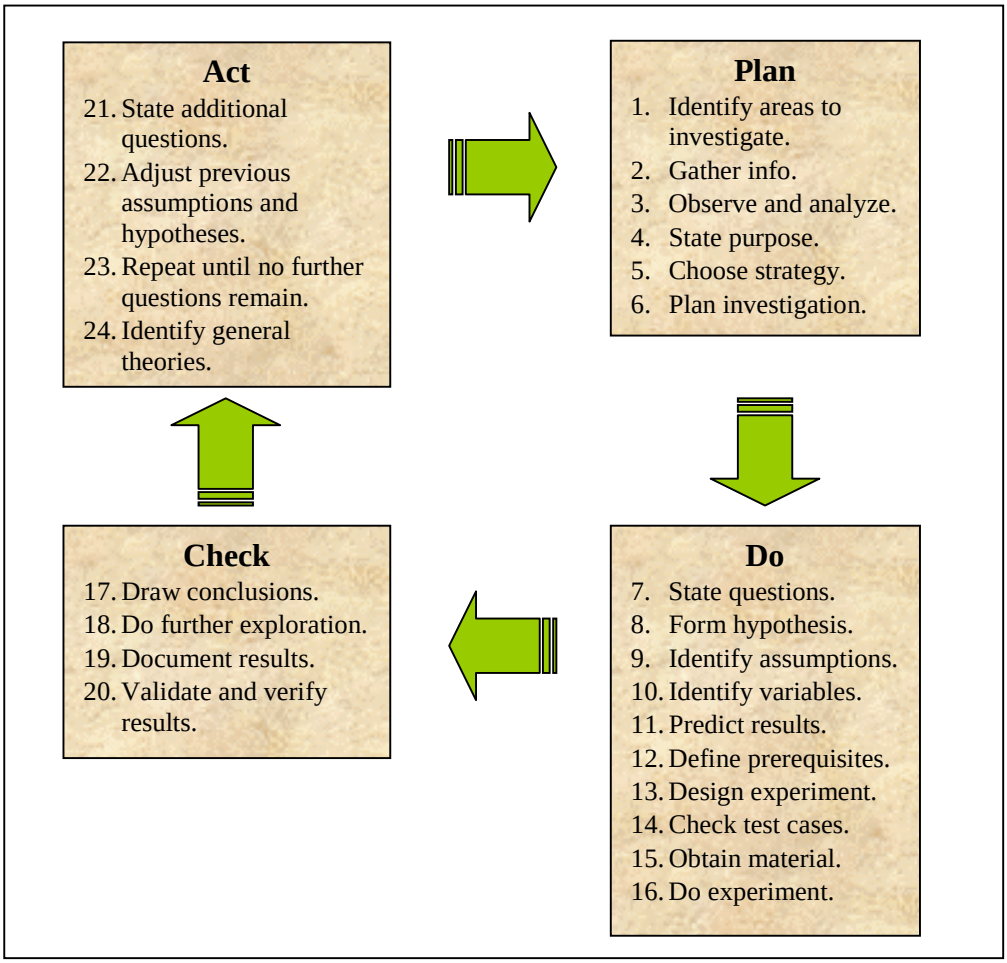


Fig. 1 The figure shows all the steps in the scientific method adapted for software testing.

The PDCA model has been chosen to provide a framework for the different steps in the process (see appendix A in [Claesson3 -01]. The PDCA cycle is an iterative model, which apply to the nature of the scientific method (i.e. form hypothesis, test, draw conclusions, new hypothesis, test, etc.).

Plan

1 **Select areas to investigate.**
Find the most important areas to conduct a research. Start with an unbiased search for clues of where to look. Use experience, imagination and evidence of possible problem sources.

2 **Gather information (facts).**
Gather all the facts you can get in the areas to be investigated. Keep track of the source of the information. Use a variety of information sources. “Read between the lines” to find non-obvious aspects that are important to investigate.

3 **Observe and analyze the area you want to investigate (facts, problems, ideas).**
Analyze the facts you have gathered. Investigate any ideas you get. Note interesting issues that you might want to investigate further. Are there any problem areas?

4 **State the purpose of the investigation.**
Define goals for your investigation. For example: to find out the maximum number of possible users that can be logged on to, and using, the system at the same time. Also specify the prerequisites and constraints associated with each goal.

5 **Choose strategy.**
Choose a strategy suitable for the subject you want to investigate. Select which processes, methods and tools you would like to use. Identify how you want to approach the problem or issue to be investigated.

6 **Plan your investigation.**
Plan the steps necessary for the investigation. Identify needed resources, such as test tools, test equipment and human resources. Make a time schedule with a deadline. Integrate your investigation planning with your overall test plan.

Do

7 **State your questions.**
Ask questions related to the area of your investigation. How does this work/not work? The question may be raised when observing the behavior of the system, when an unexpected response/event occurs or by critical thinking.

8 **Form a hypothesis.**
Give plausible explanations (answers) to what you have observed or the possible cause and effect of a problem. What, how and why a specific action may cause a failure in the system or how to determine what a correct behavior of the system is.

9 **Identify general assumptions.**
Identify all assumptions you have made regarding the investigation. Which parts/factors of the subject are assumed to have little or no influence on the result? Which parts will not be changed during the experiment?

10 **Identify variables.**
Identify or make an educated guess of what things might affect the subject under investigation/test. Which are the most important variables that are involved in the cause and effect relationship you want to study?

11 **Predict the results of the hypothesis.**
What is the expected result if the hypothesis is true? What data is expected? What system behavior, printouts, response information on the screen, error messages do you expect? Is there anything that should not happen?

12 **Define the prerequisites for when the hypothesis can be considered true.**
Which system configurations will be used in the experiment (in order to make the outcome valid for the hypotheses made)? Which functionality must be delivered and installed?

13 **Design experiments to test your hypothesis.**
Prepare test cases to show if the hypothesis was correct or not. If required also prepare test scripts (for load tests or other automated tests). Prepare both positive and negative test cases to obtain more reliable results to make the hypothesis easier to validate.

14 **Check test cases and test scripts**
Check that the test cases are well described and reflect all the questions and hypotheses you want answers to.

15 **Obtain materials, equipment and prepare for the experiment.**
Gather required test equipment (tools, hardware, cables, software). Load any test data needed. Start the log functions in the system (e.g. error logs, system logs).

16 **Do the experiments and record the data.**
Perform required actions to reach the preconditions of the test cases. Execute your test cases. Navigate your way through the execution to know where you are, where you have been, where you are going and the stops you have made along the way. Record all steps taken.

Check

17 **Draw conclusions from the experiments.**
Draw conclusions from the test results to find out whether the hypothesis was correct or not. Did any failures occur? What is considered being a failure? Write trouble reports both on the product and the process. Are the results consistent with earlier results, general theories and related hypotheses?

18 **Perform further exploration of the hypothesis if needed.**
If a hypothesis is supported by the results of the test you might want to consider other hypotheses that make the same predictions (what other actions may give the same result?).

19 **Document your test cases and your observations, results and conclusions of the experiment.**
Keep track of what you have done. Save log files, document test cases, document/save issues, events, risks and knowledge gained during the experiment/test. Write down your conclusions.

20 **Validate and verify the results and the conclusions.**
Validate your results. Were the results reliable? Did the results represent a valid relationship between cause and effect or was it mere accidental?

Act

21 **State new or additional questions based on the experiment results and your conclusions.**
Identify areas where you want to obtain more knowledge based on the results from the current experiments. In most cases you will need to ask more questions.

22 **Adjust previous assumptions and hypotheses.**
Make adjustments to your initial assumptions and hypotheses based on your results and conclusions from the experiments. Design additional experiments that support or reject your assumptions and hypotheses.

23 **Repeat the preceding steps until no further questions remain.**
Repeat some or all preceding steps to gain more knowledge. Any number of iterations of the steps in the process are possible as long as you keep the structure and know what you are doing (follow the goals and strategy outlined for the process of your investigation).

24 **Identify general theories based on the experiments performed.**
If there is strong evidence for a hypothesis of being true based on the results from several independent tests (and re-tests), the hypothesis becomes a theory. A theory might be that the developers make the most mistakes when checking boundary values of used data.

3 Description of the different steps

Detailed description of all the steps in the scientific method.

3.1 Select areas to investigate

This is the most important part of the whole process. If you miss an important area, such as reliability or performance of the system, you might face serious problems when the system is put into service (for example see [Splaine -01]).

> ***From a testers' point of view all the systems in the world can be divided in two categories:***
> ***1. Systems that crash.***
> ***2. Systems that have not yet crashed.***

If you are uncertain of how to identify an important area you might want to start an unbiased observation/search first (to gather information and search for clues where to look). The search for clues can be performed in many ways, such as:

- Looking backward (past experiences) and forward (risk and failure mode analysis) in time.
- Viewing the system from three perspectives of the user, the system and the associated risks.
- Looking at positive and negative aspects using the six thinking hats and the PMI method [De Bono-85].

The backward/forward looking method.

Looking backward:

- Studying reports from previous projects (where did they experience problems in the past).
- A root cause analysis of the most serious problems found in the field and in previous projects [Ammerman -98], [Gano -99].
- Studying the current release of the system, its functions, characteristics, usage and problems.

Looking forward:

- A brainstorming meeting (with designers, customer support, testers, requirement authors, product management, integrators, quality assurance responsible…). To obtain ideas of important issues to be looked into refer to [Quality -01], [Thinking -98], [Focus -01].
- A technical risk analysis of the system [Claesson3 -01], [Connell -96].
- Studying the system, how it is supposed to work and how it is intended to be used.
- A Failure Mode and Effect Analysis (FMEA) [McDermott -96], [Lutz -96], [Wallace -95].

Known quality management techniques can also be used when conducting your search for areas, problems and issues of interest to be investigated, see [Quality -01], [Mears -95].

If you are an experienced tester you probably have many good ideas of what needs to be investigated. You remember what problems you faced in the previous release and you might also have been involved in trouble shooting problems reported from your customers.

Try to obtain knowledge in the following areas:

- How to use your test process, methods and tools in an efficient way.
- How your organization works.
- How to communicate with the designers and get the input information you need.
- The system you are going to test.
- Other people's experiences from previous projects.
- The customer's requirements, problems, needs, expectations and values.

Three perspectives of the system.
Important issues to be investigated can be divided in three main areas:

User Requirements, needs, expectations and values.

System Functions and characteristics.

Risks Failure to meet or fulfill the above.

When conducting further research also look at:

- The system **inside out** (system view).
  - System boundaries and interfaces.
  - Functions.
  - Characteristics.
  - Configurations.
- The system **outside in** (actor view).
  - Actors.
  - What are the possible inputs (value ranges, types and formats)?
  - What are the possible outputs (error messages, result printouts, messages on the screen?
  - Use cases/scenarios.
  - Business processes.
  - Business rules.
- Causes, conditions (system state), actions (trigger of events) and resulting effects.
- Expected outcome and benefits (main purpose) of using the system.

The PMI method.
You may also use the "PMI-method" and the six thinking hats [De Bono-85] when selecting areas to investigate.

Example:

How to consider investigating the system performance:

+ The number of maximum users is only a rough guess made at the marketing department.
+ The projection of the growth rate for the number of users has been inaccurate in the past.
+ The last project was delayed mainly because of capacity problems.
- It is very expensive and time consuming to make extensive performance tests.
- There is not enough HW available to perform any tests of the systems' capacity limits.
i Similar systems on the market don't reach the capacity levels of our system.

Select and prioritize which areas to be investigated based on:

- The highest risks, e.g. the risk of an unstable system.
- Conditions with cause and effects having the most impact for the user and the system, e.g. 1000 simultaneous users logging on to the system performing updating of their repository may have the effect of slow response times. Slow response time is considered a major issue for the users.
- The most important requirements.
- The most used (and visible) areas of the system.
- The most important areas, e.g. system restart functions.
- The most modified/changed parts.
- The parts with the most vague/unclear specifications.
- The parts using new or untested technology.
- Areas with a severe defect history (e.g. where frequent system crashes occurred in the past).

When to start the investigation:	As soon as the project starts.
Output:	A prioritized list of areas to investigate.

3.2 Gather information

Gather all the facts you can get in the areas that should be investigated (e.g. requirements).

Interview people (e.g. customer support, designers, system architects, operation and maintenance responsible). Ask them to use the six thinking hats when giving you the information (see [De Bono-85] and [Focus -01]).

Example of things to ask for:

- Facts about what the system is supposed to do and how well it will serve the customer/user.
- Facts about configuration, interfaces, usage and characteristics.
- Ideas regarding aspects to consider in an investigation, i.e. what to look for.
- Feelings and beliefs about the new system.
- Anticipated problems and risks.

Output: Input information for the investigation.

3.3 Observe and analyze the area you want to investigate

Analyze the input information you have obtained. Look for clues that might reveal problem areas or issues that need further investigation [Norman -88].

- Are there any obstacles or uncertainties in reaching the objectives of the system (regarding functionality and characteristics)?
- Are all public system interfaces defined and under configuration control?
- Have any of the interviewed persons expressed any concerns about the subject area that will be investigated?
- What is likely/unlikely to go wrong?
- What is considered critical if the system failed in a certain area?
- Are there any gaps/uncertainties regarding the requirements?
- Where do you lack input information in order to select test cases (i.e. incomplete specifications)?
- Which characteristics of the system are the most important for your customer?

- What common mistakes do the designers make?

Identify specific problems in the area to be investigated [Thinking -98]:

Identify all possible problems that stand in the way of reaching the project and system goals you may come up with.

Select the problems that have the most negative impact on the project and system goals.

Identify which problems that can be addressed/evaluated in testing.

Output: Issues, problems, risks and/or ideas to be investigated.

3.4 State the purpose of the investigation

Decide what you want to find out in the area you have selected for the investigation.

Example:

1. To find out the system limits regarding the number and types of transactions and the maximum number of users for a certain system configuration.
2. To evaluate the risks regarding the stability of the system platform.

Each purpose could be used to define a test object. Specify:

- Why are you doing this investigation?
- Which issues, problems, risks or subjects do you want to investigate?
- Which are the prerequisites associated with the goal?
- Which test phase will be involved in the investigation?
- Which depth of investigation do you aim for (i.e. test coverage)?
- What do you expect to learn?

Check that the purpose of each individual investigation is in line with your overall testing goals.

- Test coverage (of feature/function, use case, component, unit, code or characteristic).
- Risk coverage (to what extent should risks be evaluated?).
- Error trend (to what trend level should testing continue?)
- The timeframe/deadline in which testing should be completed.

Output: Purpose and goals for the subject to be investigated.

3.5 Choose strategy

Select a suitable strategy for the subject you want to investigate. Choose strategies both before the investigation starts and during the investigation after questions and hypotheses have been formulated.

- Strategy for further inquiries (i.e. a suitable approach when asking questions and forming hypotheses).
- Process, methods and tools for testing your hypotheses.
- Test steps and test phases (e.g. system test).
- How to reach sufficient test coverage and confidence in the results.
- Test environment (i.e. the scale of the system configuration to be used).
- Test techniques [Beizer-90].
- Test case selection strategy [Claesson1 -99].
- Test execution strategy [Claesson2 -01].

Output: An investigation strategy.

3.6 Plan your investigation

Plan all the steps included in the investigation that you have specified in the strategy.

- Prepare a time schedule (start and stop dates).
- Estimate needed effort (man-hours) required for the investigation.
- Specify the needed test environment.
- Allocate resources (people and equipment).
- Identify which test objects that should be included?
- Plan how to control constants and variables in your investigation.
- Are there any risks involved in the investigation?

Align your investigation planning with the overall test planning in your project.

Your plan needs to be flexible to allow changes in the overall goals, hypotheses and experiments.

You may plan the investigation by using time boxes [Stapelton -97], in 2-3 weeks of duration, where you specify:

- The tasks to be performed in the investigation.
- Input information required.
- Expected output from the investigation.
- Start date.
- Planned finish date.
- Estimated number of man-hours.
- Who is responsible.

Output: An investigation plan.

3.7 State your questions

Ask questions that focus on the purpose of your investigation and what you want to know.

Example:

System performance.

- Is the system capable of handling 1000 simultaneous users even though there are no clear requirements related to system performance?
- What if the system restarts in the middle of a high load situation?

Ask yourself as many questions as you can come up with. Use your six thinking hats [De Bono-85] in the process of discovering different approaches of what questions to ask.

Use “what-iffing” techniques [Thinking -98]. That is to ask **what if** many times related to the area to be investigated and the focus of the subject. What-iffing involves describing an imagined situation, condition, action, state or event, and then examining associated facts, consequences and effects.

- What if we do this, what would happen then?
- What if the requirements are wrong, or the estimation of the total number of users?
- How does this function actually work?
- What are the main problems previously observed that you would like to know more about?

Ask questions based on your observations and analysis of the system.

Use different views (the user, the customer, the system operation and maintenance person, other communicating systems); ask why (at least 5 times).

You may also want to continue asking questions including: who, what, which, where, when, how.

The possible answers to your questions must be measurable/testable.

Output: A set of questions for inquiry and investigation.

3.8 Form a hypothesis

What do you think the answers to your questions may be? Your hypotheses may explain the previous observations.

Example of a hypothesis

- The system is unable to handle more than 5000 users simultaneously logged on to the system while asking for information about their account.

A hypothesis may include the following premises:

1. The system involved and its configuration.
2. Reasons/arguments for the hypothesis of being true.
3. Precondition (the required system state for when the hypothesis can be considered true).
4. Actors involved (users or other systems).
5. Scenario/use case.
6. Variables and constants.
7. Data (input and output).
8. Actions/events.
9. Presumed results.
10. Postcondition (end state of the use case or scenario, e.g. error message displayed).
11. Rules and/or limitations of the functional behavior or characteristics involved.

Make a fish bone diagram to show the cause and effect relationship for the statements in your hypothesis. Example:

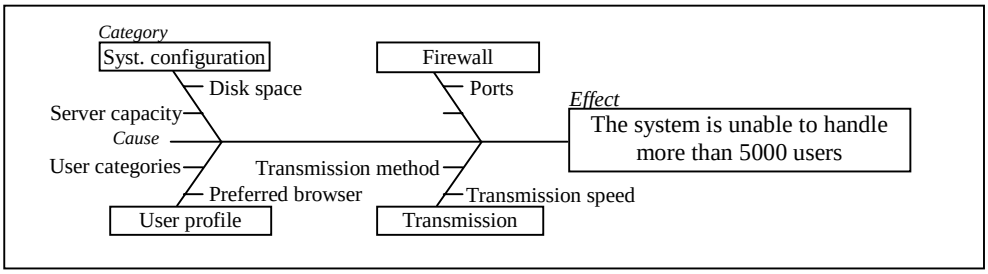


Fig. 2 Example of a cause and effect diagram where you can see what may cause/influence your hypothesis of being true.

Output: A hypothesis with a proposed cause and effect relationship.

3.9 Identify general assumptions

Identify and list all assumptions you have made regarding the hypotheses and the experiments.

- Which parts of the system are vital for your experiment?
- Which size and configuration of the test environment do you assume being sufficient?
- Which system risks do you assume being minor and therefore not tested?
- How do you assume unexpected behavior is treated in the system?
- How much do you think the designers have tested their own code in unit test?
- Which design process and methods do you assume the designers have followed?
- To what extent do you assume the code has been reviewed?

By looking at your assumptions (things you take for granted or things you don't have time to investigate) you may want to reexamine any of the assumptions that are critical for the outcome of your investigation.

Output: A (prioritized) list of general assumptions vital for your investigation.

3.10 Identify variables

Identify or make an educated guess of what factors might affect the subject under investigation.

- Define the factors as variables.
- Identify and group the variables by type and importance.
- Look at your cause and effect diagram. Try to find one or more variables in each branch of possible causes.

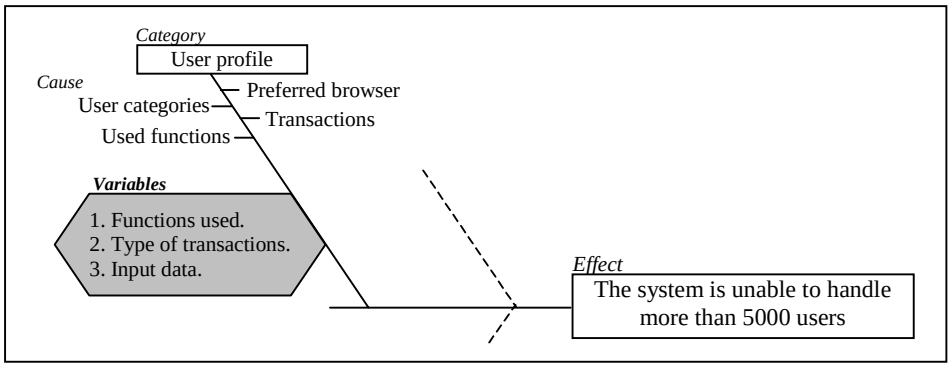


Fig. 4 Example of variables associated to the category" user profile". The user profile is one of the causes related to the stated performance effect (problem).

- Identify which variables are independent and which are dependent.
  Example of independent variables:
  - Number of users.
  - Types of transactions.

  Example of dependent variables:
  - Transactions per second.
  - The number of records in the transaction.
- Identify how the dependent variables interact in the effect of the outcome of the experiment.
- Specify the type and the value range of all the variables you will use.
- Identify how variables will be controlled (measured).

Output: A (prioritized) list of variables to be changed in the experiment.
A list of measurement (output) variables.

3.11 Predict the results of the hypothesis

The prediction is a formal way to put a hypothesis to a test in advance of the experiment. The purpose is to specify the expected result when the hypothesis is true. The prediction can be made in three steps:

1. If my hypothesis is true…..
2. Then ____ should happen.
3. When ____ is manipulated.

Example:

1. If the system is unable to handle more than 5000 users,
2. Then the system will be blocked (response time very long or infinite),
3. When trying to send 5 transactions per second from 6000 simultaneous users.

If the prediction holds your hypothesis has a good chance of being valid and testable. If the prediction does not hold you might need to reject or change your hypothesis.

Output: Hypotheses with predicted results.

3.12 Define the prerequisites for when the hypothesis can be considered true

Which are the prerequisites for the hypothesis of being true and possible to test.

- Which other hypotheses must be true to make the current hypothesis under investigation true?
- Which system configuration is needed in the experiment?
- Which functionality must be delivered and installed?
- Which corrections, patches or service packs must be loaded?
- What preconditions must be fulfilled (system state) to be able to perform the experiment/test?
- Which user/usage profile should be used?

Output:	Prerequisites for the hypotheses of being valid and testable.

3.13 Design experiments to test your hypothesis

Design an experiment to test each hypothesis. Make a step-by-step list of what you will do to answer each question. Prepare test cases that include all the steps to show if the hypothesis was correct or not.

The test case is a procedure describing the experiment including preconditions, actions and expected results. For an experiment to give answers you can trust, it must have a "control" measurement (test case). The "control" is performed as a separate experiment/test case done exactly as the other experiments/test cases. The only difference is that no variables are changed.

Your "control group" (test case) may consist of a proven and measured cause and effect relationship.

Example:

A reference load performance measurement

Number of users:…….	5000		
Number of transactions:……..	20 000		
System configuration:…….	Client	=	Internet explorer.
	Client op. system	=	Windows.
	Web server	=	Apache.
Number of transactions per second (average):…….4			
Number of transactions per second (peak):……..7			
Response time (average):……..	0,93 seconds		
Response time (maximum):…….	5,63 seconds		

Experiments are often done many times to guarantee that what you observe is reproducible. Guidelines for experimental procedures (test cases):

- Select test data representative for the parameters/variables affecting the subject under investigation.
- Only change one variable in each experiment/test case.
- The test case must tell how you will change the variable.
- The procedure must explain how you will measure the change and compare the actual result with the expected result.
- Each experiment/test case should refer to the control group or measurement for comparison to see what the change actually did.
- Prepare both positive and negative test cases for each parameter/variable considered critical for the outcome of the experiment. I.e. selecting both valid and invalid parameter values.

Output:	Test cases for the experiment.

3.14 Check test cases and test scripts

Check that the test cases are well described and reflect all the questions and hypotheses you want answers to. Check that the test cases are:

- Applicable for the selected system configuration.
- Understandable.
- Complete.
- Correct.
- Consistent.
- Unambiguous.
- Possible to execute.

Have an independent person reviewing the questions, assumptions, hypotheses, test cases, test data and the test scripts.

Output:	Verified and validated test cases and test scripts for the experiment.

3.15 Obtain materials, equipment and prepare for the experiment

Obtain what is needed to perform the experiment.

- Gather required test equipment (tools, hardware, cables, software).
- Load any test data needed.
- Start the log functions in the system (e.g. error logs, system logs).
- Check that important system parameters are set correctly.
- Prepare and test the test tools.
- Identify the priorities and logical connections between the test cases.
- Identify how to handle faults, and how to use the trouble reporting tool.

Output:	Required material/equipment and preparations performed for the experiment.

3.16 Do the experiments and record the data

Experiments are often done in series. A series of experiments is made up of separate experimental "runs". During each run you make a measurement of how much the variable affected the system under study.

1 Perform any actions to reach the required pre-condition for the current test object.

2 Run the "control group" test cases as a basis/reference for your experiments.

3 Run a selection of independent normal flow test cases with a broad coverage of your hypotheses and the area you are investigating. The purpose is to obtain knowledge of any major issues requiring further investigation early in the test execution.

4 Run the test cases associated with the highest risks.

5 Run any improvised test cases.
Improvise when:
   - An important observation is made or when you raise a question about the (expected/unexpected) behavior of the system.
   - A fault has been discovered and you want to cover surrounding areas to show that there exist no related faults close to the found fault.
   - There are uncertainties about how a function is supposed to work.
   - You have a general suspicion regarding the reliability or stability of the system (or parts of the system).

6 Run scenario based test cases or other more complex test cases.

Navigate your way through the execution to know where you are, where you have been, where you are going and the stops you have made along the way.

- Record all steps taken.
- Save log files with input/output information from the system.
- Carefully record all your observations. Even results you did not expect may be important.
- Write trouble reports for all unexpected events considered being a failure in the system.

Make sure you have an accurate record of how the experiments were conducted and the results you obtained. Replication, the ability of others to do the same experiment in the same way is very important.

Output:	Log files with data and test records from the experiments.

3.17 Draw conclusions from the experiments

Draw conclusions from the test results to find out whether the hypothesis was correct or not.

- Were your predictions confirmed?
- Did any failures occur?
- What is considered being a failure?
- Are the results consistent with earlier results, general theories and related hypotheses?
- If your hypothesis was correct, briefly explain the major factors that prove the hypothesis.
- If your hypothesis was incorrect, try to explain why. What other answers might there be to your original questions and/or hypotheses?

If the result of a test case is considered as unexpected system behavior the following steps can be taken:

1. Define the unexpected event in more detail (e.g. an error message displayed).
2. What happened before the event occurred?
3. What is the current system state/condition?
4. What information is provided by the system indicating the unexpected event?
5. What are the underlying assumptions when you considered the event/response as unexpected (i.e. the system should not behave in this way because…)?

Output:	Conclusions whether the hypotheses were correct or not based on the experiment results.

3.18 Perform further exploration of the hypothesis if needed

If a hypothesis is supported by the results of the test you might want to consider other hypotheses that make the same predictions (what other actions may give the same result?).

Example:

Original hypothesis: The system is unable to handle more than 5000 users.

Additional hypothesis: The system will get blocked when 5 of the most common transactions are sent as input at a speed of 5 transactions per second (from 3000 users).

Prediction:

1. If the system receives 5 of the most common transactions at a speed of 5 transactions per second,
2. Then the system will be blocked (response time very long or infinite),
3. When trying to send 5 transactions per second from 3000 simultaneous users.

You may also want to add more predictions (what else could happen in the system when a specific input is given).

Use the six thinking hats [De Bono-85] when you examine the results of the experiments. Useful hats are:

Red hat Emotions.
- What is your emotional response to the behavior of the system?
- Does the system feel stable or unstable?

Black hat Critical and negative.
- Were your results consistent and reliable?
- What (else) could go wrong?

Green hat Creativity and new ideas.
- What additional test cases might be useful to provide further evidence of the hypotheses being true?
- What other input data could be given?
- What additional questions might be useful to ask?

Output:	Any additional questions, hypotheses, predictions, test cases or test scripts.

3.19 Document your test cases and your observations, results and conclusions of the experiment

Keep track and document what you have done in your experiment with results and conclusions. Gather the following:

- Log-files.
- Result printouts (messages, screen dumps, etc.).
- Error messages.
- Found faults (described in the written trouble reports).
- The list of identified and evaluated risks.
- Test specifications.
- A descriptions of the additional improvised test cases that have been executed (with corresponding log-files of the test results).

The **investigation report** might include the following:

1 The area of investigation, hypotheses including the involved functions and characteristics that were tested.

2 On which system release/version the tests were performed.

3 How and when you performed the tests.

4 Which test environment was used and how it was configured.

5 Which test data were used.

6 Which test tools, methods and techniques were used.

7 Passed/failed test cases with reference to corresponding hypothesis.

8 Obtained test coverage.

9 The knowledge gained during the experiment/test.

10 Your conclusions of the results from the experiments.

11 Analysis of the current risks in the system with respect to the experiment results.

12 Identification of areas that would need further testing:
- Missing important areas of the system.
- Critical areas.
- Not enough tested areas.
- High-risk areas.

13 A list of faults found in the system during the test execution.

14 A description of the problems encountered during the investigation and the experiment.
   - Do you need to change the procedure and repeat the experiment?
   - What would you do different next time?

15 Possible errors or risks in the investigation, i.e. can the results be trusted?

16 Your recommendations on how to proceed. I.e. releasing the system or continue testing/investigating the current area or other important areas of the system.

Output:	An investigation report including the results of the experiments performed.

3.20 Validate and verify the results and the conclusions

Validate and verify your results.

- Did you investigate the right problem or issue?
- Were the hypotheses a valid and logical representation of the stated questions?
- Was the level of control of the environment and the variables sufficient to provide a reliable result?
- Were the results reliable?
- Did you select the right (valid set of) variables?
- Did the selected test cases represent a sufficient coverage of the questions, hypotheses and predictions made for the experiment?
- Were there any misinterpretations of the test results?
- Were there any unexpected events during the experiment?
- Were there any test cases not possible to execute?
- Were there any found faults impossible to reproduce?

Output:	Validated and verified experiment results.

3.21 State new or additional questions based on the experiment results and your conclusions

Identify areas where you want to obtain more knowledge based on the current experiment results.

You may want to add questions based on your previous conclusions, or to perform further investigations of the current hypotheses and predictions. Remember to focus on the purpose of your investigation when asking additional questions. Otherwise you might end up with an unstructured search where you eventually will run out of time before any sensible conclusions can be made.

You may also want to initiate investigations in new areas.

If you, for example, experience problems when testing the system performance (e.g. load capacity), you may also have seen problems with the stability of the system (under high load conditions). A new area to conduct investigations would then be “system stability and reliability”.

A number of new questions may be raised resulting in new hypotheses and experiments.

Output:	Additional questions, hypotheses or areas to investigate.

3.22 Adjust previous assumptions and hypotheses

You might have discovered that your initial assumptions, questions and/or hypotheses were wrong. As a result of the outcome from the tests you performed, your conclusions and the validation and verification of the test results, you may need to adjust the assumptions and hypotheses.

This is a normal procedure because you made guesses or assumptions in the beginning of the investigation, which you did not know the validity of (otherwise you wouldn't need to do the investigation in the first place).

If you still don't have a clue of what to look for try to be systematic in the changes of hypotheses. If you have a clue it is easier to make the adjustments.

Example:	
Previous hypothesis:	The system is unable to handle more than 5000 users.
Adjusted hypothesis:	The system is unstable when handling more than 5000 users.

You may also want to design more experiments to gain stronger evidence of your hypothesis being true or not.

Output:	Adjusted, changed or new hypotheses to be investigated.

3.23 Repeat the preceding steps until no further questions remain

Repeat some or all preceding steps to gain more knowledge.

Scientific research is a highly iterative process where experiments have to be repeated a number of times to finally reach a reliable conclusion that is well supported.

Remember to keep the structure of your investigation with a clear focus and objective.

Output:	Preceding steps to be repeated.

3.24 Identify general theories based on the experiments performed

If there is strong evidence for a hypothesis of being true based on the results from several different tests (and re-tests), the hypothesis becomes a theory. A theory might be that the developers make the most mistakes when checking boundary values of used data. The theories could be listed in a document and used as a checklist for inspections, test case selection and for exploratory testing.

Example of a general theory:
Interface problems will occur if the public system interfaces are neither coordinated, put under configuration control nor inspected by senior designers for approval.

With a set of general theories testers are able to predict problems at an early stage of development. If the project decides not to do any activities preventing known problems, you must ask and plan for more testing time. This is, of course, neither a cost nor a time effective solution, but might be the only one possible under current circumstances.

Use your theories as a **guide**, and the experimental results as the basis for what you **decide**.

Output:	General theories about the system.

4 How to apply the scientific method

As shown in the description of the steps in the scientific method it can be used throughout the whole testing cycle of a project.

It can be used to:

- Define testing goals.
- Select testing strategies.
- Identify and select areas to test.
- Select test cases.
- Execute test cases.
- Find faults in the system.
- Draw conclusions of the test results.
- Determine if the system is ready for release.

The description of the method in this paper is a rigorous approach. The purpose is to show the whole spectrum of possibilities in reasoning and conducting a controlled experiment.

When you start using the method you may not have to document as much as imposed by the description. The most important thing is how you can apply a more structured way of thinking to reach more reliable test results within a minimum of effort and time.

Use the steps as a checklist when you plan and perform your testing.

You can also start with a shorter, simplified version of the method describing the basic steps and principles:

1. **State the question.**
   What happens if..? Is the system stable when..?
2. **Form a hypothesis.**
   A proposed explanation, or assumption, of an observed or predicted condition, behavior or characteristic of the system. Example: non-alphabetical characters given as input may cause the system to crash because the input data is not properly checked.
3. **Test the hypothesis.**
   Design test cases that show if the hypothesis was correct or not. Predict the result of the test based on what would be considered a correct behavior of the system (for example a received error message). Run the test.
4. **Draw conclusions from the test results.**
   Was the hypothesis correct? Does it need to be changed based on the new knowledge obtained from the test? Analyze any unexpected events or behavior and write trouble reports if the unexpected event or behavior was considered to be a fault. Did the test results reveal any new issues that need to be investigated?
5. **State a new question based on the results from the previous.**

5 Conclusions

A scientific method provides a framework for:

- Solving problems.
- Investigating different subject areas.
- Stating questions and hypotheses.
- Testing the hypotheses.
- Evaluating experimental results.
- Drawing conclusions and gaining new knowledge.

The purpose of a scientific method is not to prescribe a routine, which cannot be varied, but rather a systematic way of asking questions, proposing hypotheses and do testing to determine the correct answers. Scientific methods can be used as a tool to complement your current test process. It is especially useful when lacking information or knowledge within an important area of the system.

6 References

[Ammerman -98]	The root cause analysis handbook	ISBN 0-527-76326-8
[Beizer-90]	Software Testing Techniques, second edition	ISBN 0-442-20672-0
[Claesson1 -99]	Test case selection for functional testing (Eurostar -99)	http://www.testingconferences.com
[Claesson2 -01]	ETP - Enea Test Process	www.enea.se
[Claesson3 -01]	A risk based testing process (QWE –01)	www.soft.com
[Connell -96]	Rapid development.	ISBN 1-55615-900-5
[De Bono-85]	Six thinking hats	ISBN 0-316-17791-1
[Edmund -00]	The scientific method today	http://www.scientificmethod.com
[Focus -01]	Focus groups	http://universe.indiana.edu/clp/rf/focusg.htm
[Gano -99]	Apollo Root Cause Analysis	ISBN 1-883677-01-7
[Jones -95]	The thinkers toolkit	ISBN 0-8129-2601-3
[Lutz -96]	Experience report: Contribution of SFMEA to requirements analysis	http://citeseer.nj.nec.com/69117.html
[McDermott -96]	The basics of FMEA	ISBN 0-527-76320-9
[Mears -95]	Quality improvement tools and techniques	ISBN 0-07-041219-7
[Norman -88]	The design of everyday things	ISBN 0-385-26774-6
[Quality -01]	The Guide to Managing for Quality	http://erc.msh.org/quality/map.cfm
[Rational -00]	Rational Unified Process 5.1	http://www.rational.com
[Splaine -01]	The web testing handbook	ISBN 0-9704363-0-0
[Stapelton -97]	DSDM: Dynamic Systems Development Method	ISBN 0-201178893
[Telles -01]	The science of debugging	ISBN 1-57610-917-8
[Thinking -98]	Tools for the Age of Knowledge	http://www.virtualsalt.com
[Wallace -95]	A Study on Hazard Analysis in High Integrity Software Standards and Guidelines	http://hissa.ncsl.nist.gov/

QWE2002 Paper 11T

Mr. Jim Bampos
(Lionbridge, USA) & Mr. Eric Patel
(Nokia, USA)

Virtual Test Management: Rapid Testing Over Multiple Time Zone

Key Points

- The challenges of managing multiple test sites.
- 5 steps to successful virtual test management.
- Mistakes to avoid/lessons learned.

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

Jim Bampos is responsible for the worldwide testing efforts of Lionbridge and VeriTest. He has 16 years of experience in software development, quality assurance, and management.

Prior to joining Lionbridge, Jim was Vice-President of Software Quality Assurance at Vanteon and was responsible for all Quality Assurance (QA) and software testing activities. He managed the QA Outsourcing line of business, in addition to managing QA and testing activities in support of the company's development business. Jim grew Vanteon's QA Outsourcing by a factor of ten in three years. Before joining Vanteon, Jim was part of Turning Point Software (now the Vanteon Boston Development Center) and implemented best-of-breed SQA/Test methodologies, standards, and practices that made the organization a quality leader in the QA & Testing industries.

Jim is involved with the Los Altos Workshop for Software Testing (LAWST) and the Software Testing Managers' Roundtable (STMR), both periodic gatherings of senior practitioners in the QA industry. He is published in Software Testing and Quality Engineering magazine and speaks at several QA/Test conferences. He earned a BS in Computer Science from Fitchburg State College.

Eric Patel is QA & Test Manager at Nokia where he leads a team that tests digital platforms and communications solutions for the home environment. He is co-founder of the Nokia Quality Forum (NQF) Boston and creator of the RapidSQA methodology for Web testing. His experience in software testing, test management, and software quality assurance spans 10 years.

Prior to joining Nokia, Eric built the QA infrastructure for WebCT. As Director of Quality Assurance, he created testing organizations in Peabody, MA and Vancouver, BC for their e-learning software product and www.webct.com. He established test labs on two coasts with both manual and automated testing capabilities. Before joining WebCT, Eric built the testing organization for Cerulean Technology. He managed the QA effort for a client/server wireless application and its transition from a DOS to a Windows product.

In addition to STAR East, NESQAF, SWaNH SWEQSIG, PSQT/PSTT East, BOSCON and the Software Quality Group of New England, Eric will be presenting at Quality Week and Testing Computer Software this year. He is a member of ASQ, IIST, and NESQAF. ASQ has certified Eric as a Quality Manager (CQM) and Software Quality Engineer (CSQE), and he teaches the CSQE Test Prep Class for the ASQ Boston Section. Eric is also a reviewer for Software Quality Professional and The Journal of Software Testing Professionals. He holds a Bachelor of Science degree in Electrical Engineering from the University of Vermont.

Quality Week Europe 2002

March 11 - 15, 2002

Virtual Test Management

Rapid Testing Over Multiple Time Zones

Jim Bampos
Senior VP, Worldwide Testing Services
VeriTest
jim_bampos@lionbridge.com
www.veritest.com

Eric Patel, CQM, CSQE
QA & Test Manager
Nokia
eric.patel@nokia.com
www.nokia.com

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

Agenda

- Introduction
- Test Manager
- 7 Habits of Highly Effective Test Managers
- A Global Perspective
- Complexities of Scale
- Virtual Test Manager
- Example Virtual Test Organization
- Challenges of Managing Multiple Test Sites
- Lessons Learned
- Rapid Testing Over Multiple Time Zones
- Best Practices
- 5 Steps to Successful Virtual Test Management
- Conclusion
- Recommended Reading

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

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

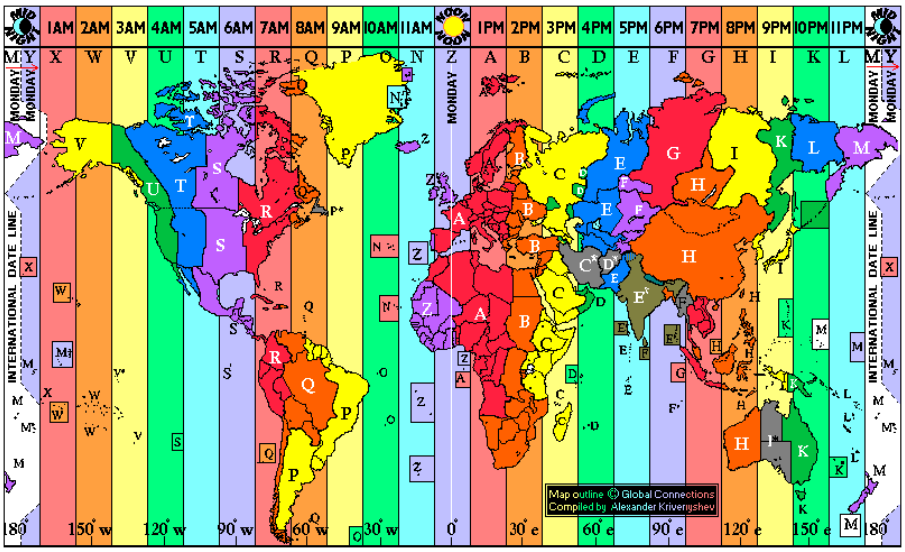
Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

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

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

A Global Perspective



Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

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

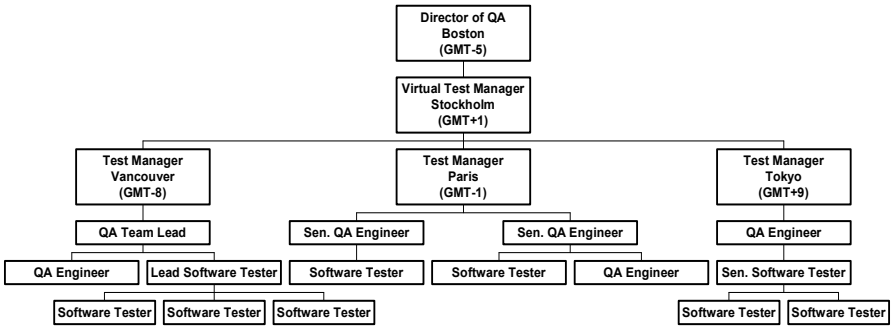
Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

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

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

Example Virtual Test Organization



Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

Challenges of Managing Multiple Test 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

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

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

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

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

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

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

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

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

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

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

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

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

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

Conclusion

- Virtual Test Manager =

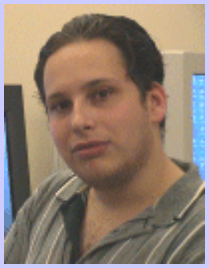
  world traveler + test professional + technical manager
- Virtual Test Manager is the "glue" that keeps all the "pieces" (the test sites) together
- No matter where you are, you will always be managing virtually
- Manage to the organization with the complexities of scale
- Communication, travel, and "face time" are essential

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.

Recommended Reading

- Bach, Jonathan. (2000). Session-Based Test Management. Software Testing & Quality Engineering, November/December 2000, 32-39.
- 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.

Virtual Test Management, Copyright © 2000-2001, Jim Bampos & Eric Patel. All rights reserved.



QWE2002 Paper 12T

Mr. Ibrahim K. El-Far, Ms. Florence E. Mottay &
Mr. Herbert H. Thompson
(Florida Institute of Technology)

Experience in Testing Pocket PC Applications

Key Points

- Learn key concepts in model-based testing.
- Apply 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 experiences.

About the Author

Ibrahim K. El-Far is a doctoral student in computer science under James A. Whittaker at the Florida Institute of Technology, Melbourne, Florida. His interests are in investigating software models, test automation and tools, adequacy criteria, and software testing education. In 2000, El-Far received an IBM CAS Fellowship supporting his research in software testing.

Florence E. Mottay is a graduate student in software engineering and 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 was awarded 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 a doctoral student in mathematics at the Florida Institute of Technology. In the past, he has worked for Microsoft Corporation as a test engineer. His research interests are in software engineering, security, and applying mathematics to computer science problems.

Experiences in Testing the Pocket PC

Software Engineering@ Florida Tech

© 2002 The authors, all rights reserved.

Overview

- Scope of the Project
- Introduction to Model Based testing
- Case Study: the Pocket Inbox
- Conclusions

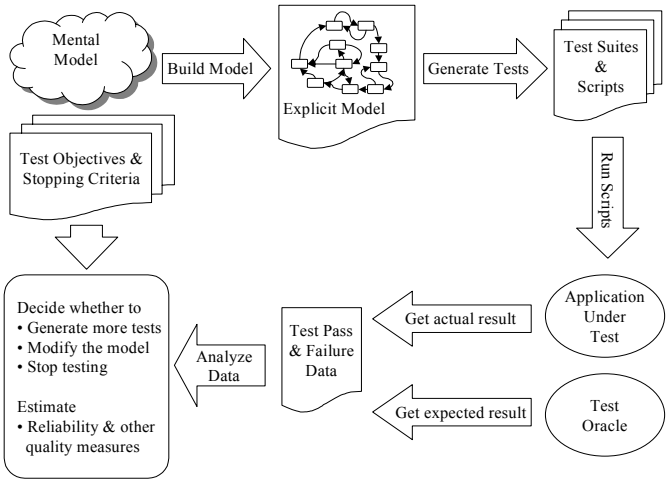
© 2002 The authors, all rights reserved.

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



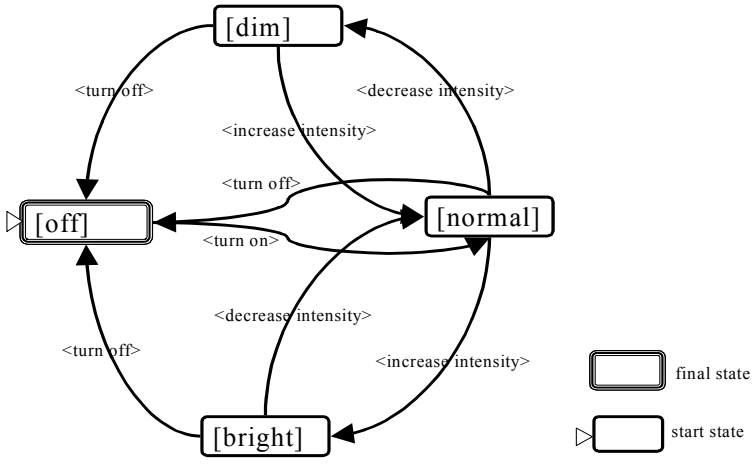
© 2002 The authors, all rights reserved.

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.

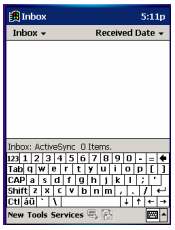
Case Study: the Pocket Inbox

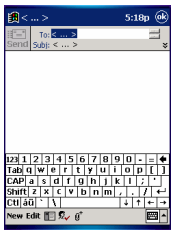
- General Information
  - 5,000 transitions
  - 1,500 states
  - Most complex application we tested
  - Design choices
    - Limited number of messages
    - Input peculiarities

© 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

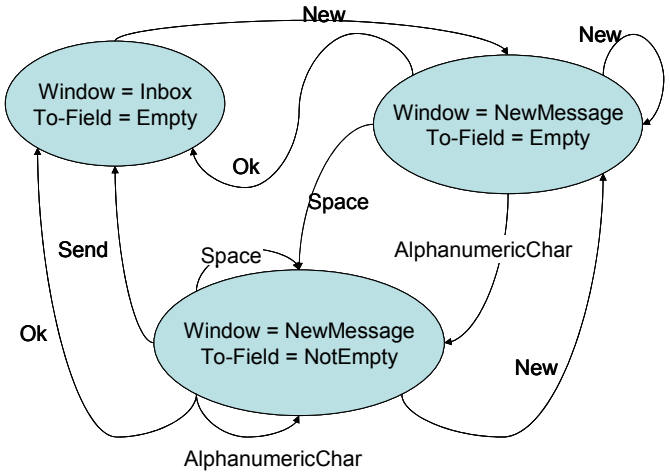
© 2002 The authors, all rights reserved.

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.*

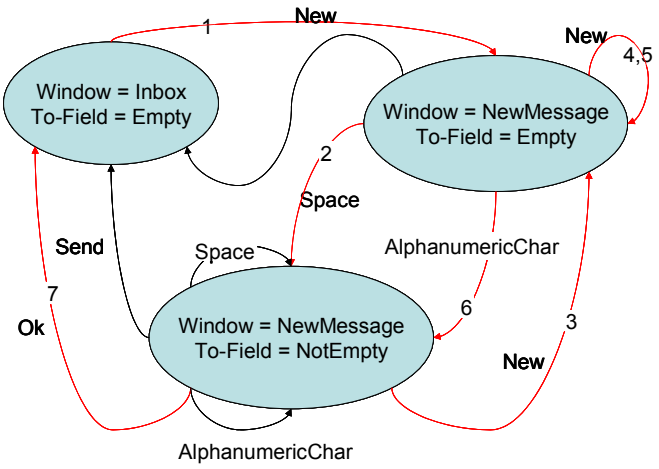
© 2002 The authors, all rights reserved.

Case Study: Transitions



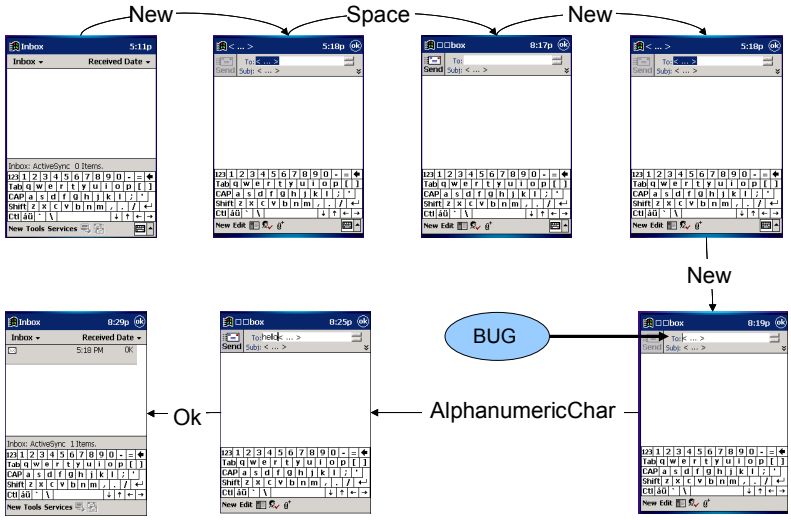
© 2002 The authors, all rights reserved.

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

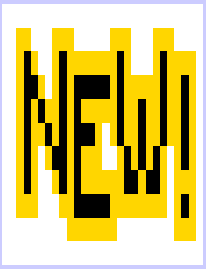
- As long as finite state machines are used, there are inescapable critical issues to be dealt with: model building and maintenance, state explosion, and model correctness. There is need for more practical pointers on how to work around, or at least reduce the impact of these factors.
- Finally, studies 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
  - Regular Old-Fashioned Mail :

    Herbert Thompson
    Department of Computer Sciences
    150 W University Boulevard
    Melbourne, Florida 32901

© 2002 The authors, all rights reserved.



QWE2002 Paper 1A

Mr. Jon Maurolagoitia & Aitor Elorriaga
(SQS S.A., Spain)

QA Success Story for Embedded Systems in Real Time Control Systems

Key Points

- Embedded Systems testing.
- QA in mature and traditional companies.
- Quality Department creation.

Presentation Abstract

This project is a success story, which managed to introduce a validation methodology in a mature and traditional industrial hardware manufacturer company, setting up a quality assurance department within that company organization. This department kept independent from the development team and was leaded by a quality mentor.

About the Author

Jon Maurolagoitia

```
Studies: Computer Engineer, University of Deusto, Spain

Work Experience:

    -Working for SQS S.A., Spain as responsible for:
        -Test Management in Embedded Software Systems
        -R&D Project evaluation

    -Working for Phillips Speech Processing, Austria as responsible for:
        -GUI Testing
        -Language testing

    -Other work experience
        -Web Site Designer and Programmer
```

Aitor Elorriaga

```
Studies: Computer Engineer, University of Deusto, Spain

Work Experience:

    -Working for SQS S.A., Spain as responsible for:
        -Test Management in Embedded Software Systems
        -R&D Project evaluation
    -Working at CARSA, writting, coordinating and participating in several European and national R&D projects
    -Working at NorSISTEMAS as Systems' Analyst in a Human Resources Management System and a Project Managemen
```

QA Success Story for Embedded Software in Real Time Control Systems

Jon Maurolagoitia

Introduction

The Presentation Overview

- Initial Approach
- Objectives of the Project
- How we started
- The Project
  - Documentation and Revision Process
  - Software Validation
  - Software Verification
  - Tools
- Results Achieved

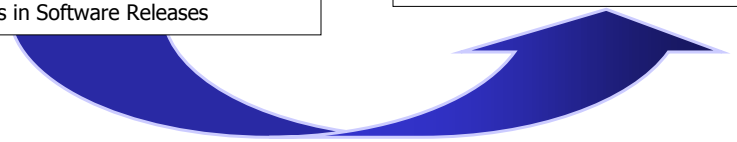
The Project

The Old Process

- Poor documentation
- No Validation task
- No Verification task
- Testing performed by Quality Dept.
- Late Software Releases
- Bugs in Software Releases

Optimized Process

- Documented Development Process
- Include Validation and Verification tasks in Development
- Improved time estimations



QA

Initial Approach

The Company Profile

- Large Hardware manufacturer
- Leading manufacturer of Real Time Control Systems
- Big growth within the last years
- Development of Embedded Software for this RTCS
- Long development processes involved in each project
- No software quality department
- **Small company development methodology**

Initial Approach

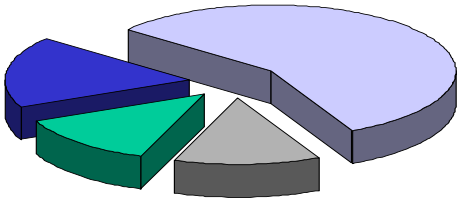
The People at the Company

- Experienced embedded software programmers
- Growing development department
- Very Involved with the company
  + Big motivation for improvement
  - Nervous of big or risky changes
- Little quality knowledge
- Motivation for introducing quality into the process

Initial Approach

In Europe

- This is not an isolated example



- Many of these companies have similar quality weaknesses
- Don't know how to start introducing QA into their organizations

How we started

Previous Step

Find the right people within the organization

- Good knowledge of the department
- People able to guide the team
- Able to motivate the rest of the department
- Able to pinpoint the best of different techniques

First Phase: Technology Audit

The Audit

- Main objective:

  Encourage motivation, communication and knowledge management in the team

- Several previous Audits:
  - Didn't take into account the motivation
  - Were not aware of the special case of the company
  - Proposed methodologies were hardly applicable
- Identify the processes inside the Dev. dept.
- Create confidence inside the department

First Phase: Technology Audit

The Result

- QA Implementation Plan
- Find the person to head the Soft. Q. Dept.
  - Experienced development figurehead
  - Convinced of the benefits of QA within his company
  - Can share his beliefs with his team
- Gradual Introduction of the Methodology
- SQS helps SoftQDept to introduce QA in development
  - Introduction of the suggested methodology
  - Adaption of the methodology to their processes

A methodology that doesn't increase the budget

The Kick Off

The Project

- Choose a project to introduce QA - two options:
  - Long Project: enables analysis of all posibilities
  - ⇨ Short Project: more adequate for introducing changes
- A Short Project allows the people to have time to think about the new methodology and how it changes their way of working
- The chosen project:
  - A small new application for a Big Product
  - The Big Product was developed years ago
  - Time constraints with the Release

The Project

The documentation

- Big problem in the department: poor documentation
- Existing documentation not updated
- Proper documentation would improve development process
  - Interdepartamental communication
  - Compose manuals of the systems, ...
- Conclusion: generate only necessary documentation
- Documents to generate:
  - Software Requirements Specification
  - Software Design
  - Document the Code

The Project

Revision of the documentation

- Get the departments involved in the project together
- Each revision will require input from different participants
- Discuss problematic points before starting development
- Only revise the main documents
- Every revision has a moderator and concrete objectives
- The moderator is responsible for achieving the objectives

The Project

The Software Validation

- Two tasks involved:
  - Revision of documentation
  - Requirements Testing
- Requirements Testing
  - Base the Tests on the Requirements
  - Obtain the desired Requirements Coverage
  - Test Management
    - Test Classes
      - Test Sub Classes
        - Test Cases
          - Test Data
  - Study the possibility of Automated Testing

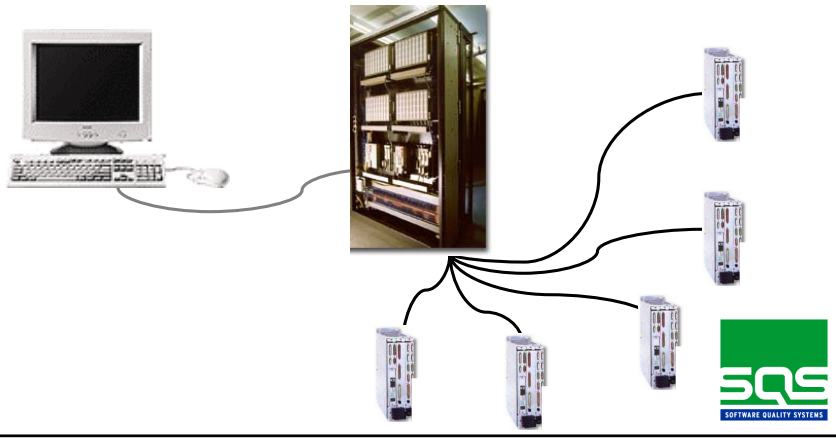
The Project

Automated Software Validation

- Introduce the concept of Automated testing
- Study the possibility of automating the tests
- Developers to be instilled with the idea of automated tests
  - System connection capabilities
  - Information to be captured from outside the system
- Start automating tests for critical Requirements
  - Requirements that need to be tested exhaustively
  - Tests that will be executed several times

The Project

The Software Automated Validation



The Project

The Software Verification

- Tasks included:
  - Code inspection
  - Unit Testing
  - Module Testing
- Code Inspection
  - Documentation/Comments included in the Code
  - Style guide inspection
  - Complexity of Code
  - The Code Inspection can be performed within the development dept

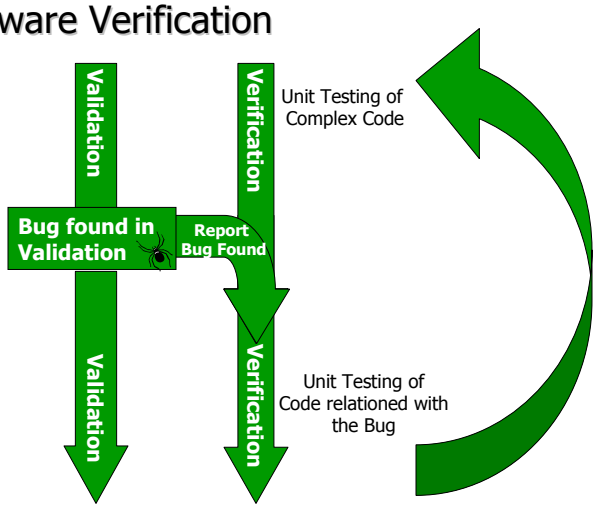
The Project

The Software Verification

- Unit and Module Testing
  - Poor documentation of previous developments
  - Only new developments will be documented
  - Therefore, it is only possible to test new developments
  - Only test the code when complexity exceeds a certain level
  - This level depends on the coverage to be achieved
  - When a bug is found in Validation, functions and modules affected by the bug will be tested

The Project

The Software Verification



The Project

The Tools

- Test Management and Execution
  - SQS TestWORKFLOW
- Code Metrix for Verification
- Defect Tracking

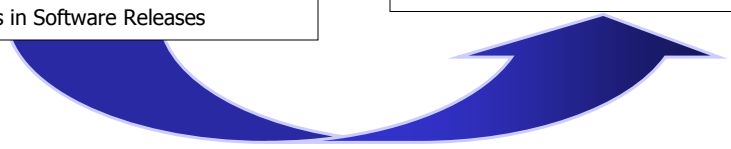
The Result

The Old Process

- Poor documentation
- No Validation task
- No Verification task
- Testing performed by Quality Dept.
- Late Software Releases
- Bugs in Software Releases

Optimized Process

- Documented Development Process
- Include Validation and Verification tasks in Development
- Improved time estimations



QA

QWE2002 Paper 2A

Dr. Rainer Stetter
(ITQ GmbH & Software Factory GmbH)

Steps to Bring the V- Model Into Real Life: A Case Study

Key Points

- Explanation of different roles in the V-Model.
- Hints to avoid pitfalls in process improvement.
- Testing of Embedded Systems.

Presentation Abstract

The V-Model defines a sound and well-designed engineering process. But real life in industry shows that it is quite hard to bring this model into the daily work flow of organizations. In this presentation I will show which steps have to be taken to install the basic principles of the V-Model. The basis of the presentation is the summary of experiences of a project which already started in 1997 and is still running. The subject of the project is the stepwise development of a complex machine which produces smart cards (id-cards and ec-cards). The software system for this kind of machine includes almost all software technologies which are known today. To bring alphanumerical data into the system you have an interface to an IBM Mainframe. The graphical data (photos and finger prints) are transferred to a windows NT system. The user interface has to handle Arabic and roman letters. All data are stored in an Oracle data base. The communication of the user interface ! with the embedded components runs via TCP/IP. Each of the (mechanical) modules of the machine has its own processor which runs under VxWorks. The low end version of this machine contains about 10 processors which have to be synchronized. The high version consists of about 20 processors.

About the Author

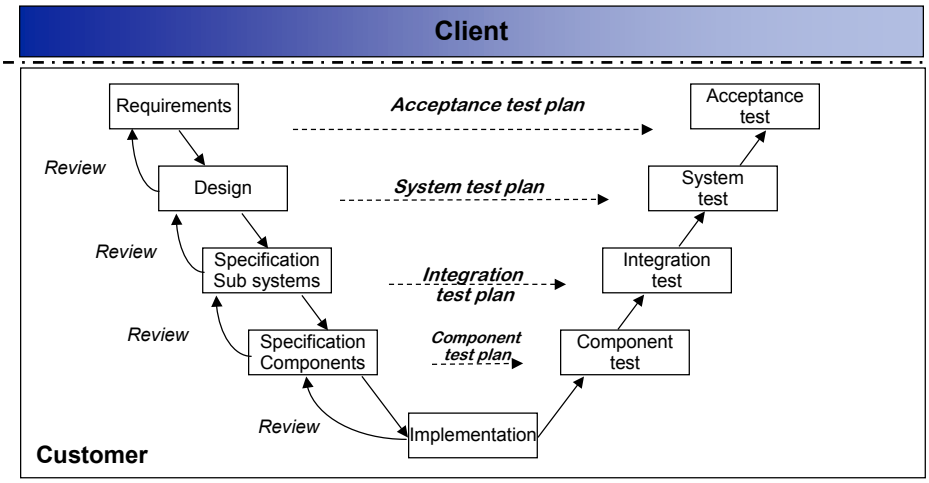
At the Technical University, Munich (Germany) Dr. Rainer Stetter studied mechanical engineering and as he was interested in software engineering he took some classes in computer sciences. While he was doing his PhD in developing a robot simulation system, which he got 1993, he improved his knowledge in software engineering. Form 1993 until 1997 he was working as a Research & Development Manager at Zwick Company, Ulm Germany.

Since 1997 he has been one of the presidents of the Software Factory GmbH, Munich. In addition, since 1998 he has been working with the Munich based firm ITQ GmbH as president. Together, Software Factory GmbH and ITQ GmbH form the Software Quality Group. With some partners of the Technical University of Munich and VDMA (German Machinery and Plant Manufacturers Association), they work on approaches to improve the quality especially in the field of embedded systems. Since 1998 Im Vice President of the VDMA Software department.

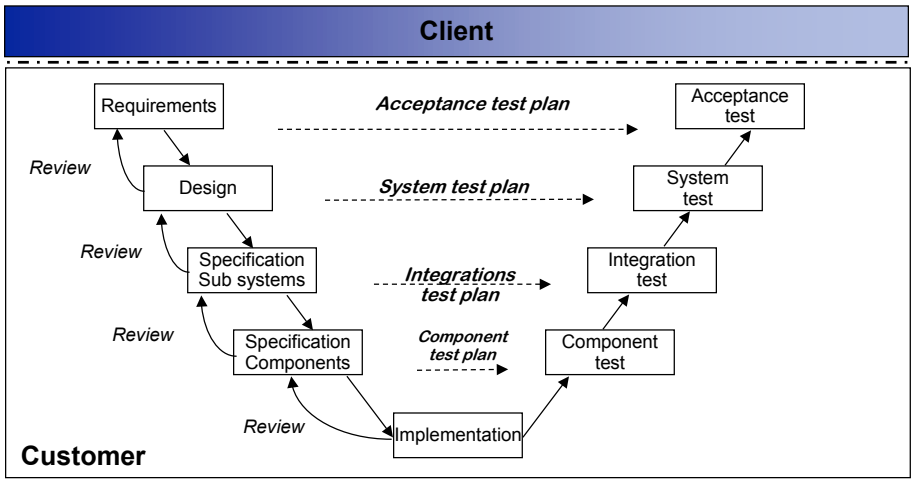
Steps to bring the V-Model into real life: A case study

Dr.-Ing. Rainer Stetter

Theory



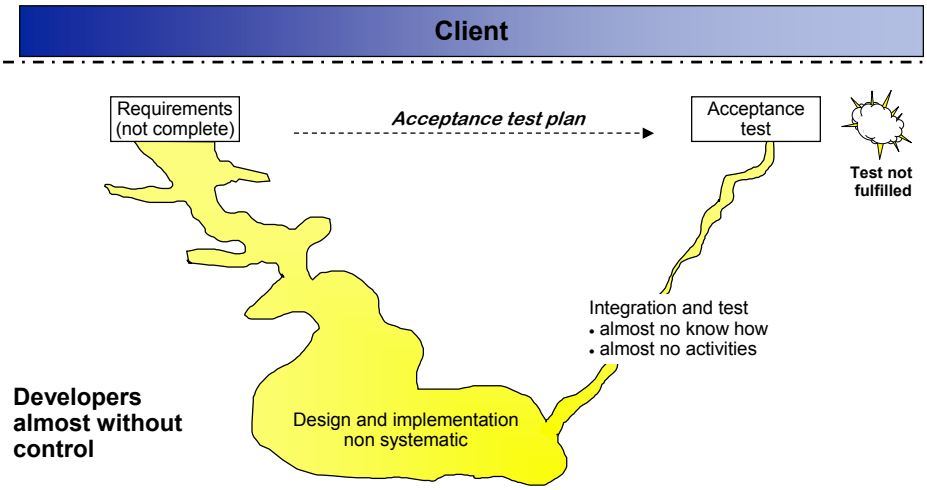
V-Model stepwise



Reality

- Project
  - Total investment about 100 million DM
  - Tough schedule:
    - About one year from idea to prototype
  - Not enough time for early phases
    - Requirements and system environment quite unclear
    - Writen documents not complete
  - Test documents
    - Not existing or very late created
    - Acceptance criteria not complete, not synchronized
  - Development results not satisfying
    - „from compiler to the client"

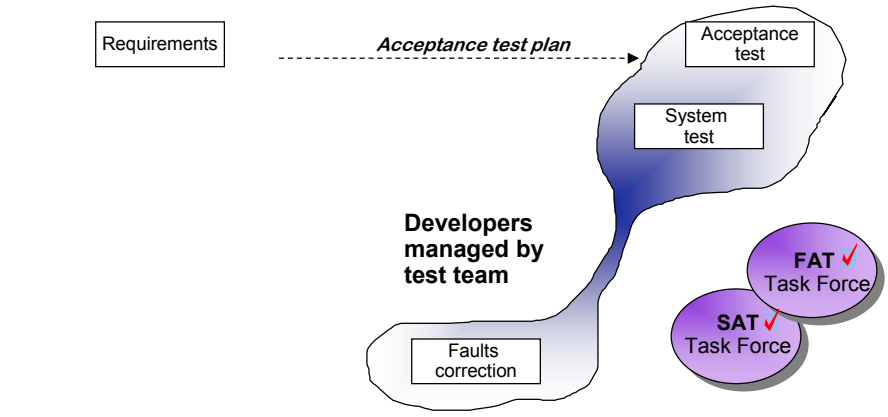
Feb. - May 97: Analysis of the situation



Measures

- Management
  - Coordination of project through test group
  - One group in Germany, another in Arab country
- Administration
  - Introduction of source code control system
  - Compability matrix hardware and software
- Test
  - Test templates
  - Checklists for testing
  - Detailed tracking of the tests
    - Error database
    - Measurement of system reliability

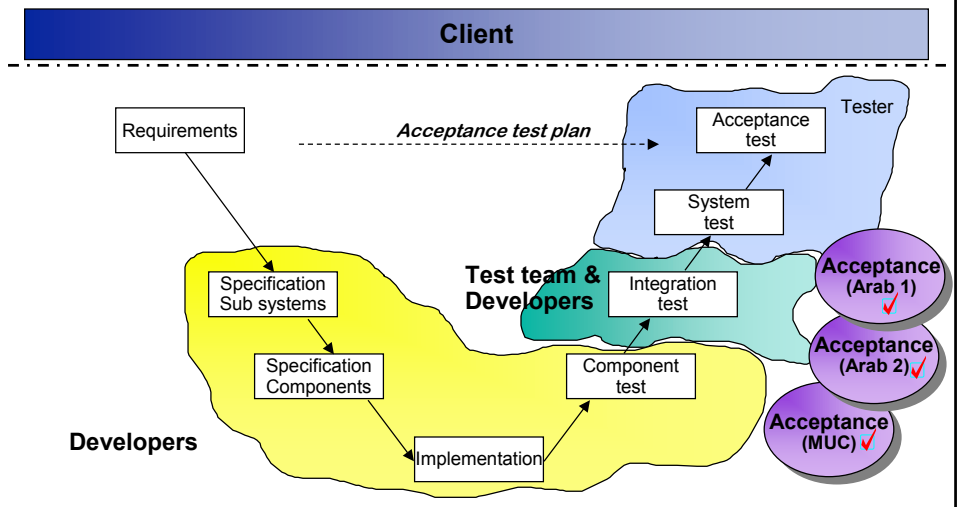
June 97 - May 98: Preparation for acceptance tests



Interim results

- Test criterias fulfilled
  - Management satisfied -> Customer had to pay
  - Necessity for tests was seen now
- Roles
  - Importance of different roles could be made clear
  - Test group was now introduced permanent
- Organization
  - Process improvement as a strategic goal
  - New projects had developers and testers

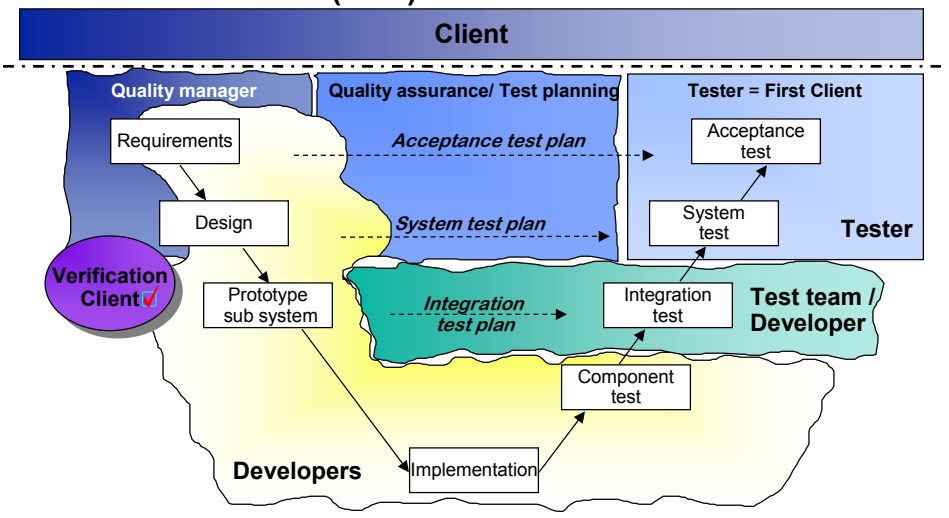
Test preparation V2.0



Measures

- Roles
  - QA-Manager:
    - Coaching project manager
    - Focussing especially on early phases
    - „Design“ of the project management
  - Test manager:
    - Test planning
    - Test design
  - Tester:
    - Testing (first client)
    - Communication with developers

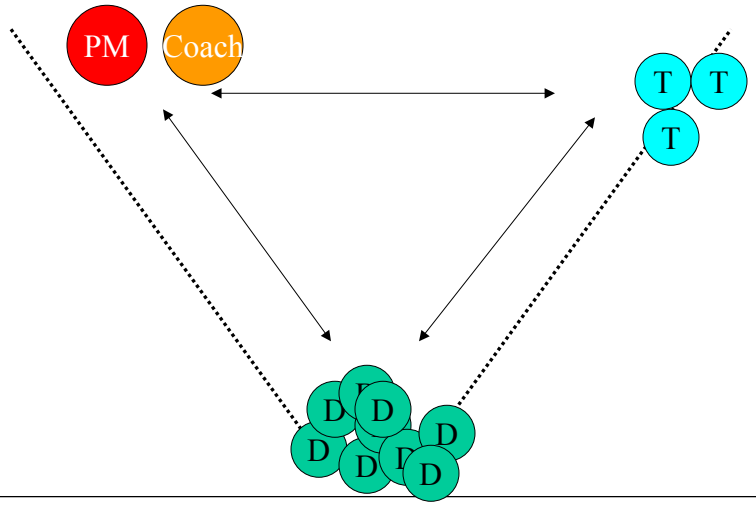
New User Interface (V3.0)



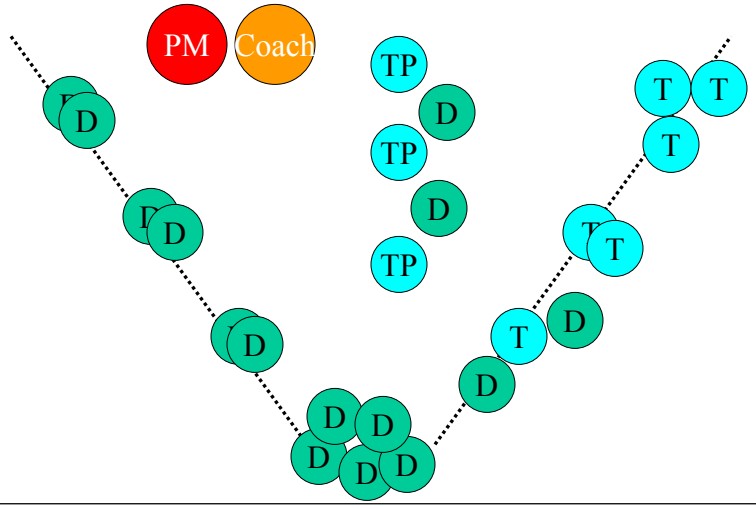
Experiences

- Different roles help to get a sound project
  - Different point of views
  - Clear responsibilities (Development <-> Testing)
- Project management
  - Has to have a clear understanding of quality requirements
  - Normally has to be coached at the beginning
- Role play
  - Has to be introduced stepwise
  - Change of roles (part-time) helps to understand eachother

First phase – Roles introduction



Second step: Roles improvement



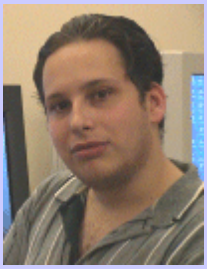
Conclusion

- There must be a strong will or an impact from outside
  - to bring the V-Model into real life
  - Occurance of a crisis is the best starting point

- Role play
  - There must be a coach and
  - mixture of experienced and un-experienced team members

  - New roles can be learned best, if you have an example
  - Role change helps to understand the needs of others

- Social aspects have the same importance as technical issues



QWE2002 Paper 3A

Mr. Ibrahim K. El-Far, Ms. Florence E. Mottay & Mr. Herbert H. Thompson
(Florida Institute of Technology)

Choosing an Appropriate Software Testing Method

Key Points

- Selecting testing methods is a critical factor for successful quality assurance.
- Analyzing software context is key to adopting an appropriate testing style.
- The factors that influence selection are many: context, cost, and organizational condi-t.

Presentation Abstract

Software testing presents us with an abundance of challenging problems. One such problem that has been little investigated until recently is the selection of a testing method that is appropriate to test a specific software application. There are scores of styles to choose from: scenario, explora-tory, model-based, stochastic, control flow, functional, and data flow, just to name a few. Making a suitable choice greatly improves a projects effectiveness and typically contributes to increased efficiency. To make reliable decisions, we need to consider, at least in part, the results of analyz-ing the context of the software under test. This work is an endeavor to bring up the critical issues in selecting a software testing method.

About the Author

Ibrahim K. El-Far is a doctoral student in computer science under James A. Whittaker at the Florida Institute of Technology, Melbourne, Florida. His interests are in investigating software models, test automation and tools, adequacy criteria, and software testing education. In 2000, El-Far received an IBM CAS Fellowship supporting his research in software testing.

Florence E. Mottay is a graduate student in software engineering and 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 was awarded 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 a doctoral student in mathematics at the Florida Institute of Technology. In the past, he has worked for Microsoft Corporation as a test engineer. His research interests are in software engineering, security, and applying mathematics to computer science problems.

Selecting an Appropriate Software Testing Method

Ibrahim K. El-Far
Florida Institute of Technology

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Overview

- Needed research areas
- Test method evaluation issues
- What to evaluate
- Hidden misconceptions in comparisons
- Factors influencing evaluation

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Categories of Quality Activities

- Going into the product
  - Good process
  - Documentation
  - Formal analysis
  - Design reviews
  - Code Inspections
- After the fact
  - Program proof
  - Fault injection
  - Mutation analysis
  - Testing

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Variety of Testing Styles

- Scenario
- Exploratory
- model-based
- Stochastic
- Control flow
- Functional
- Data flow
- Clear box
- Partition
- Black box
- Random
- Boundary Value
- Risk-based
- Requirements-based

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Needed Research Areas

- How to evaluate new test method
  - Assess worth, goals, activities
  - Estimate how well it integrates into existing practices and tools
- How to evaluate existing methods
  - Study thoroughness and cost
  - Determine where improvements can be made
- How to put methods to work in combination

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Test Method Evaluation Issues

- What is it about the testing method that we want to evaluate?
- Does the evaluation involve any entity other than the method itself?
- Is there a precise set of measures that describe what we want to evaluate?
- Are there uncontrollable factors affecting the evaluation?
- What is the scope of the results of the evaluation?

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

What to Evaluate

- Effectiveness – Is the method producing the right results?
- Efficiency – Is the method wasting as little resources as possible?
- Human elements – Is the method dependent on skills, expertise, knowledge, and personal communication?

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Comparing Effectiveness

- Number of defects found
- Delivered reliability
- Detecting defects of a specific class
- Satisfying the adequacy criteria
- Power of finding faults

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Comparing Cost

- Size of test suites
- Speed of producing test suites
- Return of investments
  - Bugs
  - Reliability Information

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Comparing Human Element

- Degree of human action involvement
- Dependency on human skills
- Dependency on domain expertise
- Amount of needed on preparation

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Factors Influencing Evaluation

The Nature of Programs

The Nature of The Method

Circumstances Surrounding Development and Test

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Contact Information

- Ibrahim K. El-Far
- ielfar@acm.org
- http://www.testingresearch.com/
- Department of Computer Sciences, Florida Institute of Technology, 150 W. University Blvd., Melbourne, Florida 32901, USA.

(c) Copyright 2002 Ibrahim K. El-Far. All rights reserved. Presented at Quality Week Europe 2002.

Copyright © 2001 Ibrahim K. El-Far. All rights reserved. This paper to appear in the Proceedings of the Fifth International Internet & Software Quality Week Europe Conference (QWE 2002), March 2002.

Selecting an Appropriate Software Testing Method

Ibrahim K. El-Far
Florida Institute of Technology

Abstract

There is an abundance of software testing styles and methods today. Some have been practiced in the industry for a long time. Others have seen limited application despite their intuitive appeal. There is a real need in our industry to study all these methods in order to decide whether to incorporate a new method into existing tools and practices and evaluate the thoroughness and efficiency of adopted methods. The same studies would also facilitate making an informed decision on combining various methods to test a particular application. Evaluating a testing method is not straightforward. There are many lines along which to make the evaluation, but we do not know enough, especially as to what aspects of the method we need to account for. Evaluation is often performed with little or no regard to the software for which the method is intended and the circumstances under which the software is developed and the method is applied.

Keywords

Method evaluation; test method comparison; test method selection; effectiveness; cost; evaluating the human element in testing

1. Introduction

Programs are some of the most complicated artifacts ever made by man. Software solutions often address problems that are very hard to understand or explain, and, sometimes, these problems are impossible to solve in a manner that satisfies every concerned stakeholder. Today, these solutions increasingly affect several aspects of our lives more than ever. Indeed, we are slowly approaching a new world of ubiquitous computing. And the software industry is going to have to tackle more hard problems and software and its craft with tend to become more complex.

Given the importance and difficulty of understanding and building software, it may surprise many industry-outsiders that building software is, to a great extent, a creative process. By that I mean that it is highly dependent on the skills and expertise of humans, who are largely responsible for soliciting, organizing, and analyzing requirements, building and reviewing designs, writing code, and numerous other related tasks. Unfortunately, but not unexpectedly, humans tend to make a lot of mistakes, and they certainly have for as long as there has been software. This has translated into products that are riddled with defects, some of which represent a real threat to productivity, safety, security, and the privacy of their users. So, as one would expect, the industry has been making improvements to the product building experience and to the quality of the product itself.

Nevertheless, despite the advent of software development tools, process and organization, formal methods, and other means of improvement, we still have a long way before we

Copyright © 2001 Ibrahim K. El-Far. All rights reserved. This paper to appear in the Proceedings of the Fifth International Internet & Software Quality Week Europe Conference (QWE 2002), March 2002.

can consistently produce high quality software that meets customer expectations, complies with its specifications, and exhibits little or no failures. Until then, what we should do is scrutinize and improve our current quality assurance efforts. At present, these efforts can be divided into two major categories: activities that contribute to engineering quality into the product and those performed after the fact. Examples of the first category include employing good process, maintaining reasonable documentation, formal design analysis, and conducting reviews such as design walkthroughs and code inspections. Program proof, fault injection, mutation analysis, and testing fall in the second category. Of all the disparate approaches to assessing and improving software products, our industry today is inclined to rely on testing.

1.1 Three Areas of Needed Research

Our industry has been doing testing in various forms since its birth, but the pursuit of serious forms of testing that transcend simple debugging have only started sometime in the early 1970s. Software testing has since presented us with an abundance of challenging problems. In the past decade, sheer practical need has forced pragmatists to address these problems in numerous contexts. The result has been a gradual proliferation of testing styles and techniques, reports about which can be found in academic and trade literature. But this very phenomenon of numerous methods and styles, which is typical of any budding field, has created the need for further research and investigation. Here are three examples.

- ***Evaluating New Methods***. There are many testing methods that have been proposed. Some of these are yet to be introduced to existing tools and practices. Others have seen limited use or have had limited community exposure, despite apparent intuitive appeal in many cases. Many in the industry would like to see if and how various methods would contribute to the betterment of and integrate well into their quality assurance frameworks. For such individuals and organizations, there is a genuine need to make an informed decision on whether to invest some time in experimenting with it, and, eventually, whether to adopt it.

- ***Evaluating Existing Methods***. There are a few testing methods that have been employed by the industry for some time. Some of these are considered – not without dispute – solid best practices. However, just because some methods have stood the test of time, or just because there are a few individuals or organizations who believe in their worth, that does not mean that the method is worthwhile. Here, there is a need to evaluate methods for thoroughness and efficiency in order to determine where improvements of any sort can be made.

- ***Combining Various Methods***. No single testing method is a silver bullet. In particular, no matter how useful a testing method, there will come a point in time at which the returns on investments will severely diminish. To address this issue, it is a common practice to use another method (e.g., scenario testing instead of state-based testing) that has different activities (e.g., design user scenarios vs. build state models), goals (e.g., test for security risks vs. verify major functions), and target areas of the software under test (e.g., individual routines vs. elements of the graphical user inter-

Copyright © 2001 Ibrahim K. El-Far. All rights reserved. This paper to appear in the Proceedings of the Fifth International Internet & Software Quality Week Europe Conference (QWE 2002), March 2002.

face). How do we decide on what combinations of methods to use in a particular situation? Again, there is a need to evaluate methods, particularly to justify the cost of applying more than one.

Consider the scores of testing styles out there: scenario, model-based, security risks, control flow, functional, stochastic, and data flow, just to name a few. Consider also the industry's dependence on testing as the primary approach toward better software products. Then, consider how many depend on software for the survival of their businesses, and, sometimes, for their own survival. The need for the above research is real. Decisions that such research would hope to influence form the basis for, among other things, determining the project timeline, allocating and committing resources, making test plans, creating and/or purchasing tools, and training personnel. Making a suitable choice greatly improves a project's effectiveness and typically contributes to increased efficiency. Implementing a poorly chosen technique, on the other hand, can only bring about awkwardness in execution, delays, and, sometimes, results that fall short of what is sought after.

1.2 What to Analyze

I would imagine that many would contend that experience is the only reliable tool to make judgments on methods. Experience can indeed improve the chances of singling out useful techniques, but can also be a double-edged sword, as biases and preconceived notions can predispose testers and management towards a particular style that may not be the most appropriate. Learning from the experiences of others can have the same positive effect, but many obstacles stand in the way of sharing lessons and best practices among organizations and sometimes among the various groups within a single organization.

In light of this, reliable decisions need to be based, at least in part, on the results of analyzing those objective entities with useful information. Such an analysis should involve all the entities, listed below, together in one context.

- ***The software under test***, including its nature, peculiarities, mission, and users
- ***The testing method***, including its activities, requirements, demands of skills and expertise, the areas of the software it targets, and the risks it addresses.
- ***The circumstances under which the software is developed and tested***, including market conditions, organizational constraints, and specifics of the project. This also includes when the method is applied, the number and quality of personnel assigned to pursue the method's activities, and the resources that are allocated specifically for the method.

The task at hand is challenging, and it requires much analysis and a tremendous amount of data, and needs to be carried out with care. Finally, I should say that, despite the magnitude of the work that is ahead of us as a community, much research has been done in the past. However, reported attempts to leverage any part of relevant work into existing practices have been limited.

1.3 This Work

This is a report of an ongoing investigation on the effectiveness and cost of software testing methods. The investigation is in its early stages. This paper identifies some areas in

Copyright © 2001 Ibrahim K. El-Far. All rights reserved. This paper to appear in the Proceedings of the Fifth International Internet & Software Quality Week Europe Conference (QWE 2002), March 2002.

software testing of which we as a community have a weak understanding. It touches on how to compare and evaluate test methods. Categories of comparison methods are outlined, and the format of some of comparison results is explained and criticized.

2. Method Comparison and Evaluation

Permit me to recapitulate. We have three distinct needs: (1) to evaluate new methods before adoption, (2) to evaluate existing methods for thoroughness and efficiency, and (3) to study the manner in which methods can be combined to achieve maximal returns on investment. The problems are indeed great, and they deserve individual treatment in their own right. As a first step toward developing answers to those problems, we need to be able to, among other thing:

- ***Evaluate a method on its own***, that is, in isolation of any other method
- ***Evaluate a method relative to another method***. For example, how does random finite-state model based testing using state variables with an objective of covering all states compare to scenario testing of all major features where each scenario involves at least three features?

Evaluation is especially ambiguous in the context of software testing methods. There are no standard lines along which methods are to be compared, although many have been proposed (I mention three later in this section). Even when those lines are known, we rarely have precise interpretations of them. For example, what are the qualities of an effective method, if we are evaluating methods based on effectiveness? Such an absence of precise interpretations translates to a lack of accurate, meaningful measures that are useful in the context of those interpretations. Interestingly enough, even when there are precise aspects of a method that are feasible to measure, there always are uncontrollable factors (especially organizational, project, and personnel related) that cast a significant doubt on whatever little we can measure.

In short, before we begin to evaluate methods, we need to be able to answer the following questions to some degree of satisfaction:

- What is it about the testing method that we want to evaluate?
- Does the evaluation of the method involve any entity other than the method itself? If so, how meaningful are the results?
- Is there a precise set of measures that describe what we want to evaluate? How can these measures be collected?
- Are there factors that affect the evaluation, but that are so subjective in nature that they are very hard to measure? Can we (and if so how) account for or control these factors?
- What is the scope of the results of this evaluation? Do they apply only to the specific instances that we used to evaluate, or does it apply in a wide variety of situations?

Although this work does not hope to answer all of these questions, only to offer some insight to what may become a part of an answer.

Copyright © 2001 Ibrahim K. El-Far. All rights reserved. This paper to appear in the Proceedings of the Fifth International Internet & Software Quality Week Europe Conference (QWE 2002), March 2002.

3. What to Evaluate in a Test Method

3.1 Effectiveness

A significant portion of the work on test method evaluation and comparison is based on how effective a method is, but what does that mean? The Merriam Webster online dictionary defines 'effective' as 'producing a decided, decisive, or desired effect.' Obviously, the meaning of effective software testing is not very clear, because it all depends on what is the 'desired effect'. For example, one of the debates that divide the testing community is whether testing is ultimately about uncovering defects or about investigating and estimating the product's reliability. Testing methods that are regarded as effective for reliability estimation may not be as effective in finding bugs; random testing methods are probably good examples of this.

In the end, an evaluation based on effectiveness is only good when we (1) know exactly what we want from the method and (2) have the means to collect enough information to judge whether the method delivers what we want. Here are a few examples of what we may want from a method:

- To detect as many defects as possible
- To gain confidence in the quality of the software. In this case, we have to talk about what aspects of quality are of interest when it comes to method evaluation: failure, performance, compliance with specifications, living up to customer expectations, etc… We also have to precisely define confidence: is it the probability that a given quality is satisfactorily exhibited over a period of time? What is satisfactory? And what is time?
- To gain insight into the security risks that the software presents. This is an example of a perfectly good reason to test an application. However, in order to use this as a basis for evaluation, we need to be more precise in stating what we want. By gaining insight, for example, we are collecting specific information about specific parts of the software that carry high security risks, and we may also be trying to uncover all security related defects. In addition, we need to detail what we mean by a security risks, if only by listing these risks so as to provide us with a rudimentary form of measure.

3.2 Efficiency

Effectiveness and efficiency, taken together, dominate the method evaluation and comparison research scenes. To be efficient, according to the online Merriam Webster dictionary, is to be 'productive of desired effects; especially: productive without waste.' So, a testing method that is efficient must also be effective to some degree, whilst the opposite does not always hold. Again, there is an ambiguity in the term 'waste.' When a method is deemed to be efficient, it is not wasting any resources, and there are many of those: time, machine resources, unnecessary activities, useless artifacts, etc… For example, a method is deemed to be efficient in uncovering defects if it is effective in finding them and if it does not waste time while finding them. But what does 'waste time' really mean? Does it mean that the method does what it is supposed to do in optimal time? Or does it mean that it operates within a reasonable distance from the worst possible time?

Copyright © 2001 Ibrahim K. El-Far. All rights reserved. This paper to appear in the Proceedings of the Fifth International Internet & Software Quality Week Europe Conference (QWE 2002), March 2002.

How would one find out what is the worst or optimal time for a certain objective to be effectively met. This can mean a lot of lengthy, error-prone analysis for some of the simplest objectives.

Furthermore, there is the question of the relationship between efficiency and effectiveness. There are many examples where being effective and being efficient may not exactly go hand in hand. Many methods are great because they are very effective in accomplishing the objectives for which they were designed, but their cost is so prohibitive that it annuls the very purpose of having the method in the first place. Note that low-cost methods are not necessarily efficient unless they are also effective to some degree.

Finally, here are some examples of what we want from our methods to consider them efficient:

- To produce small test sets
- To produce test sets fast
- To produce fast executing tests
- To produce minimal amount of tests to acquire a desired piece of information. For example, we may want the method to verify certain areas of the program of specific risks using as few tests as possible

3.3 Human Elements

The human element is perhaps the most neglected in most relevant research. There is probably no good reason for this except that human behavior, skills, expertise, and other aspects of the human mind are little understood in general but especially to researchers of this and related fields. It is very hard to account for human elements in a testing method, which are impossible to control or even measure. But accounting for these elements is a necessary evil. Talking about effectiveness and efficiency without talking about the places where humans contribute to either is intuitively suspicious. Here are some human elements that may affect how well a method works:

- ***Expertise***: in engineering, in testing, in the same organization, in the same project, in a similar product line, etc…
- ***Domain expertise***: in the field of those who will use the software, and perhaps in the fields of other stakeholders that care about the software's quality.
- ***Skills***: using tools, pursuing the method's activities, preparing the method's artifacts, programming, scripting, designing tests, etc…
- ***Knowledge***: of the underlying principles of the method, of the inner workings of the application
- ***Communication skills***: working well with groups, effective communication of ideas

4. Methods of Evaluation and Comparison

It is unfortunate that little useful data on what works in practice or theory is available. Theoretical results are not enough, because, almost by necessity, they ignore the human element, and they sometimes make too many assumptions, which consequently limits the

Copyright © 2001 Ibrahim K. El-Far. All rights reserved. This paper to appear in the Proceedings of the Fifth International Internet & Software Quality Week Europe Conference (QWE 2002), March 2002.

scope of their results. On the other hand, most of the available results on test method evaluation and comparison stem from theoretical and simulation-based observations, largely by academics. In the industry, the situation is no better, where there is strong reliance on expertise, and biases and beliefs seem to be the most critical elements in making such assessments and any consequential decisions.

4.1 Classification of Approaches

Most approaches to test comparison can be classified as one of the following:

- ***Theoretical*** approaches make use of the formal properties of methods in order to establish some kind of mathematical relationship to partially order these methods. Theoretical comparisons have the benefit of generality but the pitfall of making too many assumptions and consequently severely limiting the applicability of the comparison's results.
- ***Simulation-based*** approaches make use of a simulation of how a test method would run. This is very useful when it is too expensive and time consuming to actually apply the method to real software.
- ***Experimental*** comparison takes a software application along with tests generated by a method to generate one data point evaluating the method as it worked with that application. Experimental comparisons provide a lot of insight, but it take a lot of time and effort to generate enough data points to establish a useful pattern. Certainly, such experiments can be used to validate comparisons done under the other approaches.

Comparing test methods is a topic rich in problems and, therefore, and it is very interesting to read some of the more significant work on it. The following are good places to start: [Hamlet 1988], [Hamlet 1989], [Weyuker et al. 1991], and [Weyuker 1988]

4.2 Test Method Comparisons

The results of software testing method comparisons take on many formats, some of which have some appeal that hides the misconception underlying these formats. What follows is a simple list of such formats, grouped in three categories corresponding to comparisons by effectiveness, efficiency, and the human element (Why is method M1 better than method M2?). The comparisons grouped by efficiency and the human element are left unexplained as I felt they were clear enough. The same references cited above, especially [Hamlet 1988] and [Hamlet 1989] are great references on these and similar misconceptions. Most of the ideas stated in here should be credited to them.

4.2.1 Fallacies in Comparing Methods Based on Effectiveness

- ***M1 finds more bugs than M2***. This is appealing at first, as finding bugs is an important goal for testing in the industry. Effective testing under this goal is not about how many defects were uncovered but how many defects remain in the software. Also, this type of comparison is very misleading when we have a highly defective application, for example, where finding bugs is as easy as using the application.

Copyright © 2001 Ibrahim K. El-Far. All rights reserved. This paper to appear in the Proceedings of the Fifth International Internet & Software Quality Week Europe Conference (QWE 2002), March 2002.

- ***M1 delivers better reliability than M2***. What that means is that tests that are selected by M1 are better than those selected by M2 in providing enough information for a more accurate reliability. This is also appealing, especially for those who are interested in testing for reliability estimation. However, such results suffer from the same assumptions of reliability theory such as the availability of realistic operational profile and usually ignore such important issues as the software's size and complexity.

- ***M1 is better than M2 in detecting faults of a certain class***. This all fine, except, because of the sampling nature of software testing, it is always possible to find a fault of the concerned class that M2 uncovers and that M1 does not. In other words, just because this statement is made does not mean that the weaker method is dispensable. The same applies for the following two statements. Also, some of these results may have to do with the nature of the testing method as related to the nature of the program, in particular the latter's bugs. Here's an oversimplified example: a method whose tests cover all the statements in a program is bound to uncover one-line faults of the caliber of syntax errors.

- ***All tests generated by M2 satisfy the adequacy criteria of M1***. This is very useful when there is little time and there is a desire to apply a method with strong criteria that would satisfy other weaker criteria at the same time. For example, a method that covers all the branches of a program will also cover all the statements. However, due to the gigantic nature of the sampling aspect of testing, there is always a chance that M2 tests will find faults that M1 will not.

4.2.2 Comparing Methods Based on Efficiency

- M1 produces smaller test sets than M2
- M1 produces test sets faster than M2
- M1 yield more return on investment that M2

4.2.3 Fallacies in Comparing Methods Based on the Human Element

- M1 is less dependent on human action than M2
- M1 requires less skills than M2
- M1 requires more domain expertise M2
- M1 requires less preparation than M2

5. Conclusions

This work hopes to have established the need for test method comparison. It has pointed out to what needs to be analyzed: the software, the method, and the circumstances of test and development. It discussed some aspects of what needs to be evaluated in a testing method, and explained a classification of comparisons, along with a few examples. Future work will involve an account of the factors that can affect the results of evaluation and the degree of control one has over various variables. Future work also involve suggesting a new mechanism of selecting test methods that overcomes some of the limitations of the comparisons outlined in the previous section.

Copyright © 2001 Ibrahim K. El-Far. All rights reserved. This paper to appear in the Proceedings of the Fifth International Internet & Software Quality Week Europe Conference (QWE 2002), March 2002.

Acknowledgements

Thanks are due to Florence E. Mottay and Herbert H. Thompson for reviewing parts of his paper.

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. In 2000, Ibrahim received a fellowship from the International Business Machines Center for Advanced Studies, Canada, supporting his research in approaches to interpreting, understanding, and analyzing software test effectiveness and efficiency. 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.

References

[Hamlet 1988] Richard Hamlet. Special section on software testing. Communications of the ACM, 31(6): 662-667, June 1988.

[Hamlet 1989] Richard Hamlet. Theoretical comparison of testing methods. Proceedings of the Third Symposium on Testing, Analysis, and Verification, Key West, Florida, USA, pages 28-37, December 1989.

[Weyuker et al. 1991] Elaine J. Weyuker, Stewart N. Weiss, and Dick Hamlet. Comparison of program testing strategies. Proceedings of the Fourth Symposium on Testing, Analysis, and Verification, pages 1-10, October 1991.

[Weyuker 1988] Elaine J. Weyuker. The evaluation of program-based software test adequacy criteria. Communications of the ACM, 31(6):668-675, June 1988.

QWE2002 Paper 4A

Dr. Ina Schieferdecker, Mr. Axel Rennoch, Ms. Mang Li & Dorota Witaszek
(GMD FOKUS)

Testing of CORBA Products

Key Points

- Conformance testing is the approach to achieve interoperable CORBA ORBs.
- The VSOrb test tools for CORBA support automated test development and execution.
- A C++ and Java test suite is developed and covers OMG specification CORBA v2.3.

Presentation Abstract

The conformance test suite for CORBA ORBs incorporate many aspects of functional testing, e.g. compiler output tests, API testing and protocol testing. Test methods, configurations, required tools and their integration are presented for both the C++ and the Java test suite implementation. The test suite which covers the CORBA specification 2.3 of the OMG has been already executed to verify CORBA implementations.

About the Author

Dr. Ina Schieferdecker studied Mathematical Computer Science at the Humboldt University in Berlin and received her Ph.D. from the Technical University in Berlin in 1994. Since 1993, she is a researcher at GMD FOKUS and a lecturer at Technical University Berlin since 1995. Her technical and research work covers testing methods for distributed systems and formal methods for the design, validation and prototyping of distributed systems. Dr. Ina Schieferdecker set-up the Competence Center for Testing, Interoperability and Performance (TIP) of GMD FOKUS in 1997 and is the head of TIP since then.

Fraunhofer Institute for Open Communication Systems

CORVAL2 CORBA VALIDATION

Testing of CORBA products

M. Li, A. Rennoch,
I. Schieferdecker and D. Witaszek

FOKUS, Berlin
http://www.fokus.fhg.de/tip

©Copyright I. Schieferdecker 2001-2002

Introduction

- CORBA ORBs are key component for object-based systems, services, and applications.
- Their interoperability is essential to build a heterogeneous, practical infrastructure.
- Their quality and the likelihood for their interoperability are increased for conformance tested ORBs. Further issues are resolved in a specific interpretations process.

©Copyright I. Schieferdecker 2001-2002

CORVAL2
- an EC IST Project

- Mission: develop new test techniques, tools and test suites for CORBA 2.3 and 2.4.
- Partners
  - The Open Group (Coordinator)
  - FOKUS (Technical Development)
  - OMG (CORBA Alignment)
  - Eric Leach Marketing (Marketing)
  - Fujitsu, Object Oriented Concept (ORB Vendors)
- Started January 2000 – 2 Years Project

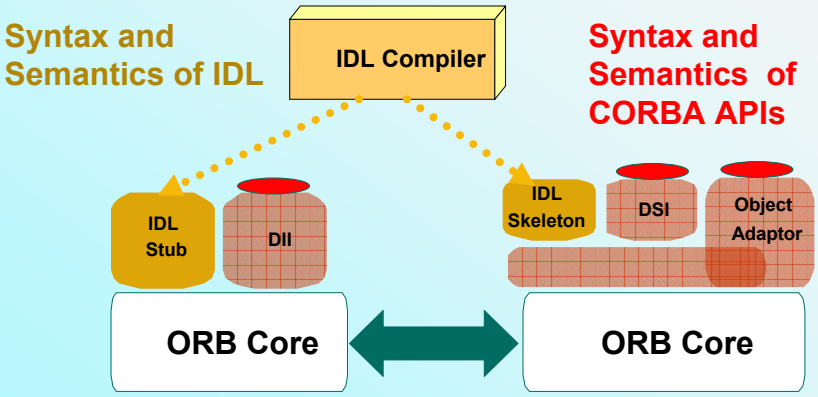
©Copyright I. Schieferdecker 2001-2002

CORBA Product Profile

- Defined by the Product Standard Definition Subcommittee (PSDS).
- A conforming ORB implementation must support:
  - CORBA core,
  - at least, either Java or C++ language mapping, and
  - interoperability protocols GIOP and IIOP.
- Not required are the support of:
  - COM/CORBA interworking,
  - interceptors, and
  - additional language mappings.

©Copyright I. Schieferdecker 2001-2002

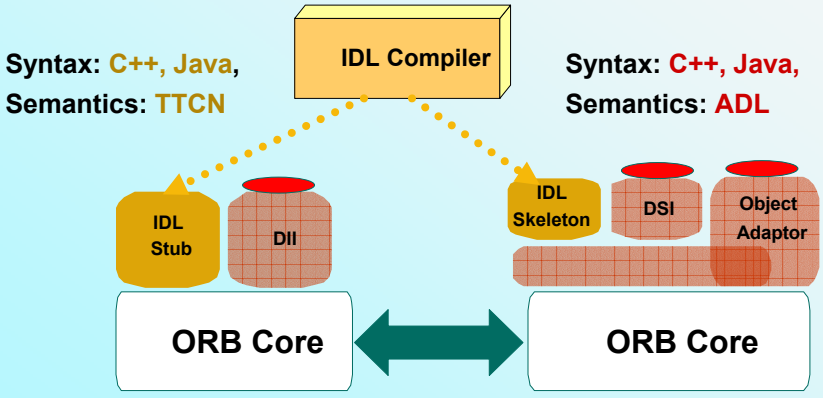
Conformance Aspects



GIOP and IIOP Protocols

©Copyright I. Schieferdecker 2001-2002

Testing Techniques



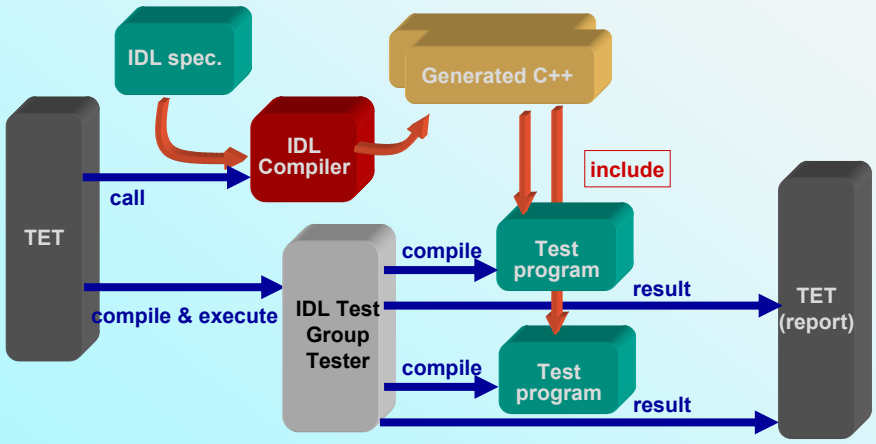
GIOP and IIOP: TTCN

©Copyright I. Schieferdecker 2001-2002

IDL Syntax Tests

- **Purpose:** to verify the syntax aspects of the code generated by the ORB's IDL compiler.
- **Focus:** static aspects of the mapping, i.e. on the type and function declarations contained in the generated source code.
- **Coverage:** about 400(C++,Java) IDL test cases available, whereas each test case covers 2-10 test purposes.

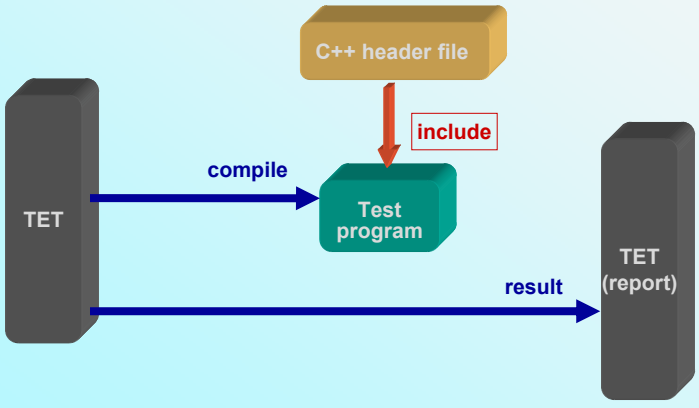
Usage of Tools



API Declarations Tests

- **Purpose:** to verify the declarations of the CORBA APIs (e.g. in form of C++ header files) provided with the ORB implementation.
- **Focus:** existence and correctness of data type, constants and interface declarations of the APIs for the target programming language.
- **Coverage:** about 1250 (C++, Java) for extensions of the CORBA 2.3 APIs, among them, value type semantics, abstract interface semantics, dynamic management of Any values, the portable object adapter, etc.

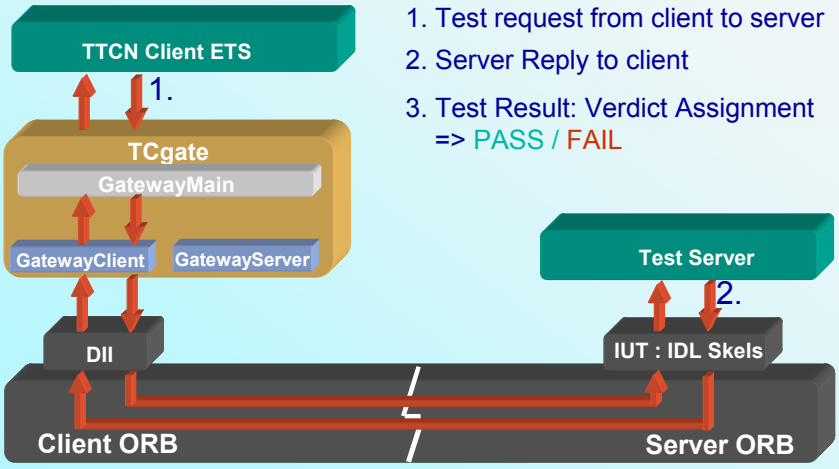
Usage of Tools



IDL Stub/Skeleton Tests

- **Purpose:** to verify the runtime behavior of the code generated by the ORB's IDL compiler.
- **Focus:** the support for the request-reply communication between clients and servers based on the stubs and skeletons under test.
- **Coverage:** appr. 380 (C++, Java) test cases for value type semantics etc.

Test Configuration



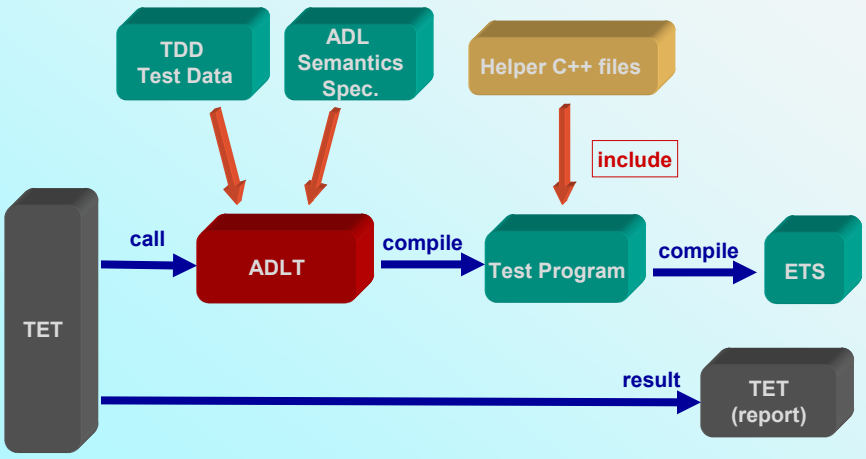
1. Test request from client to server
2. Server Reply to client
3. Test Result: Verdict Assignment
   => PASS / FAIL

API Behavior Tests

- **Purpose:** to verify the behavior of the CORBA APIs provided with the ORB implementation.
- **Focus:** semantics of operations and attributes of API interfaces.
- **Coverage:** about 450 (C++), 150 (Java) test cases for interfaces of the portable object adaptor, the dynamic management of Any, etc.

©Copyright I. Schieferdecker 2001-2002

Usage of Tools



©Copyright I. Schieferdecker 2001-2002

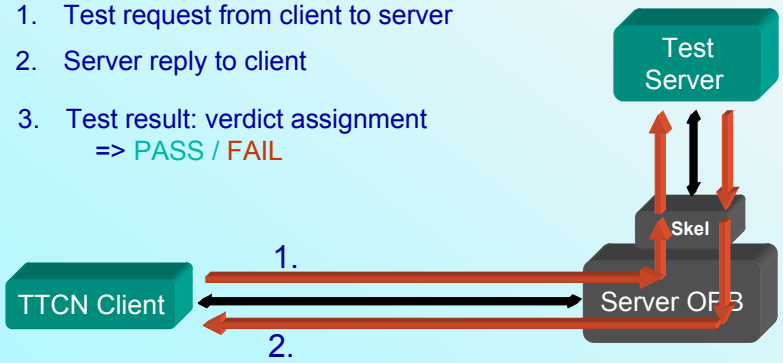
GIOP/IIOP Tests

- **Purpose:** to verify the ORB's capability in terms of the syntax of GIOP messages and their exchange over IIOP.
- **Focus:** basic functionalities supporting message ordering and CDR.
- **Coverage:** appr. 400 (C++), 200 (Java) test cases for GIOP version 1.0, 1.1 and 1.2.

Test Configuration

Server side ORB tests

1. Test request from client to server
2. Server reply to client
3. Test result: verdict assignment
   => PASS / FAIL



VSOrb

- The CORBA v2.3 conformance test suite for C++ and Java ORBs, appr. 2600 tests to evaluate the ORB correctnes

- Abstract test suite released in **Jul. 2000**
- Executable test suite for C++ released in **Jan. 2001**
- Executable test suite for Java released in **Sept. 2001**
- Branding in **Oct. 2001**

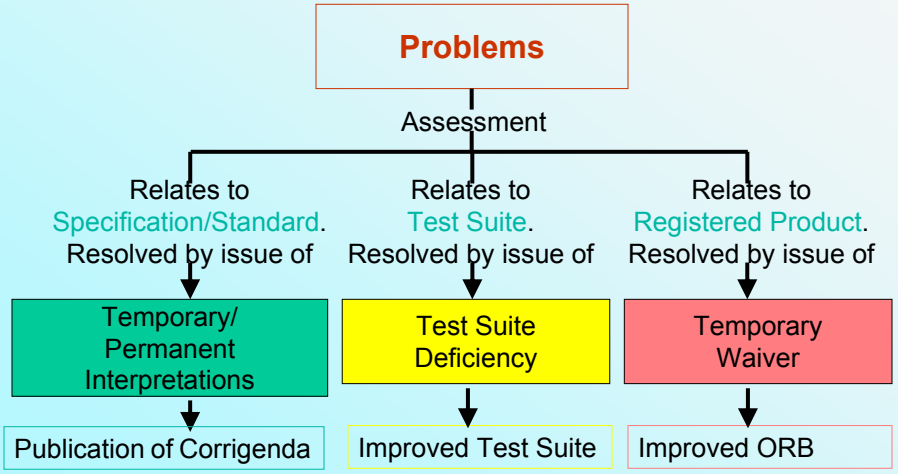
- Tests are performed with
  - ORBacus (OOC): C++, Java
  - InterStage (Fujitsu): Java
  - Mico (Open Source): C++

The CORBA Brand ...

- By The Open Group and the CORVAL2 consortium.
- Is based on the Product Standard.
- Uses a Conformance Questionnaire and VSOrb for CORBA v2.3.
- Successfully tested ORBs
  - can use the CORBA Brand Logo.

  - have to follow the interpretations process to resolve interoperability issues.

Interpretations Process



©Copyright I. Schieferdecker 2001-2002

Benefits of Conformance Testing and Branding

- Users
  - Guarantees of Conformance to OMG Specs.
  - Easier Procurement Procedures.
  - Vendor Commitment to Support Products.
- Vendors of Registered Products
  - Increased Market Acceptance and Visibility.
  - Ability to Bid for Procurements.
  - Demonstration of Commitments to Specification.

©Copyright I. Schieferdecker 2001-2002

Testing of CORBA products

M. Li, A. Rennoch, I. Schieferdecker and D. Witaszek

FOKUS
Fraunhofer Institute for Open Communication Systems
Kaiserin-Augusta-Allee 31, D-10589 Berlin, Germany
phone: +49 30 3463-7000, fax: +49 30 3463-8000

www.fokus.fhg.de/tip

Abstract

Since middleware platforms are in wide spread use for distributed systems, their quality is key to the stability and interoperability in multi-vendor heterogeneous environments. Interoperable CORBA ORBs can be achieved with conformance testing approaches. The exhaustive CORBA 2.3 Conformance Test Suite VSOrb for CORBA conformance testing checks the various system aspects of CORBA ORBs and supports automated test execution and analysis. It is applicable to both C++ and Java ORBs. This paper discusses testability aspects of CORBA ORBs and presents test strategies for them. An overview to the conformance test suite VSOrb is given.

1. INTRODUCTION

The quality of the programming and runtime platforms for distributed systems is a key issue in software development. An increasing number of vendors of middleware together with a huge number of applications arise the serious question on interoperability. In order to avoid a blind one-to-one interoperability testing of each pairs of server/customer configuration, it is proposed to follow the methodological approach of conformance testing. Conformance to standard specifications ensures interoperability and portability between products of different vendors, what is essential for the openness of distributed systems.

Testability is a key aspect for the quality assurance and success for such systems. According to [8], an ideal development of object-oriented systems is iterative and incremental. It must be accompanied by short code/ test cycles to allow iterative and incremental test planning, design and execution. This approach supports the testability of a system under development, but cannot be taken in all cases (e.g. if larger subsystems have to be integrated).

The paper considers the testability concept and approaches to test the different requirements of the CORBA core specifications. It covers theoretical and practical questions by addressing requirements on the system specification techniques to make CORBA ORBs testable and presents an approach towards the testability of ORB implementations. This paper is structured as follows: Section 2 discusses the concept of testability. In Section 3 the testing target, i.e. the special nature and requirements for object-oriented systems and CORBA ORBs specifically, is revisited. The implementation oriented view on testability for CORBA ORBs is given in Section 4. This section

describes also the selected test implementation strategies. A short overview on the developed conformance test suite for CORBA v2.3 ORBs is given in Section 5.

2. THE CONCEPT OF TESTABILITY

The overall objective of testing is to validate whether a system implementation realizes a system specification correctly. According to [18], the activity of testing consists of extracting information from a system implementation such that from this information a model of the system implementation can be constructed in sufficient detail to decide on its conformance.

The notion of correct implementation is based on a classification of faults (the so called fault model) that a system implementation might have. Faults should be detectable in order to be able to decide on their presence. Not all information about a system implementation is detectable by a test system (by an external observer). Furthermore, only a finite and typically restricted set of information about a system implementation can be collected since testing is always a finite process. Hence, one system model, which is generated in course of the testing process, may reflect different system implementations. Therefore, the assessment of the collected pieces of information might be wrong: an incorrect implementation could be assessed to be correct as its faults were not detected during test execution. Also, it is possible that a correct system implementation is assessed to be incorrect since a test system may be also faulty. An exhaustive test suite is a test suite where all passing system implementations are correct. A sound test suite is a test suite where all system implementations that do not pass are incorrect. A complete test suite is a test suite which is both exhaustive and sound, i.e. the set of passing system implementation is exactly the set of correct system implementations. Ideally, a test suite is complete, but this is not practical as an exhaustive test in finite time cannot be constructed in general. Therefore, a test suite shall be at least sound in order to be able to detect incorrect implementations.

In addition to the discussion on detectable faults it is important to take the practical view. The success of testing an implementation depends also on the quality of the reference specification the implementation has to comply with. In many cases it is due to incompleteness, contradictions or misunderstandings of the specification that conformance or interoperability tests fail. Unfortunately, even standardization procedures cannot prevent a specification from insufficiencies. Further it is obvious that the precise definition of requirements for an implementation is a key issue for the testability of a given implementation.

Often ‘testability’ is used as a synonym for ‘observability’ and ‘controllability’. In [1], a survey of these terms has been presented, considering both formal and informal viewpoints. Both, qualitative and quantitative notions are given. According to this overview, most informal definitions on ‘testability’ mean the “existence of features, properties or characteristics that facilitate the testing process of implementations”. The more formal definitions address ‘controllability’ as the “possibility of directing a system from given states to other states” and ‘observability’ as the “possibility of identifying the unknown state of a system”. The latter definitions suppose state automata as system models and the reachability property which is to reach any state with appropriate inputs. Recent studies [2] put emphasis on ‘experimental’ testability and consider the distinction between bounded and unbounded experimentation and a containment order on experiments. This approach allows to quantify testability e.g. by cost functions, such that minimum or maximum costs for refutation or validation may be computed.

3. THE TESTING TARGET: CORBA ORBS

The key component of the Common Object Request Broker Architecture (CORBA) is the ORB (Object Request Broker). It supports a distribution transparent communication between clients and server objects in a heterogeneous environment. Since its introduction, the architecture and specification of CORBA have been improved by several revisions. The Interface Definition Language (IDL), basic CORBA interfaces, interoperability protocols GIOP and IIOP, an interface for dynamic management of Any values (DynAny) and the concept of the Portable Object Adaptor (POA) were defined successively. New concepts in CORBA v2.3 include an improved POA, the Value Type and Abstract Interfaces. In addition, the number of supported languages has been increased continuously. Among the currently covered languages are C, C++, Java and Smalltalk.

While CORBA is maturing and more and more CORBA implementations and CORBA-based products emerge, the attention on interoperability and portability increases. The key of interoperability and portability is the conformance to standard specifications. CORBA specifications place conformance requirements in terms of Compliance Points. One compliance point is, for example, the whole CORBA Core that contains basic interfaces used by CORBA-based applications.

An artifact is needed to evaluate the interoperability and portability. It must be vendor independent and it should provide reproducible evaluation results. The EC IST project CORVAL2 develops and provides such an artifact. A major input was the CORBA Verification Suite (VSOrb v1.1) [23]. VSOrb is a test suite for the C and C++ mappings of CORBA v2.1. VSOrb v1.1 makes use of the Assertion Definition Language (ADL[22]) Translation System and Test Environment Toolkit (TET[24]), in order to automate the creation and execution of tests. A further input to the CORBA v2.3 test development origins from a recent work on testing techniques for CORBA-based systems, which is based on the Conformance Testing Methodology and Framework (CTMF[17]). The usability of CTMF for object-oriented systems is shown in [13]. A test management environment supporting CTMF-based test systems is also part of the input. It covers means for test management, including setup and control of tests, as well as test reporting.

The testability of CORBA systems covers different aspects of distributed object technology, the interface architecture and specification of objects, and the specific realization of implementations based on different programming languages. Major system aspects of a CORBA ORB are the. syntax and semantics of IDL definitions, syntax and semantics of the API, and the syntax and semantics of GIOP/IIOP (see also **Figure 1**). This is also reflected by the conformance requirements given in the product profile [20].

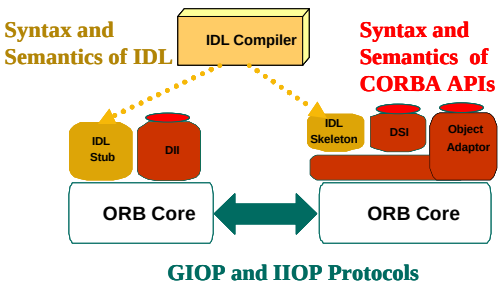


Figure 1. Relationship of different system aspects

The testability analysis of CORBA systems adopts these structuring principles, i.e. it distinguishes the following system aspects

language mapping for the mapping of interface descriptions (i.e. with an IDL compiler) to programming languages (e.g. C++ and Java) and generated interfaces (i.e. IDL stubs/skeletons)

operation based interface for APIs (e.g. ORB interface, DII, DSI, object adapter, interface repository) and CORBA services (e.g. name service)

message based interface for CORBA interoperability protocols between different ORB core implementations (i.e. GIOP/IIOP)

Requirements on CORBA ORBs for testability

The theoretical problem with respect to the testability of software - in opposite to the practical, implementation-oriented viewpoint (Section 4) - arises questions on three models:

the fault model addresses the type, location and probability of implementation errors,

the system model specifies the requirements, which an implementation has to fulfill, and

the test model covers the testing approach itself, e.g. state-based testing strategies or statistical test strategies.

It should be noted that fault models imply questions on test coverage, i.e. the objective to exercise all elements that are likely to be buggy [4]. More precisely, in [5] a distinction has been made between fault coverage and specification coverage. While the latter one reflects the relationship of a test suite and a system specification, it is the fault coverage which gives the number and the types of faults that can be discovered by testing. The following subsections are not intended to provide a study on fault models or coverage. They give rather an indication on typical faults to be discovered.

A system specification may contain information of heterogeneous nature, e.g. about reliability, efficiency, feasibility, machine environment, etc. In open distributed systems [6], the focus is on two basic constituents: the static aspects, i.e. the conceptual system architecture, and the dynamic aspects, i.e. the system behaviour. Furthermore, the need for a formal language for the description of those aspects and with a suitable execution model is claimed. It is proposed to cover both constituents within a single description. It should be noted that a formal description - in contrast to an informal (natural language) or semi-formal description - is not required but may enable unambiguous, verifiable and consistent system specifications. On the other side, formality may abstract from (i.e. hide) usability, cost or effectiveness of a system feature.

From the testing point of view: in [4], a list of requirements for the testability of a system model description has been provided. It focuses on the support of manual and automated test generation. It states that the criterion for model testability is to “devise and program an algorithm that will produce ready-to-run test cases with only the information in the model”. In addition, the following characteristics are described:

The model shall be a complete and accurate reflection of the kind of implementations to be tested. It must represent all features to be exercised.

It shall abstract details that would make the cost of testing prohibitive.

It shall preserve details that are essential for revealing faults and demonstrating conformance.

It shall represent all events (of the state model) so that it is possible to generate these events, typically as messages sent to the implementation under test (IUT).

It shall represent all actions (of the state model) so that it is possible to determine whether a required action has been produced.

It shall represent state so that it is possible to have an executable means to determine what state has been (or has not been) reached.

It is also important to consider testability already during the design phase. If testability is considered early enough, it supports the system design and enhances the effectiveness of an implementation, especially for large systems [8]. However, conformance testing is based on established specifications and can only in some cases revise a specification in order to increase testability. This is in particular true for CORBA ORB specifications, which have been adopted within a long process, where testability was not considered in advance. Some of the resulting problems are described below.

Finally, there are general requirements for system specifications (and the respective set of standards) to enable testing. These requirements include e.g. a sufficient integration in the case of multi-part specifications, a distinction between mandatory and optional requirements, etc. With the product profile by the OMG, basics of that information are provided for CORBA.

For all CORBA system aspects presented before, their definition must be unambiguous to enable testability. Formality and therefore the use of specifications techniques enable unambiguities. Therefore, specification techniques used for the definition of various system aspects are a major issue for the assessment of CORBA testability.

3.1 Language mapping

In CORBA, clients and servers are written in programming languages, but their interfaces are defined with IDL. Interface definitions are used to make client and server implementations portable and to hide implementation specific features from the user. The mapping of the interface definition to the target programming language depends on the features of this programming language (e.g. object orientation) and is defined for a variety of programming languages separately. However in CORBA, the semantic relationship between interface definition and target program are informal and base on plain text description (with examples) only.

A language mapping has to maintain the semantics of the origin interface language and to allow a usage-friendly development of clients and servers (e.g. memory management, conversion operators). Beside a possible misinterpretation of language mapping rules, there are several sources for faults to be discovered by testing. In the context of object-oriented programming, special emphasis is needed for inheritance, polymorphism, late binding, and encapsulation [4].

Tests on language mappings concern in fact the test of compilers (and interpreters). Compiler tests cover both static aspects on the error-free syntax of the generated code and dynamic aspects on the correct behaviour of this code. It has to be noted that different compilers may conform to a language mapping specification, but may produce different code however with equivalent behaviour. A language mapping compiler differs from classical programming language compilers which translate a programming test to machine code. Testing e.g. an IDL compiler has the following advantages in comparison to the latter one:

it produces an output which can easily be inspected (files with programming language code)

the input language specification does not contain any behaviour (e.g. state transition)

Thus, language mapping tests are controllable (e.g. IDL input) and observable (by the resulting files with programming language code). A core problem is how to analyse the results and how to assign a test verdict, i.e. how to compare the output with the specification of the language mapping, especially when taking into account the various possible mappings for a given language.

3.2 Operation based interface

Operation based interfaces are part of the object model of CORBA. An object (see also RM-ODP[18]) is an entity containing information and offering services. The information of an object is encapsulated. An object is only accessible through interactions based on the interfaces supported by the object. Each interface is characterized by a set of possible interactions between the object and other objects in its environment. An operation is invoked by a client object to request the execution of certain services by a server object. The invocation can be terminated by the server with or without a response.

The description of operation based interfaces covers the set of provided operations and their in/out parameter types and return types considering also allowed combinations and values. The semantics of an operation defines normal behaviour and exceptions, the ordering of operation invocations and expected return values. In testing, a distinction between the simple existence of such operations and their correct behaviour according to the specified operation invocations is made. This allows to discover with testing both missing operations or incorrect outputs at an interface.

Requirements on the testability of CORBA operation-based interfaces address basic requirements such as the completeness of provided definitions, but also further testing oriented details:

the prerequisites for operations and parameters,

the dependencies between operations under test and other operations/parameters,

the dependencies between parameters under test and operations/other parameters,

sample structures (proposed for testing) of data types,

sample values or value ranges (proposed for testing) of parameters, and

sample parameter combinations (proposed for testing) for operations under test

The fault model should consider both, valid and invalid values, structures, operation orders etc. The test approach can follow a typical ‘service testing’ approach, i.e. a classical request/reply (incl. exceptions) communication at the interface under test. The controllability in tests for operation based interfaces results from the underlying client-server principle, by which operation under tests can be invoked from the test system. The observability is restricted to the direct reactions on the operation invocation (i.e. reply or exceptions). Internals that results from an operation invocation are typically not observable.

3.3 Message based interface

In order to understand the requirements for testing a message based interface with an underlying protocol specification, a short introduction into protocol hierarchies is needed: in order to reduce the design complexity, most networks are organized as a series of layers. This insulates layers from details on how the lower layer services are implemented. Layer n on one machine communicates with

layer n on another machine. The rules and conventions used in this conversation are collectively known as the “layer n protocol”. In other words, a protocol is an agreement between the communicating parties on how communication has to proceed.

To be complete, each protocol specification should explicitly include five distinct parts [7]:

the service that is provided by the protocol,

the assumption about the environment in which the protocol is executed,

the vocabulary of messages used to implement the protocol,

the format of each message in the vocabulary, and

the procedure rules guarding the consistency of message exchanges.

Similar to specification languages, a protocol definition contains a vocabulary with a syntax format definition which needs to be known by the communicating parties. For a correct sending and receipt of messages, encoding and decoding rules for the protocol messages are required. The protocol procedure rules can be understood to be a protocol language grammar. An important aspect of protocols is that the procedure rules are interpreted concurrently by a number of interacting protocol entities. Because of this concurrency, a protocol behaviour is not always reproducible. The concurrency creates problems like timing, race conditions and possible deadlocks and hinders controllability and observability.

Controllability is realized via the upper and lower service access points of a protocol entity. Since in most cases, the lower service access point is accessible only, the controllability is restricted. Observability is given by messages that are exchanged between protocol entities. Again, only the direct reactions are observable but not internals that are caused by incoming messages. The fault model of message related system aspects corresponds to classical conformance testing approaches[17], which focuses on the identification of wrong messages, which are caused by coding errors, faulty protocol data or inconsistent behaviour w.r.t. the protocol rules.

4. TEST STRATEGIES FOR ORB IMPLEMENTATIONS

As seen from the previous sections, the testability of complex systems like CORBA is depending on a series of requirements on the system’s model, specification and implementation details. This section discusses testable features for ORB implementations and proposes appropriate test strategies and approaches.

4.1 Language mapping

IDL has a formal syntax and an informal but well understood semantics. A target language is a programming language (e.g. C++ or Java) with defined syntax and semantics. A mapping describes how the IDL syntax and semantics can be reflected in the syntax and semantics of a given programming language (considering available concepts in the used version of the programming language). The mapping gives to the ORB vendors (to a certain degree) freedom for the form of the resulting code.

The IDL compiler is testable in the sense that it can be assessed (using different IDL input) and observed (if output in readable form is available). Due to the informal character of the mapping description and the variety of possible mappings, the full automation of the test case generation and result analysis is practically not possible. It is proposed that the test case developer performs the following steps to achieve a representative set of test cases:

define a set of IDL specifications to be tested,

based on the mapping rules, the features of the target programming code should be specified manually,

compare the output obtained from the test execution with the expected output, and assign a verdict.

Test model: In the context of compiler tests, there is no complete test theory available, which solves the language mapping problem for IDL. Nevertheless, there is some general work on compiler tests [3]: language mapping (i.e. compiler) tests are black box tests, since there are no possibilities to inspect a compiler, but only to observe and run the output of the compiler for a given input specification. Following this viewpoint the output of the compiler has to be verified according to the input. W.r.t. the static aspects, the output of the "compiler under test" could be validated by the programming language compiler of the target language.

An illustration on the general process and the relationship of the involved documents in given in **Figure 2**. The IUT is an IDL compiler under test. The Test Sequence Generation has to produce (based on the IDL Grammar) a set of IDL test sequences covering all IDL constructs used in all possible context (or a representative subset of it). An output of the Test Sequence Generation is a (set of) specification(s) in IDL which will be used as an input for the IUT (these are the test cases). The IUT produces its output in a target programming language, for which the mapping rules are defined.

The following activities produce means for the test result analysis: The mapping rules are an input to Test Program Generation, which has to produce test programs. Further input is the IDL grammar and the test input in order to generate the test programs only for constructs (and their values, if any) which are currently used to test the IUT. For each IDL construct, the Test Program Generation produces test programs (including the expected results and verdict assignments). The test programs and the compiler output are used by a Test Result Analysis to compare them, assign a result and produce a test report.

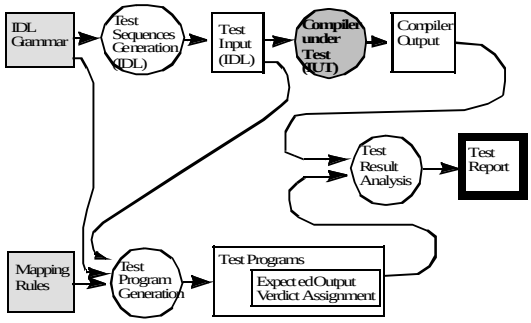


Figure 2. The development of IDL compiler conformance tests

The developed test cases (i.e. test programs) can become part of a tool for an automatic test case execution and verdict assignment for IDL compiler test. Test specifications for compiler tests (i.e. the test input) can be generated from an input language syntax to achieve a nearly complete coverage of the language constructs.

In CORBA it is due to the unavailability (i.e. there is no clear access point for the test system) of a well defined interface between the generated interface code (stubs and skeletons) on one side and any API on the ORB side, that the resulting behaviour can be tested only by a combined test of two corresponding interface programs or substitutes.

4.2 Operation based interface

The use of IDL for interface definitions gives a clear specification for the operation signatures only and leaves open the operation semantics etc. Furthermore, proposals for testing (sample parameter structures, values and combinations) are missing in the CORBA specification. With the definition of the mapping rules of IDL to programming languages like C++, guidance is given on how to implement CORBA applications, but also on the test system implementation. In particular, concrete information on the interface characteristics is given, which can be tested. It defines not only the proposed mapping, but also illegal application examples, possibilities, exceptions etc. Unfortunately, the majority of this information is informal only.

Test model: We concentrate here on the test model for features, which are new in CORBA v2.3 and have led to new test objectives. No new test strategies are needed for the (static) header file tests, but for the behaviour tests. It is proposed to use both

a test servant for the different features at the ServantManager interface, and

a new test POA to apply selected policies distinguished by the POAManager.

Similar to POA, the semantic tests for ValueType require the construction of a value type to be passed together with additional properties (state). Tests have to cover both kinds of value types, concrete and abstract (no state information). Since the ValueType concept introduces custom marshalling, a suitable sample encoding/decoding model and implementation is needed. Since Abstract interfaces do not introduce any operations, no header file tests are required. A sample abstract interface is needed to verify the Abstract interface semantics. DynAny semantic tests are based on the construction of sample any values.

The question on generating operation based interface tests targets three major aspects, namely the definition of the ordering, the test pattern, and the test values. The ordering aspects covers both, the order of interfaces under test and the test ordering of interface features (e.g. basic types, operations, exceptions,...).

Test patterns, i.e. generic and universal test case structures, are proposed to automate the test development process. Sample test patterns on the static tests have been proposed in [15]. The test body of semantics test are simpler and could be expressed using a case distinction for normal and exceptional behaviour as already supported by some test tools (like ADL[22]).

More efforts are needed w.r.t. the selection of test values. Appropriate test data filtering techniques [11] can express the requirements and allow the translation into the test programming code.

Unfortunately, developments in this research areas are still at the beginning, i.e. a lot of manual work is needed.

4.3 Message based interface

CORBA message based interfaces are not precisely defined. For example, the behaviour of a client when receiving an unexpected message is not clear. As formal descriptions of CORBA interoperability protocols are not available, again manual test development is used mainly.

Test model: Testing of CORBA interoperability protocols follows classical OSI protocol conformance testing [17]. Testing the server side ORB is straightforward since the server application is on top of the IUT, which can be implemented directly based on the server side ORB API. The client behaviour can be a part of the means of testing (e.g. realized by a TTCN based test program).

Testing the client side requires an additional information exchange: after starting a client, the application must send a request to its ORB (under test) in order to enforce it to send an GIOP request to the server side test component. After this request, the server test component will send a reply message.

Another issue is that the GIOP/IIOP communication protocol does not support status requests. Thus, there is no possibility to verify whether the client ORB (under test) has accepted and transmitted any reply to the client application, which has been received from the server side. That means that the client application (or any other component) may require some suitable means to check the client ORB (under test) status. Further, the client application must support the server side test component, which gives the final verdict, i.e. the client application has to send back any received reply (using some request message). To avoid a deadlock, a timer must be introduced under the control of the client application and a corresponding signal for the server tester side in case of a time-out has to be send to the server side.

5. VSORB V2.3 - THE CONFORMANCE TEST SUITE FOR CORBA ORBS

Based on the conformance aspects introduced above, a comprehensive conformance test suite has been designed and implemented. The focus of the test suite is the CORBA v2.3 specification including the language bindings for C++ and Java. Approximately 1000 new tests have been developed to adress the specifics of CORBA v2.3. New features of the tests cover value type, abstract interface, dynamic management of any value, portable object adapter. Also, the enhancements of GIOP/IIOP in CORBA v2.3 are considered.

Tests for language mappings are covered by IDL tests to verify the syntax aspect of the code generated by the ORB's IDL compiler and by Stub and skeleton tests - to verify the runtime behaviour of the code generated by the ORB's IDL compiler. Tests for operation based interfaces are contained in API declaration tests to verify the declarations of the CORBA APIs (e.g. in form of C++ header files) provided with the ORB implementation and in API behaviour tests to verify the behaviour of the CORBA APIs provided with the ORB implementation. Message based interfaces are tested with GIOP/IIOP tests to verify the ORB's capability in terms of the syntax of GIOP messages and their exchange over IIOP. In the following, details of the objectives, structure and techniques used by each test group are given.

The Java-based test suite has been developed following the test objectives of the C++ tests. The test strategy had to be revised for the tests bound to the syntactic aspects of the CORBA core: a class tester based on the Java language reflection utility is used to analyze the e.g. generated code from an IDL compiler. Since the tests for the dynamic semantics and protocol aspects are specified in an abstract manner, they are independent from the target language. Thus, the migration to Java required only the migration of simple test emulators. All executable tests for C++ ORBs as well as for Java ORBs are executed with the same test management system TTman [14].

The complete VSOrb test suite is a very powerful tool due to its extensive coverage of the features defined by the CORBA 2.3 specification. The high level of details allows an in-depth and due to test automation a quick error analysis. The test suite has already been successfully executed against commercial and open source ORB products.

After the presentation of the test suite at the OMG Technical Meeting in Paris, Apr. 2001, the tests have been made freely available via the CORVAL2 project web site[1]. In addition, a commercial branding programme has been launched in Oct. 2001 to offer a CORBA brand for successfully tested CORBA ORBs using the VSOrb tests.

5.1 IDL Compiler Test (IDL)

Objectives: The purpose of the IDL tests is to check whether an IDL compiler translates IDL specifications into the target language correctly, according to the CORBA language mapping standard. The focus is on static aspects of the mapping, i.e. on the type and function declarations contained in the generated source code..

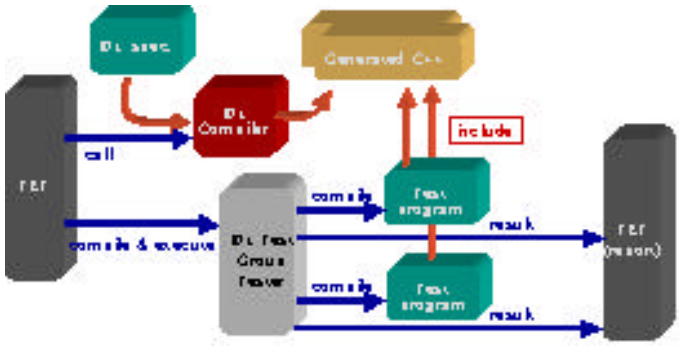


Figure 3: Structure of IDL tests

Test group structure: The tests are structured into test groups, test sub-groups and test case groups according to the logical structure of the language mapping. Each test case group consists of an IDL specification and a set of test cases, whereas each test case is a list of features to be tested. Tests for the new features of CORBA v2.3, e.g. value type, abstract value type and abstract interface, have been developed. There are about 400 IDL test cases for C++ as well as Java available, whereas each test case covers 2-10 test purposes.

Used test technique: The test suite is executed with the TET[24] testing environment via TTman[14]. The tests for C++ as target language are based on compiler error checks: for each test sub-group an IDL file is used as input to the IDL compiler under test. The generated code is then included in a

codelet file and compiled using a target language compiler (e.g. the gcc for C++ or jikes for Java). The assigned verdict depends on the success of the compilation.

5.2 IDL Stub and Skeleton Test (SII)

Objectives: The test cases cover the semantics of the stubs and skeletons generated by an IDL compiler. Focus of the CORBA v2.3 stub and skeleton tests is the value type semantics.

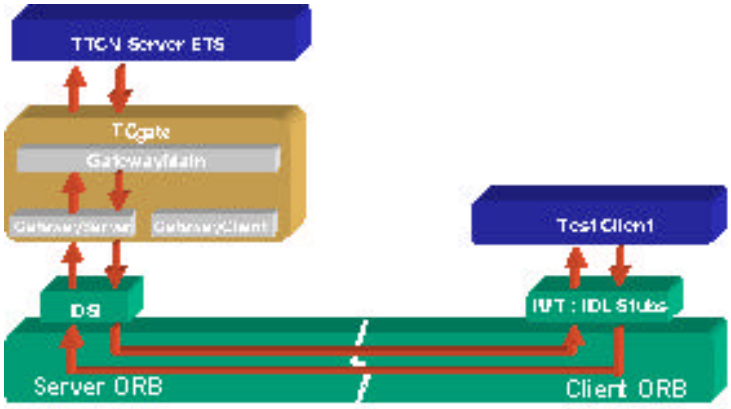


Figure 4: Structure of SII tests

Test group structure: The tests are separated into two test groups for stubs and skeletons, respectively. Subgroups for tests on substitutability of value parameters and on passing instances of different valuebox types as parameters are defined. The stub test group and the skeleton test group contain about 380 IDL test cases for C++ as well as Java ORBs.

Used test technique: The test notation TTCN (Tree and Tabular Combined Notation[17]) in combination with the TTCN/CORBA gateway TCgate[9] is used. The test suite can be executed using command-line mode or using TTman[14].

5.3 API Declaration Tests (DECL)

Objectives: The focus is on a CORBA API implementation according to its standardized requirement specification and the CORBA language mapping rules. For CORBA v2.3, new declaration tests cover value type semantics, abstract interface semantics, dynamic management of Any values and the portable object adapter.

Test group structure: The structure of the declaration tests corresponds to the CORBA specifications. There are about 1250 test cases for testing CORBA APIs of C++ as well as Java ORBs. From the CORBA v2.1 test suite [23], a number of tests have been updated according to modifications of parameter types, structure extensions etc. Some tests are removed due to obsolete features (e.g. BOA, ConstructionPolicy).

Used test technique: For the C++ declaration tests, a pattern-based approach is used. The tests use a compiler-error check, like the IDL tests, and are executed with TET[24] via TTman[14].

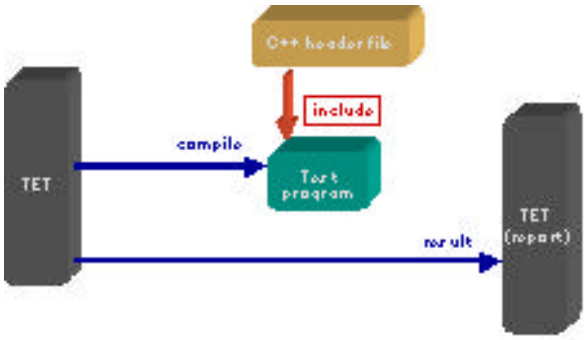


Figure 5: Structure of DECL tests

5.4 API Behaviour Test (API)

Objectives: The test cases cover the semantics specification of the CORBA APIs. Focus of the new tests are the interfaces of the portable object adaptor and the dynamic management of Any1.

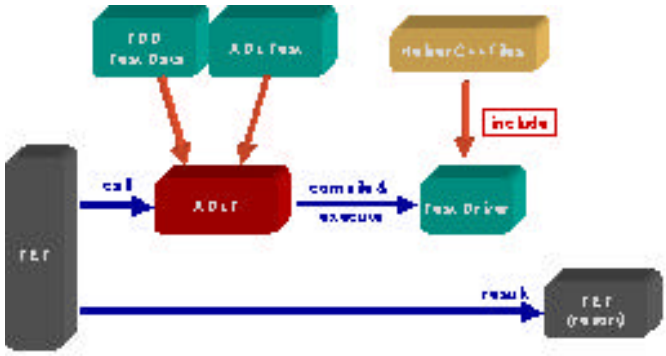


Figure 6: Structure of API tests

Test group structure: The tests cover operations and attributes of the considered APIs. Special emphasis is given to Portable Object Adaptors (POA) and value types. About 450 test cases for C++ and about150 test cases for Java are available.
Used test technique: The tests are defined using Assertion Definition Language (ADL) including the Test Data Description Language (TDD) [22]. The tests are executed with TET[24] via TTman[14].

5.5 GIOP/IIOP Test

Objectives: The tests cover the static (message format) and dynamic (message sequence) aspects of GIOP/ IIOP.

---

[1]Only mandatory operations and attributes of API interfaces are considered. The interfaces defined for the custom marshalling of value types do not provide standard behaviour and are therefore out of scope.

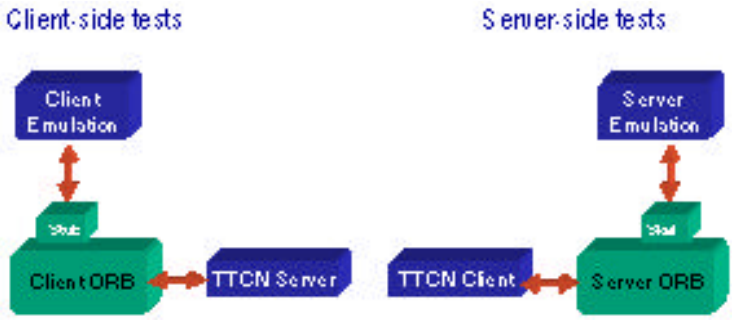


Figure 7: Structure of GIOP/IIOP tests

Test group structure: The GIOP tests are divided into server site and client side tests. Each group contains another three sub-groups on message ordering, on Common Data Representation (CDR) and on pseudo object types. The tests contain appr. 400 tests for C++ ORBs and appr. tests for 200 for Java ORBs and their GIOP implementation in version 1.0, 1.1 and 1.2.
Used test technique: The test notation TTCN[17] is used for the definition of GIOP/IIOP tests. The test suite can be executed using command-line mode or using TTman[14].

6. CONCLUSIONS

Testability of complex systems is a key aspect for the success of quality assurance for such systems. According to [8], an ideal development of object-oriented systems is iterative and incremental. It must be accompanied by short code/test cycles to allow iterative and incremental test planning, design and execution. This approach supports the testability of a system under development, but cannot be taken in all cases (e.g. if larger subsystems have to be integrated).

This paper discusses the technical approach of conformance testing the CORBA core2. According to the analysis of the different requirements in the CORBA core specification and their testability, we have to distinguish the following system aspects (see Figure 1):

language mapping, i.e. the mapping of IDL interface descriptions to programming languages,

operation based interfaces, i.e. the CORBA API, and

message based interfaces, i.e. the ORBs interoperability protocols (GIOP/IIOP).

The VSOrb v2.3 test tools check exhaustively the testable features of CORBA core implementations. The successful application of this conformance test suite is a definitive step towards high-quality, stable and interoperable CORBA infrastructures.

For syntactic tests of the language mapping and the operation based interfaces, current C++ and Java compilers are involved to run the tests. The corresponding semantic tests are defined in an abstract, implementation-independent manner to enable readability and portability of the test definitions. They are automatically transformed into executable tests.

Message based interface tests address the General Inter-ORB protocol (GIOP) in combination with the specific Internet Inter-ORB (IIOP) protocol for communication via TCP/IP. They are strictly

[2]This work is partially supported by the European IST Project Corval2 (IST-1999-11131, Enhanced Techniques for CORBA Validation, http://www.opengroup.org/corval2).

developed according to the protocol testing process defined within the CTMF [3]. TTCN, which is defined by CTMF, is used to specify the tests addressing both message formats and ordering covering valid, invalid and inopportune protocol behaviour.

Modern and powerful test development, execution and management tools have been selected, enhanced and used in the project to enable an efficient test development and to automate it. The tools have been combined and integrated to offer a common user interface for all test categories.

The techniques and tools are also applicable for functional tests of CORBA applications (server and client side), independently from a specific ORB product. In comparison to usual CORBA application test tools, the high granularity of the CORBA core test suite enables the system developer not only to know about possible failures within a product, but also to get detailed knowledge on the specific error type and location.

Furthermore, system specifications of the CORBA core, informal and formal parts, have been used to establish a catalogue of fixed test objectives. However, they do not contain sufficient information needed to develop unique tests. For example, the variety of system parameters may lead to a wide range of possible test instantiations and implementations. Therefore, there is a need to provide clear and unambiguous system specification and to enhance them with test related information. In the context of language mapping system aspects, an informal approach for test generation has to be taken since in CORBA developers are free in the design and realization of their implementations. Also, the review of available abstract interface specifications leads to the understanding that none of the currently used description techniques covers the full information required for testability, i.e. sufficient specification of data and dynamic behaviour. However, there is an approach (the reference point facet approach) for refining the specification of ODP reference point specifications. This will be used to enhance the specification and testability of object-oriented interfaces. Further, the specification of test data needs to be restricted to a practical scope using some filtering specification techniques. If this information will be specified by appropriate test description techniques, it can be used during the test generation phase, too.

There is a need to close the gap between a system requirement specification and its implementation in order to strengthen testability. The UML approach, which combines several viewpoints (including implementation) in one framework, seems in particular to be valuable in order to enhance testability and the confidence in product quality. A UML specific test profile is of specific importance to enable the direct test case development in the context of UML specifications.

7. ACKNOWLEDGMENTS

The work is partially supported by the European IST Project Corval2 (IST-1999-11131, Enhanced Techniques for CORBA Validation, http://www.opengroup.org/ corval2).

We want to thank our project partners for the fruitful cooperation in Corval2: The Open Group, IONA Tech., Fujitsu, Eric Leach Marketing Ltd., and Object Management Group. Special thanks for their contribution to the development of the test suite are given to Dr. Uwe Seimet, Cormac McKenna and John Wigram.

8. REFERENCES

[1] B. Baumgarten, H. Wiland: Qualitative notions of testability. In: A. Petrenko, N. Yevtushenko (eds.): Testing of communication systems. Kluwer Academic Publishers, 1997.

[2] B. Baumgarten, O. Henniger: Testability with unbounded testing strategies. In: G. Csopaki, S. Dibuz, K. Tarnay (eds.): Testing of communicating systems, Kluwer Academic Publishers, 1999.

[3] A.S. Boujawah, K. Saleh: Compiler test case generation methods: a survey and assessment. Information and Software Technology 39 (1997) 617 - 625, Elsevier Science B.V. Amsterdam 1997.

[4] R. v. Binder: Testing Object-Oriented systems. 1999.

[5] O. Charles, R. Groz: Basing test coverage on a formalization of test hypotheses. In: M. Kim, S. Kang, K. Hong (eds.): Testing of Communicating systems, Vol. 10, Chapman & Hall, 1997.

[6] R. Gotzhein: Open Distributed Systems. Vieweg Verlag, Wiesbaden 1993.

[7] G. Holzmann: Design and validation of computer protocols, Prentice-Hall, London 1991.

[8] H. König, A. Ulrich, M. Heiner: Design for testability: a step-wise approach to protocol testing. In: M. Kim, S. Kang, K. Hong (eds.): Testing of Communicating systems, Vol. 10, Chapman & Hall, 1997.

[9] M. Li, I. Schieferdecker, A. Rennoch: Testing the TINA Retailer Reference Point.- ISADS'99, Fourth International Symposium on Autonomous Decentralized Systems, Tokyo (Japan), 1999. (describes TCgate).

[10] R. Orfali, D. Harkey, J. Edwards: The essential client/server survival guide. 2nd edition, Wiley computer publishing, New York 1996.

[11] A. Rennoch, J. de Meer, I. Schieferdecker: Test Data Filtering. FBT'99, Munich, June 1999.

[12] J. Rumbaugh, I. Jacobson, G. Booch: The Unified Modeling Language. Reference Manual. Addison-Wesley, Reading 1999.

[13] I. Schieferdecker, M. Li, A. Rennoch: Incremental Testing at System Reference Points. - The IFIP 13th Intern. Conf. on Testing of Communicating Systems, Ottawa (Canada), Aug. 29 - Sept. 1, 2000.

[14] T. Vassiliou-Gioles, M. Li, I. Schieferdecker, M. Born, M. Winkler: Configuration and Execution Support for Distributed Tests. - IFIP 12th International Workshop on Testing of Communicating Systems (IWTCS'99), Budapest (Hungary), Sept. 1999. (describes TTman)

[15] CORVAL2: Deliverable 1 - Evaluation of the Existing Concepts and Methods Applicable for CORBA Validation, Mar. 2000.

[16] CORVAL2: Deliverable 11 - Specification of the CORBA v2.3 Conformance Test Suite, May 2000.

[17] ISO/IEC 9646-3: Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation (TTCN), edition 2, Dec. 1997.

[18] ITU-T Rec. X.901 | ISO/IEC 10746-1: Information Technology - Open Distributed Processing - Reference Model: Overview, Aug. 1997.

[19] OMG: The Common Object Request Broker: Architecture and Specification, v2.3, Jul. 1999.

[20] OMG: Product Profile, CORBA v2.3, psdef/99-11-01, Nov. 1999.

[21] OMG: CORBA C++ Language Mapping Specification, v2.3, Jun. 1999.

[22] The Open Group: Assertion Definition Language (ADL): http://adl.opengroup.org

[23] The Open Group: CORBA Verification Suite User's Guide, v1.1.1, Sep. 1999.

[24] The Open Group: TETware: http://tetworks.opengroup.org.

QWE2002 Paper 5A

Mr. Bernd Mattern
(imbus AG)

The AGEDIS Software Test Technology Project

Key Points

- The AGEDIS project and the consortium.
- AGEDIS methodology and architecture.
- AGEDIS time schedule.

Presentation Abstract

The AGEDIS project is a three year EC funded research project whose objective is to increase the efficiency and competitiveness of the European software industry by automating of software testing, and improving the quality of software while reducing the expense of the testing phase. AGEDIS will achieve this by developing a methodology and tools for the automation of software testing in general, with emphasis on distributed component-based software systems. The first phase of the project consisted of evaluation experiments with two existing test generation tools. These experiments completed in May, and the consortium is now in the throes of developing a set of tools with open public interfaces. The tools include a modelling language compiler, a test case generator, a test execution engine, and productivity tools for testers. The consortium includes research laboratories in Israel, France and the UK, and industrial partners in France, Germany, Greece and the UK.

About the Author

Bernd Mattern studied Physics from 1989 to 1996 in Erlangen, and worked as Scientist from 1996 to 1998 at Institut f r Technische Physik 2, FAU Erlangen. He has been working at imbus since 1998 as team leader, consultant, and trainer.

The

AGEDIS

Project

Automated Generation and Execution of Test Suites

imbus AG
Kleinseebacher Strasse 9
D-91096 Moehrendorf
Germany
Phone: +49 (0)9131 75180
Fax: +49 (0)9131 751850
Email: info@imbus.de
Web: www.imbus.de

© imbus AG 2001

Bernd Mattern
imbus AG
Kleinseebacher Strasse 9
91096 Moehrendorf

AGEDIS What is AGEDIS?

A utomated

G eneration and

E xecution of Test Suites for

Dis tributed Component-based Software

Information Society Technologies (IST) Programme

The European Commission
Community Research
ist information society technologies

Total budget: 4.3 M EUR EC pays: 2.3 M EUR

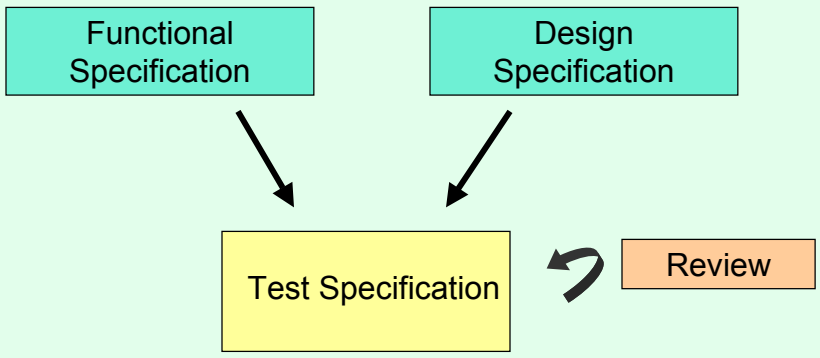
AGEDIS

Agenda

1. Motivation
2. Goals of AGEDIS
3. Overview of AGEDIS
4. AGEDIS Modelling Language
5. Other Approaches
6. Summary and Outlook

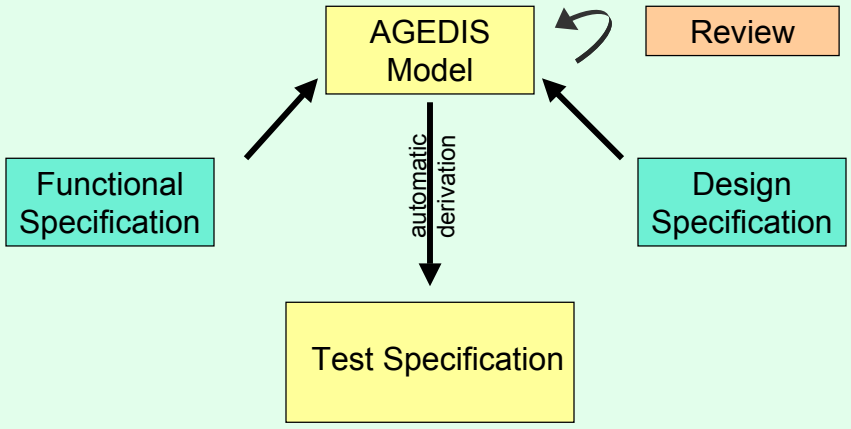
AGEDIS

1. Motivation



AGEDIS

1. Motivation



AGEDIS

2. Goals of AGEDIS

- Creation of a new testing methodology
- Cheaper software development costs
- Continuous approach to testing
  from modelling the software
  to the evaluation of the test run results
- Adoption of standards

AGEDIS

3. Partners (Institutions)

IBM Haifa, Israel: Alan Hartman, Kenneth Nagin, Sergey Olvovsky, Andrei Kirshin,...

Oxford University, UK: Jim Davies, Alessandra Cavarra

France Telecom R&D, France: Yves-Marie Quemener

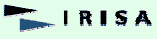
Intrasoft International, Luxembourg, Greece: Kyriakos Bechrakis, Antonis Ramfos

AGEDIS

3. Partners (Institutions)

IBM Hursley, UK: Ian Craggs

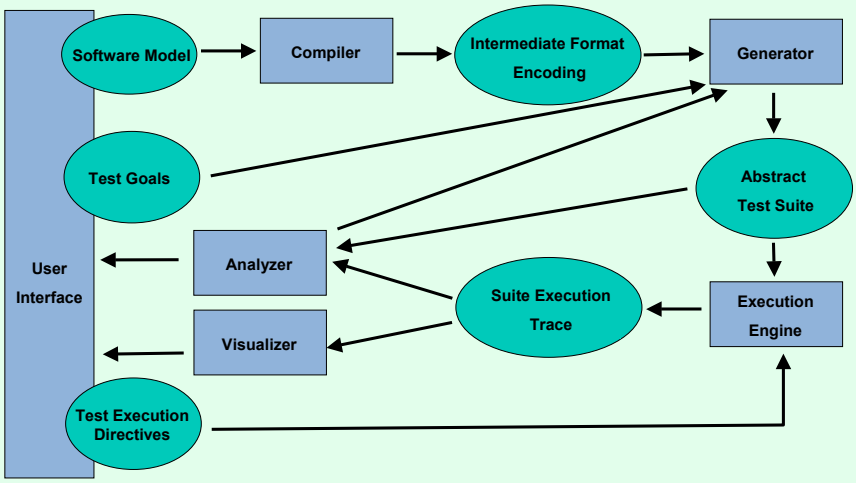
VERIMAG - Université Joseph Fourier, France: Laurent Mounier, Yassine Lakhnech



Irisa, France: Thierry Jeron

imbus AG, Germany: Manuela Heigl, Johannes Trost, Bernd Nossem, Klaudia Dussa-Zieger, ...

AGEDIS 3. Architecture



AGEDIS 3. Work Package Structure

Work Package (WP)	total effort
WP1 - Project Management	33 PM
WP2 - Software Modelling Language	25 PM
WP3 - Modelling Language Compiler	31 PM
WP4 - Test Generation Tool	32 PM
WP5 - Test Execution Tool	32 PM
WP6 - Test Productivity Tool	37 PM
WP7 - Test Automation Environment	36 PM
WP8 - Application Experiments	56 PM
WP9 - Technology Transfer	54 PM
Sum	336 PM (28 PY)

AGEDIS 3. Roles of Partners

Coordinator, leader of WP3, WP5, WP6, moderator exp 2

Leader WP2 and educational expertise

tool user, delivers test subjects

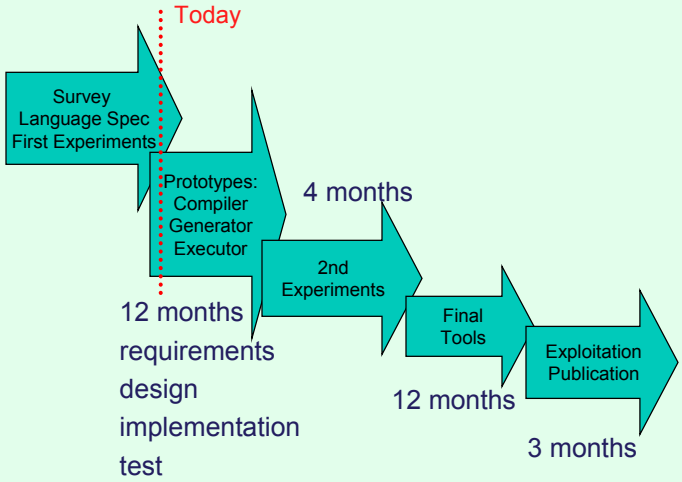
tool user, leader of WP7, integration, delivers test subjects

tool user, supplier of test subjects

leader WP4, moderator exp 1

tool user, testing expertise, leader of WP9, moderator exp 3, 4, 5

AGEDIS 3. Flow of Events



AGEDIS

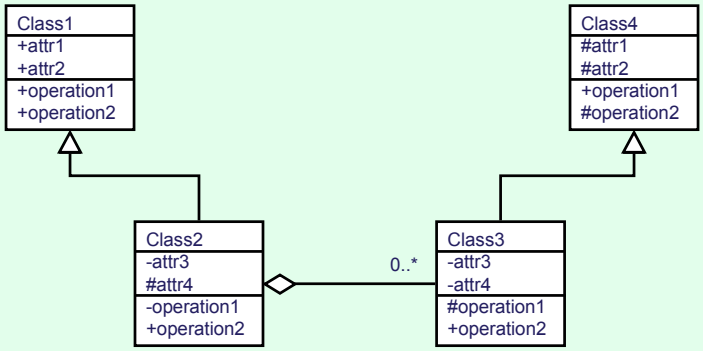
4. Modelling Language

- UML based
- Subset of UML diagrams
- Definition of special semantics
- Definition of software design
- Definition of test goals / test directives

AGEDIS

4. From Software Design

Class Diagrams:

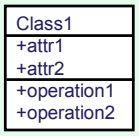


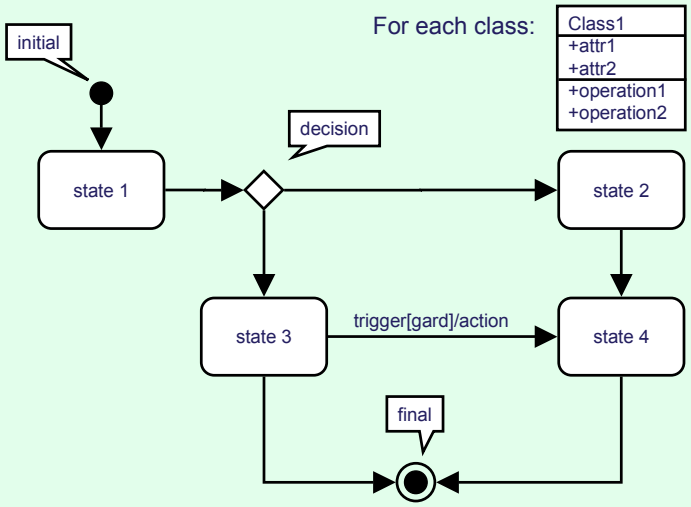
AGEDIS

4. From Software Design

State transition diagrams:

For each class:





every state has:

- entry & exit actions
- internal transitions
- **NEW in AGEDIS**:
  deferred events
  timer events

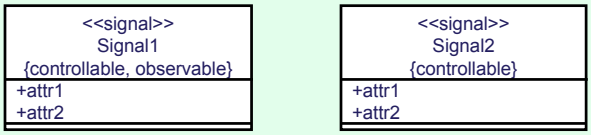
AGEDIS

4. AGEDIS Semantic extension

Stereotype <<signal>> used to specify events and actions in state diagrams:

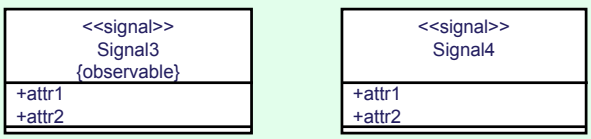
Controllable:

Signals, that can be sent to the system by the tester



Observable:

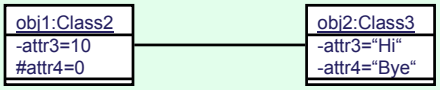
Signals from the system, that can be validated by the tester



AGEDIS

4. Test Directives

<<start>>



Object Diagrams:

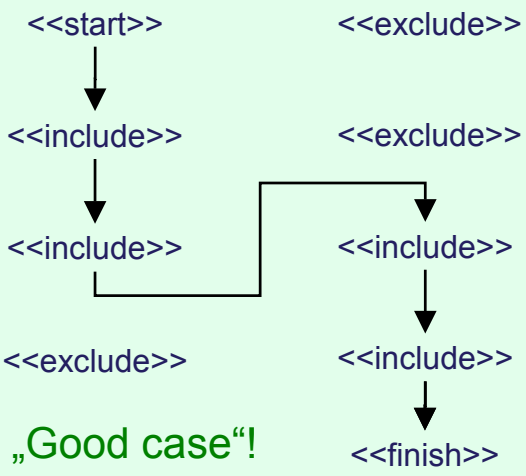
- Define states of the system during test. Attributes are defined.
- **AGEDIS Semantic extension:**
  Special stereotypes are given to diagrams:
  <<start>>, <<finish>>, <<include>>, <exclude>>

AGEDIS

4. Test Directives

What does the AGEDIS test generator do?

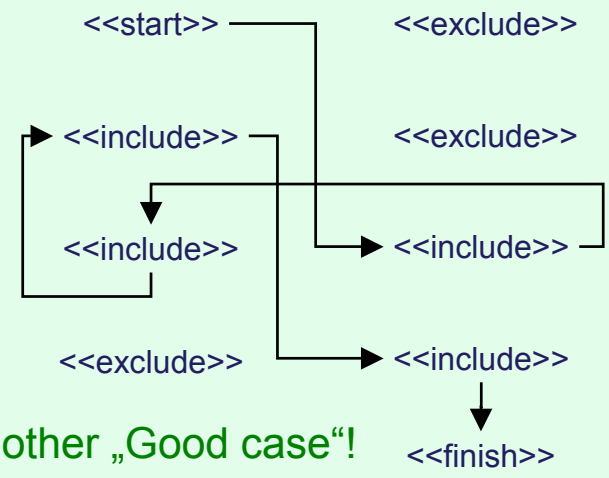
Searches for test cases:



AGEDIS 4. Test Directives

What does the AGEDIS test generator do?

Searches for test cases:

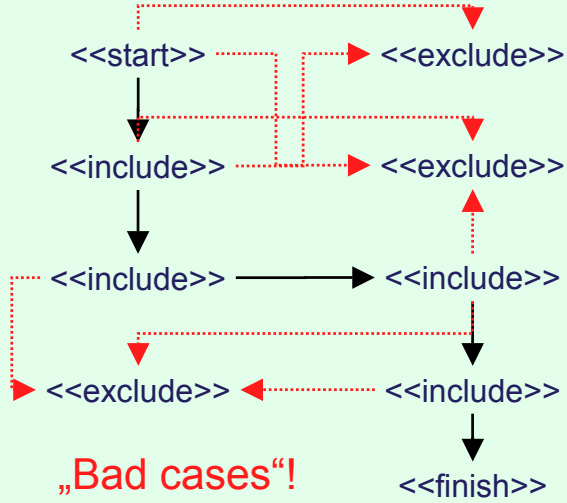


another „Good case“!

AGEDIS 4. Test Directives

What does the AGEDIS test generator do?

Searches for test cases:

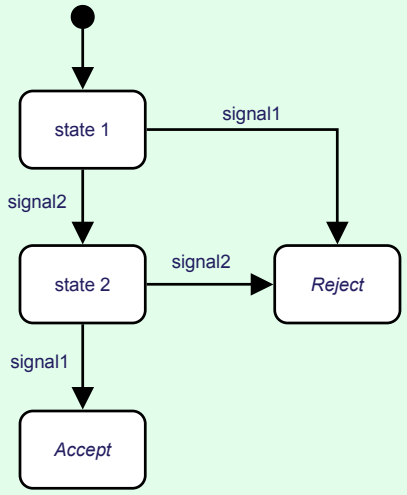


„Bad cases“!

AGEDIS 4. Test Directives

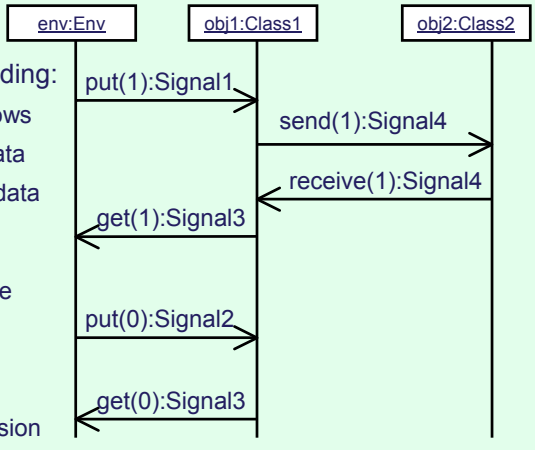
Alternative approach:

- Definition of a system-wide state transition machine
- **AGEDIS semantic extension:** „Accept“ and „Reject“ state



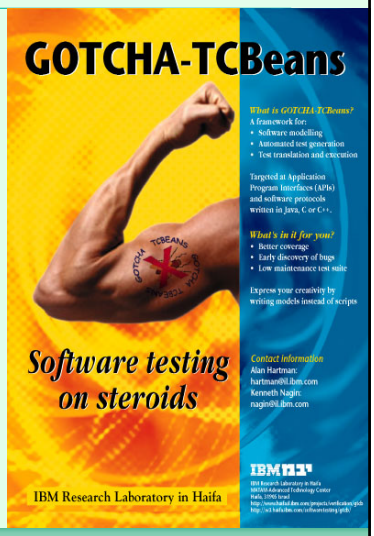
AGEDIS 4. Output from AGEDIS

- Lots of sequences including:
  - Variations in work flows
  - Variations in input data
  - Variations in output data
- Summed up:
  - An abstract test suite
- Output format:
  - XML
  - Details under discussion



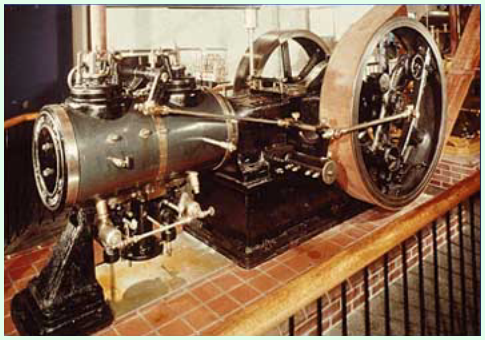
AGEDIS 4. Bringing the test suite to life

- Testbeans
  - GTC (Gotcha - TCBeans from IBM)
  - Meta-Words (WinRunner Script based method from imbus)
  - Your collection of black or white box test functions



AGEDIS 4. Test Execution Engine

- Arbitrary GUI Test tools
  - e.g. WinRunner (Mercury Interactive)



- Protocol interface driver
  - customer tools

AGEDIS 5. Other approaches

- StP/UML (Software through pictures, Aonix)
  - template technology
  - > 50% source code generation
- j-UML (Extension for Java, VTT, Finland)
  - Symbols for special Java features
  - no tool
- xUML (Kennedy Carter)
  - complete Action Language for source code generation
  - Tool bases validation of design through test cases

AGEDIS 6. Conclusion

- AGEDIS will reduce test specification and review effort
- The AGEDIS software derives test cases from UML diagram via test directives
- AGEDIS brings test documentation closer to development departments
- AGEDIS makes testing more complete
- AGEDIS makes testing more fun

Contact

Headquarter Erlangen/Moehrendorf:

imbus AG
Kleinseebacher Strasse 9
91096 Moehrendorf
Germany

Tel: 09131/7518-0
Fax: 09131/7518-50
Email: info@imbus.de

Office Munich:

imbus AG
Unter der Linde 16
80939 München
Germany

Tel: 089/3219909-0
Fax: 089/3219909-50
Email: info@imbus.de

Office Frankfurt:

imbus Rhein-Main GmbH
Zanggasse 6
65719 Hofheim
Germany

Tel: 06192/92192-0
Fax: 06192/92192-50
Email: info@imbus.de

www.agedis.de

AGEDIS

Home

Home
News
Downloads
Contact
Discussion

Partners
Background
Links

Private Area

Find on the AGEDIS newspage and links:
AGEDIS Modelling Language available, more ...
imbus QC-day, more ...
AGEDIS at EuroStar 2001, more ...

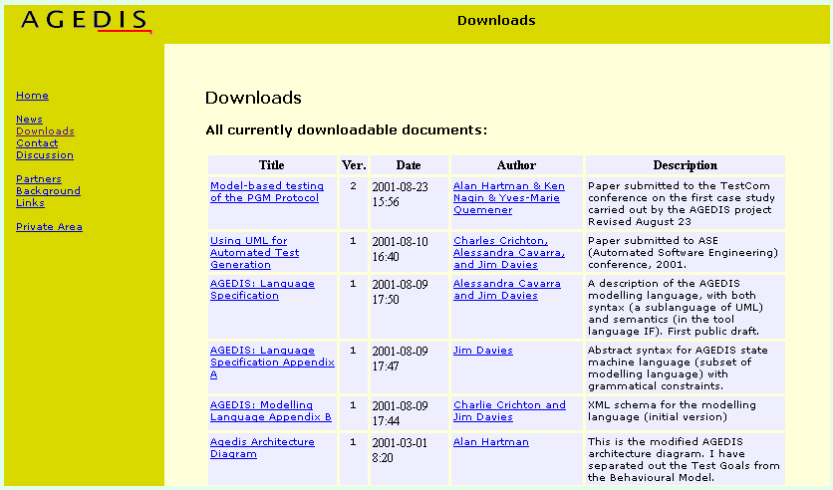
AGEDIS

Automated **G**eneration and **E**xecution of Test Suites for **DI**stributed Component-based **S**oftware

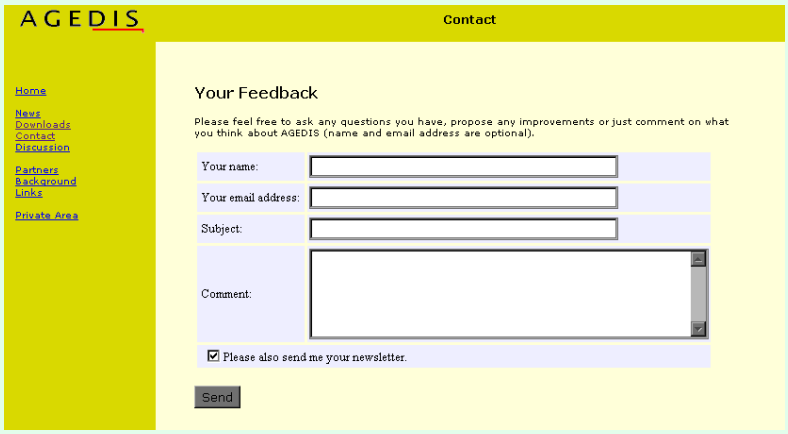
In the last week of October 2000, the European Commission signed an agreement to fund a three-year research project on the automation of software testing. The consortium that will carry out the research and development work consists of seven industrial and academic research centres in Europe and the Middle East. The consortium is headed by the IBM Research Laboratory in Haifa, and contains academic partners at the Oxford University Computing Laboratory, the Verimag laboratory at Universite Joseph Fourier in Grenoble. The major industrial partners are France Telecom R&D, the IBM development Laboratory in Hursley Park (UK), and Intrasoft International, a major software development company with headquarters in Luxembourg and Athens. The company charged with developing commercial applications of the project is a small dynamic software testing company, imbus AG, based in Moehrendorf, Germany.

The aim of the project is to increase the efficiency and competitiveness of the European software industry by automating of software testing, and improving the quality of software while reducing the expense of the testing phase. AGEDIS will achieve this by developing a methodology and tools for the automation of software testing in general, with emphasis

AGEDIS Download further information



AGEDIS Discussion and Newsletter



QWE2002 Paper 6A

Mr. Chris C. Schotanus
(CMG)

Structuring Your Tests in a Component Based Environment

Key Points

- Management: Testing the multi-layer architecture.
- Techniques: Automated Testing of Components without additional tools.
- Management: Component Manager User and/or Supplier?

Presentation Abstract

Component Based Development asks for a special approach to application delivery. In the first project that uses these techniques the developers within the project will have to create all components. But, these components are made available for reuse in other development projects. Usually, a special department will maintain the library of components that are available. It acts as a client towards component development and as a supplier towards the assemblers of the end user application. This department therefore needs to be sure of the functional quality of the components within the library when they supply them to other projects. Therefore, acceptance testing and certification of components is crucial. Automating this test makes it possible to repeat this test as often as necessary without high efforts, but needs special tools when there is no user interface available.

About the Author

Chris C. Schotanus is Principal Consultant for CMG in the Netherlands. He has over 20 years of experience in Information Technology and is strongly involved in the development of CMG's structured testing method TestFrame. He provides management consultancy services on testing and TestFrame to many major clients in the financial world, industry trading and government in Holland, Belgium, France, Nordic countries and the UK.

Quality Week Europe

2001

Structuring your Test in a Component Based Environment

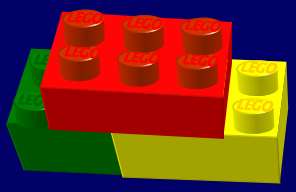
Chris C. Schotanus
Principal Consultant
CMG Division Test Management & Consultancy
chris.schotanus@cmg.nl

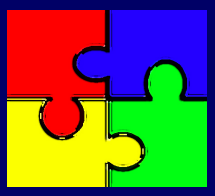
Agenda

- Introduction
- Component Based Development
- Testing in CBD environments
- TestFrame and CBD
- Techniques for Testing CBD

What is a Component (not)?

- The LEGO syndrome
- The JIGSAW metaphor





Possible definitions

A piece of software that has been specifically engineered to be reusable and is therefore only accessible via its interface...

A component is a physical and replaceable part of a system that conforms to and provides the realizations of a set of interfaces

Elementary Characteristics

- Exchangeable with another component of the same specification
- Independent deliverable package of software services
- Executable in one of the de-facto component models

Why Component Based Development?

- Time-to-Market
- Quality-to-Market
- Legacy Integration
- Migration
- Multi-channel Architecture

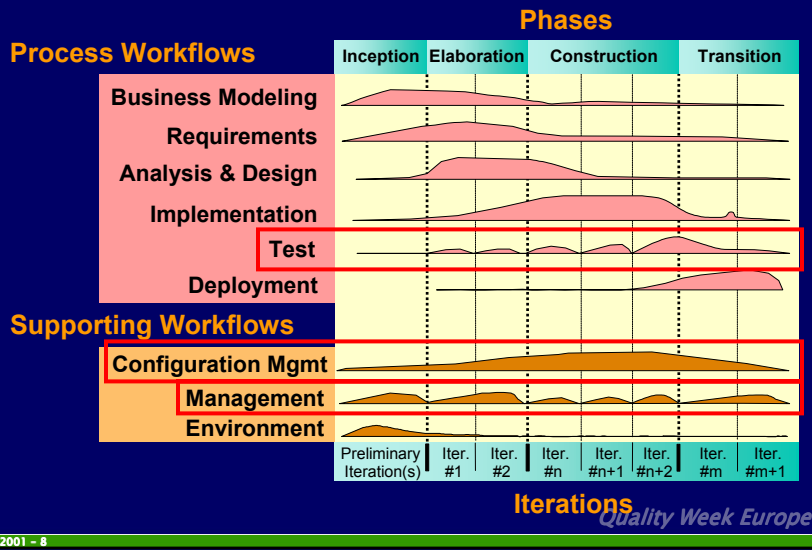
- Cost Reduction: Re-use
  Scalability
  Controllability

- ICT Issues: Flexibility
  Complexity
  Maintainability

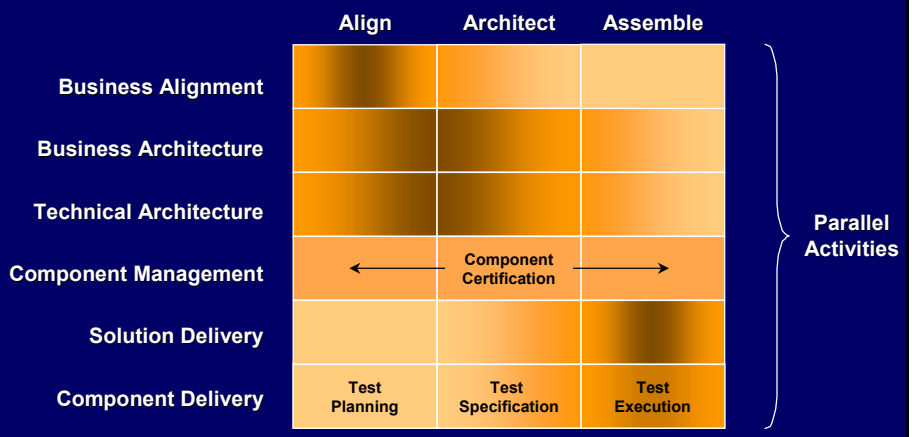
Possible Risks

- Insufficient Quality
- Single Point-of-Failure
- Organisation not Ready

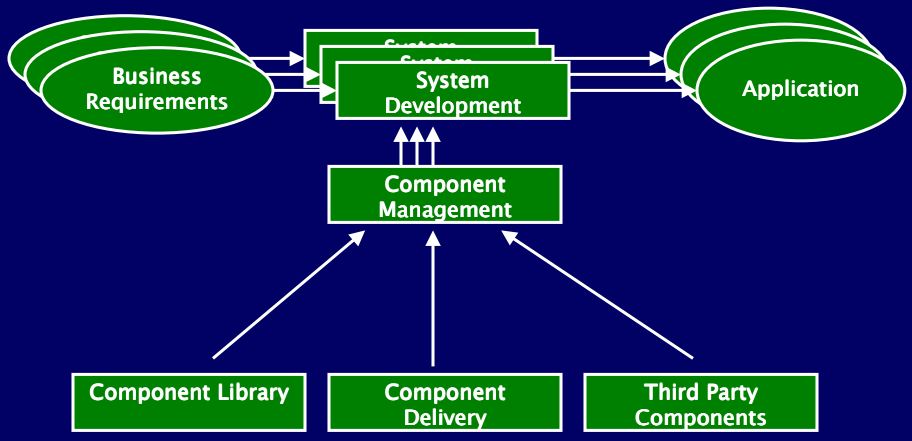
Project Phases Rational Unified Process



Project Phases Select Perspective



Component Management



Component Management

- Is Responsible for
  - Management of component library
  - Defining component standards
  - Selecting third party components
  - Development of new components
  - Acceptance of these components

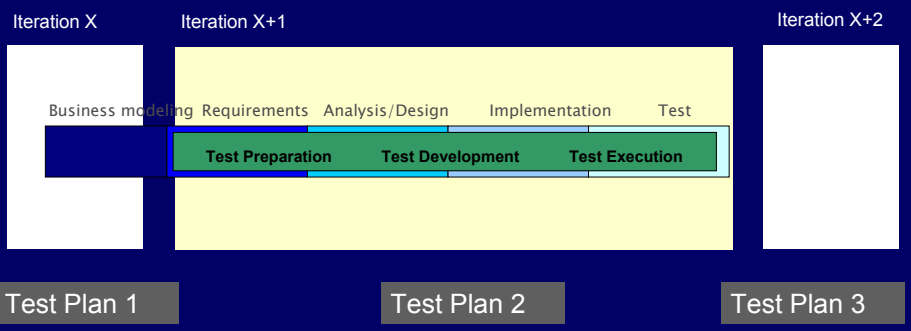
Acceptance and Regression Testing!

Some Test Types used with CBD

• Component Test • Component Integration and System Test	*Component Supplier*
• Component Acceptance Test	*Component Management*
• Assemblage Test (Integration Test)	*System Development*
• Functional Acceptance Test • Production Acceptance Test • …	*Business, Operations, System Management, …*
• ***Regression Test***	*On every level!*

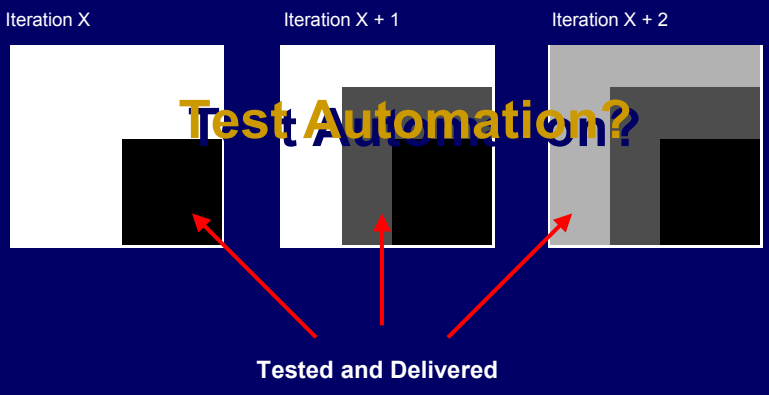
CBD: Develop in Iterations, Test in Iterations

TEST STRATEGY



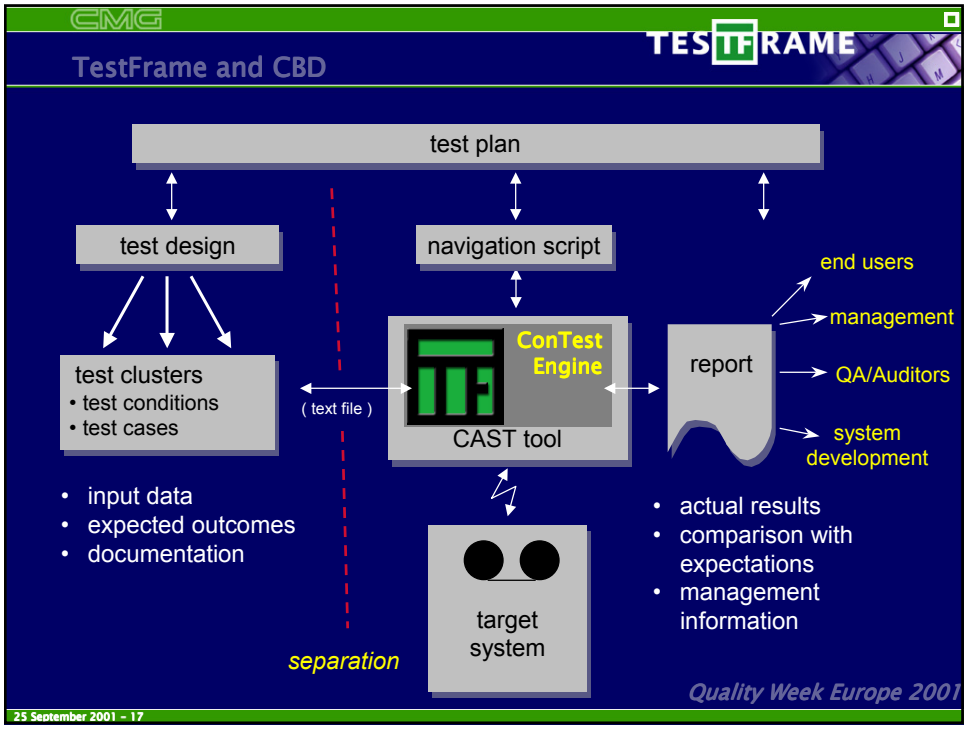
Test Plan 1 | Test Plan 2 | Test Plan 3

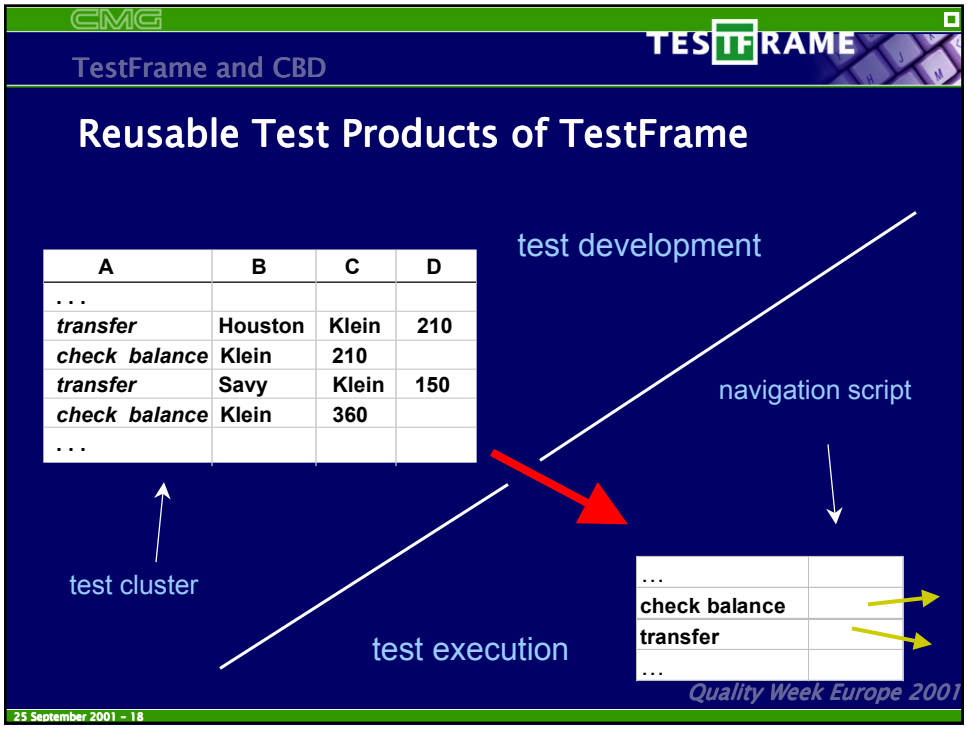
Testing Iterations --> Regression Testing



TestFrame and CBD

- Test Strategy and Planning
- Clustering/Cluster Cards
- Reviews and Inspections
- Clear test reports
- Regression tests
- Parallel development and testing of components





A	B	C	D
. . .			
transfer	Houston	Klein	210
check balance	Klein	210	
transfer	Savy	Klein	150
check balance	Klein	360	
. . .			

Proven Advantages of TestFrame

- Accessible and maintainable test products
- Shorter execution time (Automation)
- Results earlier available due to early start
- Less need of scarce resources (Business Experts, Operations etc)
- Improved control of test process
- Easy execution of regression test

Techniques for Testing CBD

- Analysis
  - Use-case as a source to Test Condition Development
  - Contract specifications as a basis to Unit/Component test
- Navigation/Execution
  - No User interface: Testing directly via the interface/middleware
  - Simulation through stubs en drivers
  - Functional testing (F.A.T./U.A.T.) via User Interface using existing tools

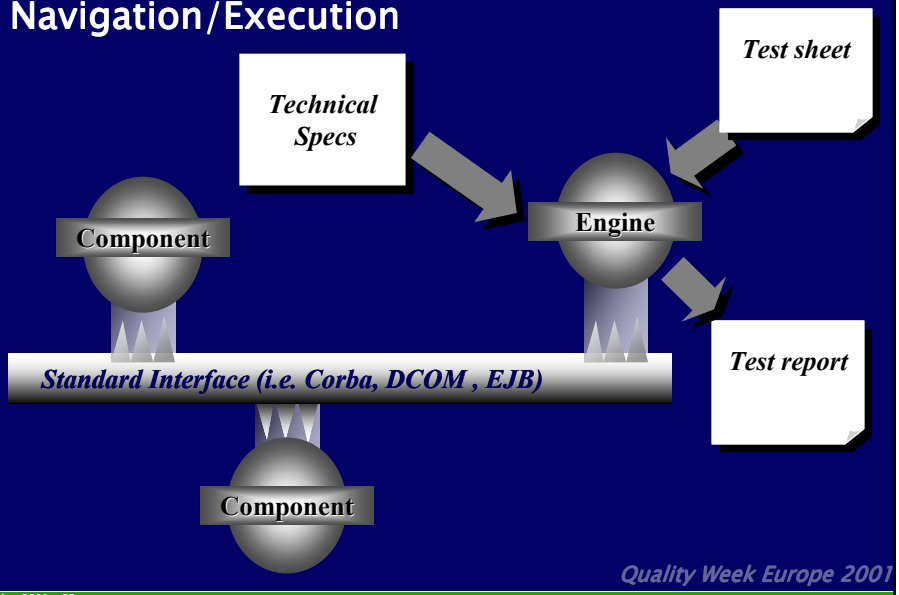
Use Cases & Testing

- Priority of use cases is important
- Planning: Which iteration, which use cases?
- Which parts of the use case in which iteration?
- Determine the depth of the use cases (requirements review)
- Extra depth in test analysis through activity diagrams
- Test analysis per use case (scenario's within use cases)
- SOAP's cover more than one use case (business functionalist)
- Additionally test of Non functional requirements (performance, load, stress, security,….)

Component Test

- Based on the Interface Specifications (contract) of the Component (*Black Box Component Test*)
- Based on the analysis and design model (Collaboration, Class, State and/or Sequence diagram) (*Integration Test*)
- Build Stubs & Drivers if necessary
- Test environment and architecture conform production as far as possible.
- Possible use of special Component Test Tool

Navigation/Execution



Some Important Issues

- Start as early as possible
- Know the contents of your DB
- Possible need for Stubs and drivers
- Prevent "dead-lock" situations
- Test on different platforms (1-tier vs multi-tier)

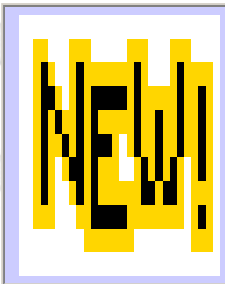
Conclusions

- From end-user point of view not different from any other test

But pay extra attention to

- Static test of design-documents
- Component Management (Requirements and Acceptance)
- Quality of Components (Re-use)
- Regression Testing

More information on www.TestFrame.com



QWE2002 Paper 7A

Ms. Nadine Pelicaen
(ps_testware)

Performance Testing: "Step On It"

Key Points

- Performance testing for websites and client/server systems.
- Software should perform within standard under varying loads.
- Ways to detect the bottleneck in your system.

Presentation Abstract

The quality attribute performance is often overlooked in the design phase of a system, in which case client and server response times can turn out to be below expectations and all kinds of resource problems pop up. It can be critical to know whether your multi-client system is performing within well-defined standards under varying loads. Moreover, how is your application holding when running under extreme conditions? In this paper we talk about the different types of performance testing and the role they play in risk management. Certain performance testing techniques allow you to detect the bottleneck in your system, in this case the area which will break first when put under stress. Different techniques and automation tools are discussed for structured performance testing of websites where load generators are used for automating user actions. But we also talk about performance test automation for client/server systems with multi-tier architecture.

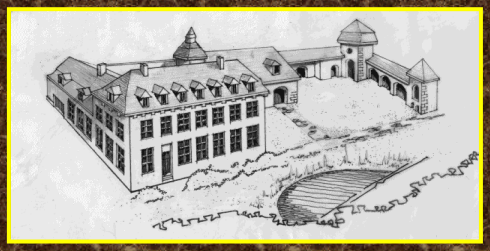
About the Author

Nadine Pelicaen is active as a Test Consultant for structured software testing at ps_testware. She has experience with performing Test Assessments for large projects and she is involved in doing research for the ps_testware competence center of test methodology. Nadine has a university degree in informatics and more than 12 years experience in the IT sector. She has worked in different functions in technical IT environments: software developer, QA engineer and manager of a software department.

Copyright © 2001 ps_testware – Quality Week Europe

Performance testing: "step on it"

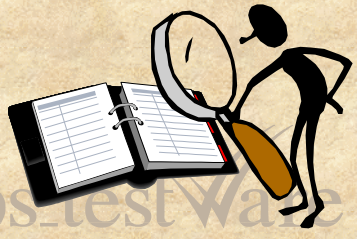
Quality Week Europe 2002



Nadine Pelicaen
ps_testware

Agenda

- Introduction
- Why? Role in overall business process
- Who? The people involved
- What? The measurements
- Way? Web Sites <-> Client/Server
- When? Incorporation in Test Strategy
- With? The tools
- Where? The test environment
- Conclusions



Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 2

ps_testware
Software Testing Services

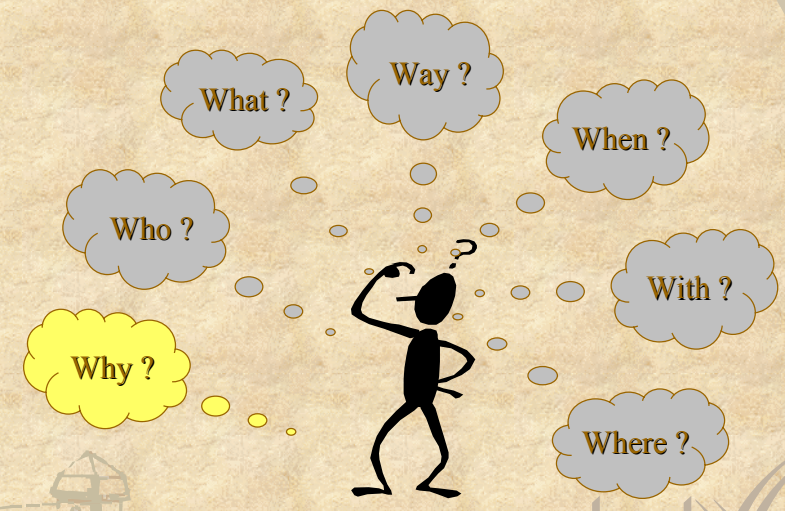
Introduction

- **Business critical knowledge**
- **Detecting bottlenecks**

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 3

ps_testware
Software Testing Services

Why? – Role in overall business process



Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 4

ps_testware
Software Testing Services

Why? – Role in business process (1/2)

- **General business benefits**
  - Live up to expectations
    - » Quality of Service
  - Business growth scenarios
    - » Budget adjustments
- **Risk management**
  - Costs of downtime
    - » Lost revenue
    - » Potential loss of future business

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 5

ps_testware Software Testing Services

Why? – Role in business process (2/2)

- **Risk management**
  - Web sites : public event
  - Carrying the risk through the project
    - » Not a one time effort
    - » Back to the drawing board
  - ROI
    - » Investment is high
    - » Calculate the risk
- **Service Level Agreements**
  - » Determining SLA's
  - » Maintenance (scalability)

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 6

ps_testware Software Testing Services

Who? – The people involved

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 7

ps_testware
Software Testing Services

Who? – The people involved (1/2)

- **Interdisciplinary team**
  - **Project leader**
  - **IT**
  - **DBA**
  - **Developers**
  - **Tool specialist**
  - **Test expert**
  - **Analyst**
  - **Configuration manager**

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 8

ps_testware
Software Testing Services

Who? – The people involved (2/2)

- Get help from
  - Marketing
  - Real users
- Different departments involved
  - Positive: Bridging the gap
  - Negative: Fingerpointing

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 9

ps_testware
Software Testing Services

What? – The measurements

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 10

ps_testware
Software Testing Services

What? – Terminology

Load testing	• Particular load • Observe behaviour • Expected peak conditions • Expected response time
Stress testing	• Extreme conditions • Breaking points • Deterioration effects
Scalability testing	• Increased load • Capacity planning
Endurance testing	• Long term load testing • Memory leaks…

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 11

ps_testware
Software Testing Services

What? – Measurements (1/2)

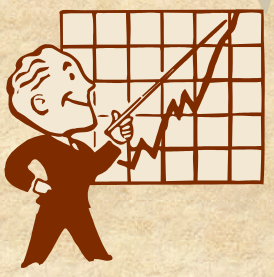
- Scientific value
- Reporting measurements
- Things like:
  - Throughput
  - Response time
  - Length of scenario
  - Nr users on moment of degradation starts
  - Errors reported
  - Nr hits, failed hits, page views
  - Page download speed

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 12

ps_testware
Software Testing Services

What? – Measurements (2/2)

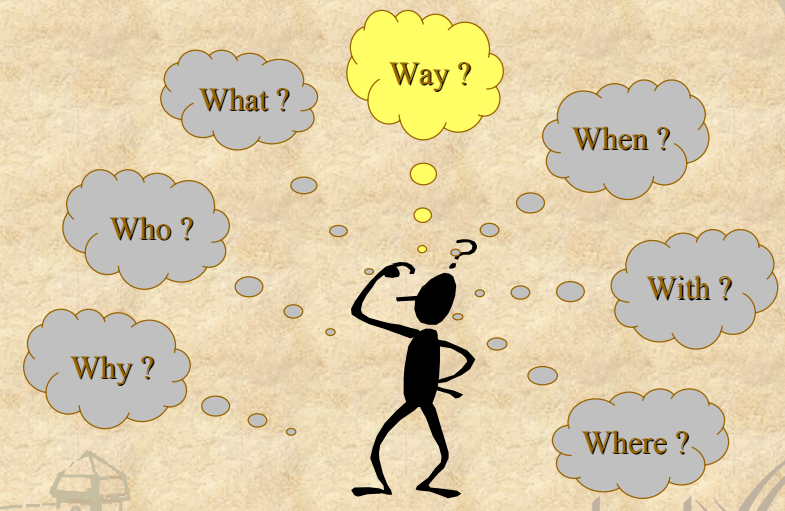
- **Harder to measure:**
  - Accessibility
  - Session independence
- **Resouce information :**
  - Memory
  - CPU
  - Database connections
  - Network saturation
  - Bandwidth information
  - Number of users in queues
  - …



Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 13

ps_testWare
Software Testing Services

Way? – Web sites vs. Client/Server



Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 14

ps_testWare
Software Testing Services

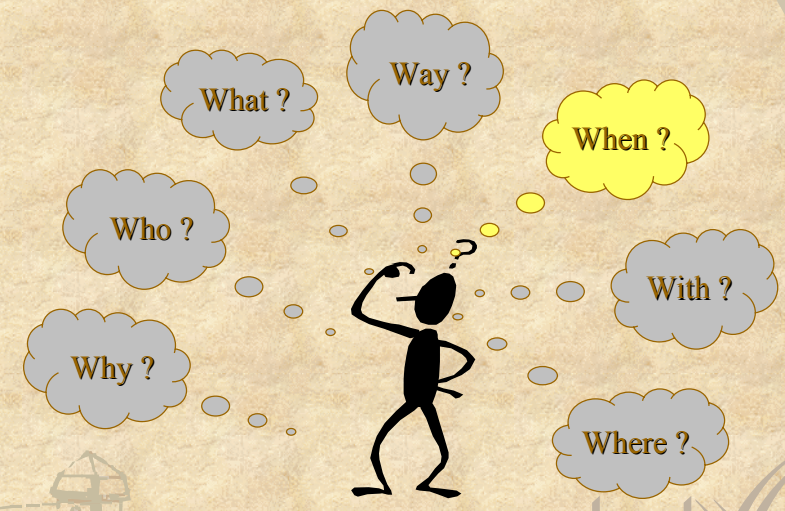
Way? – Web sites vs. Client/Server

Web Sites	Client/Server
• Security layer • Different times and locations • Dynamic vs. Static pages • Unpredictable audience • Higher demands • Higher risk • Unpredictable transaction volumes	• Overhead because of hopping • Distributed processing • Exotic OS's and protocols

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 15

ps_testware
Software Testing Services

When? – Incorporation in test strategy

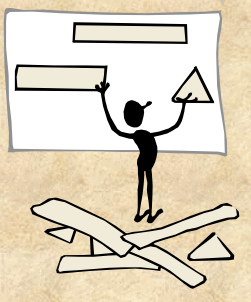


Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 16

ps_testware
Software Testing Services

When? – Incorporation in test strategy (1/2)

- Requirements
  - Nr of users, transactions, …
- Design
  - Quality attributes
  - Scalability
  - Risk
- Development
  - Performance engineering
- Test preparation
  - Coherent Test Plan in a Structured Methodology
  - Prototyping
  - Design to test the whole architecture

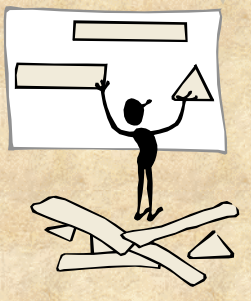


Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 17

ps_testware Software Testing Services

When? – Incorporation in test strategy (2/2)

- Test execution
  - Iterative approach
  - Small change, big impact
  - Establish benchmarks
- Maintenance
  - Needed:
    - Automated regression tests
    - Test data
    - Test environment
    - Benchmarks



Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 18

ps_testware Software Testing Services

With? – The tools

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 19

ps_testWare
Software Testing Services

With? – The tools (1/2)

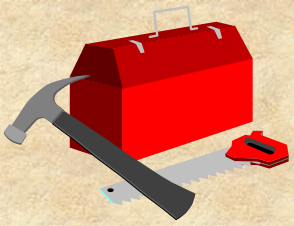
- **Paradigm:**
  - Master controller
  - Agents running virtual clients through scripts
- **Tool selection:**
  - Abilities vs. needs (check for incompatibilities)
  - Make sure the tool does not become a bottleneck
  - Costs: licenses, training, scripting,…
  - Intrusiveness
  - Facilities for analysis phase
  - Separate measurement workstation
  - For Web site testing : Emulate browser features

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 20

ps_testWare
Software Testing Services

With? – The tools (2/2)

- Scripting:
  - Recording + Parameterization
  - Realistic delays
  - All possible transactions
  - Repeatable and scalable
  - Mix of test data
  - Unambiguous pass/fail criteria
- Analysis
  - Consolidation of measurements
  - Comparing to the benchmarks
  - Pinpointing bottlenecks
- Integration
  - Test plan + test designs



Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 21

ps_testware Software Testing Services

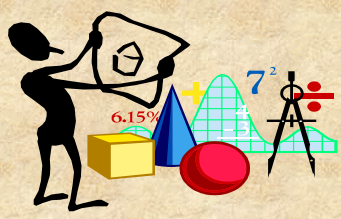
Where? – The test environment

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 22

ps_testware Software Testing Services

Where? – The test environment

- Portray production environment
  - Expensive
- Emulation + extrapolation
- Using the real production environment
  - Downtime
  - Tweak the environment
  - Mutual influence



Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 23

ps_testware
Software Testing Services

Conclusions

- Science and art
- Crucial for business
- Right mix of tool, environment, test strategy, people
- Don't give in to the chaos

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 24

ps_testware
Software Testing Services

ps_testWare
Software Testing Services

Questions

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 25

ps_testWare
Software Testing Services

Tiensesteenweg 343
B-3010 Leuven
Tel.: +32 (16) 35.93.80
Fax: +32 (16) 35.93.88
e-mail: info@pstestware.com
http://www.pstestware.com

Copyright © 2001 ps_testware – Quality Week Europe - Nadine Pelicaen — 26

QWE2002 Paper 8A

Mr. James Lyndsay
(Workroom Productions)

The Importance of Data in Functional Testing

Key Points

- Plan the data for maintenance and flexibility
- Know your data, and make its structure and content transparent
- Use the data to improve understanding throughout testing and the business

Presentation Abstract

Good data can help improve functional testing. Test data can be structured to improve understanding and testability. Its contents, correctly chosen, can reduce maintenance effort and allow flexibility. Preparation of the data can focus the business where requirements are vague.

About the Author

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 telecomms to the web, and pays keen attention to the way that his clients focus is shifting away from functional testing.

The Importance of Data in Functional Testing

James Lyndsay
Workroom Productions

Data is important to functional testing.

A system is programmed by its data.

Functional testing suffers if data is poor.

Good data is vital to reliable test results.

Good data can help keep testing on schedule.

© Workroom Productions 2001
www.workroom-productions.com

Problems caused by poor data

The following problems can be caused by poor data;

- Unreliable test results
- Degradation of test data over time
- Increased test maintenance budget
- Reduced flexibility in test execution
- Obscure results and bug reports
- Larger proportion of problems can be traced to poor data
- Less time spent hunting bugs
- Confusion between developers, testers and business
- Requirements problems can be hidden in inadequate data
- Simpler to make test mistakes
- Unwieldy volumes of data
- Business data not representatively tested
- Inability to spot data corruption caused by bugs
- Poor database integrity

© Workroom Productions 2001
www.workroom-productions.com

Topics

1) Recognising types of data
2) Organising the data
3) Involving the Business
4) Data Load and Maintenance
5) Testing the data

© Workroom Productions 2001
www.workroom-productions.com

Types of Data

Environmental data

tells the system about its technical environment.

Setup data

tells the system about the business rules.

Input data

information input by day-to-day system functions.

- Fixed input data is available at the start of the test.
- Consumable input data forms the test input.

© Workroom Productions 2001
www.workroom-productions.com

Organising the Data

Good data increases data reliability, reduces data maintenance time and can help improve the test process.

Good data assists testing, rather than hinders it.

Good data is based on;

1) Permutations
2) Partitioning
3) Clarity

© Workroom Productions 2001
www.workroom-productions.com

Data Permutations 1

Permutations are familiar from test planning

Utility Customer Care / Billing Example:

A customer can have one of three products (1,2,3). They may be billed Monthly or Quarterly, be High or Low value, and their last bill was either Paid or Unpaid. There are 3x2x2x2 = 24 combinations - and so the number of possible permutations climbs rapidly as system complexity increases.

By requiring that the list holds not all possible combinations, but all possible pairs, the list can be reduced. All possible pairs; M1, M2, M3. Q1, Q2, Q3. H1, H2, H3. L1, L2, L3. P1, P2, P3. U1, U2, U3. MH, ML, QH, QL, MP, MU, QP, QU. HP, LP, HU, LU.

The following six permutations contain all the pairs;

Customer	Account	Product	Care	Bill
1	M	1	H	P
2	M	2	L	P
3	M	3	H	U
4	Q	1	L	U
5	Q	2	H	U
6	Q	3	L	P

© Workroom Productions 2001
www.workroom-productions.com

Data Permutations 2

Permutation is appropriate when:

- Fixed input data consists of many rows
- Fields are independent
- You want to do many tests without loading / you do not load fixed input data for each test.

Permutation helps because:

- Achieves good test coverage without having to construct massive datasets
- Can perform investigative testing without having to set up more data
- Can be used to test other data - particularly setup data
- Permutation is familiar from test planning.
- Reduces the impact of functional/database changes

© Workroom Productions 2001
www.workroom-productions.com

Partitioned data

Data load/reload can be inconvenient. Data can be partitioned into:

1) Safe area

   Used for enquiry tests, usability tests etc. No test changes the data, the area can be trusted. Many testers can use simultaneously

2) Change area

   Used for tests which update/change data. Data must be reset or reloaded after testing. Used by one tester at a time.

3) Scratch area

   Used for investigative update tests and those which have unusual requirements. Existing data cannot be trusted. Used at own risk!

Data can be partitioned by machine / database / instance. Can also be partitioned by disciplined use of text / value fields.

© Workroom Productions 2001
www.workroom-productions.com

Clarity

Developers and the Business don't need to understand test data / data requirements - so some of them won't. We can make our data clearer by using available free text fields;

Customer Name	Account	Product	Care	Bill
HP1 Monthly	M	1	H	P
LP2 Monthly	M	2	L	P
HU3 Monthly	M	3	H	U
LU1 Quarterly	Q	1	L	U
HU2 Quarterly	Q	2	H	U
LP3 Quarterly	Q	3	L	P

Clarity helps because:

- Improves communication within and outside the team
- Reduces test errors caused by using the wrong data
- Helps when checking data after input
- Helps in selecting data for investigative tests

© Workroom Productions 2001
www.workroom-productions.com

Data and the Business

'The Business' is good at looking at data;

- Easier to understand than tests
- Can be compared with existing systems

Advantages

- Helps focus when requirements are vague
- Helps UAT
- Increases trust and understanding
- Helps early user identification of problems

Disadvantages

- Data creep
- Vague requirements can lead to vague data
- Incomplete data can lead to incomplete testing

© Workroom Productions 2001
www.workroom-productions.com

Ways of loading data

- Using the system you're trying to test
  - Manually entered
  - Automated tool, making keystrokes into application
- Using a data load tool
  - All new data, created for testing
  - Old data, selected for testing / filtered and migrated
  - Complete set, migrated and loaded, identical but for personal details
- Not loaded
  - Already set up for testing / Left in the system
  - You are working on the live system

© Workroom Productions 2001
www.workroom-productions.com

Frequency of data load

- At the start of testing
- With each release
- First thing Monday
- Whenever I want
- Before every test

NB: Straw poll of 'When does the data usually get loaded' also came back with the following answers;

- Before I ever got involved
- The developers left it there
- The last testers / tests left it
- Whenever we get enough time
- After we've found out what shape the database is
- When we know what it means

© Workroom Productions 2001
www.workroom-productions.com

Data Maintenance

When and why?

Replacing consumed data

Repairing broken data

Responding to change - database schema, code, requirements

New test requirements

Problems:

Sizeable task - can be a substantial fraction of overall test maintenance

Prone to error

Performed by more than one group

© Workroom Productions 2001
www.workroom-productions.com

Solutions to Loading and Maintenance headaches

1) Automate data load and maintenance where possible
2) Control / measure data change
3) Recognise and prepare for problems
4) Use good data

© Workroom Productions 2001
www.workroom-productions.com

Testing the data

Systems are programmed by their data . . .

If the functionality of your system is at all affected by the setup data;

TEST THE DATA

1) Does the current setup data affect the application appropriately?
2) Will changes made to the setup data have the desired effect?

To incorporate into functional testing, you need;

- organised data
- good load/maintenance
- business involvement

© Workroom Productions 2001
www.workroom-productions.com

Testing the data 2

Use a known dataset to:

- Show functionality correctly applied
- Assess effect of changes in setup data

Aims:

- Improve overall testing
- Help get the data right before live operation
- Increased confidence in re-configuration
- Help pinpoint bugs in live operation

© Workroom Productions 2001
www.workroom-productions.com

Further data issues

Operational Profiles

Non-functional testing

Data verification

© Workroom Productions 2001
www.workroom-productions.com

Topics: conclusion

1) Recognising types of data
   - Environmental Data
   - Setup Data
   - Input Data - Fixed and Consumable
2) Avoiding common problems
   - Recognise the problems
   - Use good data
   - Involve the Business
   - Automate loading and maintenance
   - Test your data

© Workroom Productions 2001
www.workroom-productions.com

Summary

- Plan the data for maintenance and flexibility
- Know your data, and make its structure and content transparent
- Use the data to improve understanding throughout testing and the business
- Test setup data as you would test functionality

James Lyndsay
Workroom Productions
www.workroom-productions.com
jdl@workroom-productions.com

© Workroom Productions 2001
www.workroom-productions.com

The Importance of Data in Functional Testing

Author: James Lyndsay
Version: 1.0

Author James Lyndsay, Workroom Productions Ltd.
Biography 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.
Abstract A system is programmed by its data. Functional testing can suffer if data is poor, and good data can help improve functional testing. Good test data can be structured to improve understanding and testability. Its contents, correctly chosen, can reduce maintenance effort and allow flexibility. Preparation of the data can help to focus the business where requirements are vague.
Keywords Data, Functional testing

© Workroom Productions 2001

The Roles of Data in Functional Testing

Testing consumes and produces large amounts of data. Data describes the initial conditions for a test, forms the input, is the medium through which the tester influences the software. Data is manipulated, extrapolated, summarised and referenced by the functionality under test, which finally spews forth yet more data to be checked against expectations. Data is a crucial part of most functional testing.

This paper sets out to illustrate some of the ways that data can influence the test process, and will show that testing can be improved by a careful choice of input data. In doing this, the paper will concentrate most on data-heavy applications; those which use databases or are heavily influenced by the data they hold. The paper will focus on input data, rather than output data or the transitional states the data passes through during processing, as input data has the greatest influence on functional testing and is the simplest to manipulate. The paper will not consider areas where data is important to non-functional testing, such as operational profiles, massive datasets and environmental tuning.

A SYSTEM IS PROGRAMMED BY ITS DATA

Many modern systems allow tremendous flexibility in the way their basic functionality can be used. Configuration data can dictate control flow, data manipulation, presentation and user interface. A system can be configured to fit several business models, work (almost) seamlessly with a variety of co-operative systems and provide tailored experiences to a host of different users. A business may look to an application's configurability to allow them to keep up with the market without being slowed by the development process, an individual may look for a personalised experience from commonly-available software.

FUNCTIONAL TESTING SUFFERS IF DATA IS POOR

Tests with poor data may not describe the business model effectively, they may be hard to maintain, or require lengthy and difficult setup. They may obscure problems or avoid them altogether. Poor data tends to result in poor tests, that take longer to execute.

GOOD DATA IS VITAL TO RELIABLE TEST RESULTS

An important goal of functional testing is to allow the test to be repeated with the same result, and varied to allow diagnosis. Without this, it is hard to communicate problems to coders, and it can become difficult to have confidence in the QA team's results, whether they are good or bad. Good data allows diagnosis, effective reporting, and allows tests to be repeated with confidence,.

GOOD DATA CAN HELP TESTING STAY ON SCHEDULE

An easily comprehensible and well-understood dataset is a tool to help communication. Good data can greatly assist in speedy diagnosis and rapid re-testing. Regression testing and automated test maintenance can be made speedier and easier by using good data, while an elegantly-chosen dataset can often allow new tests without the overhead of new data.

Problems which can be caused by poor data

Most testers are familiar with the problems that can be caused by poor data. The following list details the most common problems familiar to the author. Most projects experience these problems at some stage - recognising them early can allow their effects to be mitigated.

- Unreliable test results.
  Running the same test twice produces inconsistent results. This can be a symptom of an uncontrolled environment, unrecognised database corruption, or of a failure to recognise all the data that is influential on the system.

- Degradation of test data over time.
  Program faults can introduce inconsistency or corruption into a database. If not spotted at the time of generation, they can cause hard-to-diagnose failures that may be apparently unrelated to the original fault. Restoring the data to a clean set gets rid of the symptom, but the original fault is

undiagnosed and can carry on into live operation and perhaps future releases. Furthermore, as the data is restored, evidence of the fault is lost.

- Increased test maintenance cost
  If each test has its own data, the cost of test maintenance is correspondingly increased. If that data is itself hard to understand or manipulate, the cost increases further.

- Reduced flexibility in test execution
  If datasets are large or hard to set up, some tests may be excluded from a test run. If the datasets are poorly constructed, it may not be time-effective to construct further data to support investigatory tests.

- Obscure results and bug reports
  Without clearly comprehensible data, testers stand a greater chance of missing important diagnostic features of a failure, or indeed of missing the failure entirely. Most reports make reference to the input data and the actual and expected results. Poor data can make these reports hard to understand.

- Larger proportion of problems can be traced to poor data
  A proportion of all failures logged will be found, after further analysis, not to be faults at all. Data can play a significant role in these failures. Poor data will cause more of these problems.

- Less time spent hunting bugs
  The more time spent doing unproductive testing or ineffective test maintenance, the less time spent testing.

- Confusion between developers, testers and business
  Each of these groups has different data requirements. A failure to understand each others data can lead to ongoing confusion.

- Requirements problems can be hidden in inadequate data
  It is important to consider inputs and outputs of a process for requirements modelling. Inadequate data can lead to ambiguous or incomplete requirements.

- Simpler to make test mistakes
  Everybody makes mistakes. Confusing or over-large datasets can make data selection mistakes more common.

- Unwieldy volumes of data
  Small datasets can be manipulated more easily than large datasets. A few datasets are easier to manage than many datasets.

- Business data not representatively tested
  Test requirements, particularly in configuration data, often don't reflect the way the system will be used in practice. While this may arguably lead to broad testing for a variety of purposes, it can be hard for the business or the end users to feel confidence in the test effort if they feel distanced from it.

- Inability to spot data corruption caused by bugs
  A few well-known datasets can be more easily be checked than a large number of complex datasets, and may lend themselves to automated testing / sanity checks. A readily understandable dataset can allow straightforward diagnosis; a complex dataset will positively hinder diagnosis.

- Poor database/environment integrity
  If a large number of testers, or tests, share the same dataset, they can influence and corrupt each others results as they change the data in the system. This can not only cause false results, but can lead to database integrity problems and data corruption. This can make portions of the application untestable for many testers simultaneously.

Classification of Data Types

In the process of testing a system, many references are made to "The Data" or "Data Problems". Although it is perhaps simpler to discuss data in these terms, it is useful to be able to classify the data according to the way it is used. The following broad categories allow data to be handled and discussed more easily.

Environmental data

Environmental data tells the system about its technical environment. It includes communications addresses, directory trees and paths and environmental variables. The current date and time can be seen as environmental data.

Setup data

Setup data tells the system about the business rules. It might include a cross reference between country and delivery cost or method, or methods of debt collection from different kinds of customers. Typically, setup data causes different functionality to apply to otherwise similar data. With an effective approach to setup data, business can offer new intangible products without developing new functionality - as can be seen in the mobile phone industry, where new billing products are supported and indeed created by additions to the setup data.

Input data

Input data is the information input by day-to-day system functions. Accounts, products, orders, actions, documents can all be input data. For the purposes of testing, it is useful to split the categorisation once more:

FIXED INPUT DATA

Fixed input data is available before the start of the test, and can be seen as part of the test conditions.

CONSUMABLE INPUT DATA

Consumable input data forms the test input

It can also be helpful to qualify data after the system has started to use it;

Transitional data

Transitional data is data that exists only within the program, during processing of input data. Transitional data is not seen outside the system (arguably, test handles and instrumentation make it output data), but its state can be inferred from actions that the system has taken. Typically held in internal system variables, it is temporary and is lost at the end of processing.

Output data

Output data is all the data that a system outputs as a result of processing input data and events. It generally has a correspondence with the input data (cf. Jackson's Structured Programming methodology), and includes not only files, transmissions, reports and database updates, but can also include test measurements. A subset of the output data is generally compared with the expected results at the end of test execution. As such, it does not directly influence the quality of the tests.

Organising the data

A key part of any approach to data is the way the data is organised; the way it is chosen and described, influenced by the uses that are planned for it. A good approach increases data reliability, reduces data maintenance time and can help improve the test process.

Good data assists testing, rather than hinders it.

Permutations

Most testers are familiar with the concept of permutation; generating tests so that all possible permutations of inputs are tested. Most are also familiar with the ways in which this generally vast set can be cut down.

Pairwise, or combinatorial testing (see Cohen, Dalal, Parelius, Patton: The Combinatorial Design Approach to Automatic Test Generation and Tai, Lei: A Test Generation Strategy for Pairwise Testing) addresses this problem by generating a set of tests that allow all possible *pairs* of combinations to be tested. Typically, for non-trivial sets, this produces a far smaller set of tests than the brute-force approach for all permutations, The same techniques can be applied to test data; the test data can contain all possible pairs of permutations in a far smaller set than that which contains all possible permutations. This allows a small, easy to handle dataset - which also allows a wide range of tests.

EXAMPLE: UTILITY CUSTOMER CARE / BILLING SYSTEM

In this example, we look at the permutations possible in a portion of the fixed input data for a simple billing system. We're all familiar with the output of such a system; each customer is billed periodically for usage, the value of their bill being affected by a tariff, and with various other customer attributes kept in the system. Each account in our hypothetical system can have the following characteristics;

A customer can have one of three tariffs (labelled 1,2,3 below). They may be billed Monthly or Quarterly, be High or Low value, and their last bill was either Paid or Unpaid.

Characteristic	Account	Tariff	Care
Tariff	1	2	3
Billing period		M	Q
Customer Value		H	L
Last Bill		P	U

There are 3x2x2x2 = 24 combinations - and as the system complexity increases, so the number of possible permutations climbs ever more rapidly.

By requiring that the list holds not all possible combinations, but all possible pairs, the list can be reduced. The following six permutations contain all the pairs;

All possible pairs; M1, M2, M3. Q1, Q2, Q3. H1, H2, H3. L1, L2, L3. P1, P2, P3. U1, U2, U3. MH, ML, QH, QL, MP, MU, QP, QU. HP, LP, HU, LU.

Customer	Account	Tariff	Care	Bill
1	M	1	H	P
2	M	2	L	P
3	M	3	H	U
4	Q	1	L	U
5	Q	2	H	U
6	Q	3	L	P

This small, and easy to manipulate dataset is capable of supporting many tests. It allows complete pairwise coverage, and so is comprehensive enough to allow a great many new, ad-hoc, or diagnostic tests. Database changes will affect it, but the data maintenance required will be greatly lessened by the small size of the dataset and the amount of reuse it allows. Finally, this method of working with fixed input data can help greatly in testing the setup data.

This method is most appropriate when used, as above, on fixed input data. It is most effective when the following conditions are satisfied. Fortunately, these criteria apply to many traditional database-based systems:

- Fixed input data consists of many rows
- Fields are independent
- You want to do many tests without loading / you do not load fixed input data for each test.

To sum up, permutation helps because:

- Permutation is familiar from test planning.
- Achieves good test coverage without having to construct massive datasets
- Can perform investigative testing without having to set up more data
- Reduces the impact of functional/database changes
- Can be used to test other data - particularly setup data

Partitioning

Partitions allow data access to be controlled, reducing uncontrolled changes in the data. Partitions can be used independently; data use in one area will have no effect on the results of tests in another. Data can be safely and effectively partitioned by machine / database / application instance, although this partitioning can introduce configuration management problems in software version, machine setup, environmental data and data load/reload. A useful and basic way to start with partitions is to set up, not a single environment for each test or tester, but to set up three shared by many users, so allowing different kinds of data use. These three have the following characteristics:

Safe area	· Used for enquiry tests, usability tests etc. · No test changes the data, so the area can be trusted. · Many testers can use simultaneously
Change area	· Used for tests which update/change data. · Data must be reset or reloaded after testing. · Used by one test/tester at a time.
Scratch area	· Used for investigative update tests and those which have unusual requirements. · Existing data cannot be trusted. · Used at tester's own risk!

Testing rarely has the luxury of completely separate environments for each test and each tester. Controlling data, and the access to data, in a system can be fraught. Many different stakeholders have different requirements of the data, but a common requirement is that of exclusive use.

While the impact of this requirement should not be underestimated, a number of stakeholders may be able to work with the same environmental data, and to a lesser extent, setup data - and their work may not need to change the environmental or setup data. The test strategy can take advantage of this by disciplined use of text / value fields, allowing the use of 'soft' partitions.

'Soft' partitions allow the data to be split up conceptually, rather than physically. Although testers are able to interfere with each others tests, the team can be educated to avoid each others work. If, for instance, tester 1's tests may only use customers with Russian nationality and tester 2's tests only with French, the two sets of work can operate independently in the same dataset. A safe area could consist of London addresses, the change area Manchester addresses, and the scratch area Bristol addresses. Typically, values in free-text fields are used for soft partitioning.

Data partitions help because:

- Allow controlled and reliable data, reducing data corruption / change problems
- Can reduce the need for exclusive access to environments/machines

Clarity

Permutation techniques may make data easier to grasp by making the datasets small and commonly used, but we can make our data clearer still by describing each row in its own free text fields, allowing testers to make a simple comparison between the free text (which is generally displayed on output), and actions based on fields which tend not to be directly displayed. Use of free text fields with some correspondence to the internals of the record allows output to be checked more easily.

Example from billing system above:

Customer Name	Account	Tariff	Care	Bill
HP1 Monthly	M	1	H	P
LP2 Monthly	M	2	L	P
HU3 Monthly	M	3	H	U
LU1 Quarterly	Q	1	L	U
HU2 Quarterly	Q	2	H	U
LP3 Quarterly	Q	3	L	P

Testers often talk about items of data, referring to them by anthropomorphic personification - that is to say, they give them names. This allows shorthand, but also acts as jargon, excluding those who are not in the know. Setting this data, early on in testing, to have some meaningful value can be very useful, allowing testers to sense check input and output data, and choose appropriate input data for investigative tests.

Reports, data extracts and sanity checks can also make use of these; sorting or selecting on a free text field that should have some correspondence with a functional field can help spot problems or eliminate unaffected data.

Data is often used to communicate and illustrate problems to coders and to the business. However, there is generally no mandate for outside groups to understand the format or requirements of test data. Giving some meaning to the data that can be referred to directly can help with improving mutual understanding.

Clarity helps because:

- Improves communication within and outside the team
- Reduces test errors caused by using the wrong data
- Allows another method way of doing sanity checks for corrupted or inconsistent data
- Helps when checking data after input
- Helps in selecting data for investigative tests

Involving 'The Business'

Bringing QA into the development process early is a known way of reducing the cost of fault detection and resolution, and can improve time to market. The development process is initiated outside IT or QA teams by 'The Business', generally with some sort of business opportunity and associated requirements generation. A variety of different roles are encompassed by this vague and non-standard term, including those of regulator, customer, user and 'boss' - but one quality they have in common is that they known about the data rather better that they are likely to know abut the technical internals of the system. Data can enable effective communication between QA and 'The Business' in this early stage.

'The Business' is good at looking at data. Data can be compared with – and is often equivalent to – data used by existing process and systems. It can be easier to understand than tests, and it is often simpler to make a link between input and output data than it is to describe the internal process. Sharing data with 'The Business' can help to illustrate and model situations. This has the following advantages:

- Helps focus when requirements are vague
- Increases trust and understanding by improving ease of communication
- Helps early user identification of problems
- Improves familiarity and effectiveness of User Acceptance Testing

However, sharing the data has disadvantages. In sharing data, it is important to recognise and control the following problems:

- Data creep - where groups outside QA are using the Test Team's data, there may be pressure to include changes that aren't directly required by the Test Team. This can lead rapidly to large and uncontrolled datasets.
- Vague requirements can lead to vague data - although good data can help improve to focus requirements, truly vague requirements are likely to cause the data to change constantly until the requirements are firm. This introduces extra work, and can lead to out-of-date data.
- Incomplete data can lead to incomplete testing - if poor data is made a core tool for discussion, and testing only addresses known and discussed aspects of the data, the scope of testing may be too small and parts of the system can be left untested.

Data Load and Data Maintenance

An important consideration in preparing data for functional testing is the ways in which the data can be loaded into the system, and the possibility and ease of maintenance.

Loading the data

Data can be loaded into a test system in three general ways.

- Using the system you're trying to test

  The data can be manually entered, or data entry can be automated by using a capture/replay tool. This method can be very slow for large datasets. It uses the system's own validation and insertion methods, and can both be hampered by faults in the system, and help pinpoint them. If the system is working well, data integrity can be ensured by using this method, and internally assigned keys are likely to be effective and consistent.

  Data can be well-described in test scripts, or constructed and held in flat files. It may, however, be input in an ad-hoc way, which is unlikely to gain the advantages of good data listed above.
- Using a data load tool (see list at http://www.dwinfocenter.org/clean.html)

  Data load tools directly manipulate the system's underlying data structures. As they do not use the system's own validation, they can be the only way to get broken data into the system in a consistent fashion. As they do not use the system to load the data, they can provide a convenient workaround to known faults in the system's data load routines. However, they may come up against problems when generating internal keys, and can have problems with data integrity and parent/child relationships.

  Data loaded can have a range of origins. In some cases, all new data is created for testing. This data may be complete and well specified, but can be hard to generate. A common compromise is to use old data from an existing system, selected for testing, filtered for relevance and duplicates and migrated to the target data format. In some cases, particularly for minor system upgrades, the complete set of live data is loaded into the system, but stripped of personal details for privacy reasons. While this last method may seem complete, it has disadvantages in that the data may not fully support testing, and that the large volume of data may make test results hard to interpret.
- Not loaded at all

  Some tests simply take whatever is in the system and try to test with it. This can be appropriate where a dataset is known and consistent, or has been set up by a prior round of testing. It can also be appropriate in environments where data cannot be reloaded, such as the live system. However, it can be symptomatic of an uncontrolled approach to data, and is not often desirable.

Environmental data tends to be manually loaded, either at installation or by manipulating environmental or configuration scripts. Large volumes of **setup data** can often be generated from existing datasets and loaded using a data load tool, while small volumes of setup data often have an associated system maintenance function and can be input using the system. **Fixed input data** may be generated or migrated and is loaded using any and all of the methods above, while **consumable input data** is typically listed in test scripts or generated as an input to automation tools.

When data is loaded, it can append itself to existing data, overwrite existing data, or delete existing data first. Each is appropriate in different circumstances, and due consideration should be given to the consequences.

Frequency of Data Load

Different aspects of the data can be loaded with different frequencies and at different times, and consideration of this aspect of data load can be a substantial part of the Test Strategy's approach to data. The following triggers to data load are typical, and are listen in order of increasing frequency.

Not often

- At the start of overall test execution
- With each release of code / database schema made to the test environment
- At the start of each week (or any other calendar-based , rather than event-based trigger)
- When a tester asks for it
- At the start of every individual test

Often

Environmental and setup data are usually loaded least frequently, fixed input data is loaded often, and consumable input data is typically generated (but used rather than directly loaded) with each test. Frequently loaded data often has a higher cost and urgency associated with its maintenance, and it can be productive to find a method which allows fewer, less frequent loads.

Data Maintenance

All data needs work to keep it complete, clean, and up-to-date. The following list shows some of the reason for data maintenance. By monitoring these, time for data maintenance can be factored into the plans.

- Replacing consumed data
- Repairing broken data
- Responding to change - database schema, code, requirements
- New test requirements

Tasks to refresh the data should be built into the test plans. Repairs can be classified in the same way as bugs and fixes prioritised appropriately. Changes to the data can generally be planned for, but be aware that if the test scope has missed some tests, or if investigative tests need new data, the changes required may be short-notice and urgent.

Data maintenance can cause significant problems. Manual data maintenance is prone to error, and can introduce faults that lead to test failures, which are then logged and must be painstakingly resolved. Data maintenance is a sizeable task, and can make up a substantial fraction of overall test maintenance costs. However, the most common cause of problems is that data maintenance is generally (and often necessarily) performed by more than one group - different people tend to be responsible for environmental data than are responsible for fixed input data. Synchronising changes can be hard and effective management is needed to deal with differing priorities.

Solutions

Forewarned is forearmed with data problems, and recognising and preparing for problems can help in their resolution. The lists above may help with this task.

Change control and measurement can help to isolate changes and alert team members when they happen. Monitoring environmental variables, saving database contents and using configuration management can all be used to avoid or mitigate problems.

Maintaining data with automated, repeatable (and debuggable) scripts reduces the number of data maintenance faults.

Using small, well-known and comprehensive datasets as described above makes test maintenance simpler and quicker, and may also reduce the likelihood of errors.

Testing the Data

A theme bought out at the start of this paper was 'A System is Programmed by its Data'. In order to test the system, one must also test the data it is configured with; the environmental and setup data.

Environmental data is necessarily different between the test and live environment. Although testing can verify that the environmental variables are being read and used correctly, there is little point in testing their values on a system other than the target system. Environmental data is often checked manually on the live system during implementation and rollout, and the wide variety of possible methods will not be discussed further here.

Setup data can change often, throughout testing, as the business environment changes – particularly if there is a long period between requirements gathering and live rollout. Testing done on the setup data needs to cover two questions;

- Does the planned/current setup data induce the functionality that the business requires?
- Will changes made to the setup data have the desired effect?

Testing for these two questions only becomes possible when that data is controlled. Aspects of all the elements above come into play;

- The setup data should be organised to allow a good variety of scenarios to be considered
- The setup data needs to be able to be loaded and maintained easily and repeatably
- The business needs to become involved in the data so that their setup for live can be properly tested

When testing the setup data, it is important to have a well-known set of fixed input data and consumable input data. This allows the effects of changes made to the setup data to be assessed repeatably and allows results to be compared. The advantages of testing the setup data include:

- Overall testing will be improved if the quality of the setup data improves
- Problems due to faults in the live setup data will be reduced
- The business can re-configure the software for new business needs with increased confidence
- Data-related failures in the live system can be assessed in the light of good data testing

Conclusion

Data can be influential on the quality of testing. Well-planned data can allow flexibility and help reduce the cost of test maintenance. Common data problems can be avoided or reduced with preparation and automation. Effective testing of setup data is a necessary part of system testing, and good data can be used as a tool to enable and improve communication throughout the project.

The following points summarise the actions that can influence the quality of the data and the effectiveness of its usage:

- **Plan the data for maintenance and flexibility**
- **Know your data, and make its structure and content transparent**
- **Use the data to improve understanding throughout testing and the business**
- **Test setup data as you would test functionality**

References

Webpages and online papers:

Cohen, Dalal, Parelius, Patton: The Combinatorial Design Approach to Automatic Test Generation http://www.argreenhouse.com/papers/gcp/AETGissre96.shtml, Telecordia Technologies, pub. IEEE Software September 1996, pp. 83-87

Tai, Lei: A Test Generation Strategy for Pairwise Testing http://www4.ncsu.edu:8030/~ylei2/myData/htmlfiles/ipo_j01/ipo_j01.html, Department of Computer Science, North Carolina State University

Updates to this paper may be found at www.workroom-productions.com

This paper © Workroom Productions Ltd. 2001

QWE2002 Paper 9A

Mr. Leo VanDerAalst
(Gitek nv, Belgium)

Testing Challenges of Incremental Component Based Development

Key Points

- Explanation of incremental component based development iCBD.
- Understanding the test challenges of an iCBD environment.
- How to deal with these challenges, lessons learned, best practises.

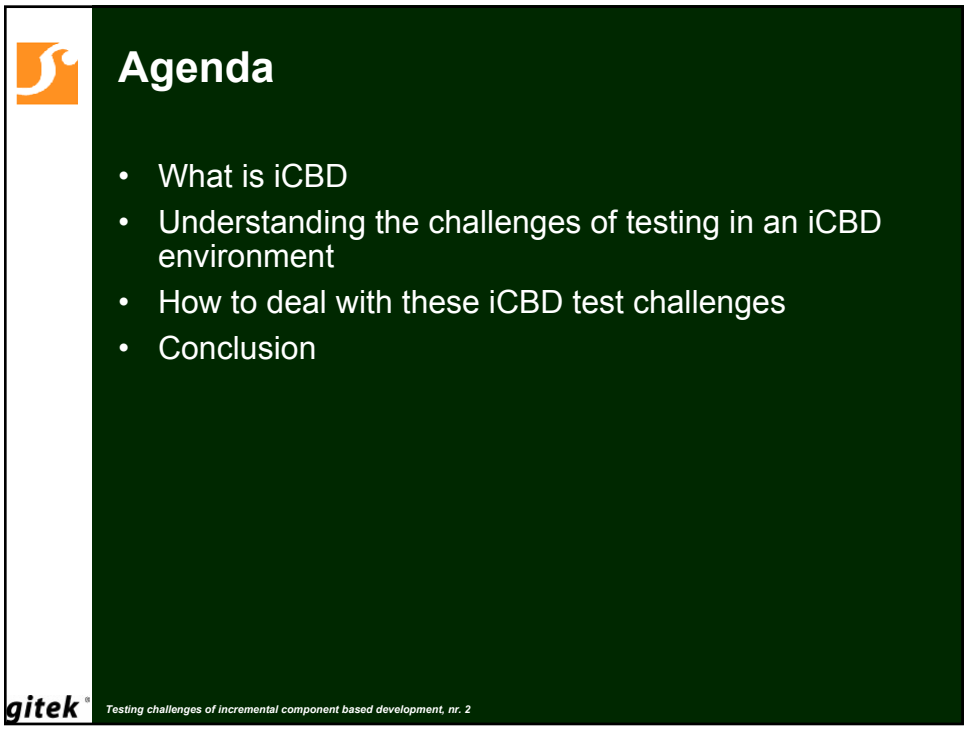
Presentation Abstract

For developing software, two trends are very popular at this moment iterative and incremental system development and component based development. Several organisations try to combine both approaches. What are the consequences for testing in this so called iCBD environment? What is iCBD? Why CBD? Why incremental system development? What is a component? What is CBD? What is iterative and incremental system development? Understanding the challenges of testing in an iCBD environment. How to deal with these iCBD test challenges. For the challenges, several possible solutions are presented.

About the Author

Leo van der Aalst is a member of the R&D-team of IQUIP Informatica B.V. in the Netherlands, a company with over 400 dedicated testers and owner of the structured testing approach Tmap®. He has more than 14 years experience with structured testing in all of its aspects: strategy, co-ordination, preparation, realisation, consulting and innovation. One of his activities in the R&D team was to develop a generic model for implementing test service centres at customer locations. This model has been used several times since then, for instances in a credit card company in the Netherlands. At the moment Leo implements a testing approach in an incremental component based development environment. This approach is developed and implemented in collaboration with one of the largest financial institutions in the Netherlands. He frequently presents at conferences (ING Bank 2000+2001, Dutch Association for Engineers 2000+2001, TEST Congress 2000) and is (co-)author of several articles.

Testing challenges of incremental component based development (iCBD)
Quality Week Europe 2002, Brussels
by
Leo van der Aalst
gitek
Testing challenges of incremental component based development, nr. 1


Agenda
• What is iCBD
• Understanding the challenges of testing in an iCBD environment
• How to deal with these iCBD test challenges
• Conclusion
gitek
Testing challenges of incremental component based development, nr. 2

What is iCBD

For developing software, two trends are very popular at this moment:

- Component based development
- Iterative and incremental system development

Reasons for iCBD

Required: flexibility!

To be improved:

- Adaptibility (system)
- Reusability (system)
- Manageability (project)

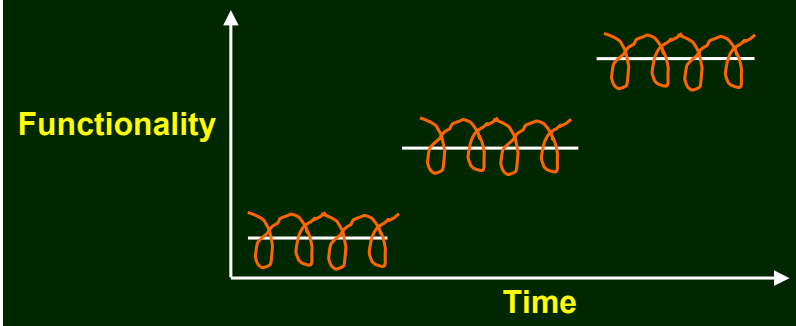
What is a component

- Encapsulated piece of software
- Offers specified services
- Accessible via well-defned interfaces only
- Business or technical
- Independent of system development method, tool or programming language
- Independent of other components

(See also: Butler, Forrester, Gartner, etc.)

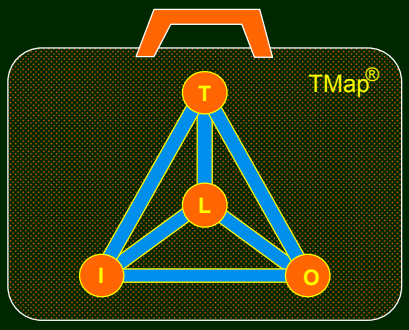
Traditional system development

Iterative and incremental system development



Understanding the challenges of testing in an iCBD environment

- Lifecycle — What, when ?
- Test organisation — Who ?
- Test techniques — How ?
- Test infrastructure & tools — Where, etc. ?

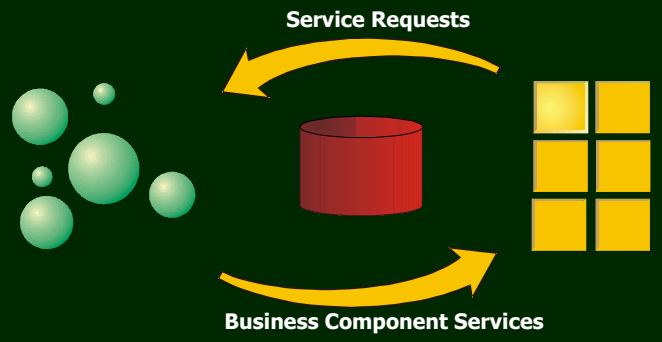


TMap® : Test Management approach is a registered trademark of IQUIP Informatica B.V.

Differences

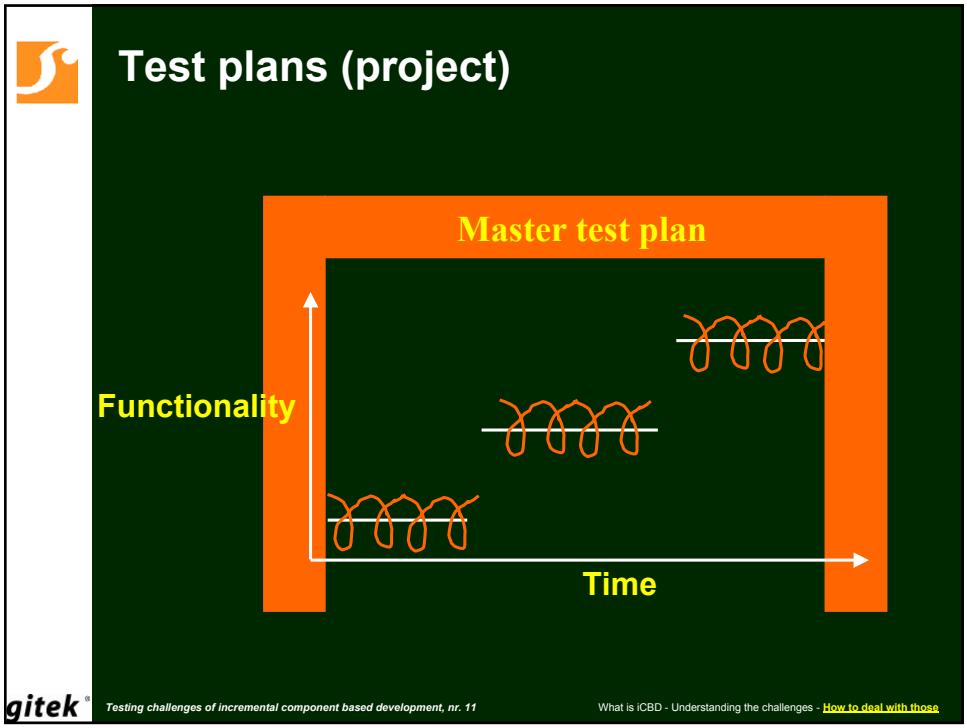
	Traditional ("waterfall")	Iterative Incremental
Lifecycle		
Test plans and test levels	+	++
Test strategy	+	++
Timebox	-	++
Organisation		
Test roles	-/+	++
Tester trapped in team	-/+	++
Independent testing	++	-/+

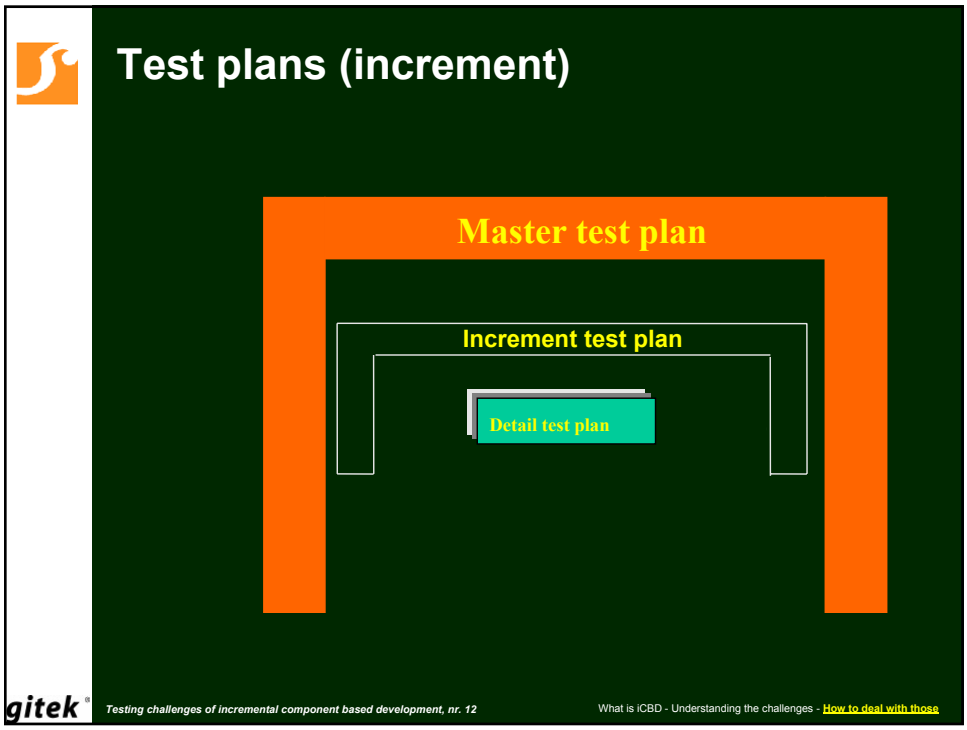
How to deal with these iCBD test challenges



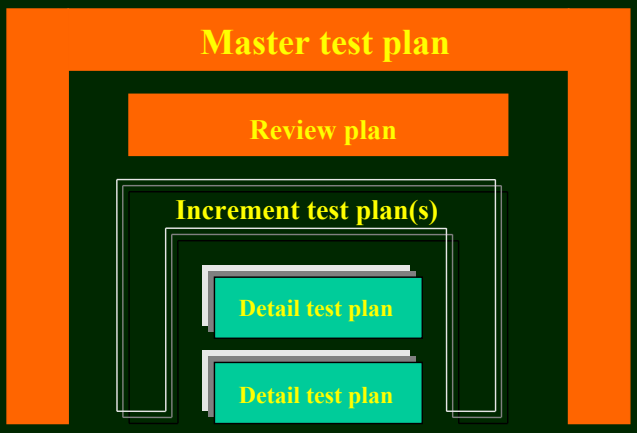
Supply **Manage** **Consume**

Select Perspective is a registered trademark of Aonix

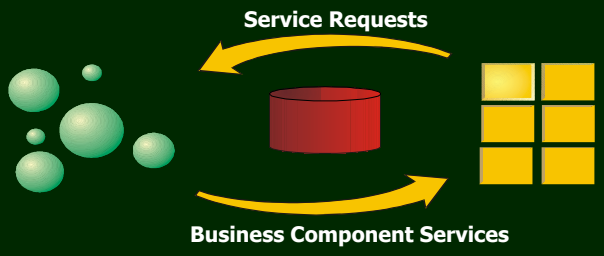

Test plans (project)
Master test plan
Functionality
Time
gitek
Testing challenges of incremental component based development, nr. 11
What is iCBD - Understanding the challenges - How to deal with those


Test plans (increment)
Master test plan
Increment test plan
Detail test plan
gitek
Testing challenges of incremental component based development, nr. 12
What is iCBD - Understanding the challenges - How to deal with those

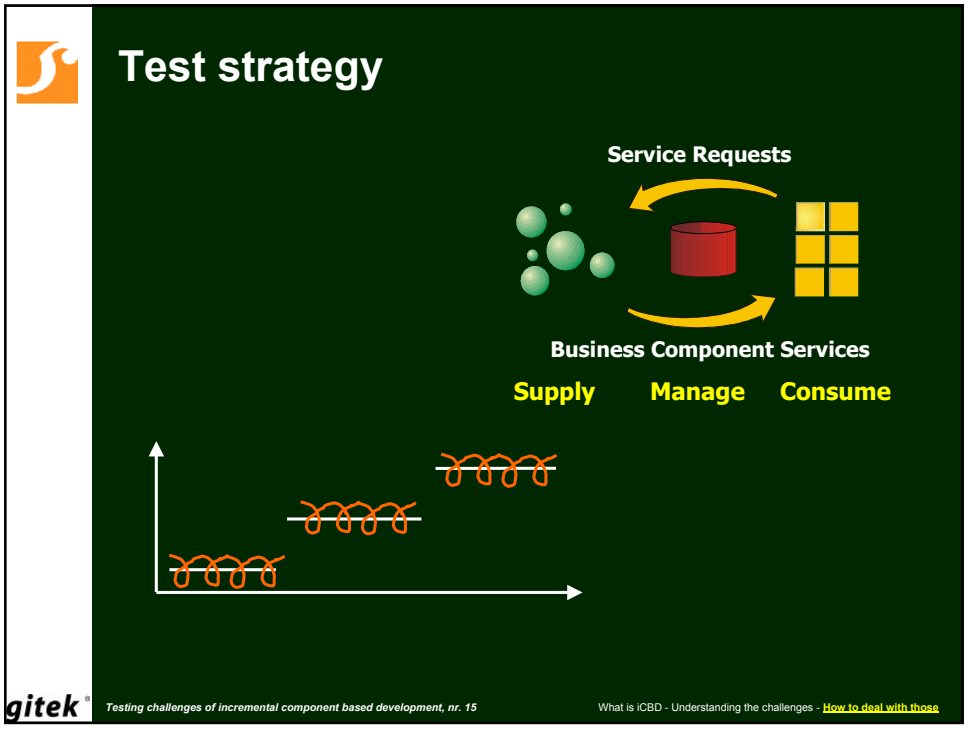
Test plans (overview)

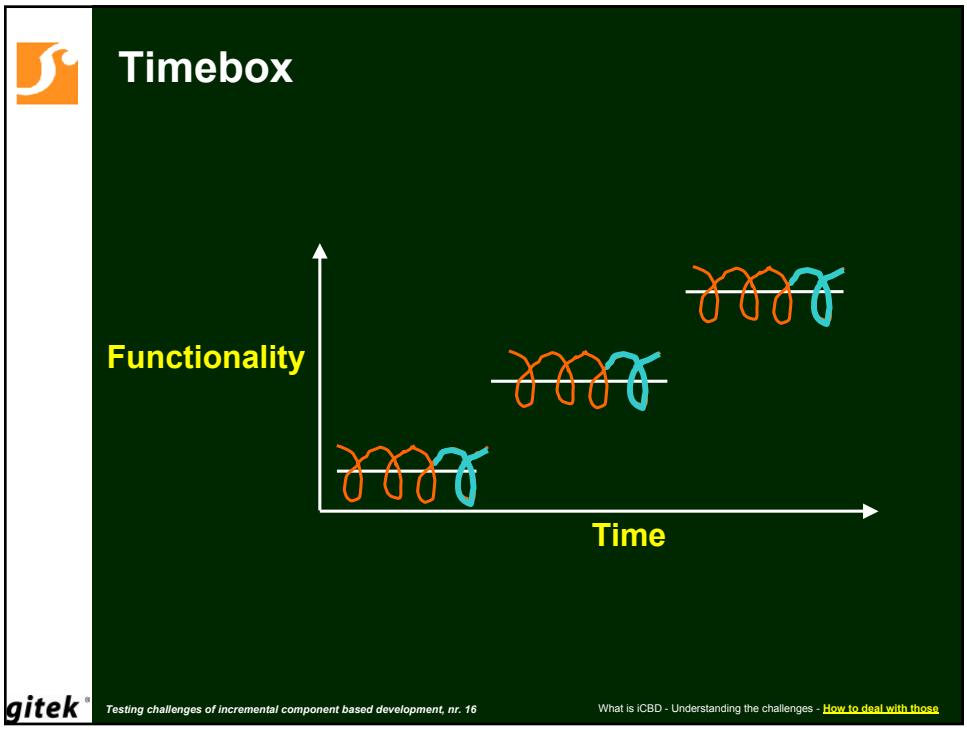


Test levels



Supply	Manage	Consume
Unit Test Integration Test System Test	Acceptance Test (Component Certification)	Unit Test Integration Test Combined Test User Acceptance Test Operational Acceptance Test


Test strategy
Service Requests
Business Component Services
Supply
Manage
Consume
gitek
Testing challenges of incremental component based development, nr. 15
What is iCBD - Understanding the challenges - How to deal with those


Timebox
Functionality
Time
gitek
Testing challenges of incremental component based development, nr. 16
What is iCBD - Understanding the challenges - How to deal with those

Test roles

component manager

project manager

solution developer

component developer

technical architect

test analyst

business domain analist

test manager

data service developer

data base developer

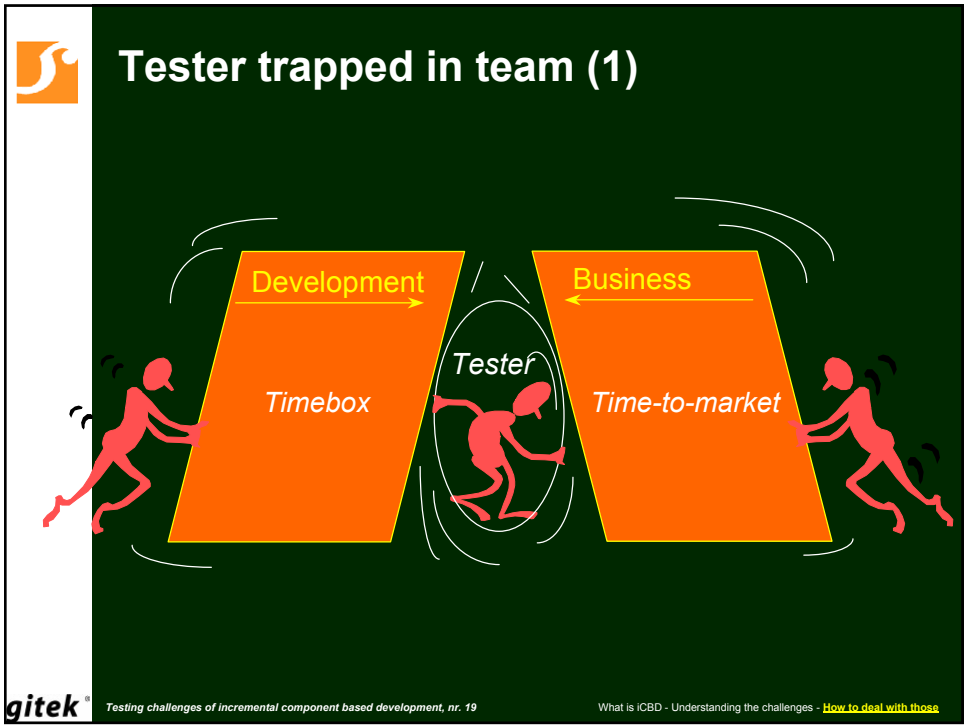
reuse advisor

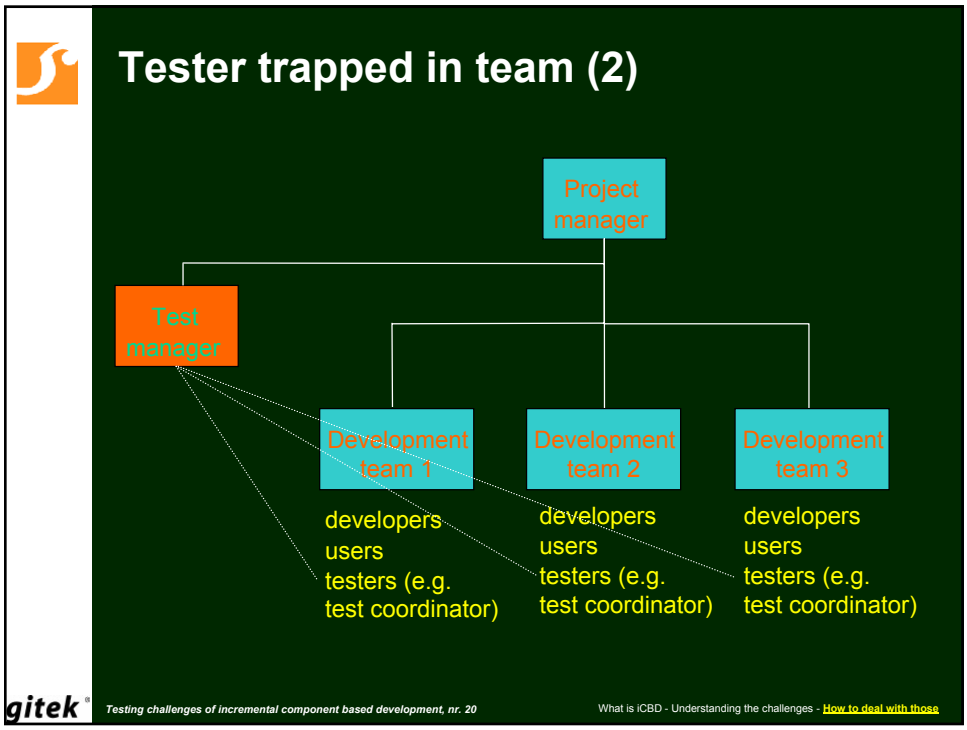
advisor user

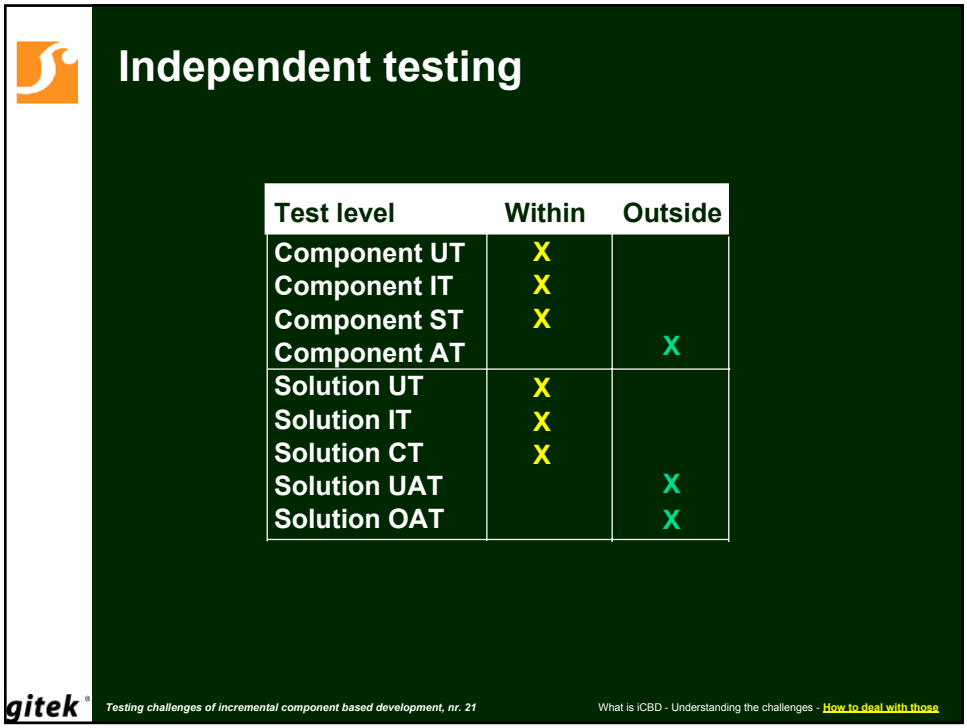
tester

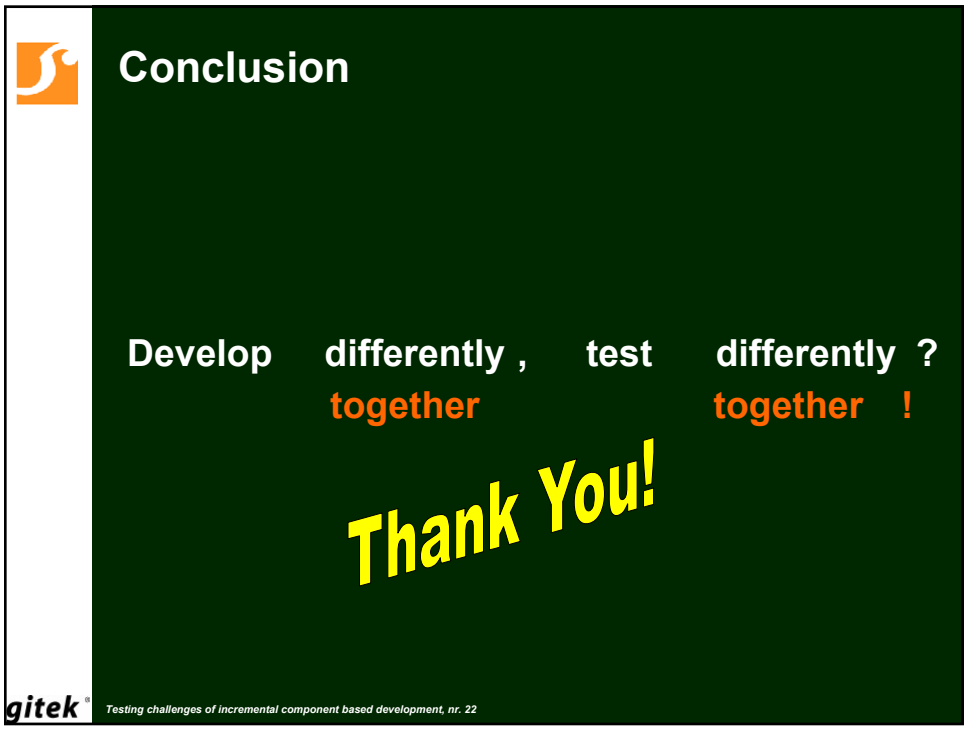
Test roles

Who is the tester?


Tester trapped in team (1)
Development
Business
Timebox
Tester
Time-to-market
gitek
Testing challenges of incremental component based development, nr. 19
What is iCBD - Understanding the challenges - How to deal with those


Tester trapped in team (2)
Project manager
Test manager
Development team 1
Development team 2
Development team 3
developers
users
testers (e.g. test coordinator)
developers
users
testers (e.g. test coordinator)
developers
users
testers (e.g. test coordinator)
gitek
Testing challenges of incremental component based development, nr. 20
What is iCBD - Understanding the challenges - How to deal with those


Independent testing
Test level
Within
Outside
Component UT
Component IT
Component ST
Component AT
Solution UT
Solution IT
Solution CT
Solution UAT
Solution OAT
X
gitek
Testing challenges of incremental component based development, nr. 21
What is iCBD - Understanding the challenges - How to deal with those


Conclusion
Develop differently , test differently ?
together together !
Thank You!
gitek
Testing challenges of incremental component based development, nr. 22

Testing challenges of incremental component based development (iCBD)

Contents

Contents ii

1 Introduction 1

2 iCBD 2
- 2.1 History 2
- 2.2 Component Based Development 2
  - 2.2.1 Architecture 3
  - 2.2.2 Obtaining components 3
  - 2.2.3 Component management 3
  - 2.2.4 Assembling 4
- 2.3 Iterative and incremental development 4

3 iCBD and testing 7
- 3.1 Methodologies and Testing 7
- 3.2 Risks 7
  - 3.2.1 Test strategy 7
  - 3.2.2 Component ownership 7
  - 3.2.3 Theory: a lot of testing 7
  - 3.2.4 Will the real tester rise? 8
  - 3.2.5 Documentation 8
  - 3.2.6 Independent testing 8
  - 3.2.7 Time box effect 8
  - 3.2.8 Tester not experienced 9
  - 3.2.9 Solutions 9

4 Structured testing in iCBD 10
- 4.1 Introduction 10
- 4.2 Test strategy 10
  - 4.2.1 Strategy considerations for components 10
  - 4.2.2 Strategy considerations for assembly 11
- 4.3 Test levels, evaluation and master test plan 13
  - 4.3.1 Test levels 13
  - 4.3.2 Components and the quality of low-level tests 15
  - 4.3.3 Evaluation 15
  - 4.3.4 Master test plan 15
- 4.4 Test basis 16
- 4.5 Lifecycle 17
- 4.6 Techniques 17
- 4.7 Infrastructure 18
  - 4.7.1 Test environment 18
  - 4.7.2 Test tools 18
- 4.8 Organization 19
  - 4.8.1 Who? 19
  - 4.8.2 Management 20

Bibliography 21

1 Introduction

Two significant trends may be distinguished to develop systems at this time: incremental development and component based development (CBD). In practice both trends occur linked with each other. Several clients wanted to know the best way to organize testing in such environments.

Because existing variants of TMap® like RAT, Rapid Application Testing [Broekman, 98] or TComp, [Koomen, 99] only answer this partly, it is decided to specify a new more dedicated variant, based on the most important and most popular design methods: Select Perspective® (SP), DSDM® and Rational Unified Process® (RUP)[1]. In this paper knowledge of TMap® is considered to be known by the reader. In this document, the subject incremented development of component based systems, is abbreviated to iCBD.

[1] TMap is a trademark or registered trademark of IQUIP Informatica, Select Perspective is a trademark or registered trademark of Aonix, Rational Unified Process is a trademark or registered trademark of Rational Software Corporation, Dynamic Systems Development Methodology is a trademark or registered trademark of DSDM Corporation

© 2001 IQUIP Informatica B.V., Diemen (The Netherlands)

2 iCBD

2.1 History

From the origin of the information technology, a large need to create information systems more rapid, better and cheaper exists. Although significant progress is being made, this need still exists. In addition: this need is still growing in the last years.

A reason for this is that information technology is continuously getting more important for management purposes. A development like e-business can not exist without ever-increasing automation. When information systems are delivered too late or have insufficient quality, this directly affects primary business processes negatively.

The question arises if the ICT-field is in a position to meet this demand of ever growing speed of system development. Two possible solutions are quite interesting at this moment:

- *Component Based Development (CBD)*, in which system development makes use of already existing components on a large scale, at which the functioning of the component is more important than the contents of it;
- *Iterative and incremental development* combines waterfall- and "pure" Rapid Application Development (RAD) system development.

These solutions will be explained in more detail, in under mentioned paragraphs.

2.2 Component Based Development

In order to explain what components are, one can make an analogy with a music center. This center has several components like a CD-player, speakers, cassettedeck, tuner and amplifier. These components are connected to each other with standard plugs, the functioning of each component is known, but it is not clear what is in each component. And when at a time a different component is needed, say a DAT-recorder, the only thing needed is just to replace the cassettedeck and change the plugs ("plug-and-play").

In traditional computerization the music center (=information system) will be replaced generally in total, because adjustment is too difficult. Continuing the analogy: the back side of the music center is a snarl of plugs and wires, each connection has its own specific size and the front size of the music center shows that the cassettedeck is controlled through the buttons of the amplifier.

Because the term "component" is quite recent within ICT, a standard definition doesn't exist yet or, there are a lot of different definitions. Several characteristics of components are listed below, gathered from common definitions:

- *Encapsulated;*
  The internal functioning, the “how”, is shielded from the public.
- *Offers (specified) services;*
  The service, the "what" needs to be described clearly.
- *Access through (specified) interfaces, only;*
  Interaction between the component and other software is only by specified interfaces.
- *Independent of develop method, tool and language;*
  Purists say that a component should be object oriented, others think this is irrelevant as long as preceding characteristics are matched. The use of

object oriented technology consists of some powerful mechanisms (inheritance, polymorphism, information hiding and encapsulation) which fit the CBD-principles well. A disadvantage is the fact that object oriented requires a different way of thinking and because of that it is not easy to combine it with other - not object oriented - methods and technologies within organizations.

- *Independently useable as a "whole for composition by third parties;*
  It should be possible to (re-)use components by third parties. This in fact implies that both executables and documentation need to be available. The origin of standard middleware makes it easier to run systems on a diversity of hardware and software, look at the Internet for instance! The size of a component is bigger than a program and smaller than an ERP-package.

Although there are many definitions of a component, this report considers the preceding characteristics of a component as a starting point.

The development of systems using components should lead to qualitative high level information systems, which can be corrected fast, and lead to a reduced and better control of the development process. A general requirement is that the solution integrates with the existing information systems. Most organizations have invested a lot in existing information services. Even though it is hard to maintain those systems, generally, these *legacy systems* are important to the management and so extensive that replacement is nearly impossible and not advisable.

The generic process steps of CBD discussed next:

2.2.1 Architecture

The first step is to determine the ICT needed to comply with the business objectives of the company. Parts of this step are setting up a business process model, defining the requirements of the system, the technical architecture on a global level, the components needed, and selecting the standards to be used.

2.2.2 Obtaining components

It is examined if components can be bought, reused or need to be constructed. Contrary to traditional system development, with CBD it is a matter of course to obtain components of third parties or reuse them from the own organization.

In case the component needs to be constructed, it will be specified, build and tested. In case it is meant to be reused, requirements to the documentation and quality of the component will be automatically of a higher level and more attention will be paid to check these requirements.

In addition the so called wrapping-variant can be used; alternative interfaces are built connecting to existing, non-component, programs (like legacy-programs) in a way that these can be combined with other components. Later on this semi-component may be replaced by a real component. This makes a gradual transition to a component based architecture possible.

In case the component is be obtained from a third party, the component will be tailored to fit the requirements during this step. This can be done by tuning the parameters of the component or by creating supplementary parts. It is tested as a whole. The custom work is considered as self building.

2.2.3 Component management

The components used in an organization need to be managed, for instance with help of a component library. This library describes not only the functioning, the interaction with other components and the technical specifications but also data about interfaces, configuration, different versions, naming and test ware.

© 2001 IQUIP Informatica B.V., Diemen (The Netherlands)

2.2.4 Assembling

With the help of the bought, reused and constructed components the required system is put together. Hereby a lot of programming still needs to be done. Apart from the building and adjusting the interfaces, the design of the user interface, the control of the workflow, compiling and linking and testing of the whole system are carried out this phase. The result of all this, the new system, needs to be accepted. In order to accomplish this the acceptance test will be performed.

2.3 Iterative and incremental development

An IT project with conventional development methods, sometimes called waterfall, often starts with the defining and detailing of the global wishes of the organization. This results in a specification of the system to be build. These specifications act as a contract between client and supplier. The supplier transforms these details into a design which then is constructed. Next he performs tests in order to secure his work meets the specifications. After final delivery, the client will make sure he gets what was agreed upon by performing an acceptance test. The relation between the waterfall method of system development and testing, is usually shown by the V-model (see figure 1). On the right side of the V several test levels are drawn. On the left side the in between products are drawn, which define the test basis for test levels on the right side.

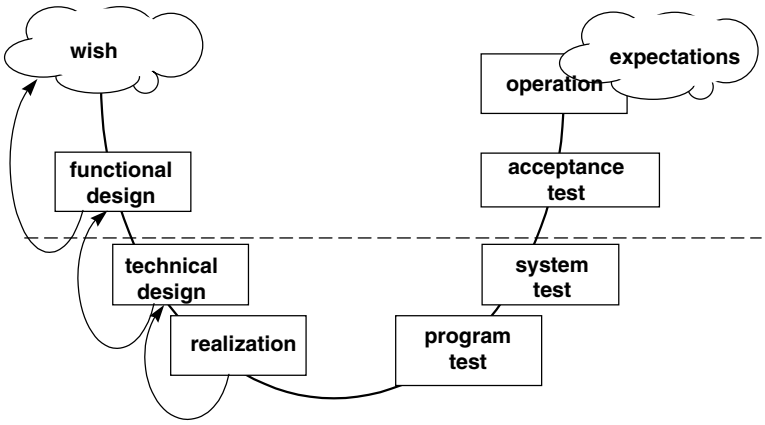


Figure1, The V-model shows the classic relation between testing and system development.

Often this waterfall method leads to irritation and criticism like:

- The process of system development takes too much time in order to meet the continuously changing demands of the market.
- The developers generally spend too much time writing documentation, which doesn't describe the precise needs of the user.
- The involvement of the user is too low and he is involved too late. It looks as if the user knows what he wants after the building of the system.
- Projects overrun its time and budget because many necessary changes are recognized late in the design phase.

This lead to the introduction of new development methods which accept sketchy requirements but require commitment of the user. The best known example for this is Rapid Application Development (RAD). SP, DSDM and RUP are all three variants to RAD. Figure 2 shows the generic development cycle of the method.

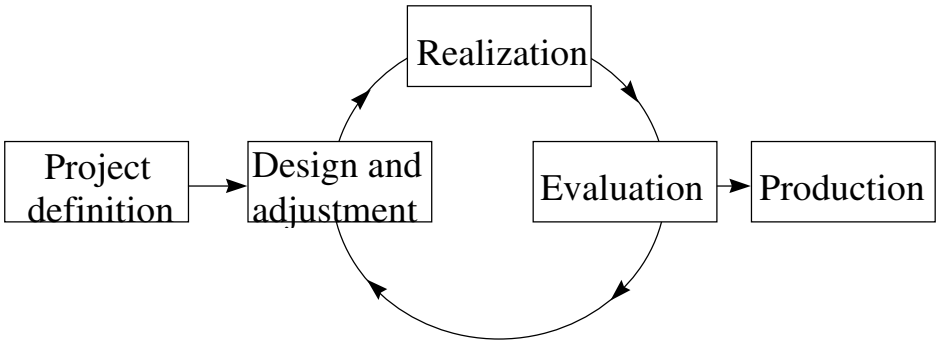


Figure 2, the RAD development course with cycles

At first the assignment and the scope are specified. Based on this global definition, the system will be designed, constructed and tested. These activities will be repeated until a satisfying result is gained within the estimated time. This will be taken into production and the cycle starts again.

Although several RAD-flavors exist, they have the following characteristics in common:

- Iterative and incremental;
  The actual construction of a system already starts based on sketchy requirements. By way of short feedback cycles the requirements are detailed and the system will be adjusted accordingly (iterative). Also, the development is directed to a fast delivery of the first usable version, after which in next versions the functionality will expand (incremental).
- Intensive involvement of the user in the working groups of system development;
  Users have an important role within the small development groups. These teams are responsible for the deliverance of a specific part of the system. The users are authorized to decide about functional content and requirements of that part of the system. Because the users are intensively involved at the development, there is no need to describe the functioning of the system extensively. By the way, RAD is usually initiated from the system development department, not from the business departments. As a consequence business departments will be informed insufficiently and do not fully understand the heavy impact of this new method of system development on their involvement.
- Time boxing;
  The development cycle is short and more driven by time than by functionality. A shopping list of components to be built is formulated. The items on this list get a priority and will be controlled by the working group for development. Beyond all this is the 80-20 rule: it generally costs 80% of the time and budget to construct the last 20% of the system. So first develop 80% of the system in 20% of the time. The result is taken in production, in order to give the users a working system early. The last 20% of the system will be build in the remaining time (80%).

The objective of RAD-methods is to deliver an adequate system within a very short time. It may not be the perfect solution, but it sufficiently meets the needs of the organization, is acceptable for the users and delivered on time!

As a disadvantage of the methods it is said that the high speed of development affects the quality of the system developed negatively.

There isn't a sharp boundary between RAD and waterfall methods. When both methods are placed next to each other, SP, DSDM and RUP position themselves within the group of RAD methods on the waterfall side. In this way one tries to overcome the mentioned disadvantage of RAD-methods as much as possible. The

negative association of RAD is the reason to use the term iterative development instead of RAD in this document.

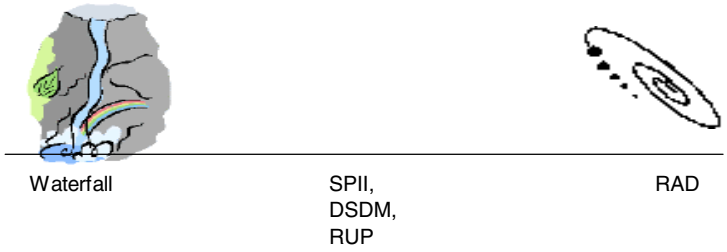


Figure 3, SP, DSDM and RUP

3 iCBD and testing

3.1 Methodologies and Testing

In the iCBD methodology testing is seen as an integral part of system design. Testing starts very early in the design time line and interacts with other system design activities. iCBD also boasts a risk analyze based test strategy, because of the knowledge that good testing implies making choices.

The fact that testing is seen as an integral part results in naming test related subjects like test levels, test types, test activities, test tools and test techniques.

Because of the iterative character of the methodology the reusability of test ware is highly valued. Reusability can be achieved by managing test deliverables and test automation.

The iCBD methodology has its own view on activities and products and recognizes several traditional test types, although those test types are assigned other names than usual.

3.2 Risks

The in iCBD used ways to look at testing equals the existing beliefs in testing and are promising. One would expect that testing in iCBD environments is easier than testing in traditional system development. In practice this is not the case. Evidently other factors exist that contribute to a more difficult testing.

3.2.1 Test strategy

Testing everything is not possible, this fact is widely known. Therefore testing on basis of risks and priorities is common practice and understood by the iCBD methodology. In which way that is going to be done is left unclear. The methodology restrains itself to a list of activities and components that will be more important, it does not describe in which way a more important component can be tested more thoroughly.

3.2.2 Component ownership

A component that will be used in more than one system must be of excellent quality. To test such a component it is necessary to put in relatively high amount of test effort. The question is who is going to pay for the extra test effort. The manager of the project that is going to be the first user of the component is not readily willing to pay for extras that he is not going to benefit from. The department that is going to administer the component wasn't allocated budget for design activities. So this leads to the questions: who benefits the most from a high quality component and who’s prepared to pay for it?

3.2.3 Theory: a lot of testing

The methodologies use a lot of test related concepts such as prototype testing, unit testing, integration testing, regression testing, system testing, system integration testing, user acceptance testing, operational acceptance testing, performance testing, usability testing, security testing, operability testing, conversion testing, etc.

© 2001 IQUIP Informatica B.V., Diemen (The Netherlands)

In practice it is found difficult to separate test types and test levels and using that many test concepts afflicts a contra productive attitude: *we cannot do this all, we will make our own choices*. People end up in defining a programmers test, a user acceptance test and an operational acceptance test without looking at test strategy, quality characteristics, risks and attuning the test levels.

3.2.4 Will the real tester rise?

In the iCBD methodology the tester has a role not a function. According to the methodology one needs to possess a certain level of test knowledge and test experience to fulfil this role. In the abstract this is true, but in practice this creates a problem. Due to the small teams that are formed, each member has to fulfil several roles in the team. Overlooking the importance of testing the test roles are easily skipped or not adequately fulfilled. For users taking up the test role often seem to have overlooked several parts of the system. They concentrate on the behavior of the system at normal day to day tasks and are not equipped to analyze a found failure. Does a user who is fulfilling a test role know what he is going to accept? In evaluation sessions the engineer demonstrates a new version of the product and asks for a users acceptance. If the user accepts the new version, what is he accepting precisely? Is he accepting the window layout or all the fields shown on the window? Has he checked if the standard data is shown, that the input fields are correctly validated or that certain options are released under certain conditions, etceteras? The risk exists that a user will accept the product based on a limited number of aspects while the rest of the team assumes he is accepting the whole product.

3.2.5 Documentation

To prepare the physical test cases the tester needs to know what are the expected results at certain input to the system. This calls for detailed specifications and enough time to transform the specifications to physical test cases. At iterative system development the specifications seem to be produced or finished at the end of a time box. Also is aimed at a minimum set of documentation. As a result of this the discussions between team members and the decisions they have made are documented in a limited way. The late availability and minimum quantity of the documentation have an impact on the test techniques that can be used. A lot of techniques demand a certain form of documentation and can not be used at this way of system development.

This documentation issue is also a problem in administering reusable components. In order to decide if a component is reusable, it should be documented thoroughly.

3.2.6 Independent testing

The receiving organization often expects an independent user acceptance test. This will result in formal and massive tests which will overlap the test activities within the development team. In acting this way they show a mistrust in the users who are members of the development team and will the speed and cost gains of iterative development be minimal.

3.2.7 Time box effect

Time is the leading factor at incremental development. The products have to be delivered at agreed moments in time, the amount of functionality or quality is seen as less important. The time box introduces the risk of extending time working on realization of as much functionality as possible, leaving little time for adequate testing.

3.2.8 Tester not experienced

Working together as a team asks a lot of the person responsible for testing. Other members of the team treat this person often as an obstacle and a slowing down factor. If this person is inexperienced or less sustained, he is easily going to be influenced by the opinions of others. Risks warned for as a result of test activities are being skipped or seen as harmless. Not until the time of implementing the product the poor quality of the product will show. This risk will be a lot smaller when a professional tester is added to the team.

3.2.9 Solutions

The next chapters describe in what way testing in the iCBD environment can be implemented. The risks described in the above paragraphs will be answered in different sections. In the following table the risks are related to a solution.

Risk	Solution
Test strategy	4.2, 4.3
Component ownership	4.2.1, 4.3.1
Theory: a lot of testing	4.3.1, 4.3.4
Will the real tester rise?	4.3.1, 4.8.1
Documentation	4.4, 4.6, 4.8.1
Independent testing	4.3.1, 4.3.4
Time box effect	4.2.2, 4.3.1
Tester not experienced	4.2.2, 4.3.1

© 2001 IQUIP Informatica B.V., Diemen (The Netherlands)

4 Structured testing in iCBD

4.1 Introduction

This chapter is about the issue "in which degree do developments implicate the way of testing?". Knowledge of TMap® is a starting point for this item. The following paragraphs describe the aspects that are specific for iCBD in relation to the TMap® cornerstones.

Successively we will describe:
- Test strategy,
- Test levels,
- Test basis,
- Lifecycle,
- Technique's,
- Infrastructure,
- Organization.

4.2 Test strategy

In this paragraph diverse aspects of the test strategy are dealt with. A distinction is being made in considerations that deal with testing of components and considerations that have to do with testing in an iterative environment. At the same time there is the following general consideration:

- Good use of iCBD means that the project and organization have to meet up to a number of starting points, such as sufficient commitment of users and organizational layout in order to make reuse of components possible. In practice this means that only few of these starting points are met. For the test strategy it is important to anticipate that problem by taking the possible measures.

4.2.1 Strategy considerations for components

An important characteristic from Component Based Development is that a system is constructed from a mixture of bought, reused and newly build components. For a number of reasons, this requires high quality components. Here the term quality does not only imply the right functionality of the system, but also quality characteristics such as maintainability, performance and portability:

- The builders of these components are often (no longer) available. If one or more of the components shows a shortcoming of quality, this can have serious consequences for the total system. Repairing the components can be extremely difficult or even impossible. This is more valid when components are bought or reused than self build components.
- It is difficult and expensive to find the cause of a problem that occurs in assemblage phase or thereafter because knowledge of the internal working of the underlying components is missing. High quality of the components in combination with good specifications and interface descriptions reduces the chance of problems and shortens the time searching for the cause.
- One component may be used by several systems; therefore insufficient quality of the components may be a problem for those systems. Beforehand it isn't known in which way and in which systems a component will be used. That makes it difficult to analyze all the risks and required quality during the stage of development. Is, for instance, performance testing relevant? Maybe not for

the first system that will make use of this component, but what about the future systems that will make use of it?

Therefore it is important to determine in an early stage if a high level of quality is required from the components. Testing a self-build or wrapped component can be compared with testing of regular tailor-made software. Primary subject of the test is if the build component meets the specifications. Depending on the risks additional tests will be executed to check if and how much the component meets the non-functional quality requirements.

The main differences with regular tailor-made software are:

- The quality demand 'suitability' does not only affects the organization but also the architecture (i.e. the use of standard interfaces);
- From a component it is often expected, that it can easily be replaced, for example that this component can be replaced during run-time. This is an issue when several other systems or components use the component that has to be replaced. Changing the component should be possible without noticing by the other systems or components.
- It is often expected that a component is reusable. To be reusable more severe demands are made like correct results, understandability (documentation!), flexibility (in order to make the component widely useable) and portability (i.e. platform-independent, use of standard interfaces). By the way, it may be possible to deliberately build components that will not be reused. Then these higher demands are not applicable.

If the mentioned quality characteristics need to be tested and how much test effort this may cost depends entirely on the risks, the required insight in the quality characteristics and the owner of the component.

The above mentioned points are differences between tailor-made software and self-made components Besides that, components may be reused or bought. Testing reused or bought components is mainly the same as testing a standard software package. This means that when testing this product, less emphasis lies on functionality, and more on the non-functional quality-characteristics such as suitability. We may expect from the (internal or external) supplier that the component meets the specifications. This can be proven because the component is already used by the organization or elsewhere. This trust can be strengthened if the component is delivered with test ware that shows the coverage of the tests. So testing the functionality is less obvious than with self build component. Nevertheless it may be important to test this, for example when the component is in use only on a small scale yet or when a unique composition is used. The functioning of the component is often tuned by parameter setting or extension with parts of tailor-made software. Of course these adjustments should be tested.

4.2.2 Strategy considerations for assembly

The separate components are integrated into the desired system during the assembly process. In the assembly process the test strategy focuses on the product risks of the system. This is the same as the traditional test strategy. Nevertheless the following aspects should be taken into account:

Integration

We need to be aware that component based development implies a displacement of complexity further backwards. Even if each component is very well developed and tested, not until the moment of total integration it will turn out whether it yields to a usable system.

Integration takes place between extremes. One extreme holds the big-bang philosophy. Herewith all separate components are connected simultaneously, where after is tested if everything functions properly. Disadvantage is that when the unity does not function, much puzzling is required to retrieve which

component or interface causes this. The supplier too, is not often willing to analyze the component when it is insufficiently clear that his component causes the problem. Another extreme implicates that the interfaces with each additional component need to be tested first. Not until it appears that those interfaces function correctly, integration will be continued. Disadvantage here is the great quantity of time and effort needed to complete the integration. In practice, depending on the components available and risks, a middle course is steered.

Iterative developing

The iterative character of system development means that the product will not be finished at once but continuously evolves. Each evolution takes place in a short period of time, sharply defined by the time box principle.

The short lead-time is seen as a disadvantage to testing: so much to test in so little time. The counter argument is that the developer is limited in time as well. The problem is therefore situated in the fact that the developer overruns its development time at the expense of the available test time, because the deadline of the time box is " chiseled in concrete ". This problem is for that matter recognizable too for traditional testers. Good proactive management of the test process and operating from a test strategy based on risks, like TMap® describes, are essential. Fortunately the problem is also recognized by some of the methodologies. DSDM recognizes a threesome iterations: investigative, refining, and consolidating, whereby it is not allowed to build a new functionality in the last iteration. When in spite of this all the testing part still is jeopardized, the only thing remained to do is to report the risks as clear as possible.

Another problem is the quantity of successive evaluations. When the product after all passed several evaluation sessions, is it still complying with the corresponding agreements? A part of the product will be finished and agreements, necessary for the next cycle, have to be made. A possible risk here is that parts, accepted and agreed upon in a previous cycle, may be changed as a result of side effects of agreements made later on in time.

The project manager normally gives the tester the responsibility to control this regression. This implies that on a large regular base regression tests need to be executed. This in order to verify continuously that the system still is performing as agreed upon. The tester needs to assure himself of the fact that agreements made, can be reproduced in exact the same way in next evaluation sessions. They have the duty to retain to a specific example that shows exactly what the desired behavior of the system is. The tester deals with this example as if it is a test case. At later evaluation sessions these established test cases could be used to control whether the previous appointed agreement is still performing in the new product versions. Mainly this regards the testing of the quality characteristic “functionality”. For this purpose reusable test ware is of very great importance. At the moment of test set up this aspect should be taken into consideration.

The regression test is therefore an important candidate to test automation. Modern 'Capture & Playback' tools could be of tremendous service to record test cases. During the development of such (automated) regression test the tester needs to aim at a high level of portability. The organization can gain great advantage out of these regression tests in a later stage of the project, for example when the automated test suite can be executed at the integration test and at the production acceptance test.

Another responsibility of the tester is to help the user to take the right decision and not to forget components when evaluating these. Hereby the tester can make use of a checklist of aspects to be tested. Furthermore he can make use of his experience and expertise to indicate the weak spots and risk fields, which therefore ask more attention. It is the responsibility of the user to decide whether the product is accepted or not. It is the role of the tester to identify weaknesses that are apparently overlooked.

Because documentation in the iterative system development is late and minimal, an early involvement of the tester in the lifecycle, in particular the process to be geared to between developer and user, is required. This enables the tester to control agreed items.

Finally: not everything can be negotiated between developer and user. A tester needs to be aware of this to avoid discussions afterwards.

4.3 Test levels, evaluation and master test plan

4.3.1 Test levels

The iCBD methods acknowledge more or less the traditional test levels. When a programmer has written his program it has to be tested (program test). When a certain number of programs are integrated into a workable part, the functionality of this part is tested (integration test). After this, the developer likes to test the complete system (system test), before it is submitted for approval to the customer(s) (acceptance test).

The important question at each test level is who is responsible for it. At program- and integration tests it is clear, but at acceptance tests there are several options. The table below gives an overview.

Test level	**Who?**
Program test	Developers (within the team)
Integration test	Developers (within the team)
System test	Developers (within the team)
(Functional / user) acceptance test	Users (within or outside the team)
Production acceptance test	System management (mostly outside the team)

“Within or outside the team” means that a test can be executed within the project team, but that it is also possible that a separate test team outside the project team can execute some test levels. It is usually decided to execute the production acceptance test outside the project team. This test requires specific (system management) expertise of the testers and a test environment that is as production like as possible. Both items often are not present within the project.

The strong involvement of decision authorized users and the collaboration within the team will minimize the (traditional) barrier between developers and users/acceptors. In case the functional acceptance test is executed within the project, it may be combined with the system test.

Executing the acceptance test outside the project is not in accordance to the principles of iterative development. Only in case of great risks, executing the acceptance test outside the team is a better option, in spite of the harm done to iterative principles. Examples of such risks are: users which are insufficiently decision authorized; project manager or developers who “dominate” the users; the system is critical in such an extend that the tests need to take place in a sufficient controlled, stable production-like environment.

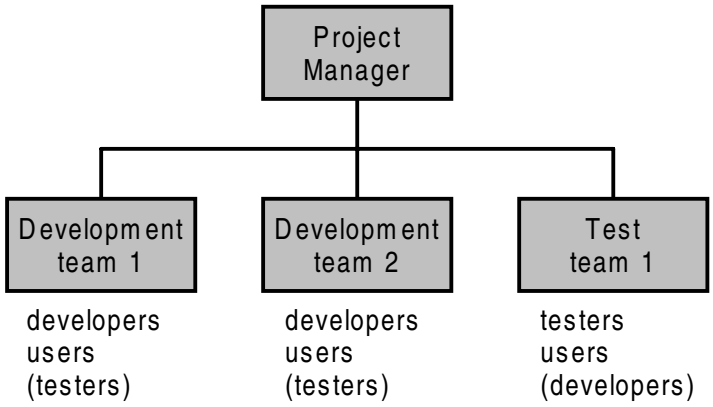
The engineering of components is normally organized in a separate process, parallel and separated from the assembly process. This separate process means fundamentally that the same test levels, starting with the program test to the acceptance test, are applicable. Depending on the scale, complexity and risks of the component all levels will be handled or for example only the program test and system test. An acceptance test of a separate component is possible, but not

obvious. It depends on the fact whether the owner can be clearly distinguished. In case the project is owner the acceptance of the component implicitly takes place as part of the overall user acceptance test. In case the owner is the department "Component Management", it may be necessary to conduct an acceptance test on the component itself in order to determine whether the component is of sufficient high quality. The department also can choose to accept a component on the basis of an agreed profundity of the system test. A production acceptance test of a component is normally not executed, because a component will never be implemented itself, but always be part of a system.

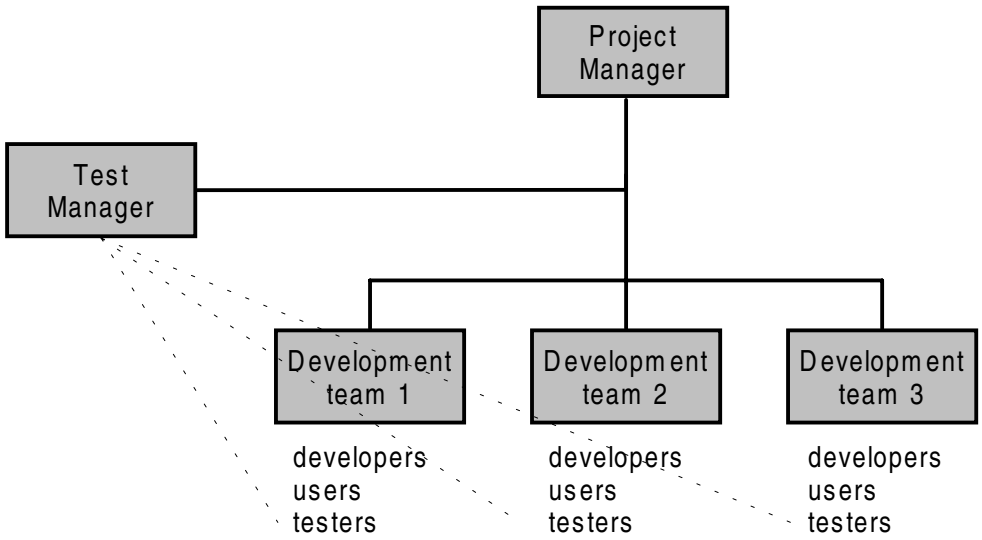
The choice of test levels always depends on the situation, per organization and even per project.

When putting a high-level test team together, the fact that "the tester" is a role has to be taken into account. Both developers and users can fulfil this role. To guarantee sufficient test expertise, two alternatives are possible:

1. Initialize a special test team, managed by an experienced test manager, the traditional variant. This possibility is obvious when separate development- and test teams are desirable, for example to judge as objective as possible the quality of the object to be tested.



2. Attach an experienced test manager to a staff function with functional relations to the development teams. The testers within the team report to the test manager. The advantage is that the gap between developer, user and tester in the teams is minimized, while the test manager can point out problems in case the testing in a team becomes a bottleneck due to time pressure or other circumstances.



In both cases it is advisable to allocate professional testers to the team.

4.3.2 Components and the quality of low-level tests

To accomplish the desired high level of component quality, testing thoroughly is of great importance. Theoretically this can be achieved by putting a lot of effort in test levels like the system- and acceptance test. For certain reasons this makes no sense. Firstly these test levels are relatively late in the lifecycle. Defects discovered by these test levels are relatively expensive to repair. Furthermore the components appear to be difficult to test, for example because they don't have a user-interface, or quite a lot of technical knowledge is required for a good test. Therefore it is easier to make agreements to improve the organization of program- and integration tests. Some aspects to improve on these test levels are:

- Introduce a test lifecycle;
- Employ the test strategy to the program- and integration tests;
- Produce evidence of achieved testing (as a part of the test strategy);
- Allocate test expertise;
- Install an Application Integrator with responsibility for quality of the products and progress of the integration;
- Automate the test preparation and executing;
- Report in a simplified manner.

4.3.3 Evaluation

As stated earlier, at iCBD, testing and evaluating are seen as integral parts of system development. Evaluation may be seen as reviewing in between products, like the functional design. The importance of evaluating before testing is that defects can be detected at an early stage of the product. Because of this the costs of repair are much lower. Evaluation is also very easy to organize, because no programs need to run, no environment needs to be set up, et cetera. Familiar forms of evaluation are reviews and inspections.

Evaluation proved to be a very efficient method, but is very often hard to carry out in a right way. To deal with more manageable evaluation, the following indications are given:

- Make somebody responsible for the evaluation activities. This person needs to organize and supervise the sessions and needs to get enough time. The project leader is therefore an undesirable choice. The test manager or QA-manager is a more obvious choice.

- Employ techniques, like process descriptions and checklists to maximize the effectiveness of the evaluations.

- Involve testers in the evaluations so that the in between products are evaluated from the point of view of testability. In general, testers have a critical attitude, which makes the evaluation more effective.

4.3.4 Master test plan

Aligning the test levels is a must when aiming at an optimal test strategy: to find the most important defects as early and as cheap as possible. At this alignment the above-mentioned evaluation activities need to be considered too. The way to achieve this is by establishing (and monitor) a master test plan. The master test plan makes an overall coordination of the test process possible. The big advantage here is that scarce knowledge and resources can be optimally allocated to the different test levels.

© 2001 IQUIP Informatica B.V., Diemen (The Netherlands)

4.4 Test basis

Within iterative development the behaviour of applications is determined within the projectteam during workshops. Within iCBD documentation is usually based on the Unified Modeling Language (UML). UML is a collection of modulating techniques whereof use case and class diagrams are the most widely known. The test cases are then çreated based on use cases and use case diagrams, even though these are highly abstract.

The quantity and quality of the system documentation is often deliberately minimal due to the involvement of the user in the development process. When user and developer agree and the developer at the same time records it in the computer software, then why should they write it down?

The arguments to keep the system documentation to a minimum are usually that:

- it will take too much (lead-)time to make documentation.
- it is very expensive and time consuming to keep this documentation up to date and
- without documentation there is a better communication between developer and user.

However there are a few drawbacks to this decision of minimal documentation:

- a higher risk of misunderstanding
- no documentation of knowledge
- insufficient transferability
- insufficient maintainability
- a big need for verbal communication

For this reasons this approach is specially fit for less complex projects, while the expectation is that many iCBD projects are extremely complex. Also having and keeping the same people in the development and maintenance phase, makes for bigger chance of success in the task: with every new person that needs to be trained, there are above mentioned drawbacks to be felt.

A variable on above mentioned is that the documentation is drawn up after the fact, as soon as the application is stable. For testing this makes little difference, because who is going to determine that the application is stable (and which sponsor wants to wait with testing until the documentation is ready)?

Testing is one of the first activities that notice negative consequences of the decision for less documentation. Testing is the comparing of an object to be tested (most times the programs) with a frame of reference, the test basis (usually the functional design). When this test basis is documented insufficiently, the test team and the sponsor have a choice: minimal testing or supply the tester with more knowledge about the test basis (whereby the knowledge of individual persons forms the test basis), so that enough test cases can be put together to give confidence in the quality of the test. In both cases the test team needs to report very clearly about the risks. Although generally the second variation is chosen, this has a few disadvantages that are in addition to the previous mentioned disadvantage:

- The needed communication costs a lot of extra time, when it was meant to save a lot of time.
- Because a joint base with the developer is absent (except in the head of the sponsor), the risk is greater that the application doesn't work according to the expectations of the testers.
- The testers preferably need to have a lot of knowledge about the subject matter. The combination of lots of test and subject mater knowledge in one person is rare.

- Sometimes it is tempting to turn the documented material into design. A tester is never allowed to sit on the chair of the designer and put up a construction base! A tester is not qualified for this and gives an unwanted mixing of the responsibilities.

Preferably one wants to prevent that the documentation is inadequate. A good rule is to arrange for the early involvement of the testers to judge the testability of the documentation. This makes the disadvantages of insufficient documentation often insightful, so that still can be decided to deliver more documentation.

4.5 Lifecycle

There where test activities take place, TMap® can help out: Planning and Management, Preparation, Specification, Execution and Completion.

We have to keep in consideration that the phase Preparation in comparison to the other phases can become lighter or heavier:

- When getting a usable test basis is more time-consuming than normal, like explained in the previous paragraph, the phase becomes heavier.
- The detail intake can sometimes be part of the testing. In that case the phase Preparation becomes lighter.

4.6 Techniques

The available test basis is of great influence on the choice of test specification techniques. In practice the minimal documentation results especially in the choice for informal test specification techniques.

The popularity of UML as replacement of traditional modeling techniques creates by many people the expectation that the existing test specification techniques also have to be replaced. The argument for this is that the test techniques are only useable for intentional traditional techniques. This is a misconception. For traditional modeling techniques like Entity-relationship Diagram or Data Flow Diagram there are no specific test techniques. Likewise there are no separate test techniques for Use Case diagrams or Class diagrams. Commonly test techniques use functional specifications, data verification and screen layouts, the place where these items are stored is of less importance.

Used techniques are:

Technique	Necessary information	Test basis
Data flow test	Input to (one or more functions of) the system + expected results	Use cases Use case diagrams Class diagrams Class descriptions State transition diagrams Business scenarios
Process cycle test	Interfaces between business processes and information system	Use cases Use case diagrams State transition diagrams AO-descriptions

Technique	Necessary information	Test basis
Real-life test	The expected exploitation needs (frequency, level of database load)	Use case diagrams Description of expected exploitation needs
Syntactic test	Screen behavior and primary input validation	Screen layouts Data constraints (in classes)
Semantic test	Validation of relations between objects or data	Relations between data (in classes) State transition diagrams
Error guessing		
Program interface test	Interfaces between programs or components	Component interface description Classes

In most situations test cases are based on Use cases or Business scenarios. Checklists with a list of test and review aspects of components, based on earlier experiences, help in this process. At iCBD other aspects than in the traditional system development environments can be of great importance. For instance: does the component comply with stated architecture rules? The checklists will have to be updated with those new aspects.

One of the latest named test techniques is called Exploratory testing, which is a variant on Error guessing. At this technique test specification and test executing are mixed up. The tester is expected to investigate the test object at first, so in that way he can get a feeling of the weak spots of the object and adjust the test to that. The benefit is that a small test basis will do, the disadvantage is that test executing can only start at the moment the component is released. For this reason Exploratory testing is seen as an addition to the known set of test techniques. More about this technique can be found on http://www.testingcraft.com/exploratory.html.

4.7 Infrastructure

4.7.1 Test environment

Components are based on a specific architecture. The availability of a test environment, representative for this architecture, is strongly recommended. The architecture, however, may be a very heterogeneous and complex environment. The simulation of such a production-like situation may be hard to realize in testing and usually is an expensive matter. That is why an optimum should be find between the representativity of the test environment and the corresponding costs, at the performing of the tests.

Because the architecture has a certain stability and is meant to be used in the long term, this is an important argument to have the test environment at the disposal permanently. With a permanent environment at hand, no need exists to do the time consuming install and equip process more than once. Specific at this is that around this permanent test environment a hierarchical organization is organized with personnel, working procedures, test techniques, tools et cetera.

4.7.2 Test tools

As is stated in the beginning of this document, the every increasing concern of ICT requires a higher level of flexibility of that same ICT for the organization.

The ICT will be adjusted more often and more deeply than now, using components of high quality. The high frequency of adjustments and the high concern of good quality are a challenge for testing. Constantly further going automation of the test process is one way testing can meet this challenge.

The actual kinds of test tools may be suitable for the testing of components generally. Expected is, that new kinds of test tools come into market, directed to the integration between components, far instance by giving insight in the interfaces between components. Next, several important ways of test automation will be discussed.

4.7.2.1 Regression testing

Both the iterative way of work and the fact that components will be reused, imply the need for retest on a regular base. Automation of these (regression)tests is obvious. Capture & Playback or scripting tools give support at this. Point of attention: the maintainability of the scripts recorded is crucial to the cost-effectiveness of the automation. Techniques like data-driven enlarge the maintainability.

Anyway, the automation of the test is more obvious at the composing process than at the assembling process. The functionality of a component is expected to be more stable that the functionality of the system as a whole, which consists of several components and is constructed iteratively. This stability is again an important requirement to cost-effective automation.

4.7.2.2 Automating the conversion of test specifications into test cases

In the iterative system development environment the specifications are not frozen, but are ever changing and growing. For testing a direct relation connection between test specifications and test cases is a big benefit: while changing specifications it is clear which test cases should be changed also, at finding a failure with a test case it is clear which specification is not correctly implemented.

Automating the process of converting the test specifications into test cases helps to keep the conversion process manageable. Tools for requirement management and tools for test management will have an automatic interface.

4.7.2.3 Drivers and stubs

Components often lack user interfaces. To test a separate component “drivers” and “stubs” are used. Drivers are able to trigger a component and hand over certain input to the component. Stubs are used if from within a component a call is made to a function, program or other component, while the called function is not yet available or hard to implement. Drivers and stubs will be delivered together with the component.

The quality requirements for drivers and stubs must be stated. Drivers must be parametric in order to test the component under several conditions.

4.8 Organization

4.8.1 Who?

Collecting the right mixture of system knowledge, material knowledge and testing knowledge in a high-level testing team is a critical success factor.

Therefore recruitment of only users and developers is not enough. Besides the specific knowledge and skills that testers need to have, a bit of "psychology" also plays a part because the basic attitude of a tester is essentially different from the

attitude of a user or a developer. A testers job is to show the lack of quality in the system and in order to do this, he will actively search for errors. The attitude of a developer or user is more pointed at proofing that the system works correctly.

As stated in paragraph 4.3, it is recommended to let professional testers participate in the testing team. Besides that, the importance of test automation means that also expertise in this area is needed in the team. This doesn't mean that developers or users have become over complete as participants of the tests, in the contrary. But these people do have to be educated in testing: structured testing in general and the testing techniques especially. It is preferred that each member of the testing team has a specialization (system knowledge, material knowledge, or testing knowledge), and basic knowledge of one or both of the other knowledge areas. The same is applicable for the professional tester in the team: he needs to have some material - and / or system knowledge. The choice for minimal documentation means that this knowledge can be acquired by early involvement in the project. Participation in Reviews and high-level tests in an early stage are useful activities

The tester gets slightly different tasks than in a traditional project. The accent lies more on controlling the appointments that are made and supporting the user by helping to identify the possible weaknesses. This is described in paragraph 4.2.2.

The text above is about putting together of the testing team for the high-level tests. In general, indirect involvement of professional testers is for low-level testing the best suitable approach. For example this can by advising about the best way to organize the test, or by training and coaching low-level testers in necessary testing knowledge and skills.

4.8.2 Management

Several times we have emphasized the fact that components and the iterative way of working will be reused repeatedly, therefore tests also will be repeated more than once. Good management of test ware is an important precondition for repeatability of tests.

For components counts that management of test ware is part of managing the component. Herewith a clear insight can be given in the quality of the component and can be prevented that components have to be tested over and over again.

Bibliography

- Aonix, (2000), *Select Perspective Workflows*, version 1.0
- Aonix, (2000) *Perspective and Testing*, version 0.1
- Aonix, (2000) *Aonix' Select Perspective*, version 1.0
- Broekman, B., (1998), *RAT, Rapid Application Testing*
- Bunk, E.B., (1999) *Testen binnen DSDM,* Ernst & Young Consulting
- Butler Group (1998), *Component-Based Development Report*
- DSDM Consortium, (1997), *Dynamic Systems Development Method, version 3*
- DSDM Consortium, (1999) *DSDM: practical testing guidelines (white paper)*
- Koomen, T., (1999) *TComp, testen bij component based development*, versie 1.1
- Kruchten, P. (2000), *The Rational Unified Process, An Introduction, Second Edition*, Addison-Wesley, ISBN 0-201-70710-1
- Pol, M., R. Teunissen, E. van Veenendaal (1999), *Testen volgens TMap*®, Tutein Nolthenius, 's-Hertogenbosch, ISBN 90-72194-58-6
- Rational Software Corporation (1997), *UML Summary version 1.0*
- Willems, R. (2000), *DSDM and TMap®, a working team 1.0,* Cap Gemini Ernst & Young Nederland B.V.

© 2001 IQUIP Informatica B.V., Diemen (The Netherlands)

QWE2002 Paper 10A

Mr. Erik Simmons
(Intel Corporation)

Product Triage: A Medical Approach to Predicting and Monitoring Product

Key Points

- The field of Trauma Triage provides many useful metaphors for software quality.
- Software risk assessment and monitoring can improve with similar 'Triage' techniques.
- ROC curves and other mechanisms can be used to monitor the process.

Presentation Abstract

The medical profession has long relied on triage to predict which patients require enhanced care or monitoring based on presenting risk factors or indicators. By comparing the field of trauma triage to software product development, several new and useful insights can be gained into risk assessment and monitoring. In addition, the methods and work done in developing the triage criteria used to classify and assess patients can be translated to software engineering, providing a robust way to establish, utilize, and maintain criteria for software.

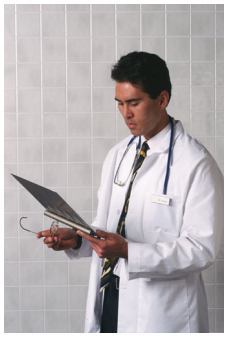
About the Author

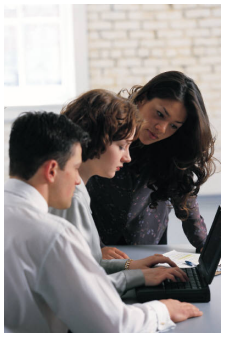
Erik Simmons has 15 years experience in multiple aspects of software and quality engineering. Erik currently works as Platform Quality Engineer in the Platform Quality Methods group, part of the Corporate Quality Network at Intel Corporation. He is responsible for Requirements Engineering practices at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

Product Triage

A “Medical” Approach to Predicting and Monitoring End-Product Quality

Erik Simmons, Intel Corporation





Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

For information, contact:
Erik.Simmons@Intel.com

Trauma Triage

Triage is the process of prioritizing responses to patients based on the severity of the presenting symptoms.

In only a few seconds, an EMT on scene must decide whether a trauma system activation should be called for an injured patient.

This decision is based on experience and about two dozen Triage Criteria – true/false conditions that are associated with the need for trauma system activation.

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

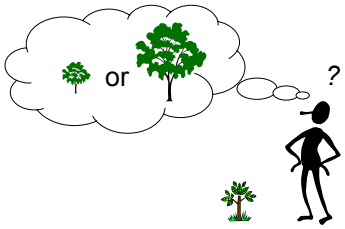
Truths about Prediction

There are two fundamental but conflicting truths about predictions and estimates:

- The value of the prediction increases with distance from the event being predicted

BUT…

- The difficulty of making accurate predictions also increases with distance from the event being predicted



Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Measures, Metrics, and Indicators

A **measure** establishes the "extent, dimensions, capacity, etc. of anything, especially as determined by a standard"*

A **metric** is a typically a composite of two or more measures

An **indicator** is the result of a comparison of a measure or metric with a baseline quantity or expected result

For Example

Measures: Number of software failures, time

Metric: Mean Time Between Failures (MTBF)

Indicator: MTBF < 200 hours

*Adapted from *The Handbook of Software Quality Assurance, 3rd Ed.,* by Schulmeyer and McManus, PTR 1998

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Measures, Metrics, and Indicators

There are two type of indicators: *Sentinel* and *Rate-based*

Sentinel indicators are triggered by any occurrence of the condition

Rate-based indicators are triggered when a metric falls above or below established limits

Example:

Sentinel Indicator: Any patient death occurring outside of the Intensive Care Unit

Rate-based Indicator: Average on-scene time of more than 10 minutes prior to patient transport

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Sensitivity and Specificity

The **sensitivity** of an indicator or test is how well it detects the problem of interest – the percentage of all true cases captured.

The **specificity** of an indicator or test is how well it excludes test subjects without the problem of interest – the percentage of all false cases rejected.

A good indicator has high sensitivity *and* high specificity

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

ROC Curves

Receiver Operating Characteristic (ROC) Curves were created to plot signal-to-noise ratios in electronics, but they have also been widely used in medicine to analyze diagnostic tools.

ROC Curves used in trauma care usually contain Sensitivity plotted against Specificity Loss (1-Specificity) for various probabilities of a case being positive.

A ROC Curve shows how Specificity Loss varies with Sensitivity.

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Sample ROC Curve Data

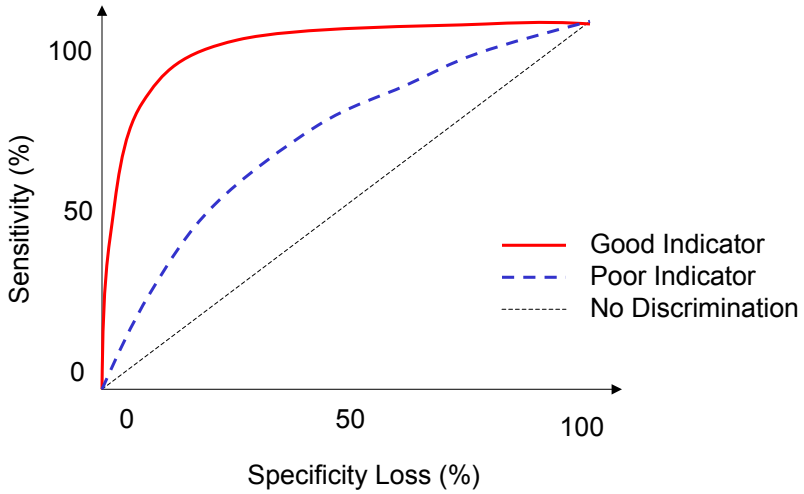
Probability Cutoff for Positive Cases	Resulting Sensitivity	Resulting Specificity Loss
90%	15%	5%
75%	50%	15%
50%	75%	25%
25%	89%	35%
10%	93%	95%

In this example, the “sweet spot” is at 25% probability of occurrence, where sensitivity is 89% and specificity loss is 35%.

Increasing sensitivity to 93% increases specificity loss to 95%.

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Sample ROC Curves



Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Using ROC Curves

A **good indicator** comes close to the upper-left corner of the plot. This means that a large gain in sensitivity is available without much specificity loss.

A **poor indicator** lies closer to a 45-degree line in the plot, indicating that sensitivity gain and specificity loss occur at similar rates.

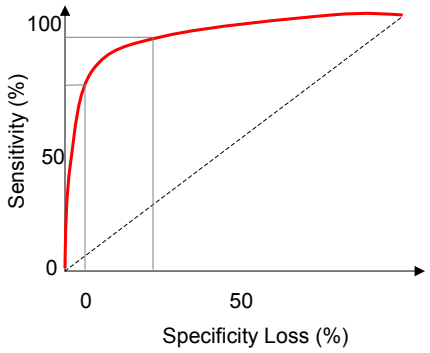
Where to set the probability cutoff depends on things like the cost of false negatives and false positives.

For example, two HIV tests were developed:

- An inexpensive first pass with high sensitivity and poor specificity
- An expensive follow-on with high sensitivity and high specificity

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Using ROC Curves



Q: What is the change in specificity loss when moving from a sensitivity of 75% to 90% for this indicator?

A: About 25%.

If we increase the sensitivity, what will the increase in false positive cases (associated with greater specificity loss) cost the system?

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Using ROC Curves

Like human diseases, most problems in product quality affect only a small proportion of the base population.

An increase in sensitivity of a certain percent (say 15%) will cause some additional specificity loss (say 25%).

Since the negative population is so much larger than the positive population, an additional 20% false positives is a large number compared to additional true positives, and may burden the system heavily.

True Population	75% Sensitive, 5% Spec. Loss	90% Sensitive (+15%), 30% Spec. Loss (+25%)
N=1000, 100 Positive 900 Negative	75 pos. cases, 45 false pos.	95 pos. cases (+15), 270 false pos. (+225)

15 more positive cases found, but at what price?

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Implications for Product Quality

A Few Tough Questions

- Are the indicators used to measure product health known to be associated with customer and end-user quality?
- Which indicators are the best predictors? How do we know?
- Are the indicators evaluated for predictive validity on a regular basis?
- How are new indicators located and deployed?
- Are there combinations of indicators that are more powerful than individual ones?
- Does the experience of a Quality Assurance Engineer, Project Manager, etc. contribute to (or even outweigh) the power of indicators?

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Implications for Product Quality

Imagine a world where:

- A model with high sensitivity and low specificity loss is used early and often in the product lifecycle to predict the need for enhanced quality monitoring, quality assurance, or project management.
- The model is not only proven accurate, but is re-evaluated periodically and adjusted for optimum performance.
- This performance adjustment is made based on a ROC Curve or similar data-driven device so that the expected effects of the adjustment are known before the fact.
- The quest for new indicators is ongoing, and indicators are added based on known predictive validity.

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Product Triage Criteria

Trauma Triage Criteria are broken into 4 categories:

- **Physiology** – Vital signs & alertness problems
- **Injury Anatomy** – Burns, longbone fractures, paralysis, and other types of dangerous & life-threatening injury
- **Injury Mechanism** – Falls > 20 feet, vehicle rollover, high-energy transfer, etc.
- **Co-Morbid Factors** – Pregnancy, age < 12 or > 60, hostile environmental conditions, etc.

What are the parallel conditions in product development?

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Product Triage Criteria

Possible Parallels

- **Product "Physiology"**: Team size, product size, estimated project length, total budget, budget expenditure rate, etc.
- **"Injury" Anatomy**: Schedule slippage, change in scope, significant rework, project restart, missed milestone, loss of sponsorship, excessive turnover, etc.
- **"Injury" Mechanism**: Market changes, unrealistic estimates, inadequate requirements engineering, uncontrolled baselines, & other poor engineering practices
- **Co-Morbid Factors**: Development in multiple locations or cultures, complex products, inexperienced team, inadequate stakeholder access, green business, multi-tasked resources, etc.

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Creating a Model

1. Develop a list of candidate Product Triage Criteria.
2. Define precisely what is meant by ‘bad outcome’ – cancelled project, late by x%, overbudget, etc.
3. Understand the costs involved (tests, false positives, and false negatives) in order to make a good model.
4. Measure the ability of the candidate criteria to predict products that are at risk of a ‘bad outcome’
5. Develop and tune a predictive model based on costs, acceptable sensitivity, and acceptable specificity loss.
6. Pilot and deploy the model, adjusting as needed.
7. Maintain, re-evaluate, and tune the model over time with new data.

Copyright © 2000 Intel Corporation. No part of this presentation may be copied without the written permission of Intel Corporation.

Product Triage:

A "Medical" Approach to Predicting and Monitoring End-Product Quality

Erik Simmons
Intel Corporation
JF1-46
2111 NE 25th Ave.
Hillsboro, OR 97214-5961
erik.simmons@intel.com

Version 1.0, 03/29/01

Prepared for Quality Week 2001

Key Words: Product Quality; Metrics; Leading Indicators; Risk

Author Biography

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. He holds a Masters degree in mathematical modeling and a Bachelors degree in applied mathematics from Humboldt State University in California.

Abstract

The medical field of trauma care provides interesting and valuable metaphors for product quality. By adapting some of the tools and concepts used within trauma care to product quality rather than patient outcome, a new and disciplined approach to predicting and monitoring product quality emerges. As in trauma care, the decision to render specialized care and interventions in product triage is governed by the use of triage criteria. These criteria are binary conditions that are known through analysis and studies to be correlated with poor outcomes and the need for enhanced care. Sensitivity and specificity are defined and used to quantify the performance of indicators. Receiver Operating Characteristic (ROC) Curves are introduced as a very good way to quantify the predictive power and cost of a system of indicators. Parallels between trauma triage and product development are presented and explored. A process for developing a set of product triage criteria is given.

Introduction

Motor vehicle crashes are the leading cause of death in the United States among children and young adults 5 to 27 years of age, and crashes ended more than 41,000 lives overall in the US in 1999 [NHTSA99]. More than three million people were injured in collisions during that same period. Aside from motor vehicles, thousands more people die each year from falls, burns, intentional violence, and other sources of injury.

Trauma care began on the battlefield where severe injury is common, and rapid, effective treatment is essential. The practices and techniques have been refined and extended over time, and many states now have trauma systems established to deal with the challenges posed by traumatic injuries.

Differentiated care for victims of serious injury within trauma systems has been demonstrated to lead to improved outcomes and reduced mortality. Patients with minor injuries tend to do well regardless of the level of treatment rendered, but severely injured patients require treatment in specialized centers containing skilled specialists, enhanced equipment, and advanced techniques to reduce the risk of death and long-term disability. Beyond improved outcomes, trauma centers also shorten the recovery time of patients.

As part of systematized trauma care, Emergency Medical Technicians (EMTs) arriving on the scene assess the condition of the victims using triage. Triage is the process of prioritizing treatment to patients based on the severity of the presenting symptoms. EMTs use a set of triage criteria to perform an assessment of observable or easily measurable risk factors known to be associated with the need for treatment within a trauma center. Although this assessment must be done very rapidly, the accuracy of the system is impressive. During a study performed in the state of Oregon in 1995, 87% of the patients that needed trauma system care were correctly entered into the system by EMTs using a combination of the triage criteria and their experience [Simmons95].

New product development is commonly a risky endeavor, judging by the number of high-profile failures that continue to occur – especially in the field of software engineering. Trauma triage provides some interesting insights into how risk for failure might be measured early, quickly, and accurately. Before these insights can be explored, a few concepts need to be defined and discussed.

Measures, Metrics, and Indicators

Though they are sometimes used interchangeably, the terms measure, metric, and indicator are all distinct. Good definitions are provided in sources like [Schulmeyer98]:

- A **measure** establishes the “extent, dimensions, capacity, etc. of anything, especially as determined by a standard”
- A **metric** is a typically a composite of two or more measures
- An **indicator** is the result of a comparison of a measure or metric with a baseline quantity or expected result

For example,

Measures: Number of software failures, time
Metric: Mean time between failures (MTBF)
Indicator: MTBF < 200 hours

Meaningful decisions are made from metrics through indicators. There are two major categories of indicators: sentinel and rate based. Sentinel indicators are triggered by any occurrence of the condition, while rate-based indicators are triggered when a metric falls outside of established limits. For example, trauma hospitals often have indicators like the following:

- **Sentinel indicator**: Any death occurring outside the Intensive Care Unit (every one is reviewed)
- **Rate-based indicator**: Average on-scene time of more than 10 minutes before patient transport (procedures are reviewed when the rate is exceeded in any review period)

Most indicators are not definitive in demonstrating a problem or condition, but instead define levels of metrics that are associated with that problem or condition (that is, they *indicate* either healthy or unhealthy conditions, but *prove* nothing). A "normal" temperature or blood pressure reading indicates a healthy condition, while a high temperature or blood pressure reading is associated with many different conditions, almost all unhealthy.

The level chosen for an indicator is sometimes set based on a goal or opinion rather than any solid clinical evidence – a practice that can lead to serious problems. Indicators like those used in trauma triage are proven associated with conditions requiring the capabilities of a trauma system hospital. Many indicators have been created in the realm of software engineering, but only a few good efforts to evaluate them have been made, most notably the COCOMO II model for creating cost estimates [Boehm01]. The COCOMO II variables encompass risk reduction efforts and a host of factors known to affect cost, and might serve as a point of departure for any efforts to create a system as described in this paper. Other items from which to work include the published risk taxonomies from various sources, and the program management work of the NASA Software Engineering Laboratory (SEL).

Sensitivity and Specificity

The performance of an indicator can be objectively measured using sensitivity and specificity. The **sensitivity** of an indicator or test is how well it detects the problem of interest – the percentage of all true cases captured. The **specificity** of an indicator or test is how well it excludes test subjects without the problem of interest – the percentage of all false cases rejected.

A perfect indicator would have 100% sensitivity and 100% specificity. However, in practice, such indicators rarely if ever exist, and some specificity loss is accepted in order to achieve an acceptable sensitivity. For example, in the previously cited study on trauma triage criteria [Simmons95], the system performed at 87% sensitivity, and 83% specificity. The 17% specificity loss represents patients entered into the trauma system unnecessarily (also called *over triage*). These patients represent a significant cost to the system, as do the 13% of patients that were not entered into the system but needed the care it provides (*under triage*). In this case, the price of under triage can be human life, but tuning a system of indicators to improve sensitivity can have serious side effects on the system, as demonstrated later.

Individual indicators are Boolean; that is, a condition is either present or absent or a rate is either exceeded or not. This means that individual indicators have only a single sensitivity and a single specificity (though there may be some diagnostic errors or similar factors at work). Using a individual indicator, there are only two possible probabilities for a case being positive: zero or one. In systems where many indicators are used together, such as the one used in trauma triage, each indicator contributes to the sensitivity and specificity of the overall measurement. This means that there are several possible probability levels for positive cases, each with a corresponding sensitivity/specificity pair.

This is important because the various possible sensitivity/specificity pairs can be plotted together in order to understand how the indicators work together as a system. This allows the optimal probability for a case being positive, but more than that, it allows the system to be tuned based on an understanding of what effect changing the probability threshold for positive cases will have. This can be represented visually using Receiver Operating Characteristic (ROC) Curves.

ROC Curves

ROC Curves originated in electronics to plot the signal to noise ratio of receivers, but since an indicator or indicator system seeks to separate signal (positive cases) from noise (negative cases), they work well in this arena also [Hanley82]. A ROC Curve contains sensitivity plotted against specificity loss (1-specificity) for all possible probabilities for positive cases within a given indicator system. Simplified example ROC Curve data are shown in Table 1.

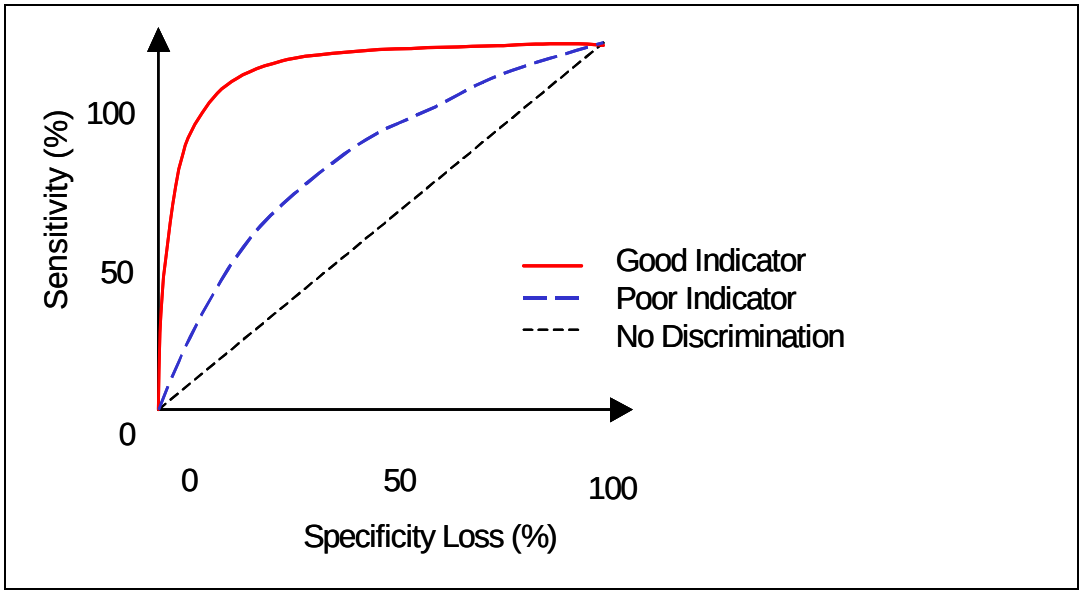
Table 1

Probability Cutoff for Positive Cases	Resulting Sensitivity	Resulting Specificity Loss
90%	15%	5%
75%	50%	15%
50%	75%	25%
25%	89%	35%
10%	93%	95%

In this data, the “sweet spot” is located at 25% probability of occurrence, which yields 89% sensitivity with 35% specificity loss. Note that decreasing the threshold probability to 10% does increase sensitivity to 93%, but at the cost of increasing specificity loss to 95%. This is akin to a system that flags almost every case as positive, locating all but 7% of the positive cases but also taking in all but 5% of the negative cases. In this example, there is a practical limit to what the indicators can do without exorbitant cost to the system.

Figure 1 shows two ROC Curves. In this example, the curve represented by the upper solid line represents a good indicator system like the one in Table 1, and the one represented by the middle dashed line represents a poor indicator system. The diagonal line represents a system with no discrimination power at all – that is, increases in sensitivity are matched with equal loss in

Figure 1



specificity

.

The closer that a ROC Curve gets to a diagonal line, the worse the corresponding indicator system. The more the ROC Curve deviates from the diagonal towards the upper left corner of the plot, the greater the sensitivity gain available for a small specificity loss.

Once the sensitivity/specificity data have been plotted, the question of what threshold probability to use can be answered. The decision is usually made based on several factors, including the cost of diagnostics associated with the indicators, the cost of false negatives, and the cost of false positives. For example, consider a situation where there are two tests used as indicators of the same condition. The first test is inexpensive, highly sensitive, but also has a high false positive rate. The second test is expensive, highly sensitive, and highly specific to the condition. Under these conditions, the first test might be used as an initial screen, followed by the more expensive but more accurate test performed on just the positive population from the first test.

Tuning Indicator Systems and the Cost of Specificity Loss

Like human diseases, most problems with product quality affect only a small percentage of the total population. While sensitivity is very important, specificity loss is often overlooked (at least at first) when designing and tuning indicator systems. Suppose that increasing the sensitivity of an indicator system by 15% from 75% to 90% (through adjusting the threshold probability of a positive case) results in a specificity loss of 25% from 5% to 30%. This sounds OK on the surface, but as Table 2 shows, the effects are startling when the true positive population makes up only 10% of the total:

Table 2

	Before:	**After:**
True Population N=1000	75% Sensitivity 5% Specificity loss	90% Sensitivity (+15%) 30% Specificity loss (+25%)
100 positive 900 negative	75 positive cases found 45 false positives	90 positive cases found (**+15**) 270 false positives (**+225**)

In this example, an additional 15 positive cases were located, but at the cost of 225 new false positives! This would overwhelm a quality organization tasked with deeper diagnosis, monitoring, or intervention. A better test or indicator system created to identify the true at-risk population must be found.

Product Triage

Trauma triage is based on several categories of indicators. Though there are variations between the indicator sets used in different trauma systems, the set used in Oregon is typical and is given in Table 3.

Table 3

Group	**Criterion**	**Description**
Physiology	1*	Systolic blood pressure < 90 mm Hg
	2*	Respiratory rate < 10 or > 29 per minute
	3*	Glasgow Coma Scale < 13
Injury Anatomy	4*	Penetrating wound mid-thigh to head
	5*	Burns to more than 15% of total body surface, or to face, feet, hands, or genitalia (in conjunction with other injuries)
	6*	Amputation proximal wrist or ankle
	7*	Spinal cord injury or paralysis

	8*	Flail chest
	9*	Two or more obvious proximal long-bone fractures
Mechanism of Injury	10*	Death of same car occupant
	11*	Ejection from an enclosed passenger space
	12*	Complex extrication lasting more than 20 minutes
	13	Fall of 20 feet or more
	14	Pedestrian hit at 20 mph or more or thrown at least 15 feet
	15	Vehicle rollover
	16	Motorcycle, ATV, or bicycle crash
	17	Significant impact or intrusion to passenger space
Co-morbid Factors	18	Age < 12 or > 60
	19	Hostile environment (e.g., extreme heat or cold)
	20	Medical illness (e.g., chronic lung diseases, heart failure)
	21	Presence of intoxicants
	22	Pregnancy

* Mandatory system entry for these criteria; for all others system entry is at EMT discretion.

Notice that the triage criteria are broken into four categories: physiology, injury anatomy, injury mechanism, and co-morbid conditions (complicating factors). What parallels can we find to these criteria within product development?

Physiology
These criteria represent the patient's physiology – the 'vital signs'. The vital signs of a product development effort might consist of items like:

- Development team size
- Product size
- Project length
- Total budget
- Budget expenditure rate

Injury Anatomy
These criteria describe the visible signs of the underlying injury. Visible signs of problems for product development include:

- Schedule slips
- Scope changes
- Significant rework
- Project restart
- Missed milestones
- Loss of executive sponsorship
- Excessive turnover

Injury Mechanism
Certain mechanisms are known to be associated with a likelihood of severe injury. Some well-known injury mechanisms for product development efforts are:

- Changing market conditions or stakeholder needs
- Inadequate requirements engineering
- Pressured or otherwise unrealistic estimates (or no estimates at all)
- Uncontrolled baselines
- Other inadequate software engineering processes or practices

Co-morbid Factors
While not injury causes or symptoms themselves, these factors can exacerbate or complicate injuries. For product development efforts, such factors include:

- Development in multiple locations or cultures

- Highly complex products
- Inexperienced team
- Insufficient stakeholder access
- Multitasking resources on more than one project
- Lack of product domain experience

These categories represent new ways of thinking about product risk. In triage, the emphasis is on early, predictive risk assessment. Using the results, product development efforts could be entered into a kind of "trauma system" offering differentiated care as indicated by the project's symptoms.

Imagine a world where:

- A model with excellent sensitivity and low specificity loss is used early and often in the product lifecycle to predict the need for enhanced quality monitoring, quality assurance, project management, or similar activity during development.
- The model is not only proven accurate, but is re-evaluated periodically and adjusted for optimum performance.
- This performance adjustment is made based on a ROC Curve or similar data-driven device so that the expected effects of the adjustment are known before the fact.
- The quest for new and improved indicators is ongoing, and indicators are added based on known predictive validity.

While establishing such a vision will not be simple, the process to create the system is straightforward:

1. Develop a list of candidate Product Triage Criteria.
2. Define precisely what is meant by 'bad outcome' – cancelled project, late by x%, over budget, etc.
3. Understand the costs involved (tests, false positives, and false negatives) in order to make a good model.
4. Measure the ability of the candidate criteria to predict products that are at risk (as defined in step 2) and need enhanced monitoring, management, or other intervention[1].
5. Develop and tune a predictive model based on costs, acceptable sensitivity, and acceptable specificity loss.
6. Pilot and deploy the model, adjusting it with each experience.
7. Maintain, re-evaluate, and tune the model over time with new data.

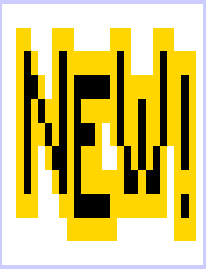
References

NHTSA99 *Traffic Safety Facts 1999,* National Highway Traffic Safety Administration DOT document HS 809 092

Simmons95 Simmons, Erik, et al, *Paramedic Injury Severity Perception Can Aid Trauma Triage*, Annals of Emergency Medicine 26:4, October 1995

Hanley82 Hanley, J.A., and McNeil, B.J., *The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve*, Radiology 1982; 143:29-36

[1] Logistic regression, classification and regression trees, and other statistical techniques can be used for this step, depending on the nature of the indicator data.



QWE2002 Paper 11A

Mr. Kie Liang Tan
(CMG)

Configuration Management in a Test Centre

Key Points

- Getting test process under control
- Management of services by test centre
- For good quality and good time-to-market of SW

Presentation Abstract

This author will present the method to manage the configuration of the testware and the items to be tested in a test centre. A brief description of the method is as follows:
* define the configuration item of the physical and logical components;
* define the level of detail of the hardware, software and documentation;
* develop the data model and the relationships of the configuration items;
* define the information to be stored in the database (configuration management database);
* develop Dynamic Software Library and Dynamic Test Library with different environments for development, test and production;
* set-up the change management process and release management process which are the important processes as interface to the clients; these are actually the processes to provide the services to the client of the test centre;
* implement the Configuration Management Database, the Dynamic Software Library and Dynamic Test Library;

About the Author

Mr. Kie Liang Tan has studied (1971 1977) in the Technical University of Delft as Electrical Engineer with field of specification Information Technology and Data Communication. He has 23 years experience in ICT (Information Communication Technology) in different areas, i.e: Telecommunication, Medical Systems, Space Systems, Finance, Trade, Transport and Industry. He has significant experience in the area of test and verification coordination and management. He has joined CMG since 1998 and worked in the TestFrame, Software Testing and Test Centre as senior management consultant.

Configuration Management in a Test Centre

Kie Liang Tan
CMG – TestFrameTest Centre
kie.liang.tan@cmg.nl

Agenda

- Test Centre
- Purpose of Configuration Management
- Configuration Management Processes
- Configuration Item Identification
- Configuration Control
- Configuration Status Accounting
- Configuration Auditing
- Conclusion

2

Test Centre

- Provide test services (test outsourcing)
- Test environment management
- Testware management (test scripts, test cases, test data)
- Test preparation, execution and reporting
- Test consultancy

3

Purpose of the configuration management

- to establish and maintain the integrity of the products throughout the whole lifecycle.

4

Configuration Management processes

- **Configuration Item (CI) Identification**
- **Configuration Control**
- **Configuration Status Accounting**
- **Configuration Auditing**

5

Configuration Item Identification

- Determine Configuration Item and the level of detail
- Determine directory structure and naming convention
- Determine required documentation

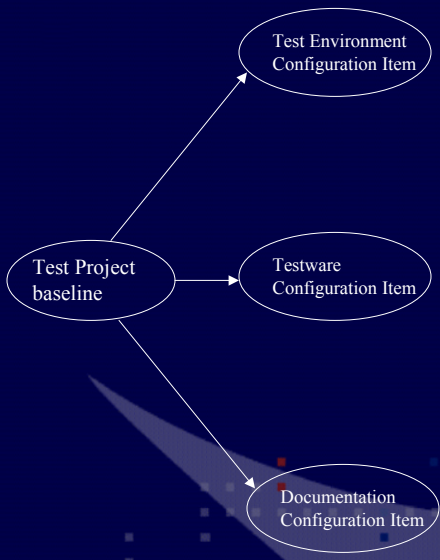
6

Determine Configuration Item and level of detail

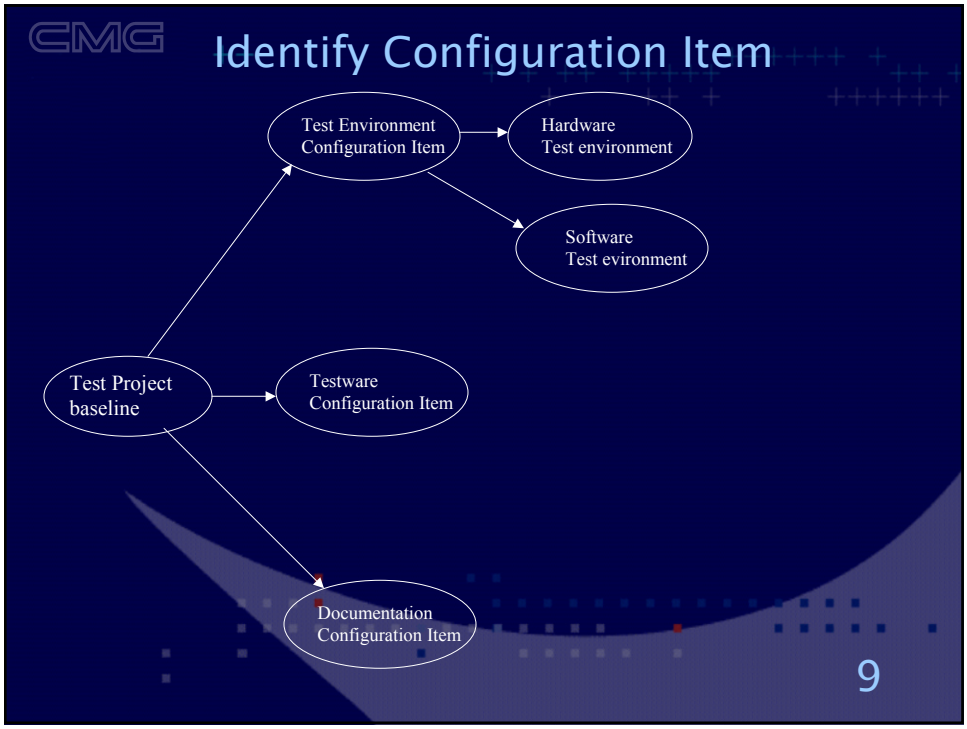
- Test environment
  - Hardware and software
- Testware
  - Test cases, test scripts, test data
  - Test results
- Documentation
  - Requirements, design, test
  - Standards and procedures

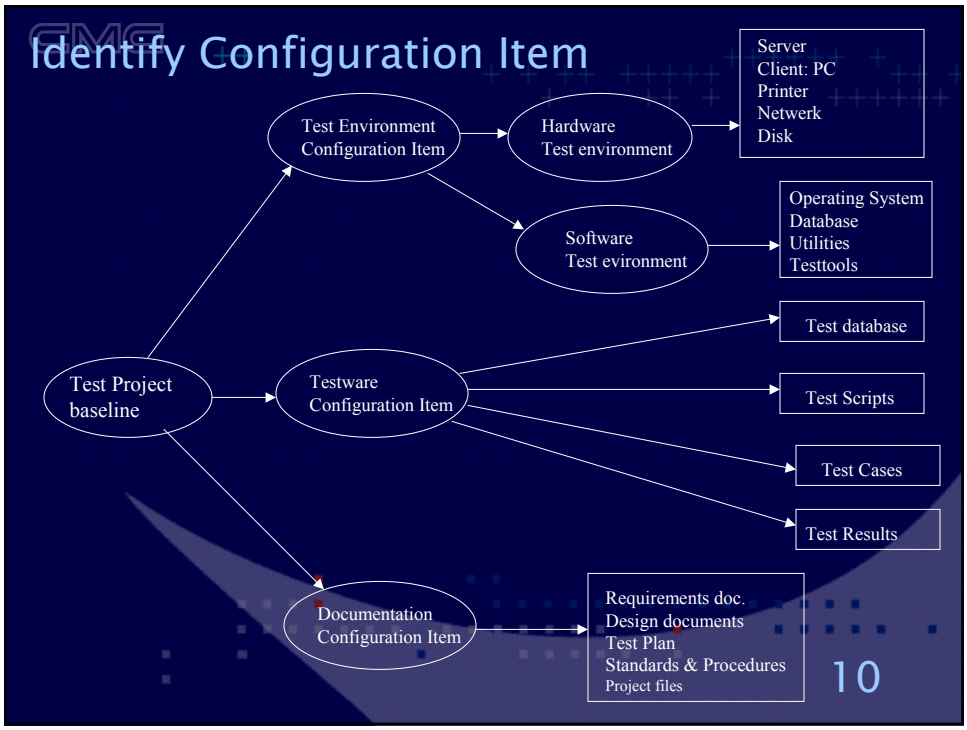
7

Identify Configuration Item



8


Identify Configuration Item
Test Environment Configuration Item
Hardware Test environment
Software Test evironment
Test Project baseline
Testware Configuration Item
Documentation Configuration Item
9

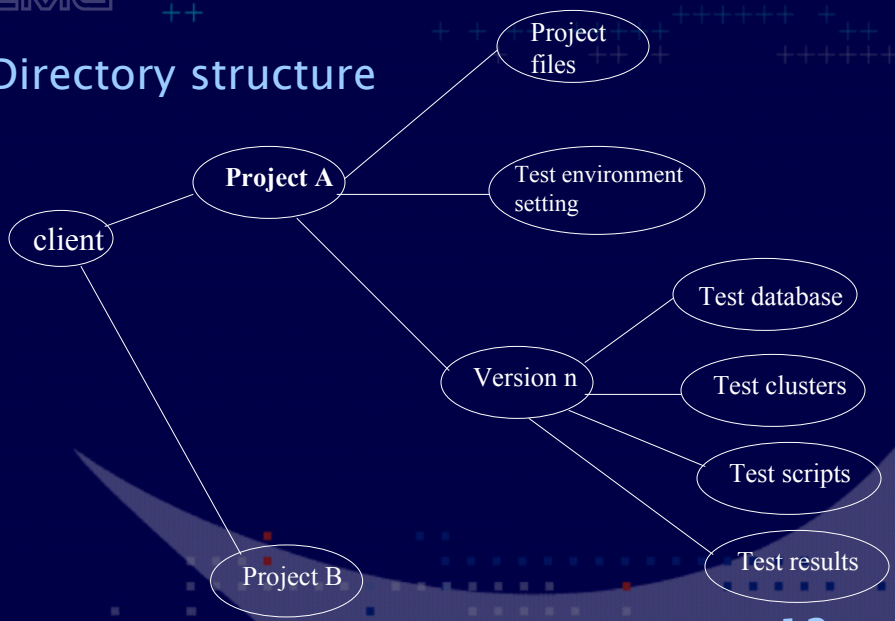

Identify Configuration Item
Server
Client: PC
Printer
Netwerk
Disk
Test Environment Configuration Item
Hardware Test environment
Operating System
Database
Utilities
Testtools
Software Test evironment
Test database
Test Project baseline
Testware Configuration Item
Test Scripts
Test Cases
Test Results
Documentation Configuration Item
Requirements doc.
Design documents
Test Plan
Standards & Procedures
Project files
10

Configuration Control

- Submitted service request (with release note of the application)
- Evaluate the sevice request
- Review the service request
- Implement the changes in the testware
- Test and verify

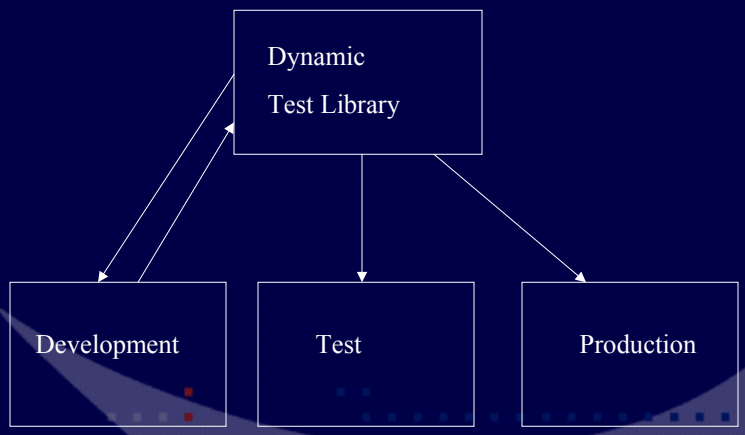
11

Directory structure



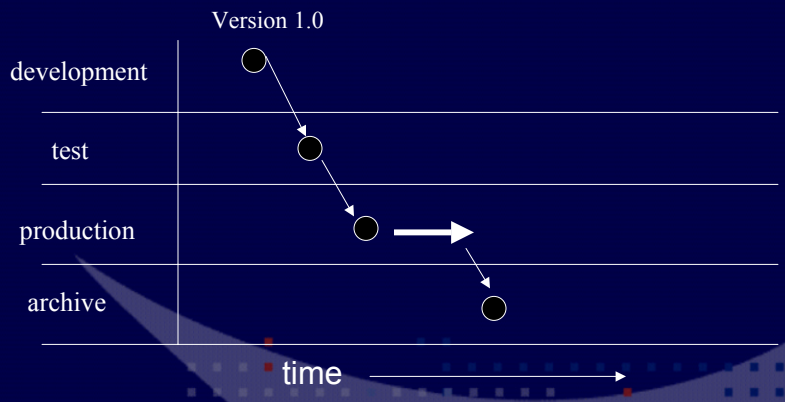
12

Dynamic Test Library



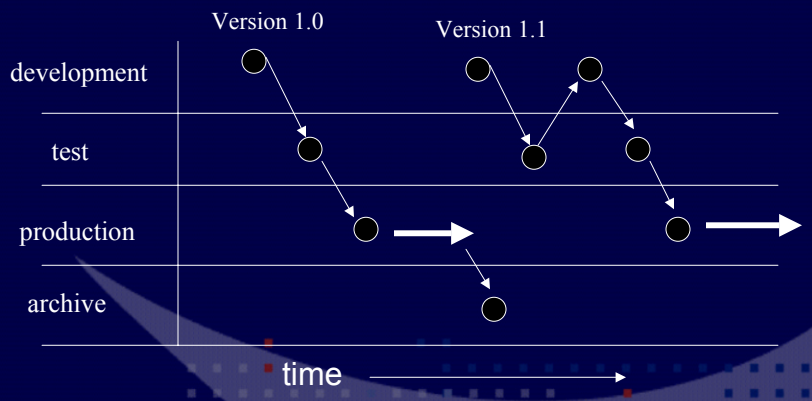
13

Version Control



14

Version Control



15

Configuration Status Accounting

- The recording and reporting of information needed to manage configuration
  - listing of the configuration item
  - the baseline
  - status of the changes to the configuration

16

Configuration Auditing

- Project baseline audit
- Test audit
- Conform to 'Test Centre Audit Procedure'
  - Audit results are reported to all involved
  - Action items are tracked to closure

17

Project baseline audits

- Performed on a regular base (e.g. every month)
- Check the structure and contents of the Test Library against the documentation baseline

18

Test Audit

- Conducted at end of the test related to a service request
- Examine all change requests to ensure that they were appropriately approved or incorporated
- Examine test failures and ensure that corrective actions have been taken
- Verify that test results indicate successful completion of the test
- Verify that test reports represent the actual version of the system

19

Conclusion

- Configuration management ensure the integrity of the product
- Configuration management effect quality
- Good configuration management is essential for a Test Centre

20

Configuration Management in a Test Centre

Summary

This paper describes the Configuration Management of the test environment and the testware (test scripts, test cases, test database and documentation) for a Test Centre. This is based on our experience in various projects in our Test Centre.

A Test Centre is a centralised test environment and test organisation to perform test services for the clients.
The services are:

- Managing the test environment (test infra-structure, i.e.: hardware, software, test tools);
- Managing the testware;
- Performing test preparation, execution and reporting;
- Giving support in test consultancy.

The following processes are relevant for a Test Centre:

- Configuration management of the test environment and the testware: software control and distribution management, change management and documentation management;
- Facility management: test environment infrastructure management, computer operations, network management, automatic operation and preventive/ corrective maintenance;
- Availability, capacity management and contingency planning;
- Help desk, problem management and customer liaison;
- Test management: test plan, test preparation, test execution, archiving test results and test reporting.

The purpose of configuration management is to plan, organise, control and coordinate the identification, storage and change of software, hardware and documents through the whole lifecycle (development and maintenance phases).
No matter how large the project may be, software configuration management has a critical effect on quality. Good configuration management is essential for efficient development and maintenance, and for ensuring that the integrity of the software is never compromised.

According to the definition of IEEE Standard, configuration management process is:

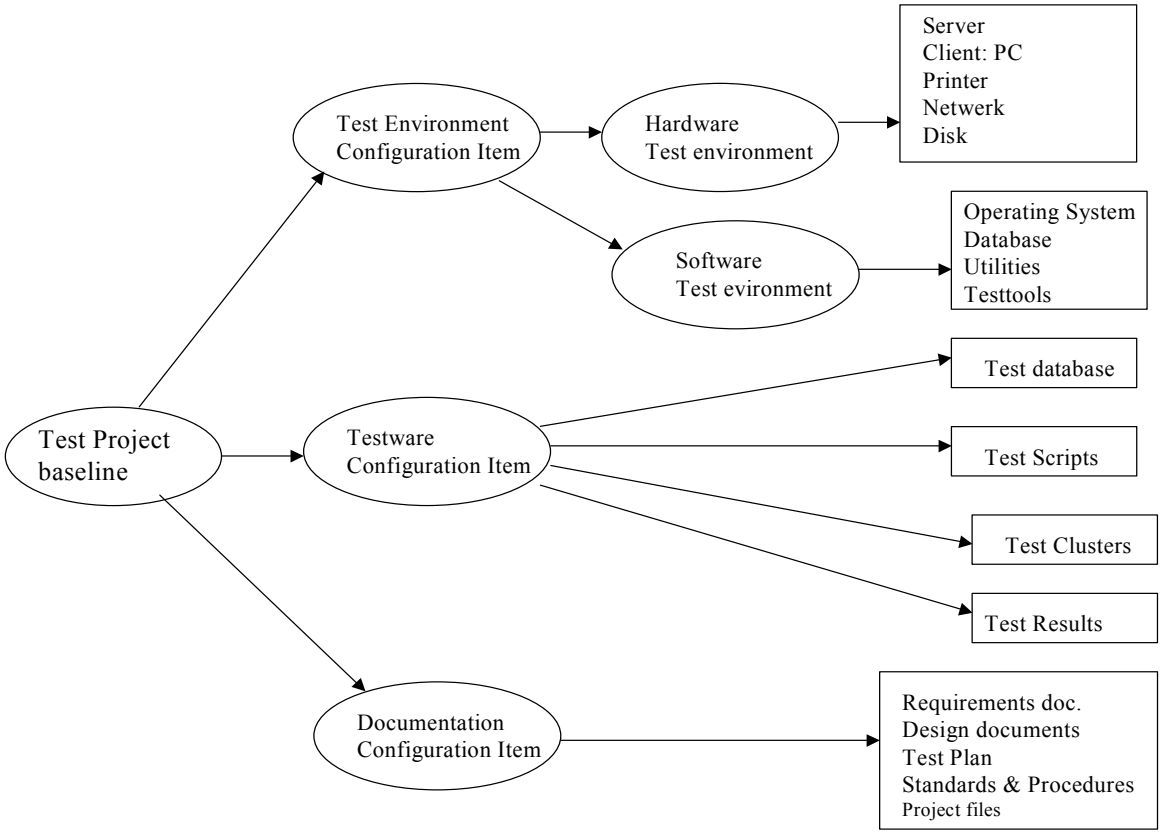
1. process of identifying and defining the configuration items in a system;
2. controlling the release and change of these items throughout the whole lifecycle;
3. recording and reporting of the status of configuration items and change request;
4. verifying of the completeness and correctness of configuration items.

1 Identifying and defining configuration items

The subject matter of all configuration management activities is the 'configuration item' (CI).
A CI is a collection of hardware or software elements, treated as a unit, for the purpose of configuration management. A CI could be physical or logical component.
The level of detail of the CI needs to be defined, i.e.: hardware, software or documentation with the level of detail for specifying them.
For example: we define a PC as a CI or the mouse, the memory or the graphics card as a CI.

The Configuration Item for the Test Centre (see figure) are as follows:

- The first level is decomposed as test environment, testware and documentation;
    - The test environment CI consist of hardware and software;
        - The hardware consist of client, server, printer and network;
        - The software consist of Operating System, Database and Tools;
    - The testware CI consist of test clusters (test cases), test data, test scripts and test results;
    - The documentation consist of requirements documents, design documents, test plan, standards and procedures.



2 Controlling the changes and the release of configuration item

2.1 Configuration change control

A service can be requested by a client of the TestCentre i.e.: service request with release note of the application to be tested from the client containing information about the changes to the application, the planning when the application will be ready and the relevant documentation (such as: requirements document, functional design, technical design).

The following activities will then be performed by the Test Centre:

- Evaluating the service request and the release note,
  - Evaluate on correctness and completeness;
  - Check if the changes are already submitted (duplication);
  - Make a rough estimation of the impact of this service request to the testware .
- Reviewing the service request in the Test Centre Configuration Control Board:
  - Analysing the request;
  - Assessing the request with regard to the impact on the testware;
  - Assigning the request to a release;
- Implementing the changes in the testware
- Test and verifying the changes:
  - Run system test to test implementation of the changes;
  - Run regression test to test that the changes will not have negative effect on the existing functionalities.

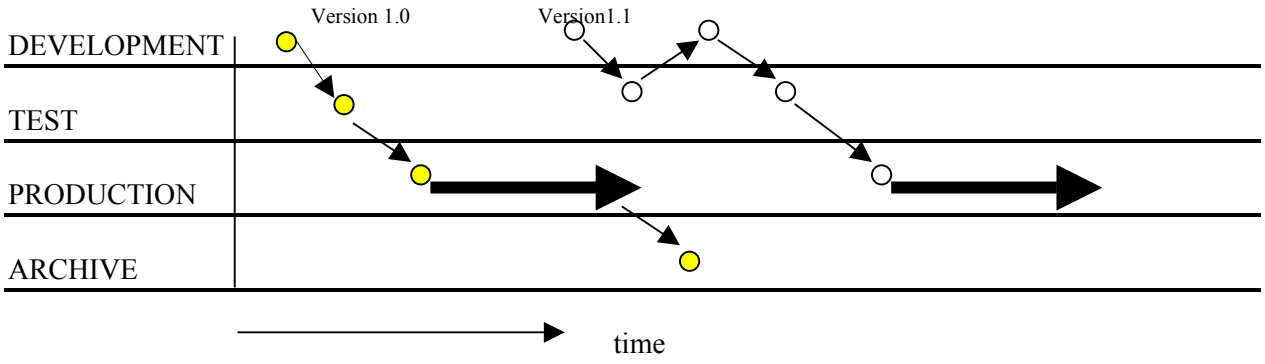
2.2 Configuration item storage

The Test Centre uses the method to store the Testware sets in the Testware Library as configuration items and having a version number.
The individual testware item that make up the content of each type of set do not have their own version numbers. The effect of this that whenever anything in the Testware set is changed, a new version of the Testware set is created containing the changed item and the unchanged items.
A baseline is something that has a version number but consists of a number of configuration items, each version of a baseline being built from specific versions of the configuration items.

The testware configuration items are stored in the Dynamic Test Library (DTL).
This DTL consists of an archive version and the current new version of the testware.
In order to have a complete picture, the application software in the DSL (Dynamic Software Library) and the related testware in the DTL, will be stored together.
If the old version has to be tested, then the application software and the testware can be restored from the archive and a test can be performed immediately.

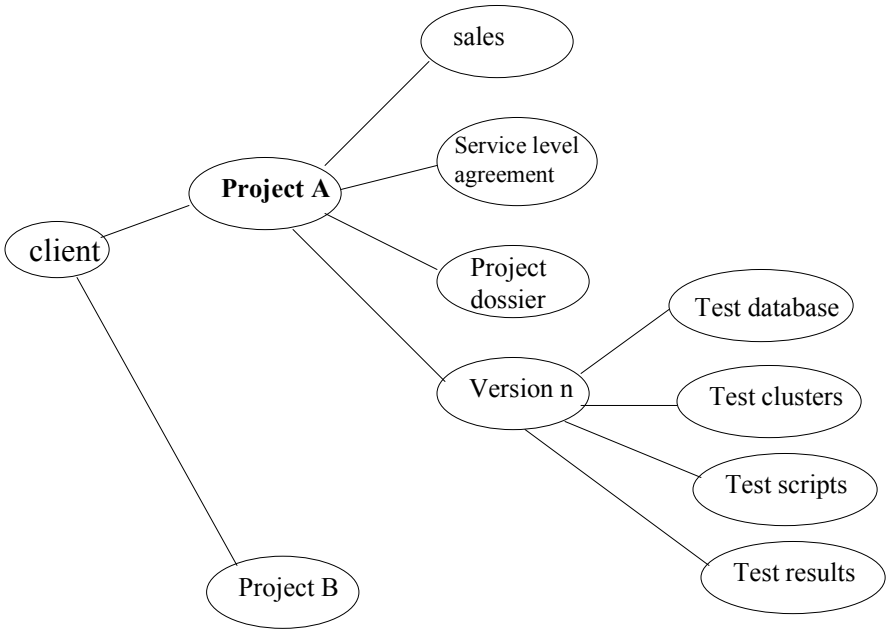
The version control is as follows:
The next phases are defined: development, test, production and archive; for each phase an environment is created. A copy of the application software and the testware may exist in one of the environments.
During the development phase, the current version of the testware is retrieved from the DTL and stored into the development environment.
If the first version is available from the development environment, this will be put in the DTL.
During the test phase, a copy of the current version of the testware is retrieved from the DTL and put in the test environment.
If the test is performed successfully then the testware is moved from the test environment into the production environment.

At the moment that maintenance has to be performed of the production environment, then a copy from the DSL/ DTL will be made available to the development environment.
At the moment that a new version exist with a new version number, the copy from the production environment will be archived.
The following figure gives an overview:



2.3 Directory structure

For each client of the test centre, the following structure will be maintained:



Project A
All information of each project is stored under project name in the related subdirectory.

Sales
In this subdirectory all sales related information are stored.

Service Level Agreement
In this subdirectory the service level agreement related items are stored.

Project Dossier
In this subdirectory all project plans, standards and templates, correspondence, minutes of meeting, progress reports are stored.

Version n
In this directory the different versions of the testware are stored.
For each version the test database, test clusters, test scripts and the test results are stored.
For the test environment, the image of the standard test station (Operating System and tools) configuration is stored on CD and on a server. The project depending configuration setting of the test environment will be stored separately.
Before test execution, the following will be loaded consequently:
the standard image of the test station;
the project depending setting of the test station;
the test database;
the test scripts;
the test clusters
After the test, the test results are stored.

2.4 Access rights to the directories

Only the Test Coordinator may change the project dossier and the test results.
Only the Test Analyst may change the test database and the test clusters.
Only the Test Navigator may change the test scripts.
The sales and service level agreement directories can only be changed by the Operation Manager of the Test Centre. The operation manager has full control of all directories.

Directory	Test Coordinator	Test Analyst	Test Navigator	Operation Manager
Sales	R	R	R	F
Service Level Agreement	R	R	R	F
Project Dossier	C	R	R	F
Test database	R	C	R	F
Test clusters	R	C	R	F
Test scripts	R	R	C	F
Test results	C	R	R	F

R = Read only
C = Change
F = Full Control

3 Configuration status accounting

Configuration status accounting is the recording and reporting of the information that is needed to manage a configuration effectively, including a listing of the approved configuration identification, the status of proposed changes to the configuration, and the implementation status of the proposed changes.
Configuration status accounts are important instruments of project control, and should be made available to all project members for up-to-date information on baseline status. When a test works one day and not the next, the first question is "what has changed?". The configuration status accounts should answer such questions quickly and accurately.
The Test Centre Quality Manager is responsible for maintaining the configuration status accounts, which are composed of records that describe the contents, dates and version/issue of each baseline;

Configuration status is presented as tables (see table).

Configuration Item	**Baseline number**				
	1	2	3	4	5
Requirement doc.	1.0	1.1	2.0		
Functional Design	1.0	2.0	3.0		
Technical Design	1.0	3.0	4.0		
Test Plan	1.0	1.1	2.0		
Test Clusters		1.0	2.0		
Test Scripts		1.2	2.0		
Test Data		1.0	2.0		
Operating System		5.1	5.1		
Test tools		4.4	4.4		
Database		3.2	3.2		
Application to be tested		1.1	2.0		

Each column of the table lists Configuration Items in a baseline. The table shows the issue and revision number of each document and the version number of the testware and the code.
The revisions to each document are required between each baseline to show how the version numbers change.
The changes to the configuration kept in the Configuration Item history.
Also a problem report database, containing the status of problem reports and their solutions , exist in the Test Centre which is also part of the configuration status accounting.

4 Configuration auditing

Audits are performed in the Test Centre by the Quality Manager conform to the "Test Centre Audit Procedure", i.e.:

- Project baseline audits: performed at a regular basis (every month).
  The structure, contents and facilities of the Test baseline library will be examined to verify that baselines conform to the documentation describing baselines.
- Functional Test Audit: performed at the end of the configuration change control flow related to a service request from a client. This audit will verify that test results indicate successful completion of all requirements scheduled for the release. Trace test results back to the requirements. Verify and validate that all requirements have been met.

The audit results are reported to all involved and the action items are tracked to closure.

QWE2002 Paper 12A

Mr. Sridhar Narayanan
(Cognizant Technology Solutions)

Metrics Cockpit - means of "viewing" the project

Key Points

- Automated Metrics collection and consolidation.
- Real time display of metrics in the form of cockpit dials.
- How this helps is proactive decision making and achieving business transparency.

Presentation Abstract

It's an old joke that IT development projects are on track until a week before the deadline, at which point the customer discovers that the project is really six months behind schedule. The developers may have worked long hours, but it often turns out that nobody fully understands the status of their project deliverables. One reason the status and deliverables are unknown is that the developers' processes are 'hidden' from the project manager and the end users; it is almost as if the developers were coding in the dark. Good project management goes hand in hand with formal, disciplined processes that are carried out consistently; everyone knows what deliverables are associated with each activity and there should be very few surprises along the way. But what about the end users who have been burned on previous projects and who are not passionate believers of CMM? They might ask, "How do I know that your developers are following a formal process? If the Project Manager says, "Here's a description of the process, and it's the same one we followed in our earlier projects." The cynical customer might still say, "We don't care about your earlier projects! How do we know you're following it this time?" There are two familiar ways to answer this question. A) Produce frequent prototypes throughout the project, and B) produce voluminous progress reports, updated weekly. But in today's 'Internet time' environment, that may not be enough. A mission critical e business project may last only 4 weeks. If it takes a full week to find out that a wrong system is being built, 25% of the time is already lost.

An alternative, as Cognizant Technology Solutions is demonstrating is to make the software process completely visible so that the customer doesn't have to wait passively for a week before the project team delivers a status report. The project teams use a formal process that is implemented in a web enabled workflow automation mechanism and the project team performs all its work within that environment. The automated process is "instrumented" with appropriate metrics in the form of a cockpit, allowing end users to literally watch the process in motion, in real time through their web browsers. This paper detail how the metrics cockpit enable end users to observe in real time, requirements being created, modified and approved, design reviews, with metrics about the number of defects that were identified, and track the defects to see if any festering problems threaten the projects' success. This paper also detail the benefits in adopting this approach, towards achieving business transparency.

About the Author

Sridhar Narayanan has over 15 years experience in the areas of Software Development, Testing, Software Quality Management (system implementation, audits, facilitation, and training) and Metrics. He specialises in Software Process Improvement and has assisted several organisations in successfully achieving higher maturity levels of CMM. He has presented papers in the 11th and 12th SEPG conferences held in USA and 5th European SEPG conference held in Amsterdam. He received postgraduate degrees in Quality Management and Operations Management from College of Engineering, Madras and graduate degree in Mechanical Engineering from the Institution of Engineers (India).

Metrics Cockpit

- means of 'viewing' the project

Sridhar Narayanan

Cognizant Technology Solutions

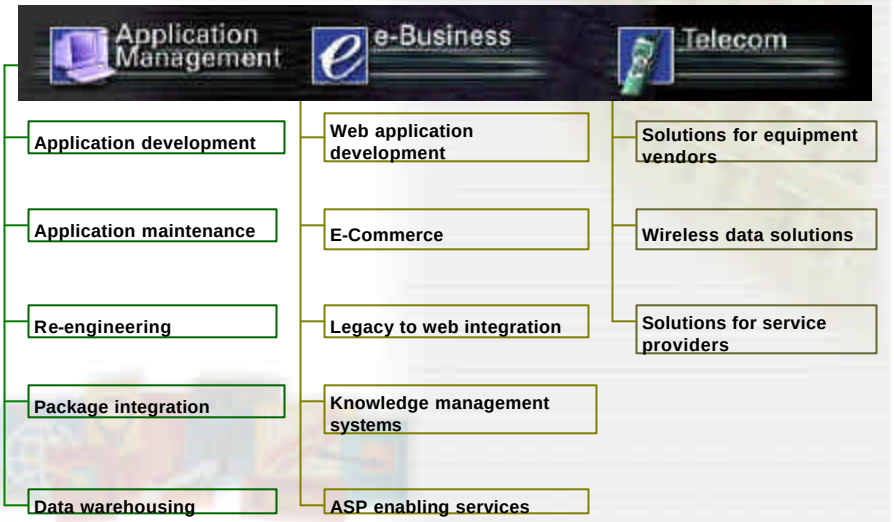
Organization Background

- Founded in 1994 as an in-house development center for The Dun & Bradstreet Corporation
- Shifted to third-party revenue in 1996
- Expanded from 50 software professionals to 3700 in 7 years
- Operation based on onsite - offshore model
- Full range of offerings in e-business and application management, across technologies

Metrics Cockpit

- means of 'viewing' the project

Solution Offerings

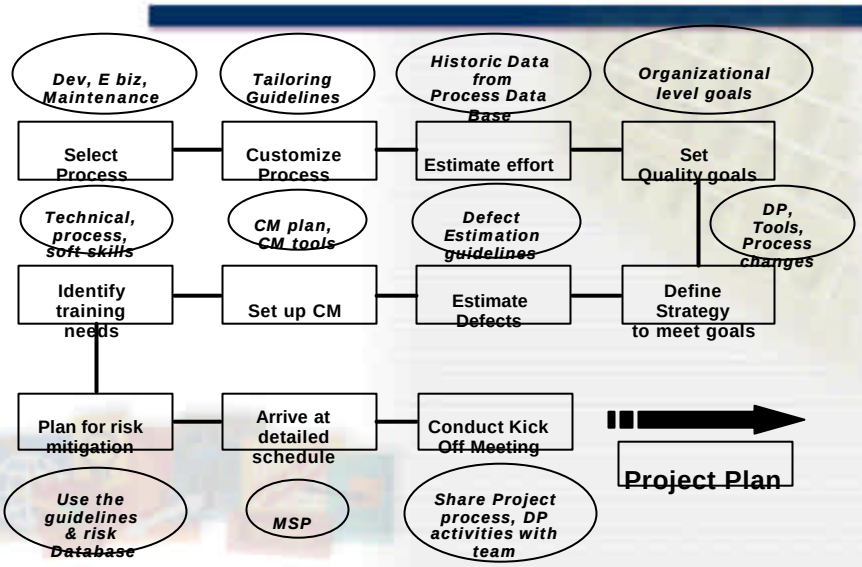


Ó Sridhar Narayanan

Metrics Cockpit

- means of 'viewing' the project

Project Activities - Planning

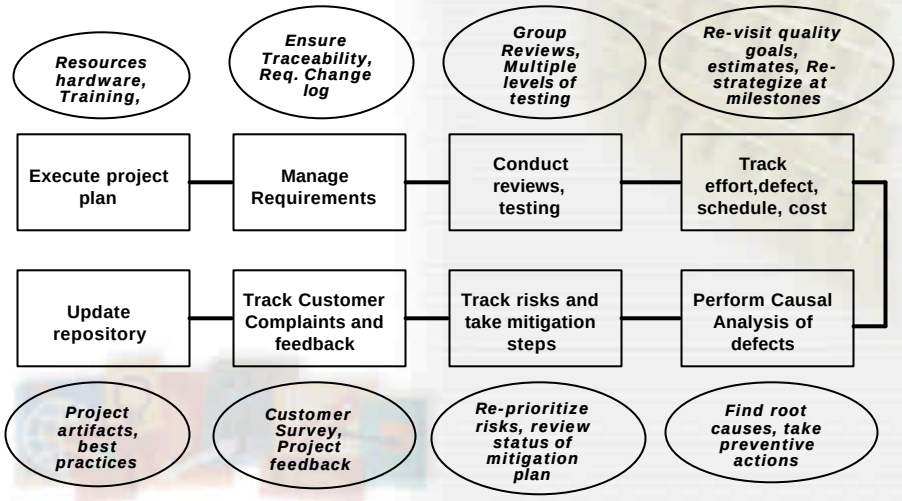


Ó Sridhar Narayanan

Metrics Cockpit

- means of 'viewing' the project

Project Activities - Tracking



Ó Sridhar Narayanan

Metrics Cockpit

- means of 'viewing' the project

Process Black box

Hidden Process

Coding in the dark

Lack of consistency and discipline

Surprises at the end

Ó Sridhar Narayanan

Metrics Cockpit
- means of 'viewing' the project

Status reporting - Usual Methods

Text based Weekly Status reports

Mostly sent through Mail

Not real time and Post facto

90% complete 90% of the time

No clear understanding of *actual* status

Ó **Sridhar Narayanan**

Metrics Cockpit
- means of 'viewing' the project

Cognizant Metrics Framework

Bottom up Approach

Measures ⇨ Metrics ⇨ Goals

Collection of

– **Core measures** (Effort, Duration, Defects, Size and Requirements) across all Operational Models, along with Customer / Project specific measures

Leading to

– Derived / Computed **Metrics**

Satisfying

– Organizational / Project-specific **Goals**

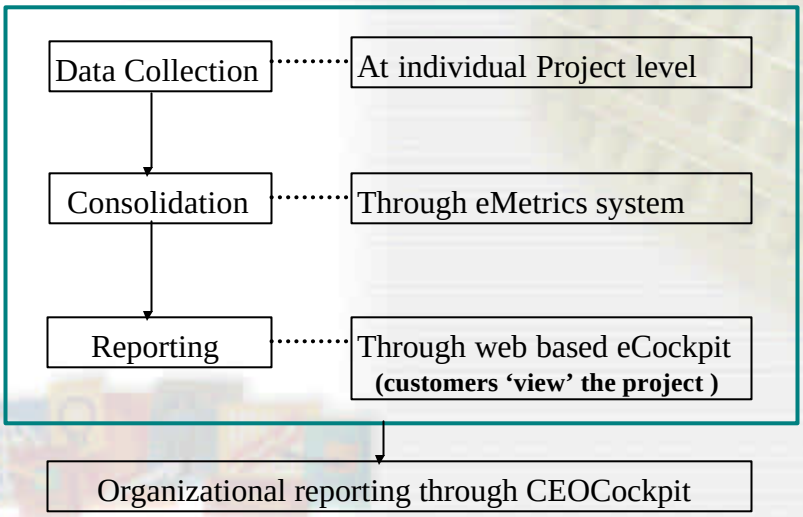
Contrary to the popular GQM approach

Ó **Sridhar Narayanan**

Metrics Cockpit

- means of 'viewing' the project

Cognizant Metrics Process

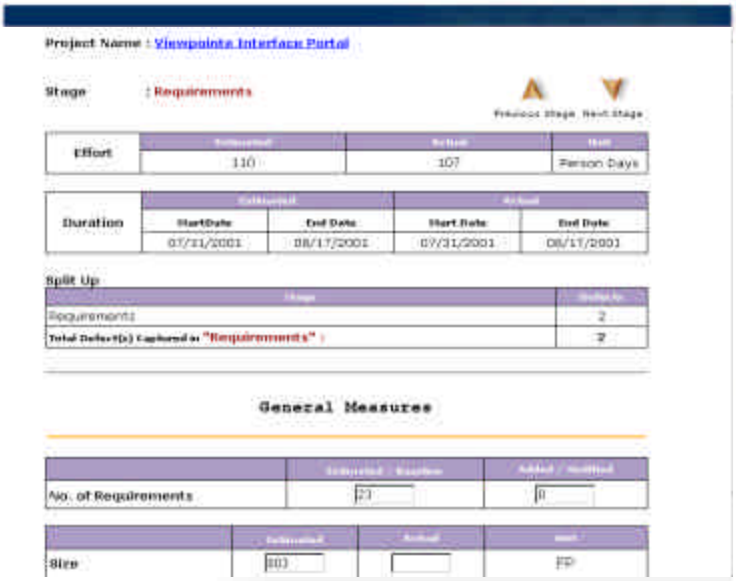


Ó Sridhar Narayanan

Metrics Cockpit

- means of 'viewing' the project

eMetrics System



Ó Sridhar Narayanan

Metrics Cockpit

- means of 'viewing' the project

Metrics Cockpit

- Metrics Cockpit represents project performance indicators in the form of a cockpit
- Similar to gauges in an aircraft cockpit, that indicate altitude, fuel, speed, etc., the gauges in Metrics Cockpit can be set to indicate project-related status such as Productivity, Effort, and Defect density, or personal metrics such as Targets, Individual progress and Cash flow
- The Metrics Cockpit visually portrays the performance of the chosen parameter through predetermined colour scheme

Ó Sridhar Narayanan

Metrics Cockpit

- means of 'viewing' the project

Cockpit Creation

Ó Sridhar Narayanan

Metrics Cockpit
- means of 'viewing' the project

Metrics Cockpit - Continued

- Dial / Gauge colour changes based on over performance or under performance of the chosen parameter
- Dials have drill down feature, allowing users to view the trend and history of performance for the chosen parameter

Ó Sridhar Narayanan

Metrics Cockpit
- means of 'viewing' the project

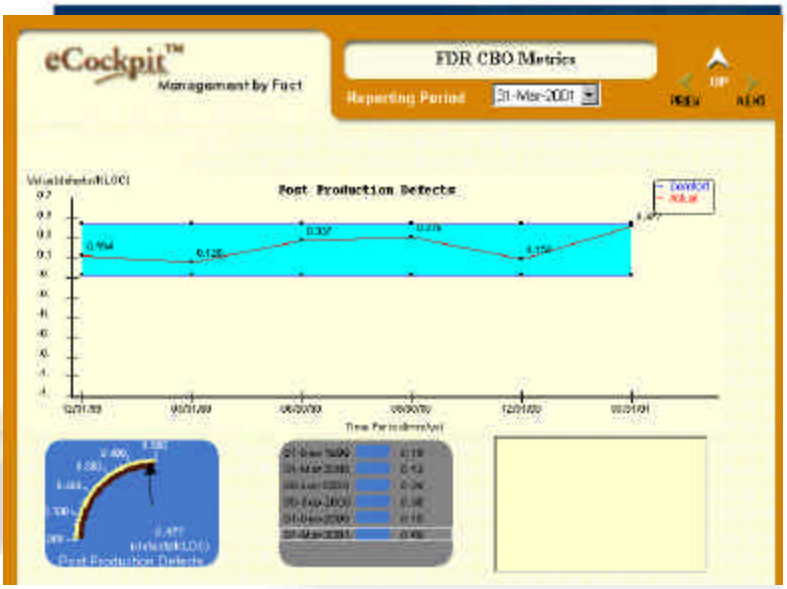
Project Cockpit

Ó Sridhar Narayanan

Metrics Cockpit

- means of 'viewing' the project

Project Cockpit - Drill down



Ó Sridhar Narayanan

Metrics Cockpit

- means of 'viewing' the project

Effective Management

- Visible insight into Project performance
- Timely analysis of Metrics for Trend
- Proactive Decision making
- Transparency in operations
- Management by Fact

Ó Sridhar Narayanan

Metrics Cockpit
- means of 'viewing' the project

Organization Performance Tracking

- Organization wide Metrics Analysis

  To determine

  – Organization Process Performance

  – Organization Process Capability

  with respect to

  – Delivery Capability

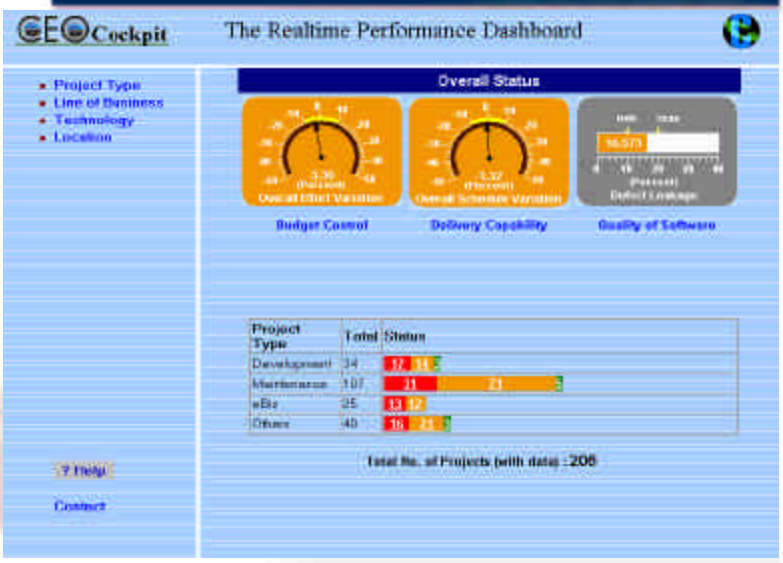
  – Budget Control

  – Quality of the Software

Ó Sridhar Narayanan

Metrics Cockpit
- means of 'viewing' the project

CEOCockpit



Ó Sridhar Narayanan

Metrics Cockpit

- means of 'viewing' the project

Availability

http://cockpit.cognizant.com

User ID : guest

Password : password

Metrics Cockpit

- means of 'viewing' the project

Contact Information

Sridhar Narayanan

Cognizant Technology Solutions

226, Cathedral Road, Chennai, India

Phone : + 91-44-8113063 Extension 2213

Mail : nsridhar1@chn.cognizant.com

: metrics_sridhar@hotmail.com

Metrics Cockpit – means of 'viewing' the project

Sridhar Narayanan
Cognizant Technology Solutions

COPYRIGHT NOTICE

Copyright (2001-2002) by the Author.

All rights reserved. These materials are confidential and proprietary to Sridhar Narayanan, the author and no part of these materials should be reproduced, published in any form by any means, electronic or mechanical including photocopy or any information storage or retrieval system nor should the materials be disclosed to third parties without the express written authorization of Sridhar Narayanan, the author.

Author Description

Sridhar Narayanan

Sridhar Narayanan has over 15 years experience in the areas of Software Development, Testing, Software Quality Management (system implementation, audits, facilitation, and training) and Metrics. He specialises in Software Process Improvement and has assisted several organisations in successfully achieving higher maturity levels of CMM. He has presented papers in the 11th and 12th SEPG conferences held in USA and 5th European SEPG conference held in Amsterdam. He received postgraduate degrees in Quality Management and Operations Management from College of Engineering, Madras and graduate degree in Mechanical Engineering from the Institution of Engineers (India)

E-mail: metrics_sridhar@hotmail.com
Sridharnarayanan@netscape.net

Summary

This presentation gives an insight into the representation of the Software Project Metrics and Performance Measures like Budget variation, Schedule variation, Productivity, Defect Leakage and Defect Density etc, in the form of web enabled cockpit dials and gauges. This representation is a totally revolutionary concept and allows for real time project status monitoring and assist in proactive decision making. Customers get a visible insight into the project status and can manage the project effectively.

Key Words

Metrics, Cockpit, Dials and Gauges, Web, Status reporting, Business Transparency, Performance monitoring.

Organization Background

Cognizant Technology Solutions (CTS) is a leading e-business and application management service provider with 10 development centers in India and marketing offices across various locations in USA and Europe. The software development and maintenance activities are carried out using the unique onsite – off shore model. The operations are carried out with major emphasis on formal disciplined process. The structured approach to quality has resulted in successful ISO certification in 1996, SW CMM level 4 assessment in 1998, ISO Recertification in 1999 and SW CMM level 5 assessment in 2000.

Post facto Status reporting & Process Blackbox – the real life scenario

It's an old joke that IT development projects are on track until a week before the deadline, at which point the customer discovers that the project is really six months behind schedule. The developers may have worked long hours, but it often turns out that nobody fully understands the status of their project deliverables. One reason the status and deliverables are unknown is that the developers' processes are 'hidden' from the project manager and the end users; it is almost as if the developers were coding in the dark. Good project management goes hand in hand with formal, disciplined processes that are carried out consistently; everyone knows what deliverables are associated with each activity and there should be very few surprises along the way. But what about the end users who have been burned on previous projects and who are not passionate believers of CMM? They might ask, "How do I know that your developers are following a formal process? If the Project Manager says, "Here's a description of the process, and it's the same one we followed in our earlier projects." The cynical customer might still say, "We don't care about your earlier projects! How do we know you're following it this time?" There are two familiar ways to answer this question. A) Produce frequent prototypes throughout the

project, and B) produce voluminous progress reports, updated weekly. But in today's 'Internet time' environment, that may not be enough. A mission critical e business project may last only 4 weeks. If it takes a full week to find out that a wrong system is being built, 25% of the time is already lost.

An alternative is to make the software process completely visible so that the customer doesn't have to wait passively for a week before the project team delivers a status report. The project teams use a formal process that is implemented in a web enabled workflow automation mechanism and the project team performs all its work within that environment. The automated process is "instrumented" with appropriate metrics in the form of a cockpit, allowing end users to literally watch the process in motion, in real time through their web browsers.

Metrics Cockpit – means of 'viewing' the project

Metrics Cockpit™ is a tool that graphically represents quantifiable data in the form of gauges, thereby providing a consolidated and intuitive "picture" of project / organization metrics. Similar to gauges in an aircraft cockpit, that indicate altitude, fuel, speed, etc., the gauges in Metrics Cockpit™ can be set to indicate project-related status such as Productivity, effort, and defects density or other metrics such as targets, individual progress and cash flow. Metrics Cockpit™ is a powerful decision support and project management tool – stated simply, it is a visible performance dashboard enabling management by fact.

In a typical project environment, metrics are collected to effectively monitor the progress of the project. Metrics Cockpit™ functions as a management information system and also aids in defect prevention.

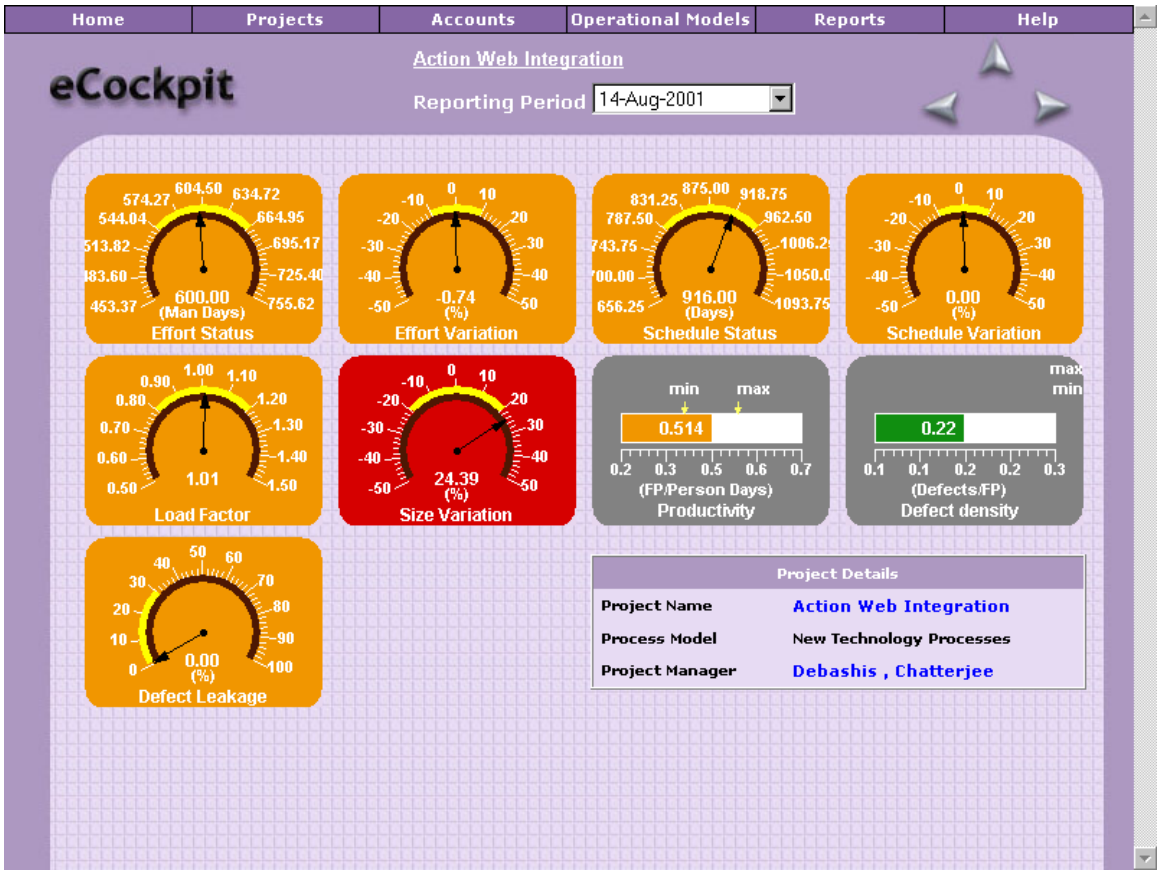
Some screen shots of the cockpit representation of project performance are shown in following pages.

Overall

The first screen gives the overall performance indicators, which include

Effort Status
Effort variation
Schedule Status
Schedule variation
Load Factor (ratio of budgeted cost and actual cost)
Size Variation
Defect Leakage
Defect Density
Productivity

The comfort zone (shown by yellow band on the dial) represents the benchmarks. The colour scheme is based on the characteristic of the indicator. e.g. For Productivity, the higher productivity is better. Hence the colour of gauge will be green if productivity is above benchmarks. Similarly, lesser defects in the software are better. The gauge colour will be green when the defect density (Total number of defects / Actual size) is less than the benchmark

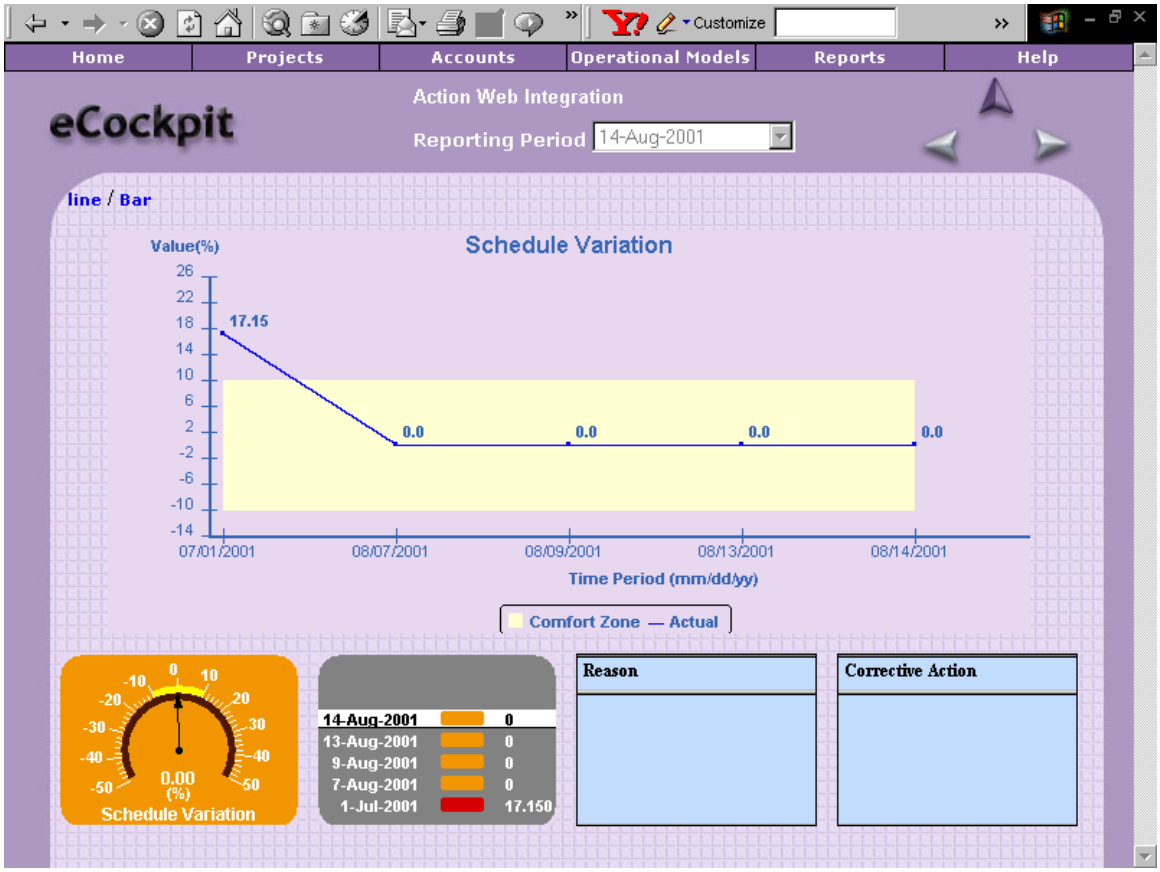


First Level Drill Down

Each gauge has a drill down which gives the performance trends over a period of time.

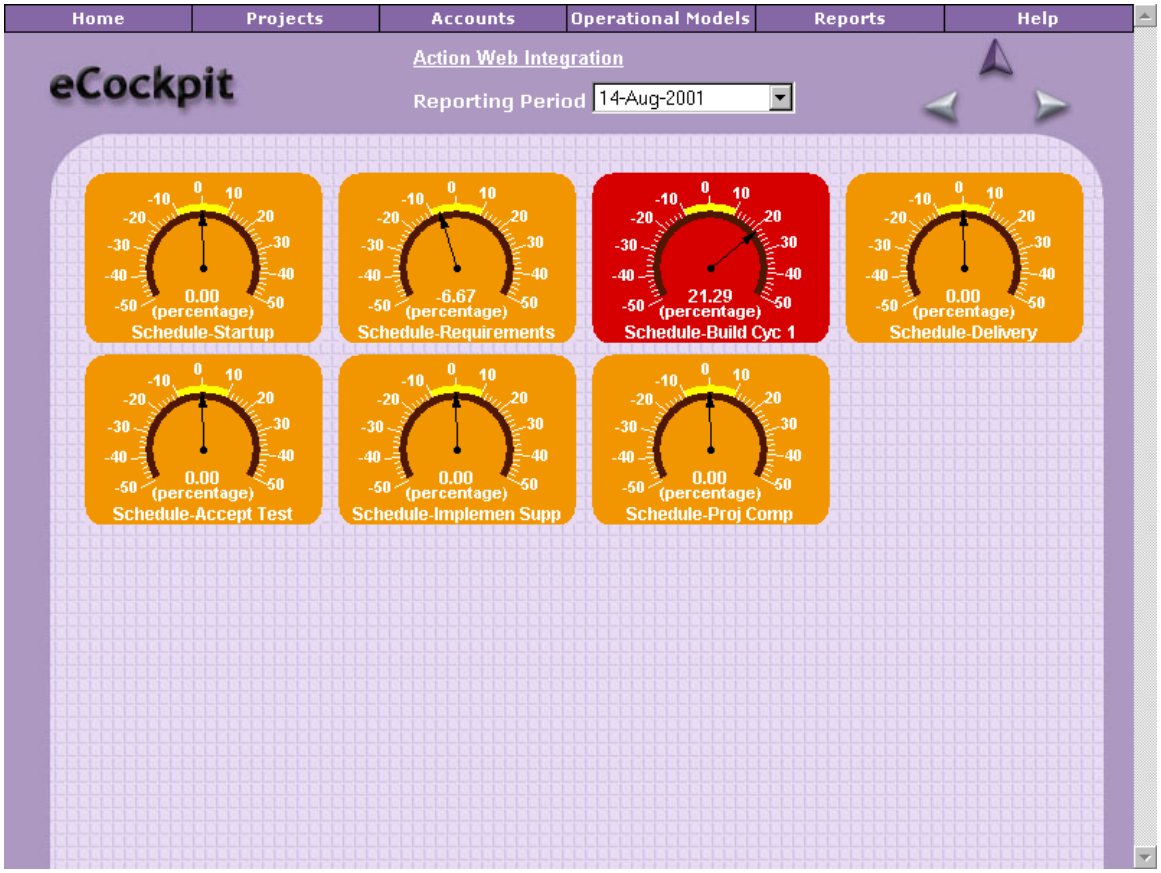
The performance is shown with respect to the benchmarks in the form of history gauge.

The reasons for deviation and corrective actions are also indicated



Second Level Drill Down

Further drill down gives details at individual stages / time period in a project.



Transparency through **Metrics Cockpit™**

Metrics Cockpit™ provides a very good insight for customers on the project's performance. The customer can get to know the schedule/ effort variation, productivity, SLA variation, defect density and many other facts about the project. This information helps the customer to effectively plan and identify risks, if any. When there is a negative trend, the customer can initiate timely action by analyzing the cause and taking remedial measures.

Business Transparency achieved through Metrics Cockpit™ has helped the customers to know exactly what's happening on the projects in Cognizant. Due to this, customers are not subjected to last minute surprises. The insight aids customers to take proactive decisions, and minimizes the Lead-time for issue resolution leading to timely delivery

QWE2002 Paper 1I

Ms. Julia Gonzalez, Mr. Gullermo Lafuente, Mr. Luis Olsina, & Mr. Oscar Pastor
(Extremadura University)

Providing Automated Support for Web Metrics

Key Points

- A Methodology to assure the quality of Websites.
- Definition of some automated metrics where data are gathered by our tool.
- A catalogue to defined metrics and reused by different evaluators.

Presentation Abstract

Automating support for many of the processes in an evaluation project allows minimizing efforts and costs as well as avoiding oversights and imprecisions introduced by the human intervention. Having as a part of the Web evaluation process a quantitative methodology that allows quantifying the quality of Web applications and a toolkit to support it are lines of research and development of our concern. In this article, we analyze about twenty catalogued automated Web metrics, as well as the Website MA (Metric Automation) tool, which provides data collection and calculation to those metrics. In addition, we illustrate the level of support given by this tool to the defined evaluation process underlying the Web Quality Evaluation Methodology (WebQEM). Effective and accurate measurement and evaluation processes require not only methodological but also technological support.

About the Author

Julia Gonzalez Rodriguez Education: Engeneering in Computer Science (UEX). Position: Assistant professor and Researcher. Currently, she continues researching about Web-site quality evaluation and the support tools (Website MA and WEbQEM_Tool). They were running case studies on museums, academic sites, and e-commerce bookstore sites. Besides, he is investigating web metrics validation, and web evaluation process models. Focused in Accesibility and Quality in use, she is making is studies of post-graduated now.

Luis Olsina Santos Education: Doctor in Science in the Software Engineering Area (UNLP) and Master in Software Engineering (UNLP). Licenciado in Information Systems (UNLu). Computer Programmer Analyst (UNLPam). Position: Professor and Researcher and GIDIS Coordinator Currently, he continues researching about the quantitative methodology for Web-site quality evaluation and the support tools (Website MA and WEbQEM_Tool). They were running case studies on museums, academic sites, and e-commerce bookstore sites. Besides, he is investigating web metrics validation, and web evaluation process models. On the other hand, he is researching distributed software architectures in order to implement an application for the control and monitoring of Web-centered domotic functionality.

Web accesibility group

Providing automated support for web metrics

Julia González
Web Accesibility Group
Computer Systems Department
Extremadura University, Spain

Luis Olsina, Guillermo Lafuente
Computer Systems Department
La Pampa University, Argentine

Oscar Pastor
Informatic Systems and Computing Department
Polytecnic University of Valencia, Spain

Contents

- Web artifact like software
- Evaluation methodology
  - Quality requirements definition and specification.
  - Metrics for quality evaluation.
  - Elementary and global evaluation.
  - Analysis of results.
- Conclusion
- Future works

Introduction

- Classification of Web artifacts by their goal
  - **Text**
  - **Hypertext: Berners-Lee**
    - *–Brochures,*
    - *–Spreding reports, etc.*
  - **Aplication with software functionality**
    - *–E-commerce*
    - *–Baking*

- Web design involves new goals
  - **Functional and non-functional requeriments.**
  - **Development and evaluation process.**

Evaluation and a new approach

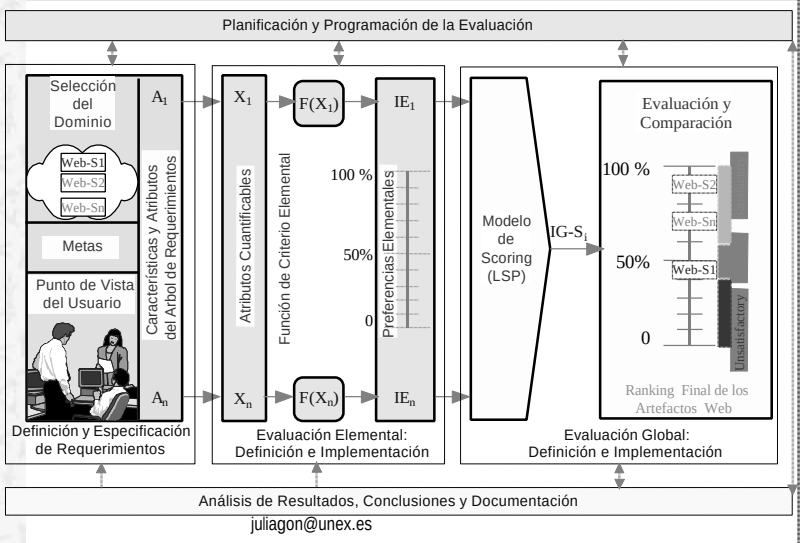
- Is the system **simple or complex**, considering quality evaluation?
- Main decisions by **experts** or by **users**? Which are **objective** and **subjective** parts?
- Evaluation is based on quality metrics and **models**, or on evaluator **intution**?
- Attributes and features valuation procedures centered in **agregation models and quantitative calculus methods** or in **quality models**?

Website QEM

- Evaluating and comparing web artifacts quality with **different complex**.
- Based on quality **metrics and models**.
- Based on attributes and features **agregation models**.
- Judment of domine **expert evaluators**.
- Evaluation process **trazeability**.
- A systematic, quantitative **methodology** focused on Software Engeenering

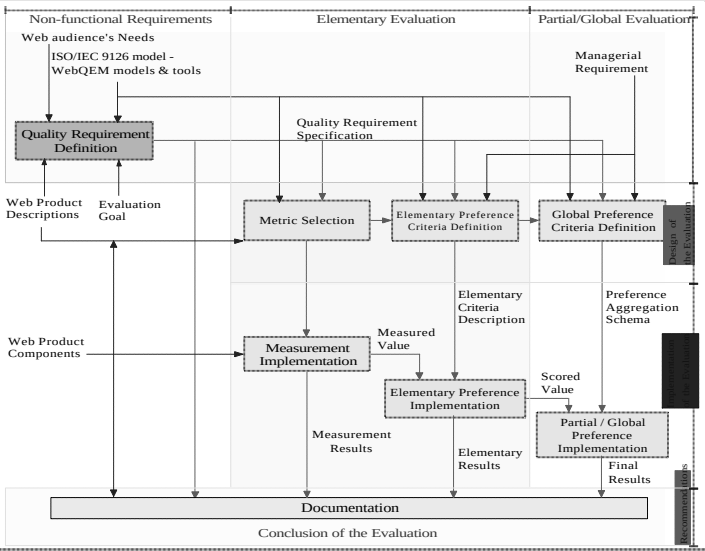
Evaluate to recommend

Website QEM



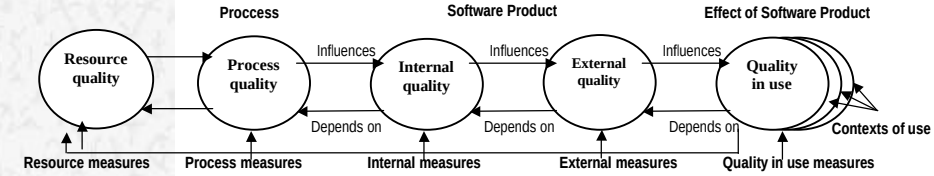
juliagon@unex.es

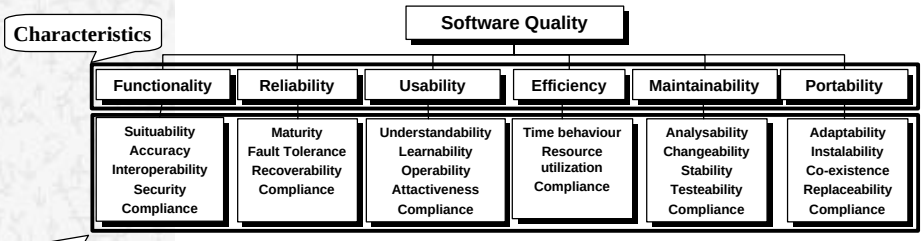
Evaluation process



Quality requirements definition and specification

Quality framework (ISO/IEC 9126-1)





Quality requirements definition and specification

Characteristics and attributes

- Usability
  - **Interfaces and stetic aspects.**
    - **Permanence and stability in main controls appearance.** Direct controls, Indirect controls, Stability
    - **Mantenance in link color**
  - **Global understability of site**
    - **Global organization schema.** Table of contents, site map, index (alphabetics, tematic)
    - **Guided visit**
    - **Image map**
  - **Help and feedback**
    - **Help.** First time visitor, searching.
    - **Directorio de Direcciones.** Telephone-Fax, e-mail
    - **Feedback based on reports.** Comments and suggestes, questionnaires
    - **Feedback based on links.** New, FAQ.

Quality requirements definition and specification

Characteristics and Attributes

- Functionality
  - **Searching and recover**
    - Limited searching, global searching, customized recover
  - **Navegate and explore**
    - Path indicator, current position label, contextual control permanence, stability, title link.
  - **From specific domine (e-commerce)**
    - **Product information.** Product description, content description, product image, catalogue, comparison of prizes, suggestion of related products .
    - **Shopping features.**
      - **Ways of shopping**: on line, shopping-cart, 1-click, check-out, secure transaction
      - **Shopping policies**: returned policies, sending information, pay policies, gift service
    - **Customizing.**
    - **Storage features.**
    - **Promotion policies.** Banners, gifts, appetizers.

Quality requirements definition and specification

Characteristics and Attributes

- Reliability
  - **No deficiency in links.** broken links, non-valid links, non-implemented links.
  - **Broken links (no different)**
    - Amount of internal/external broken links (no different)
    - Amount of all links
    - Percentage of broken links

$$PercentageBrokenLinks = \frac{AmountInternalBrokenLinks + AmountExternalBrokenLinks}{AmountLinks} * 100$$

    - Density of broken links

$$DensidadEnlacesRotos = \frac{CantidadTotalEnlacesRotos}{CantidadTotalEnlaces * CantidadDocumentos}$$

    - Concentration of broken links per node (frecuency)
  - **Broken links different (non repeated)**
    - Amount of internal/external broken links (different)
    - Amount of all different broken links
    - Percentage of different broken links

$$PercentageDiffBroken = \frac{AmountIntDiffBrokenLinks + AmountExtDiffBrokenLinks}{AmountDiffLinks} * 100$$

    - Percentage of redundance of internal/external broken links

$$IntBrokenLinksRedundancy = \left(1 - \frac{AmountIntDiffBrokenLinks}{AmountIntBrokenLinks}\right) * 100$$

Quality requirements definition and specification

Characteristics and Attributes

- Reliability
  - **Other no deficiency.** Orphan pages, pages under construtuion, spelling mistakes, etc.
    - **Orphan Pages.** Amount of pages withous any link pointed into the site.
    - **Percentage of orphan pages**

$$PercentageOrphanPages = 100 * \frac{AmountOrphanPages}{AmountPages}$$

- Efficiency
  - **Performance**
    - **Quick access pages.** t: Download tiem; s: size of the page; c: velocity in conexion.

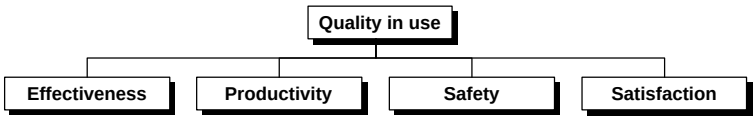
$$t_{Download} = f(s, c) \qquad g(t_{Download}) = \begin{cases} QuickAccess & t_{Download} \le t_{maximus} \\ SlowAccess & t_{Download} > t_{maximus} \end{cases}$$

    - **Accesibility**
      - **Only text version**
      - **Images with title**

$$Percentage\ ALTText = \frac{AmountALTimages}{AmountImages} * 100$$

      - **Use of frames**. To determine the higher number of views.

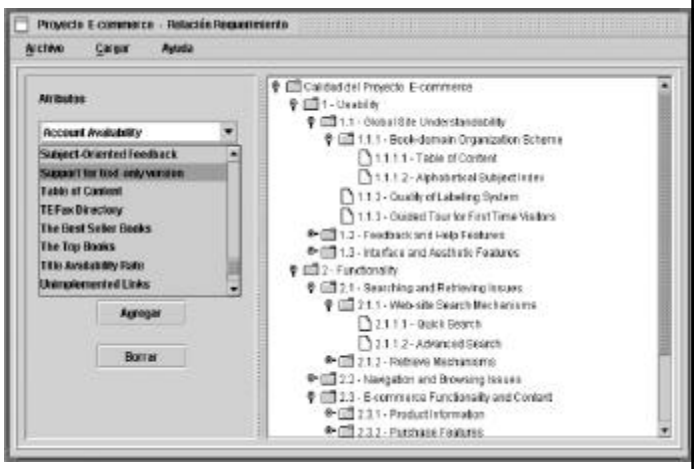
Quality requirements definition and specification



- **Effectiveness**
  - **Level of finished task.**
  - **Average of goals.**
- **Productivity**
  - **Time comsuption in the task.**
  - **Efficiency in completeness.**
  - **Efficiency in reached goals.**
- **Satisfaction.**
  - **Questionaries:** WebQual, SUMI / WAMMI.

Quality requirements definition and specification

Characteristics tree



Elementary evaluation phase

Framework to characteristics and attributes

Title: **Code:**
Type: **Characteristic**
Sub-characteristic (Code/s):
Definition / Comments:
Model to define the global calculus :
Arithmetic/Logic operator:
Used tool:
Weight:
Calculated preference Value/s:
Example/s:

Title: **Code:**
Type: **Sub-characteristic**
Super-characteristic (Code):
Sub-characteristic/s (Code/s):
Attribute/s (Code/s):
Definition / Comments:
Model to define the partial calculus :
Arithmetic/Logic operator:
Used tool:
Weight:
Calculated preference value/s:
Example/s:

Title: **Code:**
Type: **Attribute**
Higher level Characteristic (Code):
Super-characteristic (Code):
Definition / Comments:
Kind of elementary criterion:
Reference template to variables and parameters
Preference scale:
Kind of data collection:
Used tool:
Weight:
Calculated preference value/s:
Example/s:

Julia González
juliagon@unex.es

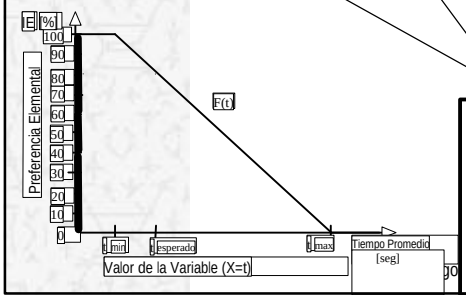
Metrics Catalogue

Código	ECON-010
Nombre	CANTIDAD DE PAGINAS MUERTAS
Objetivo/Motivación	Contar el número de páginas que no tengan enlaces internos de un sitio Web. Esta métrica (en conjunción con otras) nos puede dar información útil en las fases de desarrollo y mantenimiento indicando la ausencia de enlaces internos al sitio, dificultando una fácil de contactar navegacionalmente hacia el sitio.
Descripción	Número total de páginas que no contienen enlaces internos al sitio en las páginas artísticas del sitio analizado. Al momento del conteo de este atributo sólo se contabilizarán páginas que no contengan ningún enlace o que sólo contengan enlaces externos al sitio.
Tipo de entidad Supercaracterística	Producto->Sitio Web Confiabilidad (ISO-9126)
Fórmula	NPMu ?
Tipo de atributo	Directa/Externo
Procedimiento de aplicación	Recorrer el sitio recursivamente analizando todos los enlaces de las páginas, considerando exclusivamente aquellos que son internos y no estén rotos, y han sido visitados o no, anteriormente. Para cada nodo destino, interno al sitio, si no contiene ningún enlace interno almacenar el nodo junto al URL que lo referencia. Algoritmo en pseudocódigo: Proceso paginas Partiendo del URL inicial del sitio Web a analizar = URL Paginas muertas = 0 {i 1..Número total de páginas del sitio Web analizado Páginas muertas (URL): Paginas muertas Mientras (URL) <> Número_total_de_paginas hacer Bandera = 0 Para cada enlace(URL[i]) de la pagina con URL: URL Si URL[i] es interno entonces Bandera = 1 Fin Si Fin para Si Bandera == 0 entonces Paginas muertas = Paginas muertas + 1 Fin si Fin mientras Retorna: Paginas muertas Fin
Interpretación del valor medido	NPMu > 0, cuan más cercano a cero mejor
Unidad de medida	Cantidad (de páginas muertas)
Tipo de escala	Absoluta (contexto)
Entradas al Proceso	Sitio Web
Tipo de recolección de datos y cálculo	Automatizable
Herramienta de medición	eWebsite MA, otras
Nivel de independencia del dominio de la aplicación	TI (Totalmente independiente)
Potenciales Procesos de Uso	Desarrollo, Testing, Integración, Mantenimiento
Beneficiario de la métrica	Desarrollador, Tester, Mantenimiento
Observaciones	

Elementary evaluation phase

Data collection

Elementary quality preference



X_i ? R_i ? R , i = 1, ...,n ? F_i : R_i ? I tal que

? $IE_i = F_i (X_i)$, $X_{i\,min}$ <= X_i <= $X_{i\,max}$ IEi ? I , I = [0, 1]

IEi= 1 (o 100%) si ti <= ti min

IEi= 0 (o 0%) si ti >= ti max

IEi = (ti max – ti) / (ti max – ti min) si ti min <= ti <= ti max

Global evaluation phase

Modelos de Estructuración y Puntaje

- Agregation just additive model (Tom Gilb, y otros)

  Indicador Parcial/Global =? (Weight component x Elementary value)

  $IP/G = P_1 IE_1 + ...+ P_n IE_n$ para: $0 <= IE_i <= 1$; i = 1 ... N y, $P_1 + ...+ P_n = 1$

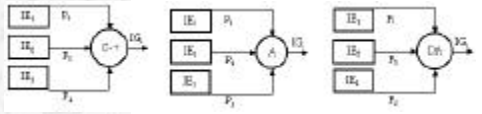
- Step-wise aggregation non-additive model or LSP (logic scoring of preference - Dujmovic)

  $IP/G(r) = (P1\ ie'_1 + P2\ ie'_2 + ... + pm\ ie'_m)\ 1/r$

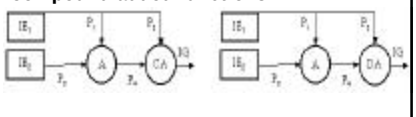
  Para: $0 <= iei <= 1$; I = 1 ... m $P1 + ...+ pn = 1$ y $p > 0$ -? <= r <= +? ;

  IP/G(- ?) = min (IE1 , IE2 , ... , iem); IP/G(+) = max (IE1 , IE2 , ... , iem);

Simple added functions



Compound added functions

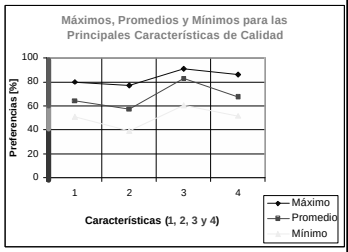


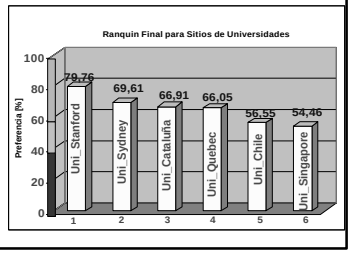
Julia González
juliagon@unex.es

Results, analysis and recommends

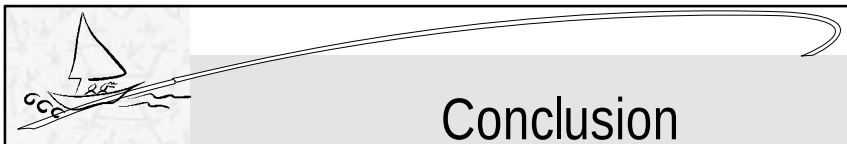
Partial and Global preference Case of study

Características y Subcaracterísticas	UPC	UChile	UTS	NUS	Stanford	UQAM
1. Usabilidad	**76.18**	**51.01**	**80.08**	**57.71**	**71.95**	**60.94**
1.1 Comprensibilidad Global del Sitio	77.74	48.5	98.17	75.70	83.17	40
1.1.1 Esquema de Organización Global	97.93	0	99.08	96.29	99.08	0
1.2 Mecanismos de Ayuda y Retroalimentación	70.12	44.6	72.3	75.8	69.4	68.79
1.3 Aspectos de Interfaces y Estéticos	80.6	69.72	83.11	54.9	75.71	64.10
1.4 Miscelánicas	73	45	33	55	33	88
2. Funcionalidad	**61.84**	**50.39**	**48.89**	**38.99**	**77.29**	**71.12**
2.1 Aspectos de Búsqueda y Recuperación	46.28	42.37	29.32	19.73	100	88.55
2.1.1 Mecanismo de Búsqueda en el Sitio	48	50.84	52.17	55.12	100	82.98
2.1.1.1 Búsqueda Restringida	24	40	44	40	100	68
2.1.2 Mecanismos de Recuperación	28	25	0	0	100	73
2.2 Aspectos de Navegación y Exploración	62.01	50.09	57.66	58.80	74.02	76.89
2.2.1 Navegabilidad	26.15	23.15	29.15	29.15	79	40.92
2.2.1.1 Orientación	15.93	15.93	15.93	15.93	70	21.75
2.2.2 Objetos de Control Navegacional	81.3	60.1	88.6	19.5	65.2	51.9
2.2.3 Frobación Navegacional	93	78	63	84	93	56
2.3 Aspectos del Dominio orientados al Estudiante	73.46	57.45	61.30	61.27	73.13	82.27
3. Confiabilidad	**60.40**	**87.62**	**90.86**	**88.70**	**83.05**	**63.61**
3.1 No Deficiencia	68.48	87.62	90.86	88.70	83.05	63.61
4. Eficiencia	**69.09**	**52.83**	**76.11**	**53.12**	**85.99**	**69.47**
4.1 Performancia	75.3	50.46	92	51.46	100	83.48
4.2 Accesibilidad	59.41	52.69	66.11	57.10	59.52	48.84
Preferencia de Calidad Global	**66.91**	**56.55**	**69.61**	**54.46**	**79.76**	**66.05**





juliagon@unex.es



Conclusion

- Web development more complex.
  - ad-hoc
  - Quality control have not been considered.
  - A new model to minimize crisis in Web development.
- Web-site QEM
  - Know the current state of working Website
  - Specify and control the quality requirements for new projects.
  - Improve Web product quality
  - Tools to achieve its goals: Website Ma and Website QEM Tool

Future work

- Automated data collection
  - Website MA
- Catalogue of metrics
- Quality in use
  - Model and framework to measure.
  - Metrics definition.
  - Developping new version of our tools.
  - Dinamic metrics.

Providing Automated Support to Web Metrics

, Julia González Rodríguez[1], Luis Olsina[2],Guillermo Lafuente[1], Oscar Pastor[3]

[1] Department of Computer Science, Polytechnic School at Extremadura University,
Avd. de la Universidad, s/n. 10.171 Cáceres. Spain
E-mail juliagon@unex.es

[2] GIDIS, Department of Informatics, Faculty of Engineering at
Universidad Nacional de La Pampa,
Calle 9 y 110, (6360) General Pico, La Pampa, Argentina
E-mail [olsinal, lafuente]@ing.unlpam.edu.ar
TE 54 2302 430497 Ext. 6501

[3] Department of Informatic Systems and Computing at Polytechnic University of Valencia,
Camino de la Vera s/n. Aptd 22.012. E-46071 Valencia. Spain
E-mail opastor@dsci.upv.es

Abstract

Automating support for many of the processes in an evaluation project allows minimizing efforts and costs as well as avoiding oversights and imprecisions introduced by the human intervention. Having as a part of the Web evaluation process a quantitative methodology that allows quantifying the quality of Web applications and a toolkit to support it are lines of research and development of our concern. In this article, we analyze about seventeen catalogued automated Web metrics, as well as the Website MA (Metric Automation) tool, which provides data collection and calculation to those metrics. In addition, we illustrate the level of support given by this tool to the defined evaluation process underlying the Web Quality Evaluation Methodology (WebQEM). Effective and accurate measurement and evaluation processes require not only methodological but also technological support.

Keywords: Product Quality, Automated metrics, Web metrics, Web Quality Evaluation Methodology, Supporting Tools.

1. Introduction

Due to the unceasing Web site and application (WebApps) growth, developers and evaluators have interesting challenges not only from the development but also from the quality assurance point of view. Particularly, regarding this last perspective, a clear definition of functional and non-functional requirements in order to measure, control, and potentially improve the produced quality is largely needed. As Web sites have grown both in interaction

and functionality, they have passed of being just static, document-oriented pages to dynamic application-oriented pages with at least the complexity of traditional software applications. One of the features of WebApps to highlight is the combination of different media and technologies to display its information and functionality that range from the simple text, voice and images, to components with complex interactions such as scripts, applets, ActiveX, among others. Consequently, an unsystematic use of this resources can cause problems to visitors who just want find information, browse, and access contents and specific functionality. For example, a low quality perception of a visited site could be due to a lack of a clear organization scheme of the main page (with missing attributes as a table of contents, a site map, or indexes, etc.), the absence of relevant contents, a low speed in loading pages, the existence of broken or invalid links, among others. All of these weaknesses can lead users giving the site up with a bad impression of it and potentially of the visited institution.

On the one hand, if there is a no defined scheme or quality model to allow evaluators orderly specifying non-functional requirements as quality, it is difficult to consider all of the optional, desirable and mandatory attributes and characteristics of quality for WebApps. WebQEM [14], from the methodological point of view, starts from a quality model consisting of a hierarchy of requirements. In fact, to follow a well-known standard, the same conceptual ISO-prescribed characteristics like usability, functionality, reliability, efficiency, portability, and maintainability are used. From these characteristics, subcharacteristics and attributes can be derived with minimal overlap.. Moreover, WebQEM was intended to perform an engineering contribution by proposing a systematic and flexible approach aimed to the evaluation, comparison and analysis of the quality of somewhat complex Webapps. By using the methodology for assessment purposes, evaluators can give recommendations by monitoring quality requirements in development projects and also by contrasting requirements in the operational phase. They can discover either absent attributes or subcharacteristics, and even unsatisfactory implemented requirements. For instance, design and implementation drawbacks related to the interface, navigation, accessibility, security, content, performance, and so forth. It is worthy of remark that the methodology has been, up to now, used in three case studies [12, 13, 15] in addition to some evaluation projects at industrial scale.

On the other hand, from the technological point of view, we have developed a first version of a toolkit in order to provide automated support to the WebQEM methodology. One of the core tools, so-called WebQEM_Tool [16], allows the administration of evaluation projects, which contains basic information of WebApps to be assessed. It also allows to edit and relate the non-functional requirements that consist hierarchically of a factor (as quality or cost), one o more characteristics, subcharacteristics, and attributes. The other tool, Website MA (Metric Automation [7]), automates the data collection and the calculation of about twenty direct and indirect Web metrics. It is important to notice that the used metrics for attributes specified in a quality model can be classified in direct or indirect metrics. The firsts, are get through a measure of a direct attribute (a non-compound attribute) of an entity; the seconds, are the result of applying a mathematical equation or model to an attribute association. Besides, data gathering can be made either manually, or semiautomatically or automatically. For automated metrics, both of direct and indirect attributes, in order to provide data collection and metric calculation an instrument can be used. For such an end the Website MA tool has been developed. Providing automated support has many benefits as elsewhere reported [2, 3, 8, 17].

Some of the automated metrics managed by Website MA, that will be analyzed later, are: *Broken Link Count*, *Link Count*, *Percentage of Broken Links*, *Number of Different Broken Links*, *Orphan Page Count*, *Number of Images of a Site with Alternative Text*, *Image Count*,

Maximum Quantity of Frames, among others. Website MA has been developed using Java2 and WebL [9] technologies. In addition to data collection and metrics calculation, Website MA yields tabular and graphical reports from the processed data. Its output serves as input to the WebQEM_Tool.

The rest of the paper proceeds as follows. In Section 2, we describe the main evaluation phases and processes of the WebQEM methodology, which gives a conceptual framework to the supporting tools. In Section 3, we analyze diverse attributes used in the case studies, focusing the attention on those automated. In addition, a cataloguing template is presented. Next, in Section 4, the main Website MA functional modules in addition to design and implementation issues are described. Finally, in the remaining section, concluding remarks and future tool enhancements are drawn.

2. The Evaluation Process in the WebQEM Methodology

We have been developing the WebQEM methodology since the late 90s, which covers the major process steps, models and procedures in the evaluation and ranking of WebApps. These process steps are grouped in the following four major technical phases:

1. *Quality Requirements Definition and Specification;*
2. *Elementary Evaluation (regarding Design and Implementation stages);*
3. *Global Evaluation (regarding Design and Implementation stages);*
4. *Conclusion of the Evaluation (with Recommendations).*

Figure 1, shows the evaluation process underlying the methodology, which includes the phases, stages, main steps, inputs and outputs. This evaluation process is inspired in the process model and enhanced process for evaluators of the ISO standards [5,6]. Next, we give an overview of the major technical phases:

2.1 Phases of WebQEM

2.1.1 The Quality Requirements Definition Phase. In this phase, the evaluators should clarify the evaluation goals (with requesters), analyze the Web product descriptions for a given domain, and precise the user view. They should select a quality model, for instance, the ISO-prescribed characteristics [5,6] as well as subcharacteristics and attributes customized to the Web domain [4,15]. The relative importance of these components should be identified considering the audience and the extent of the coverage required.

Regarding the user profile, three abstract evaluation views of quality may be defined, i.e., visitors, developers and managers views. Thus, taking into accounts the domain and product descriptions, the agreed goals, and the selected user view, i.e., the explicit and implicit user needs, characteristics and attributes should be specified in a quality requirement tree. In the end of this phase, a requirement specification document of quality is produced.

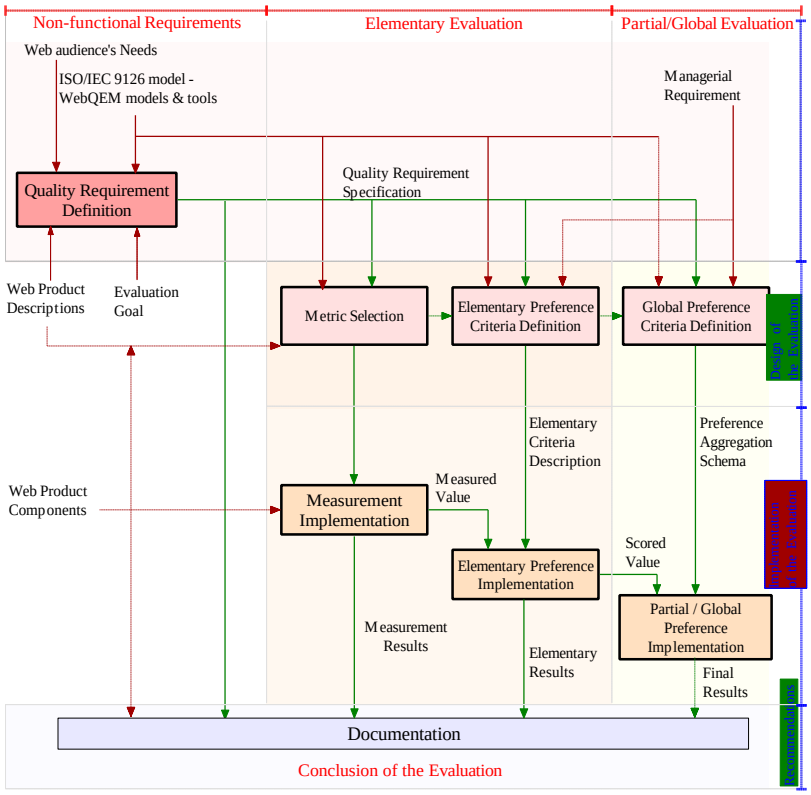


Figure 1. *The evaluation processes underlying in the WebQEM methodology. The phases, stages, main processes, inputs and outputs are shown.*

2.1.2. The Elementary Evaluation Phase. In this phase, two major stages are defined as depicted in Fig. 1: The design and the implementation of the elementary evaluation. On the one hand, regarding the quality factor, "*the quality of software products can be described in terms of quality characteristics as defined in the ISO/IEC 9126. However, the state of the art in software measurement is such that, in general, the direct measurement of these characteristics is not practical. What is possible is to assess these characteristics based on the measurement of lower abstraction attributes of the product*" [5]. For each quantifiable attribute A_i from the requirement tree, we can associate a variable X_i, which will take a numerical value from a direct or indirect metric. However, the value of this metric will not represent the level of satisfaction of this elemental requirement. For that reason, it is necessary to define an elementary criterion function that will result afterwards in an elementary quality indicator or preference value.

For instance, let us consider the *Orphan Page Count* attribute, which measure (count) pages having no return link to the site where are included in. A possible indirect metric is: X = *OrphanPageCount* / *PageCount*. Now, how do we interpret the measured value?; what are the best, worst and intermediate preferred values? The next formula represents a possible criterion function to determine the elementary quality preference EP:

EP = 1 (or 100%) if X = 0; EP = 0 (or 0%) if $X >= X_{max}$;

otherwise $EP = (X_{max} - X) / X_{max}$ if $0 < X < X_{max}$

where X_{max} is some agreed upper threshold such as 0.02

The elementary quality preference EP_i is frequently interpreted as the percentage of satisfied requirement for a given attribute, and it is defined in the range between 0, and 100%

(so the scale type and the unit of metrics become normalized).

In the implementation stage the selected metrics are applied to Web artifacts as shown in Fig. 1. Some of them can be measured just observationally, while others can be automated both the data collection and calculation. For the automated metrics, we used the Website MA tool. Once the metric value is known, the elementary quality preference is determined according to an agreed criterion as shown above.

2.1.3 The Global Evaluation Phase. Again, two major stages are defined: The design and the implementation of the partial/global quality evaluation. In the design stage, aggregation criteria and a scoring model should be selected. The goal of quantitative aggregation and scoring models is to make the evaluation process well structured, comprehensible, and accurate in order to yield partial/global indicators of quality (or cost). These can be used to help and justify optimum evaluation decisions.

There are at least two categories of these models (both supported by WebQEM_Tool). First, the evaluation procedures are based on linear additive scoring models (see [3], among others). Second, the evaluation procedures are based on nonlinear multi-criteria scoring model [1] where different attributes and characteristics relationships can be designed. In both cases, the relative importance of indicators are considered by means of weights. Anyway, once selected a scoring model, the aggregation process follows the hierarchical structure as defined in the requirement tree, from bottom to top. Applying a step-wise aggregation mechanism, a global schema can be obtained in the end. This enactable model allows us computing partial and global indicators in the implementation stage. The global quality preference represents the global degree of satisfaction in meeting the stated requirements.

2.1.4. The Conclusion of the Evaluation. In this phase, the documentation of Web product components, the specification of quality requirements, metrics, criteria, elementary and final results are recorded. Thus, the strengths and weaknesses of the assessed product with regard to established goals and user viewpoint could be analyzed and understood by requesters and evaluators. Recommendations can be suggested and justified.

2.2. Automating the Process using WebQEM_Tool.

The evaluation and comparison processes require both methodological and technological support. We have developed a Web-based tool [16] that supports the administration of evaluation projects. It allows editing and relating non-functional requirements. For instance, in our E-commerce case study, about a hundred and forty attributes and characteristics have intervened [15]. Then, by means of the automatic or manual edition of elementary indicators, it helps to aggregate the elements in order to yield a schema and calculate a global quality indicator for each site. This enables evaluators to assess and compare the quality of Web products. The results of an evaluation are shown through linked pages with textual, tabular and graphical information, which are dynamically generated from database tables. This tool was deeply discussed in [16].

In the next section, we focus mainly on some automated metrics used in the Elementary Evaluation Phase, in the case studies.

3. Automating Web Metrics.

As we underlined in the introduction, the attributes of an entity (product, product in use, process or resource), may be categorized as direct or indirect attributes. In addition, according to the type of data collection, attributes can be partially or totally automated. Precisely, the X_i value of an attribute could be determined by a manual (observational) or automated process, i.e., by means of a measurement instrument or tool.

3.1. Some Automated Web Metrics.

Although in the performed case studies many attributes values were gathered just observationally (because there was not another way to do it, e.g., for Table of Contents, Indexes, etc.), the automatic data collection and metric calculation were in many cases the only way to obtain reliable and efficient values. This was the case for attributes such as Broken Link Count, Quick Access Pages, Orphan Page Count, Average of Links per Page, among others. We are going to describe and discuss some automated metrics by tools (like Website MA).

3.1.1. Broken Link Count. This indirect attribute represents the total number of broken links both internal and external to the site, which do not include dynamically generated pages and links. Moreover, the metric of this attribute does not take into account the distinction among broken links to identical URLs, so all broken links are counted.

```
Preconditions
   Starting at the initial URL of the Website to analize URL= URL_i
   #Broken_links = 0
   #Internal_broken_links = 0
   #External_broken_links = 0
   j: 1..Total number of static pages.
Broken_links (URLj):#Broken_links
For each link (URL_ji)of page with URL= URL_j not previously analyzed
   If (URL_ji) is broken then
      If (URL_ji)is internal then
         #Internal_broken_links = #Internal_broken_links + 1
      Else
         #External_broken_links = #External_broken_links + 1
      End if
   Else
      If (URL_ji) is internal then
         #Broken_links = #Internal_broken_links + #External_broken_links
                              + Broken_links ()
      End if
   End if
End for each
    Return (#Broken_links)
End
```

Figure 2. *Specification of the algorithm that automates the* Broken Link Count *metric.*

Specified in pseudocode, we show the algorithm that automates the *Broken Link Count* metric in Figure 2. Website MA implements this algorithm and stores the destination URLs which would lead either the internal or external broken links, in addition to the current page URL where the anchor is. This allows ulterior analyses and corrections. According to the

returned HTTP state code, a broken link will be detected by this returned code. Likewise, depending on the returned state code, the amount of links that lead to non-accessible pages can be determined, among other link failures.

It is important to remark that this attribute influences the quality of WebApps from the visitor point of view, so the bigger the number of broken links is, the lesser the reliability of the site is. In addition, for maintainability issues, it is important to distinguish between external and internal links (broken or not), as we see later.

3.1.2 Link Count. The total (absolute) number of links of static pages of a site can be collected automatically. As shown in Fig. 2, by reusing this algorithm the total number of internal and external links can be calculated. Sometimes, when it is considered just internal links, the metric is called *Connectivity* in the literature ([13], p. 113).

3.1.3 Percentage of Broken Links. Using the above metrics (broken link count and link count), the percentage is calculated as follows:

$$PercentageBrokenLinks = 100 * \frac{\#InternalBrokenLinks + \#ExternalBrokenLinks}{LinkCount}$$

This attribute may be considered domain independent.

Besides, the metric of this attribute shows to some extent how reliable a site is (the reader can also consider the quotient between the number of external broken links and the total number of external links, likewise can be done for internal links). A more careful study on the quality impact of this metric would envisage the importance of broken links relating to their location in the more relevant or visited pages of a site.

3.1.4 Number of Different Broken Links. This metric is obtained analyzing the distinct URLs used in the *Broken Link Count* metric. Internal and external broken links to the same resource are just counted once. It can show useful information for the maintaining phase, and also help in the analysis of the impact on changes.

In this case, the procedure to data collection and computation as specified in Fig.2, has a slight change, i.e., just checking if the considered URL was visited before or not.

3.1.5 Percentage of Different Broken Links. Instead to the *Percentage of Broken Links* metric, the relation is established according to the non-repeated links. The percentage is calculated as follows:

$$PercentageDifferentBrokenLinks = 100 * \frac{\#DifferentBrokenLinks}{DifferentLinkCount}$$

As the reader can consider, combining distinct metrics useful information can be drawn. For instance, we can see what is the level of link redundancy regarding the quotient between the *Different Link Count* and the *Link Count* (either internal or external or both). The equation is similar as shown to the 3.1.10 attribute.

3.1.6 Number of Images with Alternative Text. Images give visual information that sometimes, for accessibility or performance reasons, should be deactivated by the user, for example, by disabling the browser's image feature. A Web site should provide suitable mechanisms to visualize images optionally, without loosing all the conveyed information.

The ALT property (in the HTML code), links an alternative text with an image (or other objects in Web pages such as applets, sounds, etc.). This contributes to the readability of the page (even the text could be read before the image is unloaded). However, the measure of this

attribute does not guarantee the quality of the alternative text. Some text can be generated automatically when editing with tools like FrontPage, among others. On the other hand, this attribute is included in the WAI guidelines of the W3C [19]. Website MA counts the number of images of the whole site that have this property.

3.1.7 Image Count. It helps to measure the amount of provided visual information (in the same way, we can alternatively measure the *Media Count* metric, where the number of media files is considered). The existence of images in a page is checked through the IMG property that is supported by the HTML code.

3.1.8 Percentage of Presence of Alternative Text. This indirect metric is calculated as follows:

$$PercentagePresenceAlternativeText = 100 * \frac{\#\ ImageswithAlternativeText}{ImageCount}$$

A more careful study on the impact of this metric it would be to observe the percentage of presence of alternative text in the images posted in the firsts levels of the tree (assuming a hierarchical structure), and/or the more visited pages of a site.

3.1.9 Different Image Count. This direct measure counts the non-repeated images in the site.

3.1.10 Percentage of Image Redundancy. The relation between the amount of different images and the image count in a site can be posed as shown in the next formula. An image repetition may be interpreted as the level of redundancy of visual information.

$$PercentageImageRedundancy = 100 * \left(1 - \frac{DifferentImageCount}{ImageCount}\right)$$

3.1.11 Page Count. It is yielded counting the total number of static pages of a site both HTML and SHTML. It shows the initial size of the site according to the number of documents or pages; however, more elaborated metrics to measure size and length can be provided (e.g., considering the page size attribute as well).

3.1.12 Average Links per Page. It is calculated as the quotient between the *Links Count* and *Page Count*. This metric gives information about the interconnection density. In other words, it indicates how an average page is interconnected toward destination nodes (see [14] for more details).

3.1.13 Page Size. The size of a (static) page is measured considering all its graphics, scanned images, sounds, videos and textual components. For each page, the size in bytes can be obtained. The size of pages is an important issue in order to appreciate the site efficiency as we posed in the following metric.

3.1.14 Quick Access Pages. The unloading time (T) is related with the size of a page p and the speed in the established connection line (c).

$$T_{Unloading} = f(p, c)$$

This time is proportional to the page size and inversely proportional to the speed of a given connection line. A function may be created in order to classify pages as quick or slow access

pages, according to a minimum threshold of time (e.g. 10 seconds) for a given speed of a connection line. (Readers can find information about recommended page sizes depending on the speed of communication lines for example in [11]).

$$g(x_{Unloading}) = \begin{cases} QuickAccess & x_{Unloading} \le x_{max} \\ SlowAccess & x_{Unloading} > x_{max} \end{cases}$$

This criteria is a simple way to measure the performance (or predict it at design time), however, it does not reflect the actual performance, as the reader might surmise.

3.1.15 Maintainability of Link Color. There are some desirable attributes as those which help users in maintaining the coherence and uniformity in the visualized information. The maintenance of link colors is one of them and it can be counted how many times link color changes, depending on if it was visited or not. It can be considered the design guidelines of blue tonalities for non-visited links and purple tonalities for visited ones. Hence, this metric counts the changes in both categories analyzing the color code for the LINK and VLINK properties (visited and non-visited, respectively).

3.1.16 Orphan Page Count. A dead-end or orphan page has no internal link to the site where is included in. When a visitor accesses an orphan page through an external URL, he/she is unable to navigate inside the site. This kind of page has no internal navigational functionality and its utility depends rather on its content exclusively.

3.1.17 Percentage of Orphan Pages. This indirect metric is calculated as shown in the following formula. As we discuss in subsection 2.2 an upper threshold such as 2% can be agreed, in addition to a criterion function to obtain the quality preference. This metric affects the reliability characteristic.

$$PercentageOrphanPages = 100 * \frac{OrphanPageCount}{PageCount}$$

3.2. Template for Cataloguing Web Metrics.

Currently, we are cataloguing Web metrics mainly those where data gathering can be automated. Providing evaluators and other stakeholders a catalog of metrics potentially allows to have a framework for reuse and consultation starting from a clear specification of the metric entity, objective, criterion, procedures and also the automated query mechanism. In this work in-progress, we have catalogued up to ninety direct and indirect automated Web metrics. Table 1 shows a template for cataloguing Web metrics.

Table 1. *Template for cataloguing product metrics exemplified with the Orphan Page Count metric.*

Code	Some code related to the non-functional requirement tree
Name	**Orphan Page Count**
Objetive / Motivation	It counts the number of pages that have no internal links to the Web site where they are included in. This metric (in conjunction with others) can give stakeholders useful information both in the development and maintenance phases indicating the absence of links that

	allow a smooth site navigation.
Description	This attribute measures the total number of static pages which can not access other pages in the site. They do not contain internal links, so in the time of counting, external links to the site will not be considered.
Entity type	Product-> WebApp
Characteristic	Reliability (regarding the ISO-9126 software quality model)
Formula	#OP (Number of Orphan Pages)
Attribute type	Direct and External attribute

Application procedure	From a starting URL (of a given site), recursively analyze all the pages, considering exclusively those that have at least an internal and not broken link, following this generic algorithm:
	Preconditions
	Starting from the initial URL of the Web site to analyze = URL_j
	Orphan_page = 0; j: 1.. Page Count
	
	Orphan_pages (URLj): #Orphan_pages
	For each page (URL_j) not previously analyzed
	If ?? an internal(URL_{ji}) not broken ***then***
	#Orphan_pages = 1 + Orphan_pages (URL_{j+1})
	else
	#Orphan_pages = Orphan_pages (URL_{j+1})
	end if
	end for each
	end

Interpretation of the measured value	#OP > = 0, the closer to zero the better
Unit of Measure	Number of orphan pages
Scale type	Absolute (it is a counting)
Input to the Process	A WebApp
Type of Data Gathering and Calculation	Automated
Measurement Tool	Website MA, among others
Level of Independence of the application domain	TI (Totally Independent) Note: For other attributes, it can also be PD = Partially Dependent or TD= Totally Dependent
Potential Processes of Use	Development, Testing, Integration, Maintenance
Potential Beneficiaries	Developer, Tester, Mantainer

4. Website MA: Design and Implementation Details.

As previously said, the Website MA tool gives support to the evaluation process for Web metrics. It allows the automation of data collection and calculation of metrics (as those described in section 3.1), in addition to the generation of graphical and tabular reports. Website MA is able to manage different evaluation projects each one represented by its URL, the selected metrics and the date of assessment, etc., so all this information is stored accordingly. For example, it is possible to evaluate a site at different times, where the evaluation strategy can be a snapshot or longitudinal analysis, or it is possible to evaluate multiple sites in order to perform cross-sectional analysis. On the other hand, the evaluation of a site can be done on-line but is also possible to do it with a mirrored copy as well.

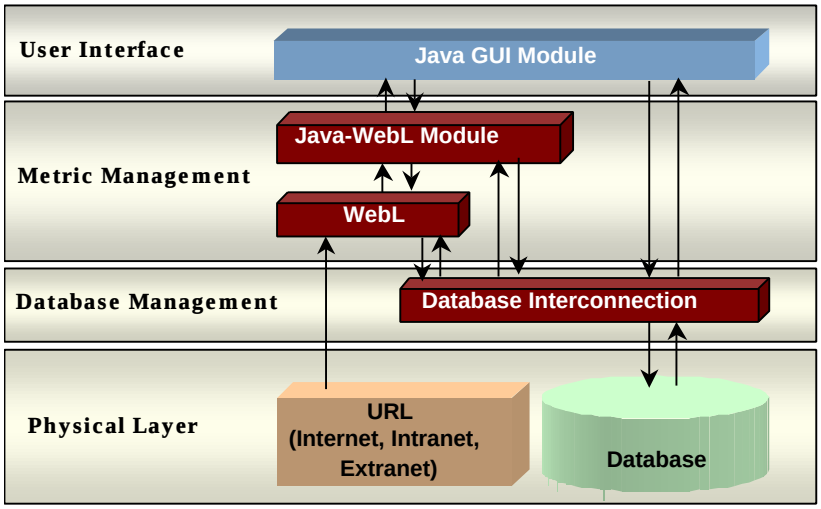


Figure 3. *Main layers and modules of Website MA.*

The tool has been designed in three main layers (figure 3), which are briefly described:

- *User Interface:* This layer allows the interaction with the application by a graphical user interface (GUI). For design purposes, we used the *Swing* package of Java.
- *Metric Management:* this is the core layer of Website MA. It manages all the configurable metrics from the GUI, in addition to performing data collection and metric calculation from the specified URL. Besides it works with the database management layer.
- *Database Management:* This layer manages all the information that is supplied by user through the *User Interface* layer. Some of the data given by the evaluator are: the project name, the Website URL, the evaluation starting and ending date and time, the customized configuration of metrics for each project, among other information. This layer is responsible to manage the diverse queries and report generation.

For the layers showed in the figure, the modules of the application have been accordingly developed. In order to analyze HTML documents and perform the basic functionality, the Java 2 platform and the WebL language were used – this language has been developed specifically to manage HTML and XML objects and attributes. The system compises the following modules: The *Java_GUI* module, the *Java-WebL* module, the *WebL* and the *Database Interconnection* module (for more implementation details see [7]).

5. Discusion and Future Work.

As Pfleeger said, "*metrics are welcome when they are clearly needed and easy to collect and understand*" [17]. Moreover, automating support for many of the processes in an evaluation project allows minimizing efforts and costs as well as avoiding oversights and imprecisions introduced by the human intervention. Having as a part of the Web evaluation process a quantitative methodology that allows quantifying the quality of Web applications and a toolkit to support it are lines of research and development of our current concern. As aforementioned, effective and accurate measurement and evaluation processes require not only methodological but also technological support.

In this article we have analyzed about seventeen automated Web metrics, as well as the Website MA tool, which provides data collection and calculation to the discussed metrics. On the other side, we have envisaged the level of support given by this tool to the defined evaluation process underlying the Web Quality Evaluation Methodology and its supporting tool (WebQEM_Tool).

Besides, we are currently cataloguing up to ninety direct and indirect automated Web metrics. Providing evaluators and other stakeholders a catalog of metrics potentially allows to have a framework for reuse and consultation starting from a clear specification of the metric entity, objective, criteria, procedures and an automated query mechanism as well.

So far, there are some documented works related to Web metrics in Mendez et al. and Scharl [10, 18]. Metrics in [10] are mainly used as input to a predictive model for the project effort estimation, meanwhile in [18] they are mainly used for longitudinal analysis. Both sets are not complete as we have observed in our current research. On the other hand, there are different tools (as LinkScan? , WebAnalyzer? , Bobby? , WebKing? , to quote just a few), which implement some of the above described metrics but none of them work with all the defined and catalogued metrics as we are doing.

One of the main purpose of Website MA is to give a more complete set of automated metrics in its follow-up version, in addition to be fully integrated to the WebQEM_Tool. Ultimately, Website MA will dynamicly provide services based on the catalogue of metrics. This will be discussed in future works.

Acknowledgments

This research is supported by the UNLPam-09/F013 research project and its extension. Also by the CYTED Program, in the VII.18. WEST (Web-based Software Technology) iberoamerican project .

References

1. Dujmovic, J.J., 1996, *"A Method for Evaluation and Selection of Complex Hardware and Software Systems"*, The 22nd Int'l Conference for the Resource Management and Performance Evaluation of Enterprise CS. CMG 96 Proceedings, Vol. 1, pp. 368-378
2. Fenton, N.E.; Pfleeger, S.L., 1997, *"Software Metrics: a Rigorous and Practical Approach"*, 2nd Ed., PWS Publishing Company.
3. Gilb, T., 1976, Software Metrics, Chartwell-Bratt, Cambridge, MA
4. IEEE Networking the World, 2000, "*IEEE Web Publishing Guide*", http://www.ieee.org/web/developers/style/
5. ISO/IEC 9126, 1991, International Standard "Information technology – Software product evaluation – Quality characteristics and guidelines for their use".
6. ISO/IEC 14598-5:1998 International Standard, "*Information technology -- Software product evaluation*" -- Part 5: Process for evaluators
7. Lafuente, G; Gonzalez, J., Olsina, L; 2001, "*Automatizando Métricas Web*", 4º Encontro para a Qualidade nas Tecnologias de Informação e Comunicações, QuaTIC'2001, Lisboa, Portugal, pp.17-24.
8. Lavazza, L., 2000, *"Providing Automated Support for the GQM Measurement Process"*, IEEE Software, Vol. 17, No. 3, pp. 56-62.

9. Marais H., 1999, "*WebL —A Programming Language for the Web*", Compaq Systems Research Center (SRC)
10. Mendez. E.; Mosley, N.; Counsell, S.; 2001, "*Web Metric –Estimating Desing and Authoring Effort*", IEEE Multimedia, V. 8, Nº 1, pp. 50-57.
11. Nielsen, J.; 2000, Designing Web Usability: The Practice of Simplicity, New Riders Publishing
12. Olsina, L., 1999, "*Web-site Quantitative Evaluation and Comparison: a Case Study on Museums*", Workshop on Software Engineering over the Internet, at Int'l Conference on Software Engineering, http://sern.cpsc.ucalgary.ca/~maurer/ICSE99WS/ICSE99WS.html, Los Angeles, US.
13. Olsina, L.; Lafuente, G.J.; Godoy, D; Rossi, G.; 1999, "*Assessing the Quality of Academic Websites: a Case Study*", In: New Review of Hypermedia and Multimedia (NRHM) Journal, Taylor Graham Publishers, UK, Vol. 5, pp. 81-103.
14. Olsina L., 2000, April, "*Web Engineering: A Quantitative Methodology for Quality Evaluation and Comparison of Web Applications*", Doctoral Thesis (in Spanish), Ciencias Exactas School, UNLP, La Plata, Argentina
15. Olsina, L.; Lafuente, G.J.; Rossi, G., 2000, "*E-commerce Site Evaluation: a Case Study*", In LNCS 1875 of Springer, 1st International Conference on Electronic Commerce and Web Technologies, EC-Web 2000, London, UK, pp. 239-252.
16. Olsina L., Papa, M.F., Souto, M.E., Rossi, G.; 2001, "*Providing Automated Support for the Web Quality Evaluation Methodology*", Proceed. of the Fourth Workshop on Web Engineering, at the 10th International WWW Conference, Hong Kong, pp 1-11.
17. Pfleeger, S. L., 1993, "*Lessons Learned in Buliding a Corporate Metric Program*", IEEE Software, Vol. 10, No. 3, pp. 67-74.
18. Scharl A., 2000, *Evolutionary Web Development*, (Applied Computing) Springer.
19. WWW Consortium, 1999, "*WAI Accessibility Guidelines: Page Authoring*", W3C Working Draft, "WAI Accessibility Guidelines: Page Authoring", http://www.w3c.org/TR/WD-WAI-PAGEAUTH/

QWE2002 Paper 2I

Mr. Kim Davis & Mr. Robert Sabourin
(My Virtual Model Inc.)

Practical Experiences in Bug Cluster Management

Key Points

- Areas of uncertainties define bugs clusters having natural affinities of causes
- Bug Clusters are easier to locate and manage than individual bugs
- Root-cause analysis & risk-based bug clusters fixes help to converge faster on e-com sites

Presentation Abstract

E-commerce development demands fast, incrementally useful, results. We propose to simplify the bug flow process by going after bug clusters, instead of individual bugs which are usually hiding one another. These clusters come from areas of uncertainties associated with missing, incomplete or rapidly changing requirements or specifications. The clusters are composed of the bugs that have natural affinities of causes.

We first identify these clusters, and track them. We help perform fast probabilistic root-cause analysis on clusters to assist in determining probable causes. We finally prioritize the clusters extermination and iterate throughout the project.

This method aims to achieve faster convergence in short timeframes. Real examples applying to e-commerce applications will be used.

About the Author

Robert Sabourin (rsabourin@amibug.com) is an author, lecturer, and president of AmiBug.Com, a firm specializing in software management consulting, teaching, and professional development. Robert is the author of the popular children's book I am a Bug (ISBN 0-9685774-0-7) which explains what SQA folks really do at work.

Kim Davis graduated in Physics in 1990, and in Computer Science & Psychology in 1994. As a Research Agent he investigated ways of speeding up the Internet by simulating network-optimisation learning algorithms. In 1997 Kim co-founded a web site design company called Inter@ction Technologies to help companies develop database-driven web sites. As Technical Lead, Director of Integration and Director of Operations of the My Virtual Model Team, Kim has helped develop the technology behind the e-commerce web sites of My Virtual Model inc. The Service he manages, called SET (Software Engineering Team), offers ways to the whole company to improve people / projects / product / processes effectiveness in terms and productivity and quality.

Practical Experiences In Bug Cluster Management

Kim Davis, MyVirtualModel, Inc.
kim@myvirtualmodel.com
www.myvirtualmodel.com

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

My Virtual Model

AmiBug.Com, Inc.

 © Kim Davis, Robert Sabourin, 2001

Overview

- Method Motivation & Project Presentation
- Risks: Business, Technical
- Potential Areas of Instability & Test objectives
- Exploratory Testing & Mapping
- From Mapped Areas of Instability to Bug Clusters
- Finding, Prioritizing and Exterminating Clusters
- Conclusion
- Future work
- Web References

My Virtual Model

 © Kim Davis, Robert Sabourin, 2001

Motivation

- 80/20 Rule: Can we save lot's of time and effort by focusing on larger structures of bugs, i.e. *Bug Clusters*?
- Apply successive approximations
- Identify the cluster – without finding all of its individual bugs!
- Identify, Prioritize and Exterminate at the Bug Clusters Level
- Fix bugs that you did not find!

My Virtual Model

My Virtual Model Community

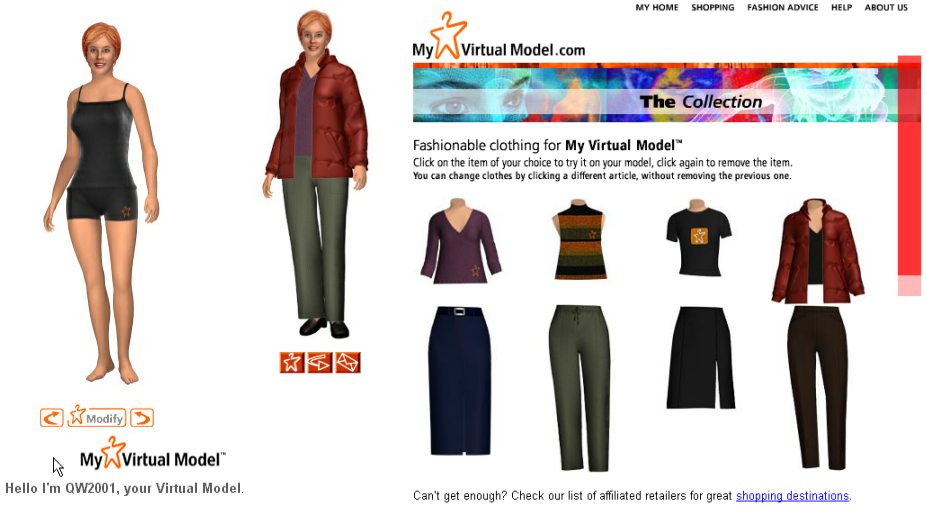
- **My Virtual Model Concept & Community**
- **3D, Web and Fashion Industries Collide**
- **Visualization**
- **Personalization – Virtual Identity**
- **Network of Interoperating Web Sites – Mobility**
- **Affiliate Services & E-Commerce**

My Virtual Model

LIMITED TOO

Sympatico LYCOS

My Virtual Model Community



Business Risks

- Fixed timeframe projects tied to industry dates for seasonal collections of garments
- Community is large and demanding in terms of consistency, performance, reliability
- It has to work and it has to be on time
- Functionality always tied to market "$$" related revenues e.g. a new type of model can be introduced to target a new market (for example Tall Men)
- New business model is evolving as project continues!
- Organizational change
- Requirement turbulence

My Virtual Model™

Example Project

- Constraint from moment clothing in-hand to release - max 8-12 weeks
- Community per client includes 100s of thousands of users - target large retail clients - we will get many many hits fast
- No forgiveness in retail fashion business - (nor Web business)!
- Trading down functionality to hit a release target is problematic
- Business requirement - release date moved up!

My Virtual Model

Technical Risks

- Technology change Management
  - Fast technology churn
  - Use technologies in production environments with limited knowledge of baseline
  - Limited time for evaluation, training
- Staff change
  - Programmer not familiar with entire code base - could miss something
  - Programmers not familiar with “live” production parameters - even if the developer knows how to make it work on his own machine (NT vs UNIX)
- Requirement turbulence

My Virtual Model

Example Technical Risks

- Many parallel activities not easy to synchronize
- New application server framework, business logic layer
  - From JServ to JRUN
- New presentation layer
  - to JSP model
- Some reused code at risk because running in a new context
- Changed requirements on technologies used

My Virtual Model

Testing Objectives & Risk

Mon Mannequin Virtuel

Project Name:	
Author:	KD
Revision:	3/30/01

ID	Summary	Note	Proposed Owner	Relative Importance Suggested % of Total Testing Effort				Consequences of Failure (Point Form List)	
				Develop-ment Lead	Product Manage-ment	Testing Lead	Average	Business	Technical
Proj-HLTO-01	Functionality	Suitability, Accuracy, Interoperability, Compliance, Security	Product Management	30	20	25	**24**	Loss of confidence - High Maintenance & patches	High Maintenance
Proj-HLTO-02	Reliability	Maturity, Fault Tolerance, Recoverability	Product Management	20	30	30	**27**	Loss of confidence - High Maintenance	High Maintenance
Proj-HLTO-03	Efficiency	System behaviour over time, Resources Usage	Development Management	20	25	15	**22**	Loss of confidence - High Maintenance	High Maintenance
Proj-HLTO-04	Usability	Understandability, Learnability, Operability	Product Management	10	15	20	**14**	Loss of user interest - Loss of customers	
Proj-HLTO-05	Maintainability	Analysability, Changeability, Stability, Testability	Development Management	10	5	5	**7**	Maintenance Difficulty	
Proj-HLTO-06	Portability	Adaptability, Installability, Conformance, Replaceability	Development Management	10	5	5	**7**	Difficulty to get clients on different platform	Lack of interoperability
			Total	**100**	**100**	**100**			

My Virtual Model

Potential Areas of Instability

- Functions particularly related to new technologies
- New stuff or changed stuff (Turbulence Index)
- Functionalities added after design due to business change

My Virtual Model

Exploratory Testing Approach

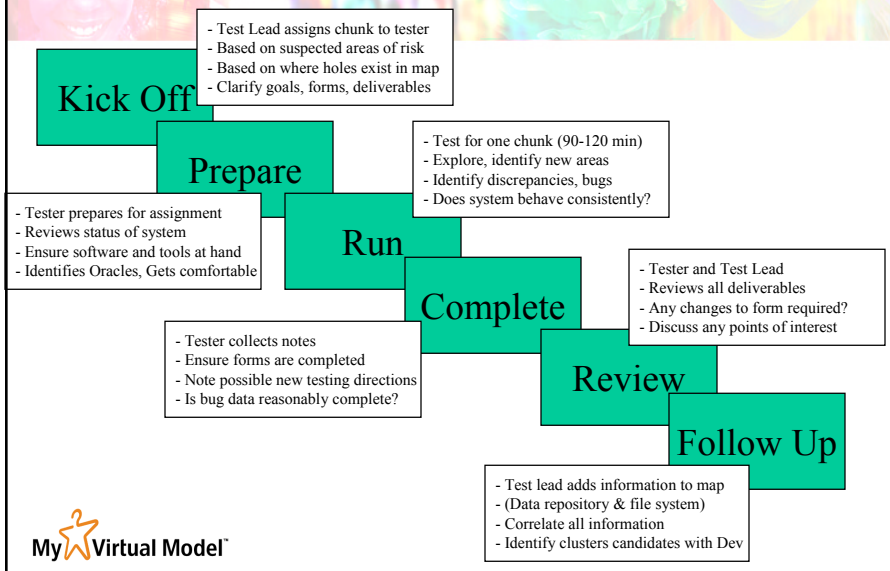
- *"In operational terms, exploratory testing is an interactive process of concurrent product exploration, test design and test execution."*

  - James Bach

- Test lead triage testing in chunks building up a "map"
- Assignment based on looking for areas of potential instability
  - Looking for clusters - observe aggregate results from all testers

My Virtual Model

Exploratory Testing



My Virtual Model

Finding a Cluster from Bugs Identified

- How should you/could you or do you identify a cluster?
- Correlation between bugs identified to get an understanding of whether they can be grouped
- Mapped Areas of Instability define clusters composed of the bugs that have natural affinities of causes
  - Requirement or Technology Turbulence
  - Copy-paste and reuse of code without checking

My Virtual Model

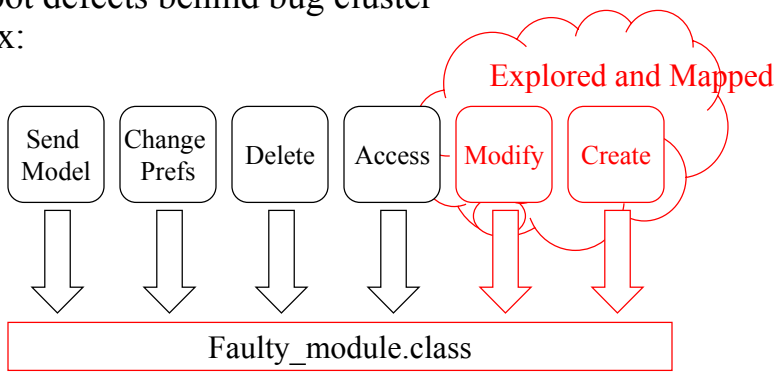
Clusters of Bugs

- What makes a Bug Cluster?
- Example Metrics Definitions
  - Functionality - several bugs related to same functionality are discovered
  - Reliability - different functions fail in similar way
  - Efficiency - several operations are similarly using resources inefficiently
  - Time - several content sources are out of sync
- Could indicate a process-related problem

My Virtual Model

Cause of Clusters

- Test team works with developers to find common root defects behind bug cluster
  Ex:



My Virtual Model

Prioritize Based on Business Impact

- Instead of prioritizing on a bug by bug basis, we work on a cluster (on a higher level)
- Less red tape
- Nice for speed!
- Can
  - Fix cluster
  - Do not fix cluster
  - Treat bugs individually
  - Detailed analytic testing

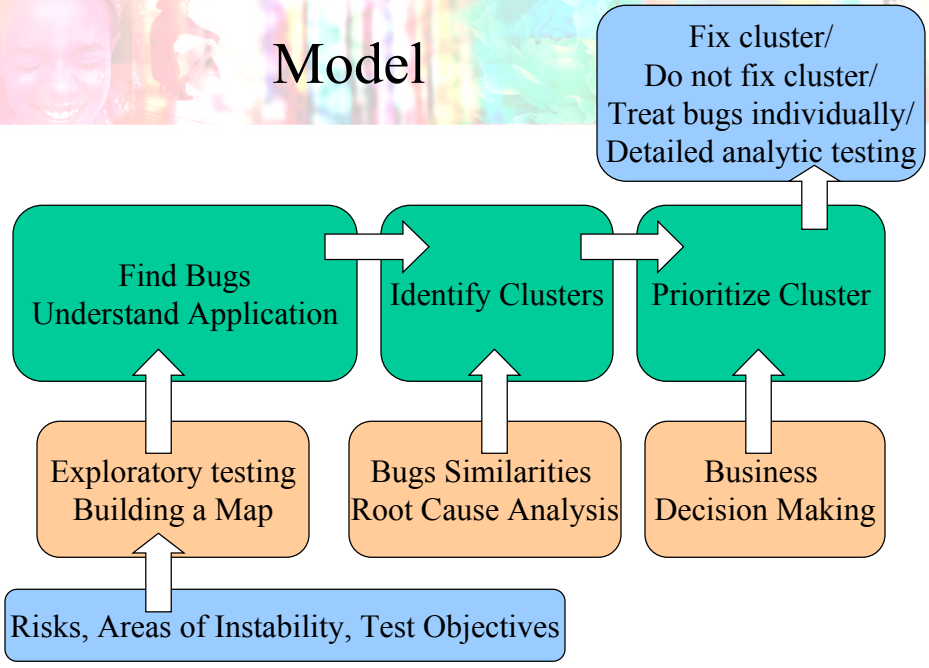
My Virtual Model

Practical Results

- Extermination techniques, developers and testers working together (fast)
- Try to identify probable root causes of sets of seemingly related bugs – Use empirical knowledge
  - Testers and developers worked well together
  - Improved developer awareness of tester role
  - Buy-in to approach
- Analysis based on root cause identified an additional 30%, so for every 10 bugs identified 13 were fixed
- **Continue using technique!**

My Virtual Model

Model



Conclusion

- Relevant in our context
- Shortening bug management overhead
- In less time we were able to deal with more bugs
- Flexible Method – Hybrid
- Buy in at all levels!
- In a turbulent environment, this will likely be a good approach on future projects

My Virtual Model

Future Work ...

- As method is applied to more projects we will better document and generalize it
- Exploratory Testing can be replaced by more analytical approaches as feeding method
- So far - developers, testers and management all like method - and the results
- Cluster Testing Method's Effectiveness & Efficiency – Using it to its fullest potential – Limits of applicability
- Will evolve based on REAL WORLD needs

My Virtual Model

© Kim Davis, Robert Sabourin, 2001

Web Reference Slide

- **www.satisfice.com**
  - **James Bach web site, exploratory and risk based testing**
- **www.testing.com**
  - **Brian Marick web site, articles about exploratory testing**
- **www.amibug.com**
  - **Robert Sabourin Web site, various presentations**
- **www.mvm.com**
  - **My Virtual Model Web site**
- **www.stickyminds.com**
  - **Much relevant content**

My Virtual Model

© Kim Davis, Robert Sabourin, 2001

Practical Experiences in Bug Cluster Management

Kim Davis, SET SQA Director, My Virtual Model inc.
Kim@myvirtualmodel.com
www.myvirtualmodel.com

Robert Sabourin, President, AmiBug.com
Rsabourin@amibug.com
www.amibug.com

Abstract

In e-Commerce applications, time attrition, fleeting turbulent requirements and lack of documentation make testing an exercise in managing people who are entangled in constraints. Our experience shows that a good risk-based bug flow process is essential. However, tracking myriads of little web site bugs in a fast-paced environment is difficult in practice. The good intentions of web integrators, scripters and web managers often disappear, in the name of competition, in front of customers demands of an ever-faster site development.

Fortunately there may be a way of dealing with this difficulty without sacrificing features or quality in such an environment: *why not stop tracking the myriad little web sites bugs* – and *start going after the bug clusters!*

Bugs are social animals, they tend to associate by affinity of causes. They may come from a variety of causes. Major causes are usually related to requirements, be them incomplete, misunderstood, unclear etc., and turbulence, i.e. the effect of change of a project variable in a short timeframe.

This results in the e-Commerce application having areas of uncertainties, which may require large amounts of rework. These areas of uncertainties define clusters composed of the bugs that have natural affinities of causes.

So instead of tracking hundreds on individual bugs in an extremely short timeframe, which is not efficient, we instead focus on finding the important areas of uncertainties, which are easier to locate and track than individual bugs. We then perform fast probabilistic root-cause analysis on them to determine probable causes pairing developers and testers to identify and correct the defects causing them.

© Kim Davis, Robert Sabourin, 2001

Motivation

80/20 Rule

We assume that generally the 80/20 rule can be applied to our testing activities. It is generally assumed that 80% of the important bugs are identified with 20% of the testing effort. It is unfortunate that we can not figure out in advance which 20% of the testing effort will find these important bugs. Can we save time and effort by focusing on larger structures of bugs, i.e. *Bug Clusters*?

One thing we know for sure is that we are severely restricted in the amount of time available for testing. So we want to focus our testing efforts on finding as many important bugs as possible in a short time, and present to management the state of the system under test so that informed business decisions can be made.

Apply successive approximations

Optimization techniques involve finding the best solution, or fit, to a problem by running a series of experiments, varying certain parameters each trial and then judging whether the results are optimal, or closer to optimal, based on our previous results and experiences. Purely statistical models are sometimes used in software testing (such as clean room based testing) to confirm that sufficient test cases have been run to prove or disprove a reliability hypothesis.

Bugs are caused by defects. Defects are introduced into a system by people using processes or tools. Defects may be introduced at any time of the development process, requirement, analysis, design, development, integration, deployment etc. The bugs found during testing are a very important input to the detection of defects in the system under test.

In the context of projects at MyVirtualModel, Inc. we know that we will fix the bugs deemed important from a business and technical point of view. Indeed the priority of bugs, indicating if and when it must be fixed, is a *business decision*.

If we find some bugs we may suspect that there is some a relationship between them. This can be considered a weak relationship which, as a result of more testing, and further bug identification, can be made successively stronger.

We use an iterative approach when testing the system. Each iteration is based on exploring the application to further enhance our knowledge of the software and validate any hypothesis about functional areas of the application exhibiting instability.

© Kim Davis, Robert Sabourin, 2001

Identify the cluster

Ideally we would like to confidently identify a bug cluster without having to finding all of its individual constituent bugs! This would save us a lot of time and effort. But how can bug clusters be identified?

Bugs are social animals, they tend to associate by affinity of causes. Our practical experience tells us that after a project, when reviewing data about bugs identified and associated defects corrected, there is generally a series of bugs related to the same root cause. If we had only found one bug of a certain "family" or cluster, then we could have fixed the lot of them without even being aware of their existence! As Brian Marick defines in "Evaluation Test Suites", Conference Proceedings, Quality Week 2000, (with James Bach and Cem Kaner) a perfectly effective test suite is one that reveals at least one failure for every fault in the program.

Prioritize and Exterminate Bug Clusters

Once suspected bug clusters are identified we can prioritize them relative to other bug clusters in the system. We can make a cluster based decision which will apply to bugs which have not yet been identified.

Fix bugs that you did not find!

By fixing the fault in the cluster of bugs we correct failures which may have existed but were not identified and recorded.

My Virtual Model Community

My Virtual Model concept and community

My Virtual Model has created a network hosting a virtual community. Members of the community create electronic replicas of themselves, called Virtual Models. Shoppers can experiment with fashion, dressing their Virtual Models with virtual garments without the intimidation of store dressing rooms.

Users log on to the site to begin building their model. By answering specific questions, users can create a virtual replica of themselves with exact measurements, skin and hair color. Users can then try on various looks using their model in a virtual dressing room. Shoppers can choose to accept advice from a "Fashion Advisor" on the basis of their body type, coloring, and fashion goals.

© Kim Davis, Robert Sabourin, 2001

Over 1.5 million models have already been created at MyVirtualModel.com affiliated sites.

Forrester Research described My Virtual Model™ as "the most viable alternative" among tools which support online apparel sales.

3D, Web and Fashion industries collide

Requirements for all My Virtual Model development efforts include 3D animation, the latest is Web GUI and interactive technologies. My Virtual Model clientele are members of the Fashion industry which operates under very tight seasonal delivery pressures. You cannot delay the spring fashion season due to a software bug! All projects have fixed timeframes.

Visualization

My Virtual Model provides the ability to mix and match garments and colors that are not only photo-realistic but are also right-sized to eliminate concerns of true fit or appropriate sizing. It includes :

- Photo realistic images of garments
- Close-up views of garments (Zoom in)
- Multiple views of garments (360 degrees rotation)
- Try-on of the garment on body with real measurements

Personalization – Virtual Identity

Virtual Models are personalized for each member of the community. All characteristics of a model can be adjusted as the end-user sees fit. If a new characteristic is modeled, then special consideration must be made for previously existing models to operate with the new characteristics. Backward compatibility is critical.

Users build what is called a Virtual Identity. The first layer is a physical representation in the form of the 3D Virtual Model, which is tied to the garment try-on service and the e-apparel market. Other layers tied to other services, for example financial, may be added later so that the model will resemble more closely the user in order to better serve it.

E-Commerce

My Virtual Model technologies are an integral part of the e-Commerce offering from client vendors. We must be able to provide specifics regarding the style, color, size and all other characteristics of garments purchased via B2B interchange of order information. Encrypted XML is used extensively to interchange the required information. My Virtual

Model must be non-intrusive to the client e-Commerce site and must be able to interact with all clients solutions.

Network of interoperating web sites – Mobility

My Virtual Model is implemented as a series of independent web servers. For each client there exists at least one of each of the following:

- Client e-Commerce
- Fashion Server Site
- Model Server Site
- Various Data Servers

A three tier architecture is used, using the familiar Web Server / Application Server / Data Server layering.

Virtual Models are mobile. Users can access their models on any of the affiliate sites, can travel from site to site, and can send their Virtual Model by e-mail. Mobility pumps the heart of the My Virtual Model Network.

Business risks

Business risks must be reviewed on a frequent basis to ensure that testing priorities are aligned with the corporate realities. We may have to react to a competitive threat or new short time frame opportunity.

Generally business risks are driven by:

- Fixed timeframe projects tied to industry dates for seasonal collections of garments
- Large and demanding User community (consistency, performance, reliability)
- Functionality always tied to market "$$" related revenues e.g. a new type of model can be introduced to target a new market (for example tall men)
- New business model is evolving (including revenue) as project continues!
- Organizational change
- Requirement turbulence

Typical Project Example

The typical project encountered has a tight time frame. From the moment clothing is in hand (hundreds of garments), to commercial deployment, takes a maximum of 12 weeks elapsed. Often projects require less than 8 weeks to complete.

© Kim Davis, Robert Sabourin, 2001

The user community per customer includes 100s of thousands of users, all clients are large retailers, thus generating many hits fast!

The retail business is similar to the Web business in that customers offer no forgiveness! If the software or site is not operational they will move elsewhere.

It is very problematic to remove, or trading down, functionality in order to hit a release target. In fact quite the opposite happens and due to compelling business requirements the release date is often moved up!

Technical Risks

The projects involve considerable technical risks. Technologies used are continuously changing and evolving. The company addresses a segment of the market which continuously demands the latest and greatest web widgets!

So our projects experience fast technology churn. The technologies being used in production releases are new to the company, marketing team, developers, testers and system integrators. Developers and testers are learning about the limitations of the solutions on the fly and often have to adapt architectures in mid project.

Technologies are often used in a production environments with limited knowledge. We have limited time for training and for evaluation of the technologies.

Staff turnover can have a devastating effect on such projects. New programmers are not familiar with the existing code base and can accidentally miss something when adding a new feature or adapting to a new embedded technology.

Developers are often not familiar with the "live" production parameters. Often the customer has not decided whether parts of the server component will work on NT or Unix based computers.

Whenever requirements change there is a risk of changing development priorities which in turn will impact the code base. How do we elegantly drop what we are doing while not accidentally breaking something!

Changes in mid-project included:

- A new application server framework, at the business logic layer, moving to an Apache server with JServ to one with JRUN.
- A new JSP driven presentation layer.

Important technical risks were also introduced due to the reuse of existing code in a new context.

Testing Objectives & Risk

Summarizing the specific technical and business risks associated with a project lead to the identification of primary testing objectives. We were able to prioritize these based on our knowledge of the areas of highest risk. Testing effort was spread across testing assignments proportional to the associated risk.

We considered the potential areas of instability to be:

- Functions particularly related to new technologies
- New stuff or changed stuff
- New functions added due to business

Exploratory Testing Approach

An exploratory testing technique was used to help find bugs, and identify problem areas of the application. To quote James Bach, *"In operational terms, exploratory testing is an interactive process of concurrent product exploration, test design and test execution."*

Exploratory testing is a systematic approach. It allows for the concurrent design and execution of tests. As the application is explored a detailed record is kept of the areas explored. Information is captured on standardized templates. Information gathered includes:

What was tested?
How was testing done?
What was discovered about the application?
What bugs were identified?
What oracles were used in an attempt to validate results?
What new discoveries may be of interest to future exploratory testers!?

A senior test lead is responsible for triaging test assignments to various members of the testing team.

The test lead uses knowledge gathered in testing combined with the risk assessment and the potential areas of instability to define the next series of test assignments.

The test lead is operating at the testing mission control. Different testers act as emissaries exploring along the directions indicated by the test lead. The test lead collects all results and saves them in a repository (file system) as a map which is being built up as the application is further explored.

Testing assignments include specific objectives. Testers must use their own judgement in following them. If a tester uncovers an area of instability on the route he may explore it or document it for future exploration. The decision has to be made on the spot.

© Kim Davis, Robert Sabourin, 2001

Bug Cluster Identification

Mapped areas of instability define clusters composed of the bugs that have natural affinities of causes. As the test lead collects results from each "chunk" of exploration, two important assessments are done with his peer in development.

1- All new bugs are reviewed to determine if they are indicative of a pattern, are they due to similar or potentially related faults? Is there a correlation between them?
2- Are newly identified bugs potentially related to previously discovered bugs? Is a pattern emerging?

When a suspected cluster is identified the testing team works closely with the development team to find the probable root cause of the problem.

Clusters can be observed in a variety of ways. Here are four examples of "distance metrics" in bug space:

Functionality:	several bugs related to same functionality are discovered
Reliability:	different functions fail in similar way
Efficiency:	several operations are inefficiently using system resources in a similar manner
Time:	several content sources are out of sync

Prioritize based on business impact

Once a bug cluster is identified it must be prioritized. Just like you would prioritize an individual bug, a cluster of bugs must be prioritized. We assigned clusters one of four priorities:

P1 - Fix immediately, next build
P2 - Fix before commercial release
P3 - Fix in some future commercial release
P4 - Do not fix

Certainly the decision to correct the defect had a lot to do with the severity and business importance of the associated cluster. If for example a memory leak is discovered on a server application which causes an occasional failure which results in a very short term delay in service, but without any loss of data, then it may be prudent not to correct it. If however the same memory leak is aggravated by increased concurrent use of the site then it may critical to fix it before going live.

By working on bugs in a collection we can make decisions more efficiently.

© Kim Davis, Robert Sabourin, 2001

Practical results

The techniques developed lead to health communication and team work between developers and testers. Developers and testers enjoyed working together to try and identify probable root causes of sets of seemingly related bugs.

Developers were much more aware of the role of testing in the process. This is especially beneficial as one of the teams goals is also to improve the Personal Software Process (PSP) of individual team members. Empirical knowledge about the processes and people in place can help since weaker points are sometimes inferred from past projects.

Management and team members all bought-in to approach. Combining exploratory testing with detective work makes sense especially when time is tight and we are always working with incomplete data. People were in sync.

We found that we were able to identify an additional 30% more software problems with this approach. More specifically we measured for every 10 bugs reported the correction of 13 defects in the code base. It is impossible to say however whether the correction of all of these defects would have been necessary.

The method is flexible and can use several alternatives as feeding mechanism to the Bug Cluster Identification phase. The one we chose was Exploratory Testing but more detailed and analytical methods may be used. This will be explored in future work.

Our three projects were released on time with minimal field reported defects. All critical problems were corrected during the final integration phase and hundreds of thousands of users are live using this software as we write this article.

We plan to continue using this technique, improving our understanding of the model and generalizing it as we continue. The effectiveness and efficiency, as well as the limits of the method will be further explored as well. This model is used in a commercial and very turbulent e-Commerce environment. It will evolve. All indications are positive and it is presently being used in several projects under development.

Some Web References

www.satisfice.com

James Bach's web site, exploratory and risk based testing

www.testing.com

Brian Marick's web site, articles about exploratory testing

www.amibug.com

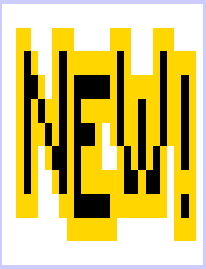
Robert Sabourin's web site, various presentations

www.mvm.com - www.myvirtualmodel.com

My Virtual Model community Web site

www.stickyminds.com

Much relevant content.



QWE2002 Paper 3I

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

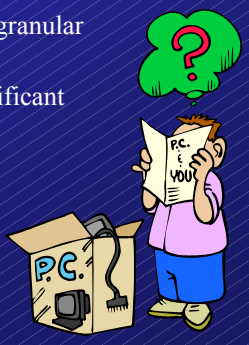
Objectives

- Review performance testing basics
- Describe fundamentals
- Explain success factors
- Review examples

Audience: Web developer, performance engineer, stress test / QA project manager

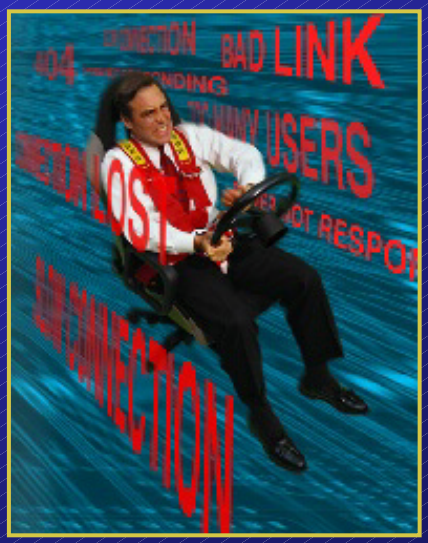
Terms & Concepts

- 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

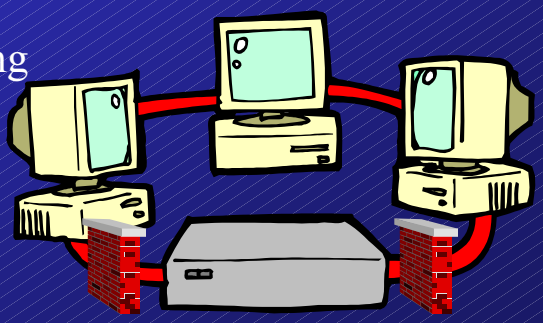
- Perform architectural walkthroughs
- Understand security methods
- Review 3rd party components
- Understand queues
- Identify caching models
- Examine session management
- Verify tool compatibility with AUT

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!

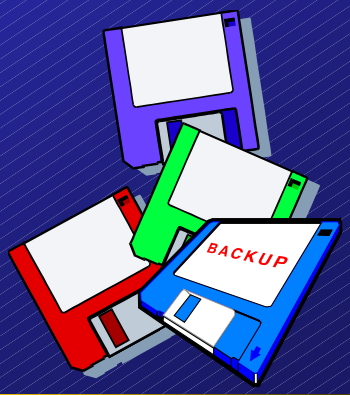
Plan - SUT Architecture

- Review physical infrastructure
- Examine firewalls
- Review connectivity
- Identify load balancing
- Review encryption



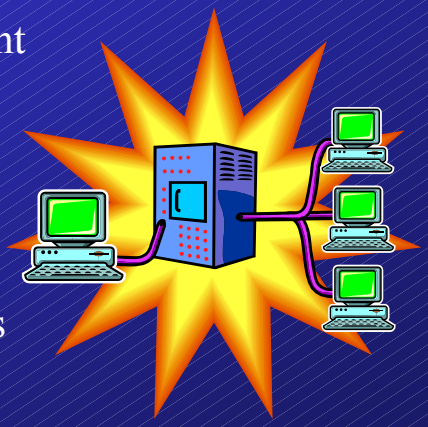
Plan - Test Data

- 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

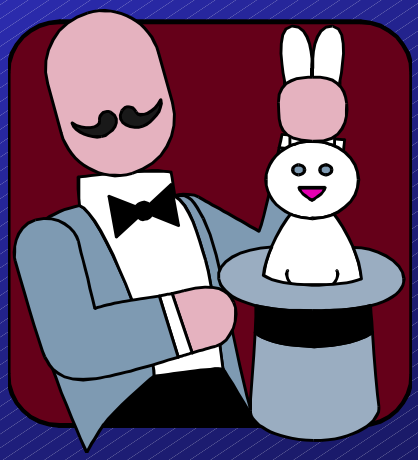
Plan - Test Metrics

- 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

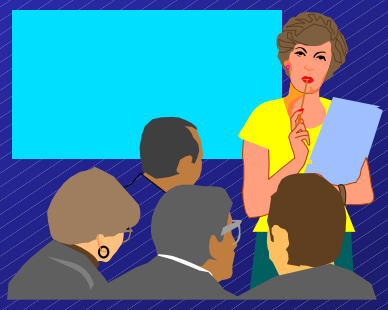
- Reduce logging to production levels
- Ramp-up virtual users
- LAN versus WAN tests
- Identify and troubleshoot bottlenecks
- Track all changes

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

QWE2002 Paper 4I

Mr. Raymond Rivest
(Computer Research Institute of Montreal)

Challenges of Automating Performance Tests for New Internet Technologies

Key Points

- Understanding the technology is vital before making a business proposal Point.
- Automate for scenario replay. Regression is not the issue any more
- New technologies need new solutions and tools

Presentation Abstract

As a testing lab, we have to adapt to various technologies. The process and requirements for performance testing are always the same, but the automation tools does not always support the technology to be tested.

We have developed an approach the narrows the automation challenge down to the performance load scenario replay. It helps understand the system under test before starting the project, thus, provides an automation solution that works.

About the Author

Raymond Rivest is a software test specialist for the Computer Research Institute of Montreal - Software Test Centre. CRIM operates in hi-tech computer application and information technology sectors. A non-profit organisation, CRIM was founded in 1985 to bolster ties between business and academia in these areas. Involved in software development and testing for over 20 years, working with multiple languages, platforms and emerging technologies. Since December 2000, he is the new FAQ maintainer for comp.software.testing and has been involved on swtest-discuss mailing list for 4 years. He initiated the swtest-discuss searchable web archive.

Challenges of Automating Performance Tests for New Internet Technologies

Raymond Rivest
Test specialist
QWE 2002
Brussels

Centre de recherche informatique de Montréal
CRIM SOFTWARE TEST CENTRE

Presentation

- Raymond Rivest is a software test specialist for the Computer Research Institute of Montreal - Software Test Centre. CRIM operates in high-tech computer application and information technology sectors. A non-profit organization, CRIM was founded in 1985 to bolster ties between business and academia in these areas.
- Involved in software development and testing for over 20 years, he has worked with multiple programming languages, platforms and emerging technologies.
- Since December 2000, he is the new FAQ maintainer for comp.software.testing and has been involved on various mailing lists for over 5 years.

Centre de recherche informatique de Montréal
CRIM SOFTWARE TEST CENTRE

Context

- As a non profit organization, the Computer Research Institute of Montreal (CRIM) provides research and development support to Quebec IT companies. CRIM Software Test Centre provides testing facilities, training and counseling.
- CRIM STC also acts as a neutral evaluation service for the health ministry of Quebec. It provides an initial impact assessment, gathering performance metrics such as bandwidth requirements and system performance, for new technologies to be deployed on the Quebec health ministry's WAN.

Summary

- How to structure the evaluation process.
- How to provide standard network, computer and application performance measures (metrics).
- How to organize a test lab to automate performance test.
- How to apply an automation process and metrics list to various technologies.
- How to define the right automation solution that will work.

Objectives

- Of the evaluation
  - Provide an Initial Impact Assessment
  - Provide standardized measures (metrics).
  - Provide a reliable and repeatable evaluation process.
- Of the automation
  - Provide a flexible network environment.
  - Provide a controlled WAN environment.
  - Provide a repeatable load scenario.
  - Provide a cost estimate of services.

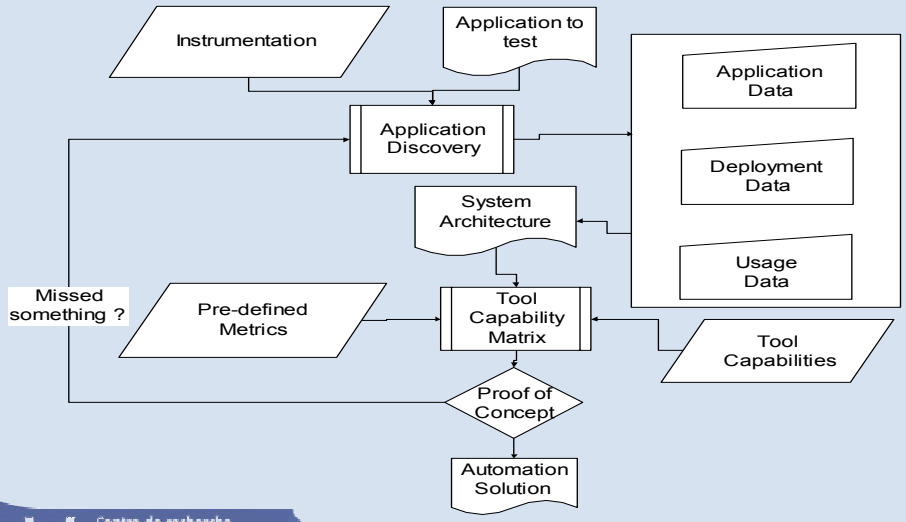
Constraints

- New technologies.
- Multiple deployment (LAN/WAN) conditions.
- Multiple application domains.
- Multiple platforms.
- Multiple system architectures.
- Already unhappy customers :-)

Process

- Define the required measurements (metrics)
- Instrument according to the metrics
- Specialize tools for specific tasks.
- Analyze the system architecture
  - Gather information about the application.
  - Gather information about its current or expected deployment.
  - Gather information about the application usage.
- Compare gathered data with tools capabilities.
- Validate the automation framework.
- Define an automation solution.

General process diagram



Predefined metrics

Computer	Database	Network
CPU Usage	Threads	Response Time
Memory Usage	Cursors	Client Time (spent on)
Disk Access	Cache Usage	Server Time (spent on)
Client Response Time	Compilation	overhead vs data (payload)
	Stored Procedures	Packet size
	Concurrent users	Application turns
	Allocated memory	Protocole usage
	Cache size	Volume/protocol
		Bottlenecks

Instrumentation

Tool	Type	Target	Metrics/Usage
Network Monitors	Agent	Network	• Network Traffic • Network Topology
Communication Analyser	Application	Communication	• Communication flow • Transaction analysis
System Monitors	Agent	Computer	• System Ressources
Database Monitors	Agent	Database Server	• Configuration performance data • Application code data
Network Emulators	• Application • Hardware	Network	• Use live data • Affect communication
Network Simulators	Application	Communication	• Use recorded data • Predict traffic & bottlenecks
Players	Agent	Application	• User output • Virtual users
Functionnal test	Application	Application	• Capture/Replay GUI action • Replay user action
Load generator	Application	• Protocol • GUI • API	• Capture/Replay at protocol level • Replay API calls at protocol level • Response time
Port mapper	Application	Computer	• Security
Version System	Application	• Computer • Database • Software	• Version management
Disk Imaging	Application	Computer	• Configuration management
Defect tracking	Application	All	• Defect tracking and reporting
Smart monkey	Application	Database	• Generate test data

Application discovery

- Use a communication analyser
  - Capture information flow
  - Figure out computer ids
  - Examine used protocols and port numbers
  - Examine packet's content
  - Get transactions size (KB)
- Check network paths
  - Hubs, switches, routers, firewalls, gateways, proxies,etc….
- Try the tools that fit best
  - Protocol capture/replay
  - GUI capture/replay

Application data

Components		
	Type	Is it a process, an application, or something else ?
	Function	What's its purpose ?
	Platform	On what platform does it run ?
Interface		
	Type	GUI, Terminal, Graphic, API ?
	Language	C++, Java, VB framework ?
	# screens	How many screen does it contain ?
	avg elem/Scr	How many elements per screen ?
Communication		
	Protocol	What are the protocols involved ?
	Port	On which ports ?
	HandShake	Any special handshake or connection sequence ?
Security		
	Login	Does it require user id and password ?
	Encryption	Does it use internal encryption ?

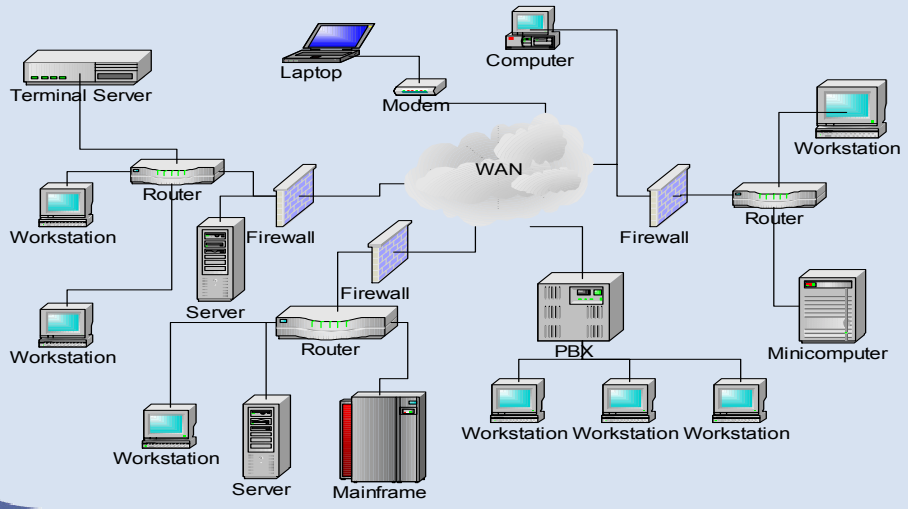
Deployment data

Component		
	Location	Where are the pieces located ?
	Lan Connection	Where is it connected ?
	UpLink	Where is it going ?
	Link Type	How fast will it go ?
Communication		
	Protocol	What are the protocols involved ?
	Port	On which ports ?
	Mode	Does it broadcast, Multicast, or use Nack/Ack ?
Security		
	Firewall	Do we go through firewall ? Which types ?
	VPN	Do we use VPN ?
	Site Certificate	Does it need site certificate ?
	SSL	Does it use Secured Socket Layer ?
	PWD Login	User specific name and password ?
	Proxy	Does it use proxies ?
Network Diagram		
		Get a network diagram

Centre de recherche informatique de Montréal
CRIM SOFTWARE TEST CENTRE

13

Network diagram example

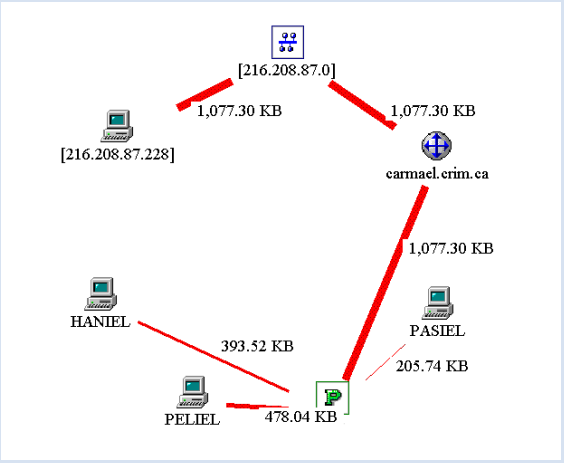


Centre de recherche informatique de Montréal
CRIM SOFTWARE TEST CENTRE

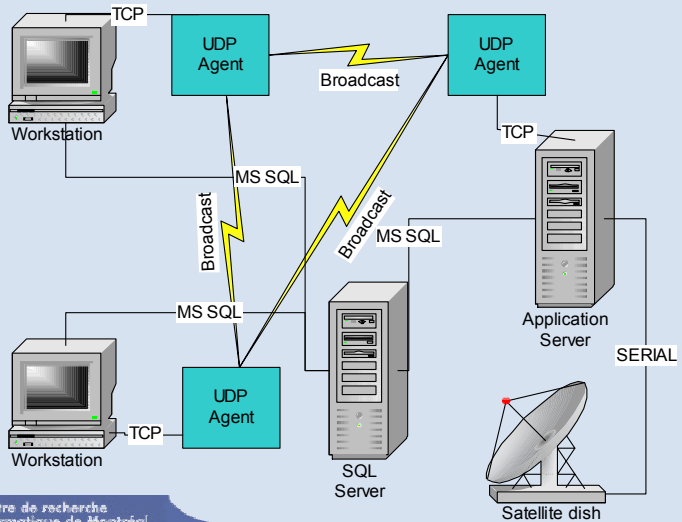
14

Find out network configuration

- See who's involved
- Well, we've got a router in the path….

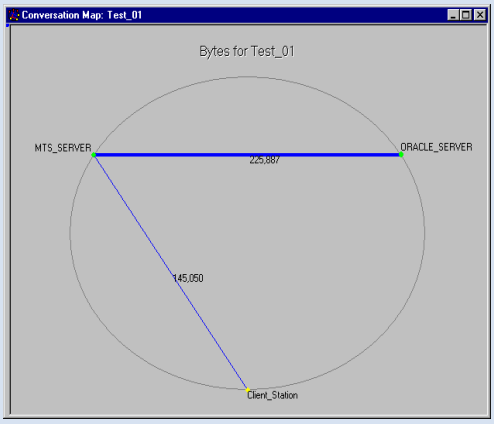


Information flow example



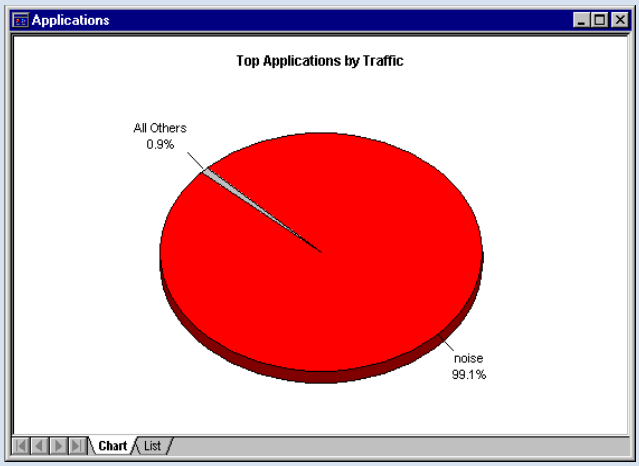
Information flow example

- Figuring out transaction flow is often the key to finding the right tool.
- This is oracle technology, but the main traffic to check is MTS…with DCOM...
- They didn't tell servers were on the same LAN!



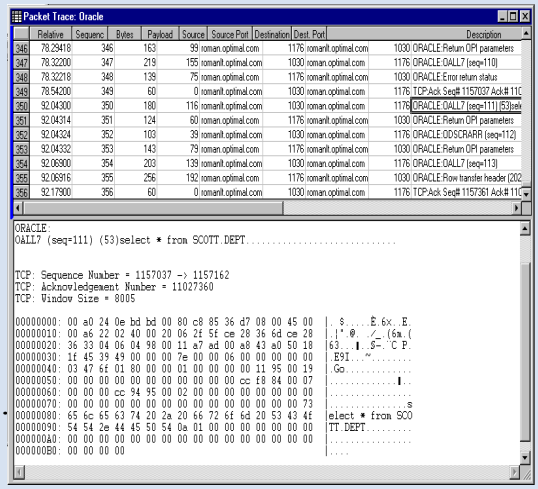
Find problems ahead

- Encryption and VPN's are a problem.
- Am I missing something ?
- From where does the noise start?



Oracle databases

- Oracle is generally readable at the communication level and easy to automate.
- Not always true…
- Some tools can't connect to the API because of application's specifics.



Usage data

Transactions		
	Type	Type of transactions
	User	Done by which users
	Count	How many times
	Pacing	How fast
	Parameters	How many variables
	Screen to go	How many screens to perform the action
	Action /Screen	How many actions per screen
	Parms /Screen	How many variables per screen
	Pre-Condition	What need to be done
	Post-Condition	How it should be afterward
	Receiving from	Where does the information come from
	Transmitting to	Where is it sent

Time Frame	
	Peek Hours
	Peek Days
	Peek Weeks
	Peek Months

Tools matrix

Tool capability Matrix	Application				Tool_x	Tool_y	Tool_z
Items	**Core**	**Agents**	**Database**	**Middleware**	Perf_x	Perf_y	GUI_z
Detail	Client App.		Oracle BD	MTS Server			
Platform	WinNT4.0		Solaris	WinNT4.0Serv	Yes	No	Yes
Type	GUI		Process	Process	Yes	No	Yes
Language	C++		C	C++	No	Yes	Yes
Technologies							
Oracle			X	X	Yes	Yes	Don't care
MS SQL							
DCOM	X			X	Yes	Yes	Don't care
HTTP							
Telnet							
VB Framework							
Java Framework							
TCP/Winsock							
Streaming							
C API							
C++ API							
Lotus Notes/Domino							
Streaming media							
Mutiple				X	Yes	No	Don't care

Tools matrix (Continued)

Tool capability Matrix	Application				Tool_x	Tool_y	Tool_z
Items	**Core**	**Agents**	**Database**	**Middleware**	Perf_x	Perf_y	GUI_z
Protocoles							
TCP	X				Yes	Yes	Don't care
UDP							
HTTP							
DCOM				X	Yes	Yes	Don't care
Winsock							
SQL			X				
Netbios							
SMB							
Mutiple				X	Yes	No	Don't care
Mapped Drives	X			X	Yes	Yes	Yes
Security							
SSL			X	X	Yes	Yes	Yes
SSH					Yes	Yes	Yes
Remote Ident					Yes	Yes	Yes
VPN					Yes	No	Don't care
Internal encryption	X				No	No	Don't care

Tools adaptability & Learning curve

- Any special configuration needed ?
- Any special modules needed ?
- How much time to deploy ?
- Can we add parameters ?
- Is it reusable ?
- What are the hardware requirements ?
- Do we build our own tool ?
- Do we need training or have the qualified staff ?
- Do we have the product, how much does it cost ?

Other aspects to look at

- Wan parameters
  - Latency
  - Bandwidth
  - Routing
  - Packet lost
  - Congestion
- Test environment
  - Production/Development Server
  - Production/Sample Data
  - Live/Close circuit network
  - Backup/Restore database

Proof of concept

- Try it
  - Make a small transaction capture.
  - Replay it.
  - Try to set variables.
  - Replay it.

Does it work ?

Automation solution

- Technology is understood.
- Infrastructure is known.
- Load objectives are defined.
- The appropriate tool is selected and verified.

- You've got the big picture.
- You are now ready to make a cost estimate.

Centre de recherche
informatique de Montréal

CRIM CENTRE DE TESTS
DU LOGICIEL

550, rue Sherbrooke Ouest, bureau 100
Montréal (Québec) H3A 1B9
Téléphone : (514) 840-1234
Télécopieur : (514) 840-1244
http://www.crim.ca

Quality Week Europe 2001
Brussels

Challenges of Automating Performance Tests for New Internet Technologies

Raymond Rivest
Test Specialist
Raymond.rivest@crim.ca

September 2001

INTRODUCTION

Key Points

- Understanding the technology is vital before making a business proposal
- Automate for scenario replay. Regression is not the issue any more.
- The right tool for the right job is the key to success.
- New technologies need new solutions and tools.
- A business proposal based on facts.

Abstract

For a test lab, having a repeatable process is probably the key factor to deliver solid results. Since the products and technologies are always different, the process is more important than the tool. The tool is used for scenario replay and load generation, not for regression tests. But how do you select the right tool? Can we rely on any given tool to fit all possible needs?

The technique we're presenting will help define an automation solution in order to do performance tests for new Internet technologies. By examining all aspects of the project, from software components to network configuration, from information flow to security issues, we gather information in order to get a clear picture of the system under test. All aspects are looked at. It is crucial to understand where monitoring agents should be located.

Knowing what information the tool provides is crucial to understand and define where metrics are. Defining adequate metrics, not drowning valuable information under a mass of useless data is also crucial. What network segments are critical? Where can the bottlenecks be? At the servers end, hubs and switches, firewalls, network bandwidth? The objective is to evaluate a system performance before deployment, not to test a product before a release cycle. Will the system cause chaos in your network when deployed?

It is often better to have smaller, more specialized tools that provide well-defined metrics. One can look at network traffic, the other at computer performance and one for scenario replay. If the scenario generation is the only task required from the automation tools, it provides a smaller objective to fill. There are tools for WAN emulation, some for capture and replay, and others for network monitoring. Some of them are free and sometimes, building a brand new tools is more cost effective.

This technique usually takes a day or two to apply, for each new projects and can save lots of human effort, frustrations and money. It requires knowledge about automation, test tools, programming, networking and the Internet. Finding that your automation tool doesn't work for the technology under test, once the contract is signed and you have to go

All rights reserved © 2001 CRIM

to production, is very costly. Once the project is over, the next one will probably be totally different.

By applying some basic problem solving skills, we are able to define an automation solution that fits the performance test process rather than adapting the test process to the automation tool. A two days investment before signing a contract of many weeks is worth it. It will prevent financial disaster, and allow to deliver on time and on budget.

About the author

Raymond Rivest is a software test specialist for the Computer Research Institute of Montreal - Software Test Centre. CRIM operates in hi-tech computer application and information technology sectors. A non-profit organization, CRIM was founded in 1985 to bolster ties between business and academia in these areas.

Involved in software development and tests for over 20 years, he has been working with multiple languages, platforms and emerging technologies.

Since December 2000, he is the new FAQ maintainer for comp.software.testing and has been involved on swtest-discuss mailing list for 4 years. He initiated the swtest-discuss searchable web archive.

All rights reserved © 2001 CRIM

1. CONTEXT

As a non profit organization, the Computer Research Institute of Montreal (CRIM) provides research and development support to Quebec IT companies. CRIM Software Test Centre provides test facilities, training and counseling.

CRIM STC also acts as a neutral evaluation service for the health ministry of Quebec, to provide performance metrics such as bandwidth requirements and system performance of new systems to be deployed on the Quebec government WAN.

The Quebec Health Ministry has established a complete infrastructure to support full connectivity over the entire health infrastructure of the Quebec province,

In order to monitor the needs and growth of its WAN, the Quebec Health Ministry has decided that application would need the be evaluated and certified as proper for WAN usage, by getting information on their bandwidth requirement ahead of deployment.

2. SUMMARY

How to structure the evaluation process.

One of the main challenges for a test lab, is to provide support for various technologies while measuring network, computer and application performance. The challenge resides in being able to operate multiple projects of many sizes and forms.

When dealing with a certification process, you have to be neutral. You also have to provide a uniform process in order to get always comparable results.

How to provide standard network, computer and application performance measures (metrics).

In order to get standardized metrics, you have to be able to rely on a rigorous process that makes abstraction of the automation tools. The tools have to provide the required metrics.

You are now adapting the automation to the process. Each part of the project can be downsized to provide the required metrics.

How to organize a test lab to automate performance test.

Once the project is divided according to predefined metrics, and automation is defined accordingly, the rest is just a matter of building the load scenario. Some tools are aimed at providing the network metrics, others at the computer or database performances. What's left ? The load scenario automation.

How to apply an automation process and metrics list to various technologies.

All rights reserved © 2001 CRIM

The idea is to gather information in order to understand all aspects of the application and identify which parts of the system are going to use the WAN infrastructure.

By identify each component of the system, its location and relation to other components, you are able to set targets for the metrics and tasks you've defined.

How to define the right automation solution that will work.

Basically, in an automation project, you will select the tool that fits your needs, according to the technology your developing in house. In a test lab, you're faced with throw-away project. Regression is not the issue, only being able to provide a repeatable load scenario is critical.

You need to make sure your automation will work, but for that particular project. But you've got only one shot at it, because, once the contract is signed and milestones are set, you cannot change your tools.

3. OBJECTIVES

3.1 Of the evaluation

Provide an Initial Impact Assessment

Imagine the WAN as a big ecosystem. You've got an ideal environment where all the actual species (applications) live happily without disturbing one another. Now you've got these new species that want to drop in your ecosystem. Are they predators that will destroy your most precious and vulnerable specie, the one that costs a fortune to adapt to the ecosystem? Are they species that carry various infections that will cause chaos in the ecosystem? Or are they so fragile that they will not survive their new environment ?

That's basically what you want to know. Will the application still work when deployed on the WAN and what will be its required bandwidth? Will it disturb important sub-systems on which other important applications depend? Will response time go the unusable heights? Will bandwidth requirements be to expensive?

Provide standardized measures (metrics).

Basically, you want to provide the same type of measures for each project. In order to do so, you'll first identify what measures are standard to most tools.

Metrics can be pretty tricky to define if you don't know at first, what you are looking for. There are some general rule of thumbs you can use.

All rights reserved © 2001 CRIM

Provide a reliable and repeatable evaluation process.

How you work, the metrics you will gather and what you will report have to be standardized. Using checklists and questionnaires is a good way to provide such standards. Using specialized tools is also great at reducing the scope of each part of the process.

3.2 Of the automation

Provide a flexible network environment.

You want to be able to run multiple projects without having interference between them. You need to have access to all of the tools, and make sure there are no surprises like someone doing heavy FTP sessions while you're running your tests.

Provide a controlled WAN environment.

You want to control the WAN so you know how it reacts, what are the elements it is composed of and vary certain aspects of it to evaluate the reactions of the application. Is it sensitive on bandwidth, latency or packet lost? You want to make sure, the environment will be the same for every sequence so you can compare the results.

Provide a repeatable load scenario.

You want to be able to reproduce your workload over and over, while varying the WAN conditions, so you can isolate which parameter will most affect the application. Often, with web application, you will use a load test tool that will operate with protocol capture and replay, so you can multiply the number of virtual users, without requiring the same amount of machines as virtual users.

Provide a cost estimate of services.

You want to be able to estimate properly the cost of automation before signing the contract. But you can't do that, if at very beginning of the automation process, you end up having an automation tool that doesn't work with the technology under test.

4. CONSTRAINTS

New technologies.

Most of the technologies you'll end up testing are brand new. How could test tools be ready for such technologies when they're not even public yet? We know that testing tools are changing at a steady pace, but can they be ahead of future technologies ?

All rights reserved © 2001 CRIM

Multiple deployment (LAN/WAN) conditions.

Sometimes, you'll have the front ends distributed along 4 or 5 various hospital and 10 to 20 external clinics, the middle tier server will be located somewhere between the medical university and the specialized hospital, and the back end server will be located at the very heart of the WAN.

Not to forget the 200 or 300 patients that are connecting remotely from their homes via the internet using good old WWW.

Multiple application domains.

You go from basic accounting systems, up to real time medical imagery system or real time heart beat monitoring system, passing along hospital meals and diets managing systems then back to good old e-mail under Lotus Notes & Domino. The domains change from one project to the other.

Multiple platforms.

Sometime you have cross platform systems where the client front end is a Visual Basic application that connects to a Microsoft Transaction Server using DCOM technology, which in turn connects the an Oracle Database Server on an UltraSparcs server or a Dec-Alpha server.

You can also get a front-end that sits on a PC, but connects through a Linux authentification server, before open an HTTP channel that allowes acces to the AS-400 for Telnet session, while the HTTP sends back graphical update to the front-end. You can also get a front end that is on a Linux box, using Netscape and javascripts, connected to a CORBA/IIOP server that talks back to a mainframe. All the fun stuff.

Multiple system architectures.

The fun part of it. It ranges from basic client/server, 2 tiers, 3 tiers, to multi tiers. All situations you've ever dreamed of.

All rights reserved © 2001 CRIM

5. PROCESS

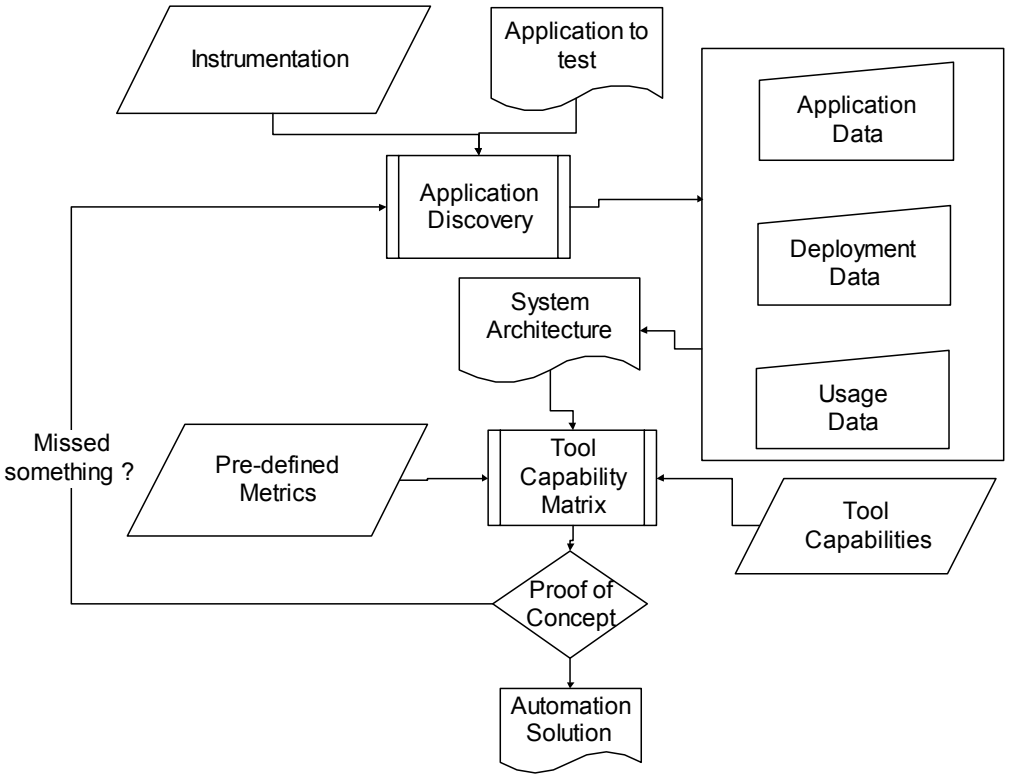


Figure 1 Process diagram

5.1 Define the required measurements (metrics)

Basically, you will need specific measurements for performance tests. These measures can generally be categorized in 3 basic types:

- Computer performance
- Database performance
- Network performance

Inside these 3 categories, there are some general measures that are practically standards among most tools. Some can simply be calculated using basic measures.

All rights reserved © 2001 CRIM

Computer	Database	Network
CPU Usage	Threads	Response Time
Memory Usage	Cursors	Client Time (spent on)
Disk Access	Cache Usage	Server Time (spent on)
Client Response Time	Compilation	overhead vs data (payload)
	Stored Procedures	Packet size
	Concurrent users	Application turns
	Allocated memory	Protocole usage
	Cache size	Volume/protocol
		Throughput
		Bottlenecks

Table 1 Predefined metrics

5.2 Instrument according to the metrics

Once the metrics are defined, you can define the instrumentation. Instrumentation is mostly **"finding the right tool for the right task or metrics"**. Again, you can define some categories in tools.

Some tools are highly specialized in measuring network performance. Most ISPs use these tools to maintain Level of Service (LOS). These tools provides information about band-width usage, protocol distribution, etc.

Some tools are specialized at gathering database performance. They will provide insights about recompilations, stored procedures usage, opened tables, concurrent access, etc.

Other tools are aimed at computer performances, like NT Performance monitor, or rstat on Unix systems but using more sophisticated storage mechanism so you can preserve your data and analyze it afterward.

Other tools can be used for configuration management, defect tracking, etc. The following table provides a brief list of tool types and metrics or tasks they are specialized at.

All rights reserved © 2001 CRIM

Tool	Type	Target	Metrics/Usage
Network Monitors	Agent	Network	• Network Traffic • Network Topology
Communication Analyser	Application	Communication	• Communication flow • Transaction analysis
System Monitors	Agent	Computer	• System Ressources
Database Monitors	Agent	Database Server	• Configuration performance data • Application code data
Network Emulators	• Application • Hardware	Network	• Use live data • Affect communication
Network Simulators	Application	Communication	• Use recorded data • Predict traffic & bottlenecks
Players	Agent	Application	• User output • Virtual users
Functionnal test	Application	Application	• Capture/Replay GUI action • Replay user action
Load generator	Application	• Protocol • GUI • API	• Capture/Replay at protocol level • Replay API calls at protocol level • Response time
Port mapper	Application	Computer	• Security
Version System	Application	• Computer • Database • Software	• Version management
Disk Imaging	Application	Computer	• Configuration management
Defect tracking	Application	All	• Defect tracking and reporting
Smart monkey	Application	Database	• Generate test data

Table 2 Instrumentation

5.3 Specialize tools for specific tasks.

You can now select the tools that are the most portable and cover your needs. Integration is a bit less of an issue since you will have a high specialization of metrics from each tool. It will also be easier on the load generation tool.

You can also aim the tools at cross platform capabilities, so you can use the same tools on most commonly used platforms.

All rights reserved © 2001 CRIM

5.4 Analyze the system architecture

This is probably the trickiest part, but also the single most important one, in order to make proper cost estimates. You need to get the big picture. The technique presented hereafter is a bit derived from "Practical theory for not testing in the dark" by Johanna Rothman and many other approaches to automation.

Using a reverse engineering approach is often a good solution to find the "gotcha's" before they get you. See the chapter "Application discovery" for the detailed approach.

Gather information about the application.

Understanding the application, its components and its purpose is a good place to start. Try to get these information from both users and programmers perspective. As a general rule of thumb, programmers have a way to see thing, end users have a different view. If you're lucky, you can even get to talk with a tester.

Gather information about its current or expected deployment.

Get information on where each an every component of the system will be located. Get information about possible firewalls, security handshakes, etc. Get information about link types, etc.

Most important, try to get a diagram of the intended architecture so you can visualize each an every pieces of the system.

Gather information about the application usage.

End users and trainers are probably your best source of information as for expected load scenarios. A good paper on the subject is probably Cem Kaner's pattern on Scenario Testing.

5.5 Compare gathered data with tools capabilities.

Once you have a clear understanding of the system, you can look at your tools and examine where your network agents have to be located, which computers you'll want to monitor, which database server needs monitoring.

5.6 Validate the automation framework.

Examine the architecture and validate the connectivity aspects. Firewalls, VPNs, SSL, etc. are problems you can foresee.

Try the tool that best fits the need to build up the load scenario.

All rights reserved © 2001 CRIM

5.7 Define an automation solution.

Once you've tried you load scenario tool, you can make a cost estimate.

6. APPLICATION DISCOVERY

Application discovery is vital to understand the technology under test. If you don't have a clear understanding of the entire system, your scenarios won't be realistic, your infrastructure won't be relevant and your automation will probably fail.

The technique described hereafter, is close to reversed engineering. By looking under the hood of the system, you can see what people may forget to tell you, or for that matter, see what's not written in the documentation.

You are often faced with a situation where the source code and flowcharts aren't available. But you need that information to understand how information will flow and select the right performance test tool.

6.1 Use a communication analyser

A communication analyser, also named sniffer, is very usefull in your toolbelt. These nice little tools can pick-up packets, decompose them, and give you a whealth of information.

Capture information flow

Most analyser can decompose packets into vital information:

Information	Description
From Point	The packet started from there
Port	On this specific port
Protocol	Using this protocol
End Point	Received by this station
End Port	On this port
Packet Size	The amount of bytes transmitted
Header Size	The amount of bytes used to direct the data from one point to the other.

Table 3 Information from analyser

Figure out computer ids

Identify the computers or resources that are relevant. A simple DNS server, under a DNS intensive application, can become a bottleneck or weak point.

All rights reserved © 2001 CRIM

Examine used protocols and port numbers.

By examining the protocols and ports, you'll be able to configure your test tool more efficiently. No sense trying to capture half of the actual exchange because you're not trapping all the communication channels.

Examine packets content.

Protocol based tools often work best when you can read the packets like the source code. It will also be simplier to set parameters because everything is clearly readable.

Measure transactions size (KB)

Get some measurements for the most common transactions. By measuring their sizes, you'll be able to figure out possible peaks or bandwidth problems ahead. It will also help in defining the proper load scenario.

6.2 Try each tools

Protocol capture/replay

These are your standard load performance tool. They hook up to the communication ports and record the transactions. Once recorded, transactions can be adapted so parameters can be used instead of hard-coded values.

The problem with these tools, is when your dealing with multiple protocoles that need some kind of synchronization. As for encryption, this is your nightmare. Unless you have access to the algorithm or can make the capture before the encryption takes place, you won't be able to use a performance tool.

The major advantage of these tools, is that they bypass the actual application, so you can run multiple users (in large amounts) on a single box.

GUI capture/replay

GUI capture/replay tools will always work, unless dealing with some new sophisticated environment or too dynamic GUI. They don't care about encryption or multiple protocoles.

The major disadvantage of these tools, is that you usually need a box for each user. But, it is possible to generate an adequate load, if you use spare time on each machine to split the actual transaction rate between all the boxes.

6.3 Check network paths

Hubs, switches, routers, firewalls, gateways, proxies, etc. are all components of a network. You have to take to account difference between these components when trying to identify bottlenecks.

All rights reserved © 2001 CRIM

Hubs are known to be collision generators, thus, congestion on a LAN. Firewall are generally fast enough, but when the demand gets too high for all the filtering required, it can become a bottleneck.

7. APPLICATION DATA

Application data is information about the internals of the system. Core language, components, GUI or character based, etc. The following table is a partial list.

Components		
	Type	Is it a process, an application, or something else ?
	Function	What's its purpose ?
	Platform	On what platform does it run ?
Interface		
	Type	GUI, Terminal, Graphic, API ?
	Language	C++, Java, VB framework ?
	# screens	How many screen does it contain ?
	avg elem/Scr	How many elements per screen ?
Communication		
	Protocol	What are the protocols involved ?
	Port	On which ports ?
	HandShake	Any special handshake or connection sequence ?
Security		
	Login	Does it require user id and password ?
	Encryption	Does it use internal encryption ?

Table 4 Application Data

All rights reserved © 2001 CRIM

8. DEPLOYMENT DATA

Deployment data is the network infrastructure that will be used for the system. You need to clearly understand how and where information will flow. The folowing table is a partial list.

Component		
	Location	Where are the pieces located ?
	Lan Connection	Where is it connected ?
	UpLink	Where is it going ?
	Link Type	How fast will it go ?
Communication		
	Protocol	What are the protocols involved ?
	Port	On which ports ?
	Mode	Does it broadcast, Multicast, or use Nack/Ack ?
Security		
	Firewall	Do we go through firewall ? Which types ?
	VPN	Do we use VPN ?
	Site Certificate	Does it need site certificate ?
	SSL	Does it use Secured Socket Layer ?
	PWD Login	User specific name and password ?
	Proxy	Does it use proxies ?
Network Diagram		
		Get a network diagram

Table 5 Deployment Data

Understanding the network layout can help you locate the flow of communication. It is important to take the right measures, at the right place.

All rights reserved © 2001 CRIM

Network layout

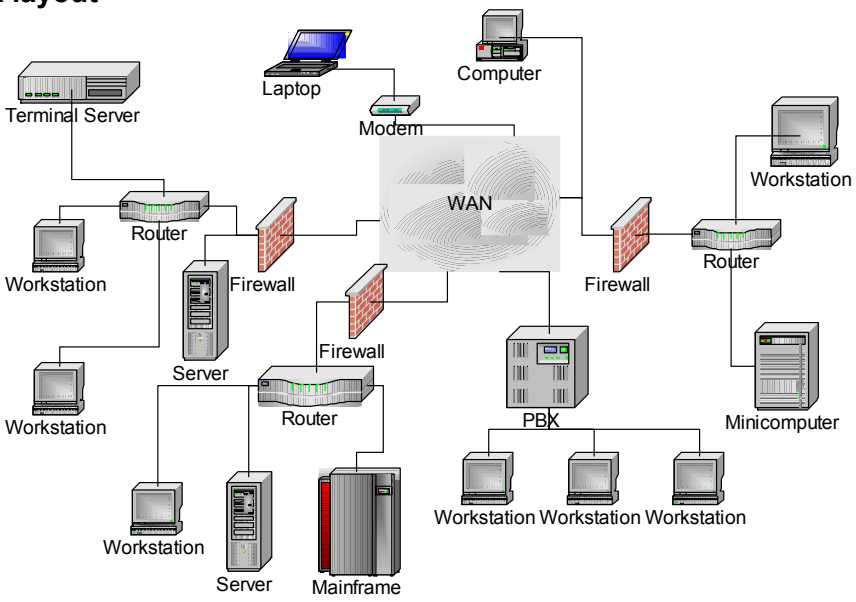


Figure 2 Network example

See who's involved

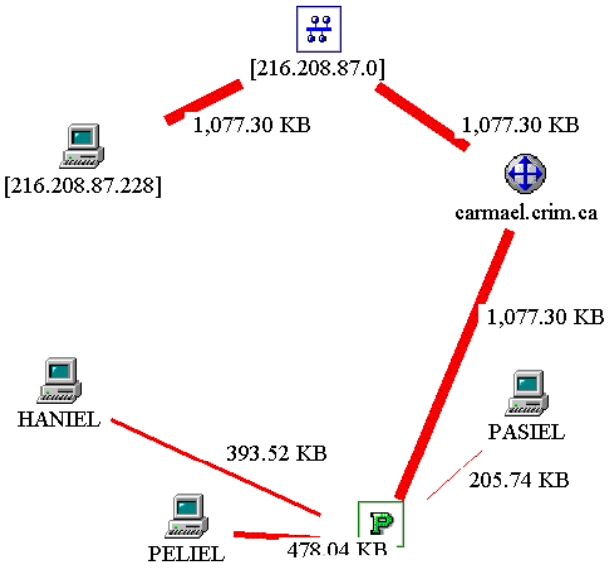


Figure 3 Network path example

Well, we've got a router in the path….

All rights reserved © 2001 CRIM

Information Flow

Information flow is your key to understanding how to build your load. This information will tell you if some protocol transmissions need to be synchronized. It will also tell you information about which information simply broadcasted.

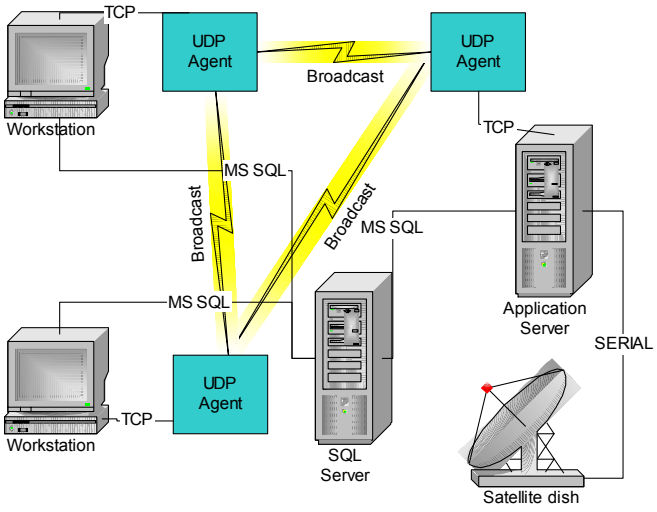


Figure 4 Information flow example

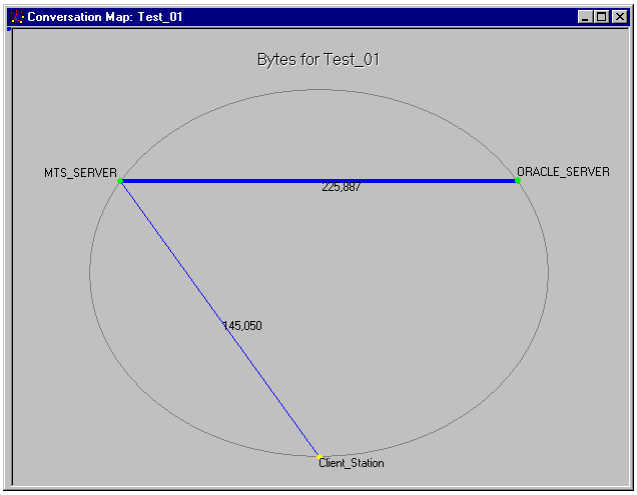


Figure 5 Information flow example

Figuring out transaction flow is often the key to finding the right tool. This is oracle technology, but the main traffic to check is MTS…with DCOM…They didn't tell both servers were on the same LAN! So Oracle is the traffic you need to monitor on the WAN. Computer and Database profiles may be interesting for bottlenecks identification.

All rights reserved © 2001 CRIM

Find Problems Ahead

Encryption and VPN 's are a problem.

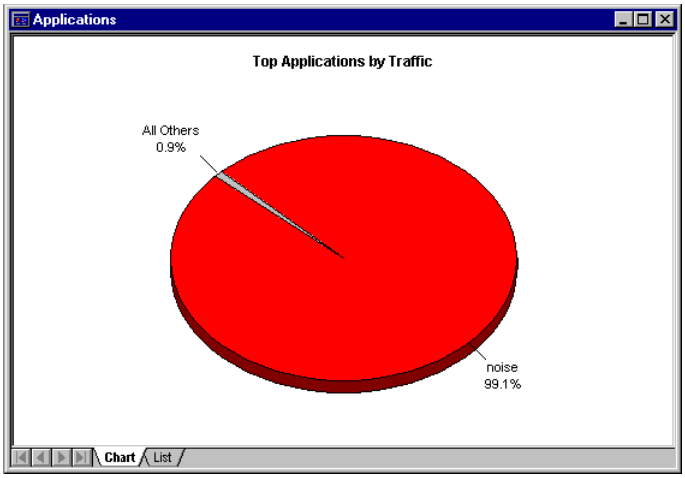


Figure 6 Encryption example

Am I missing something ? From where does the noise start? Encryption renders most of the tools useless. Unless you can capture traffic flow before and after encryption, you'll have a hard time taking measures. You'll need to establish a baseline.

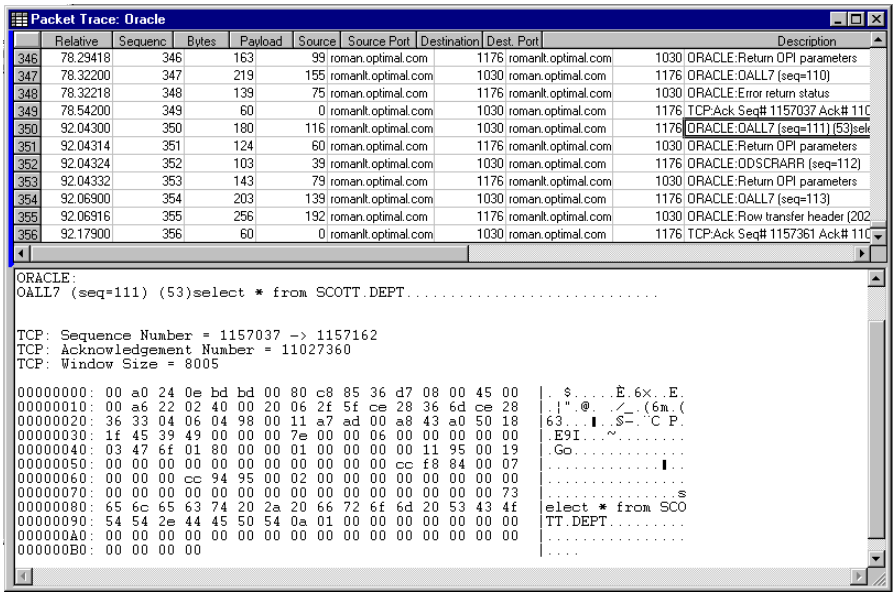


Figure 7 Oracle databases

All rights reserved © 2001 CRIM

Oracle is generally readable at the communication level and easy to automate. Not always true…Some tools can't connect to the API because of application's specifics. Custom components often cause problems when recording, either from the GUI or protocol. They sometime interfere with your tool that cannot hook itself to the API because of sharing violation.

9. USAGE DATA

Usage data is the actual load scenario information you need. This is what will define the amount of virtual users you will need to build up the load, and a key factor to performance tool selection. Again, the following table is just a partial list.

Transactions		
	Type	Type of transactions
	User	Done by which users
	Count	How many times
	Pacing	How fast
	Parameters	How many variables
	Screen to go	How many screens to perform the action
	Action /Screen	How many actions per screen
	Parms /Screen	How many variables per screen
	Pre-Condition	What need to be done
	Post-Condition	How it should be afterward
	Receiving from	Where does the information come from
	Transmitting to	Where is it sent
Time Frame		
	Peek Hours	
	Peek Days	
	Peek Weeks	
	Peek Months	

Table 6 Usage Data

10. TOOLS MATRIX

Tools have many capabilities, but they also have limitations. It is good practice to clearly understand what they can do and how they work. The following table gives you a general list of capabilities that can help classifying test tools.

All rights reserved © 2001 CRIM

This is where you cross-reference what you need and what you have. The following table is just an example of how you can select the right tool for the right job.

Tool capability Matrix	**Application**				**Tool_x**	**Tool_y**	**Tool_z**
Items	**Core**	**Agents**	**Database**	**Middleware**	Perf_x	Perf_y	GUI_z
Detail	Client App.		Oracle BD	MTS Server			
Platform	WinNT4.0		Solaris	WinNT4.0Serv	Yes	No	Yes
Type	GUI		Process	Process	Yes	No	Yes
Language	C++		C	C++	No	Yes	Yes
Technologies							
Oracle			X	X	Yes	Yes	Don't care
MS SQL							
DCOM	X			X	Yes	Yes	Don't care
HTTP							
Telnet							
VB Framework							
Java Framework							
TCP/Winsock							
Streaming							
C API							
C++ API							
Lotus Notes/Domino							
Streaming media							
Mutiple				X	Yes	No	Don't care
Protocoles							
TCP	X				Yes	Yes	Don't care
UDP							
HTTP							
DCOM				X	Yes	Yes	Don't care
Winsock							
SQL			X				
Netbios							
SMB							
Mutiple				X	Yes	No	Don't care
Mapped Drives	X			X	Yes	Yes	Yes
Security							
SSL			X	X	Yes	Yes	Yes
SSH					Yes	Yes	Yes
Remote Ident					Yes	Yes	Yes
VPN					Yes	No	Don't care
Internal encryption	X				No	No	Don't care

Tableau 7 Tools matrix

All rights reserved © 2001 CRIM

11. TOOLS ADAPTABILITY & LEARNING CURVE

Some other aspects about automation and tools have to be looked at carefully :

- Any special configuration needed ? Are there tweaks needed to make it work?
- Any special modules needed? Do you have all the modules available?
- How much time to deploy? What will be needed to install the test tool?
- Can we add parameters? Does the tool allow easy access to variables?
- Is it reusable? Can we use it on other projects using the same type of technology?
- What are the hardware requirements? Do we have what it takes ?
- Do we build our own tool? What about the ROI (Return On Investment)?
- Do we need training or have the qualified staff? Do we have someone to do the job?
- Do we have the product, how much does it cost ? Any budget to by a new tool?

12. OTHER ASPECTS TO LOOK AT

Wan Parameters

Latency. Will it affect our tools?

Bandwidth. Will it be a bottleneck?

Routing. Do we know where our application will go?

Packet lost. Do we need special error handling?

Congestion. Should we go from hubs to switches?

Test Environment

Production or Development Server.

Production or Sample Data.

Live or Close circuit.

How to backup and restore databases.

Access to the tools, license managers, etc…

All rights reserved © 2001 CRIM

13. PROOF OF CONCEPT

That's the crucial step. Try you tool correctly.

- Make a small but representative transaction
- Play it
- Modify the transaction so you can put in parameters.
- Play it a few times

Did it work ?

If it doesn't, either you mist something or there's something wrong.

14. AUTOMATION SOLUTION

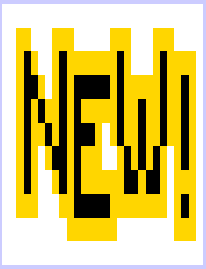
Now that you understand the technology you have to test, you're ready to make the cost estimate.

You checked out all important aspects of the system, network infrastructure, so you have the big picture.

You know how to build your load and which tool you're going to use to generate the load.

You know where the important parts are located, and where to get your metrics.

All rights reserved © 2001 CRIM



QWE2002 Paper 5I

Mr. Michael Avni
(Mercury Interactive UK)

Testing Inside and Outside the Firewall

Key Points

- Testing outside the firewall is the only way to find 35% of the bottlenecks in performance.
- 98% of websites find a critical performance problem.
- they scale up to 15% of the planned capacity on average.

Presentation Abstract

By using traditional load testing tools alone, testers could be missing up to 35 percent of bottlenecks caused by firewalls, routers or ISP peering issues. In Mercury Interactives recent white paper Common Web Site Problems Identified by ActiveTest, testing experts revealed that these bottlenecks are detected only by testing from outside the firewall. This finding strongly indicates that companies need to expand their traditional testing regimen to include load testing from outside the firewall on production systems. Only then can they be assured that all bottlenecks have been eliminated from their Web applications prior to going live. This is what our customers have experienced....

About the Author

To Be Supplied.

Load Testing Inside and Outside the Firewall

Michael Avni
European Field Marketing Manager
Mercury Interactive
mav@mercury.co.il

© 2001 Mercury Interactive Corporation

Global Package Carrier Saves $1.5M in Unwarranted Hardware and Software

- Package tracking system wasn't living up to their performance expectations.
  - The site was supposed to have throughput of 40Mbps, but it was only reaching 6Mbps.
  - The site could support 150 users, only half of their target of 300 users.
  - At peak loads, the application servers were only running at 70% utilization.
  - Response time averaged six seconds, much higher than their goal of two seconds or less.
- Purchased additional hardware to meet their performance and capacity needs.

© 2001 Mercury Interactive Corporation

Global Package Carrier Saves $1.5M in Unwarranted Hardware and Software

- The performance problems were negatively impacting customers, revenue and image.
- Made the decision to use Mercury Interactive's ActiveTest.
- Dramatic improvements in performance
- Canceled purchasing $1.5M hardware & software

Performance Summary:

	Before	After	% Improvement
Hits per Second	20	78	290% Increase
Throughput	1.8 Mbps	7.4 Mbps	311 % increase
# of Users Application Will Support	150	300	100 % increase
Response Time: Browse (at 150 users)	6	2.2	63 % decrease

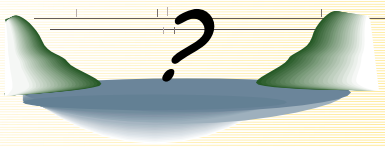
© 2001 Mercury Interactive Corporation

Core Business Risk for IT
Getting Application Deployments to Be Profitable

84% of mission-critical projects fail

Standish Group '99

Software Application Initiatives

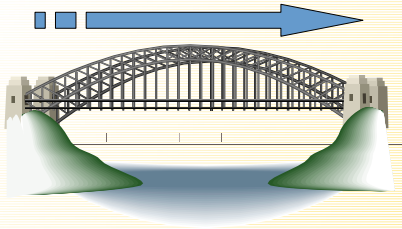


Business Benefits

- Systems don't satisfy business requirements
- Applications don't scale & perform in production
- Limited understanding/control of infrastructure & spending

© 2001 Mercury Interactive Corporation

Automated Testing & Performance Management Value Proposition



Software Application Initiatives

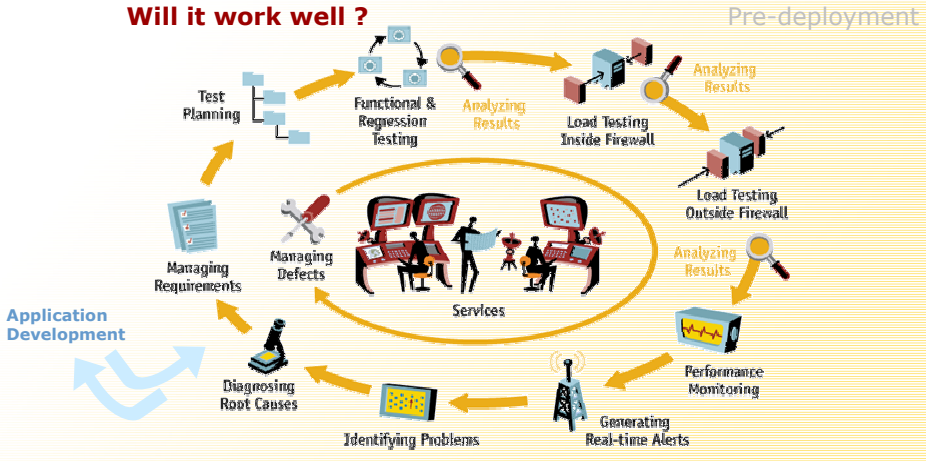
Business Benefits

- Save $millions when deploying new applications
- Maximize performance of existing environments
- Protect revenue streams
- Minimize overall technology investment

© 2001 Mercury Interactive Corporation

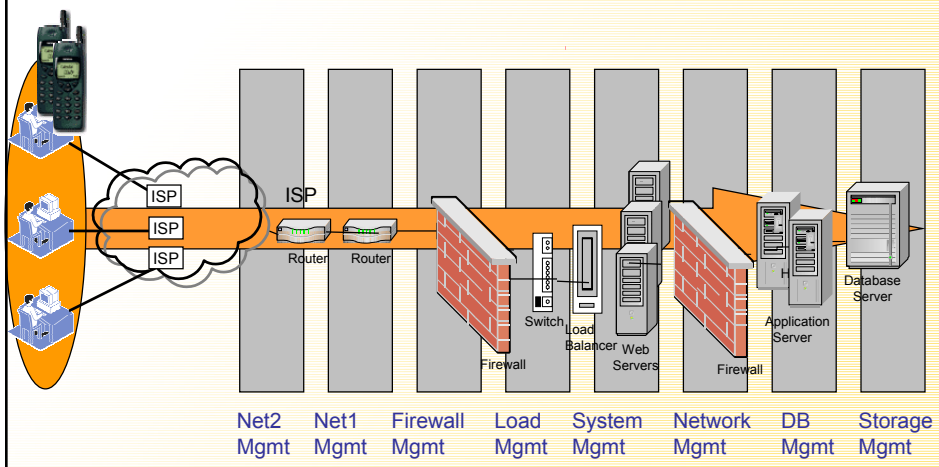
The Performance Management Lifecycle

Integrated approach to deployment



© 2001 Mercury Interactive Corporation

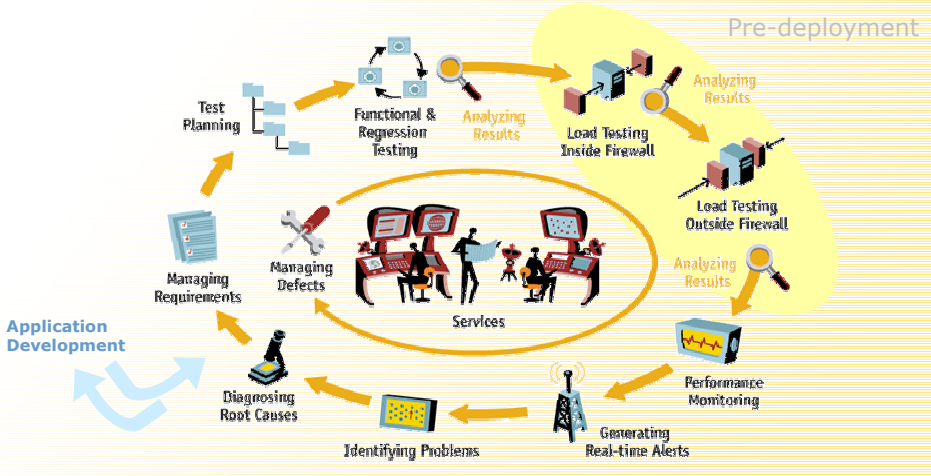
Need to Focus on End-User Experience; Yet,Cut Thru Infrastructure to Isolate Problems



© 2001 Mercury Interactive Corporation

The Performance Management Lifecycle

LoadTesting Inside and Outside the Firewall



© 2001 Mercury Interactive Corporation

Performance Statistics

- 98% of all sites have a critical problem
- Sites only scale to 15% of design capacity
- 35% of all bottlenecks occur outside the firewall

Source: Mercury Interactive
www.mercuryinteractive.com/products/activetest/whitepapers/

© 2001 Mercury Interactive Corporation

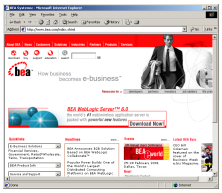
Load Testing Is Mandatory

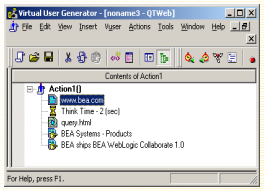
- Saves money
  - Maximizes current infrastructure investments
  - Eliminates spending on unwarranted HW/SW
- Delivers outstanding ROI
  - Improve performance 400%
- Improves end user experience
  - Customers WILL leave if the experience is poor
- Protects revenues & prevents downtime
  - 98% of sites have critical problems
  - Lost business impacts bottom line

© 2001 Mercury Interactive Corporation

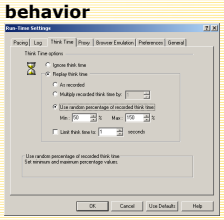
The Load Testing Process

1. Record user interactions against client application into test script

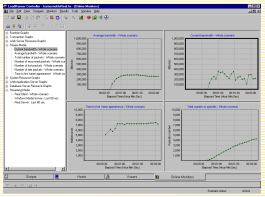




2. Turn scripts into Virtual Users by adding test data and user behavior



3. Run Virtual Users to generate load



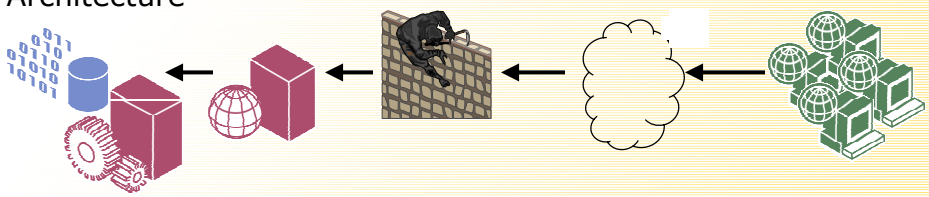
4. Monitor system performance during load test

5. Analyze data and pinpoints bottlenecks

© 2001 Mercury Interactive Corporation

The Biggest Challenge is Testing Inside AND Outside the Firewall

System Architecture | Infrastructure | Firewall | Internet | Users



Phase 1
QA environment
(Test Inside the Firewall)
Controlled environment
Application Logic, Bottlenecks etc.

Phase 2
Production live system
(Test Outside the Firewall)
Real world environment
ISP peering, bandwidth etc.

© 2001 Mercury Interactive Corporation

Top Reasons For Bad Performance

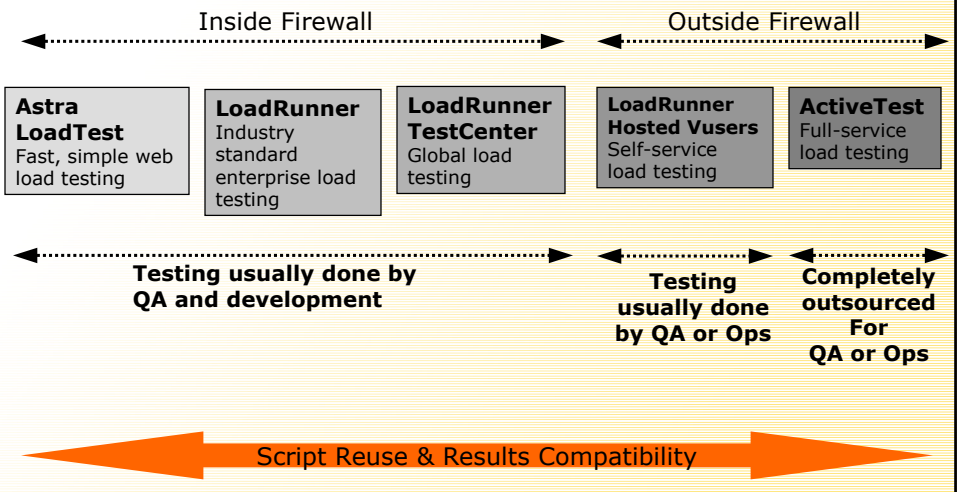
Inside Firewall

Outside Firewall

- Application Server
  - Inefficient use of DB connections
  - Excessive use of DB queries
  - Inefficient caching controlling the number of object instances
- Database Server
  - Locking
  - Long queries
  - Indexes not properly defined
- Web Server
  - Scripting languages are server intensive
  - Excessive use of HTTPS
  - Hardware limitations (memory and CPU)
- Network / Internet infrastructure
  - Badly configured load balancers
  - Insufficient network bandwidth
  - Configuration of network devices
  - ISP peering

© 2001 Mercury Interactive Corporation

A Comprehensive Load Testing Solutions Inside & Outside Firewall



© 2001 Mercury Interactive Corporation

Load Testing - Best Practices

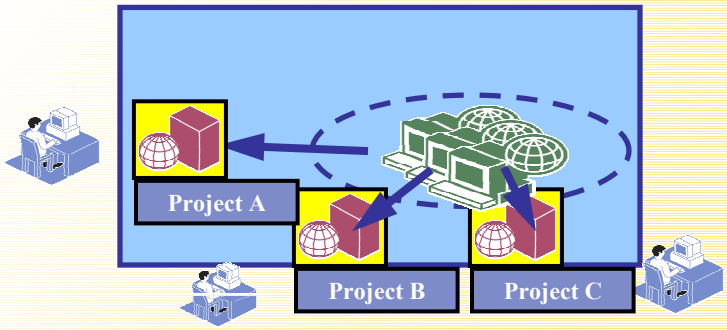
- Clear realistic performance requirements
- Test each component individually
  - Web server, image server, network, firewall, app server, EJBs, database server
- Test inside & outside the firewall to isolate problems across the entire infrastructure
- Test with every application change
  - New release, upgrade or fix (software, hardware, network)
  - Configuration changes (tuning, rollback, caching)

Test early, test often and never stop testing

© 2001 Mercury Interactive Corporation

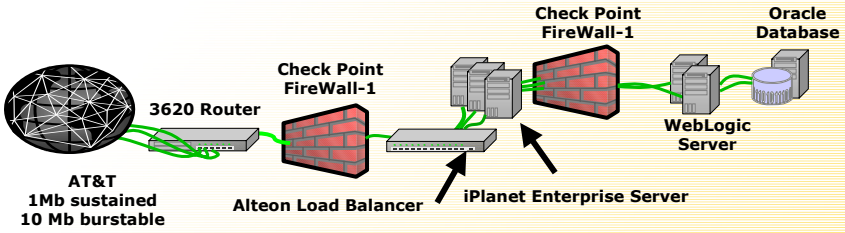
LoadRunner TestCenter - A Global Enterprise Load Testing Tool

- Runs multiple concurrent testing projects, geographically dispersed, across the company
- Eliminates redundancy of resources & shortens testing time
- Increases test efficiency and saves money



© 2001 Mercury Interactive Corporation

Hotel Chain : Results After 3 ActiveTest Test Runs



- 15x improvement in scalability
  - From 20 to 350 users
- 10x improvement in response times
  - From 2 minutes to 10 seconds
- Saved $373K in unnecessary hardware purchases
  - 4 HP servers, BEA licenses, and Alteon switch

© 2001 Mercury Interactive Corporation

Mercury Interactive's
Exemplary customer benefits

Consumer Manufacturer
Online Commerce and Promotion
- Scale from 300 to 25,000 users
- Test B2B integration

Government Agency
Integrated Employment System
- 25 days to update tax handling
- Test 15K transactions, not 750

Leading Photography Supplier
E-Business SAP Implementation
- 4 week delivery on 6 month schedule
- 3,000 users with no disruption

© 2001 Mercury Interactive Corporation

Mercury Interactive's
Exemplary ROI

Oppenheimer Fund

Functional Testing ROI 1,888% - every $ invested they had $19 return

Load Testing ROI 1,398% - every $ invested they had $14 return

A reduction of 4-5 weeks in time-to-market for each product release.

Significantly increased productivity of the functional testing process.

Quality, consistency, and scalability in performance testing which previously was either difficult or impossible to achieve.

The ability to segment performance testing results and quickly identify unexpected scalability problems that would have been devastating had they been encountered in production.

IDC white paper August 2001
Measured ROI over 3 years of
3 large enterprise customers

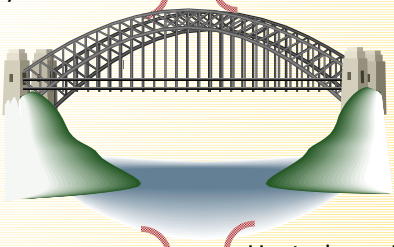
© 2001 Mercury Interactive Corporation

Automated Testing & Performance Management Delivers Breakthrough Value

Test Management determines app readiness, focuses testing efforts and shortens testing cycles

Automated functional testing delivers greater coverage and eliminates human error

Software Application Initiatives



Business Benefits

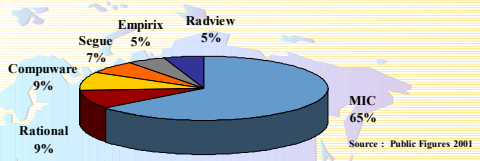
Load testing prior to deployment (inside/outside the Firewall) ensures uptime and positive user experience

Hosted services complement product solutions and offer 400%+ performance improvement

© 2001 Mercury Interactive Corporation

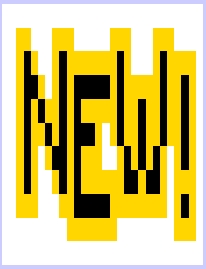
Mercury Interactive
Enterprise Testing and Performance Management

- Dominant market share
- 25,000 customers WW
- Global organization
  - Headquarters in Sunnyvale, California
    Regional Operations in Israel, UK, Singapore
  - 60+ offices in 20 countries
  - 1,400+ employees
- Financially strong
  - Multi-billion dollar market capitalization
  - Public since 1993
  - Member of the S&P 500 and Nasdaq-100



Source : Public Figures 2001

© 2001 Mercury Interactive Corporation



QWE2002 Paper 6I

Mr. Paul McBride
(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 properly.

About the Author

Mr. Paul McBride is Director of European Operations for VeriTest. Paul has a background in software design and engineering with Microsoft corp. from 1990 to 1994 and has been running business operations for VeriTest in Europe for the past four years. Prior to this, he managed a Business Unit of Lionbridge Technologies with responsibility for delivering software globalisation solutions. His recent experience is in the certification, test and deployment of software on digital media devices and together with his team, is developing QA solutions for applications under development which will be deployed on the next generation of wireless internet devices.

QWE2002 Paper 7I

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 (Paper 7I)

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

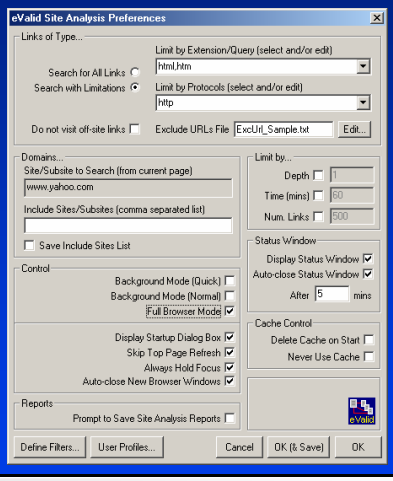
Technology Base

- 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



eValid, Inc.

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

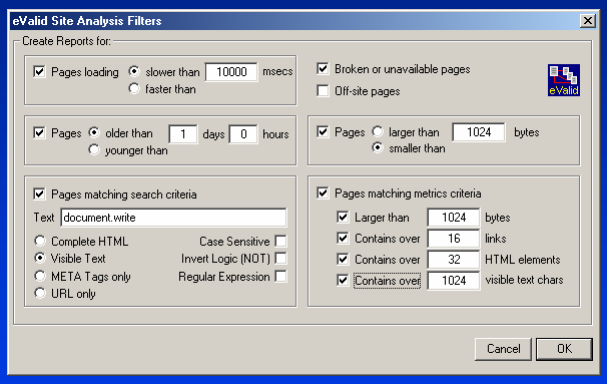
Search Mode

- Browser Mode (Foreground)
- Background Mode (Normal)
- Background Mode (Quick)
- Performance Differences
- Result Differences
- Performance Issues

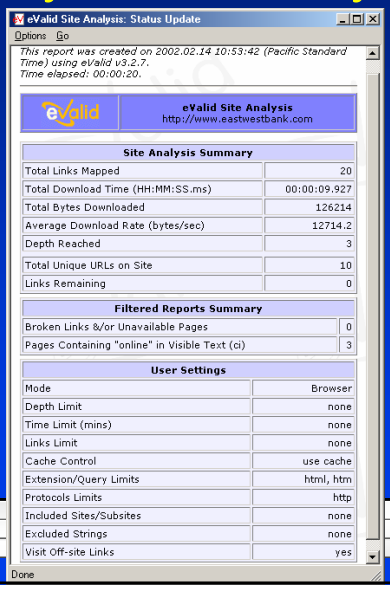
Site Analysis Filtering

- Unavailable Pages
- Slow Pages
- Old Pages
- Large Pages
- External Pages
- Matching String
- Matching Regular Expression
- Metrics
- Case Study

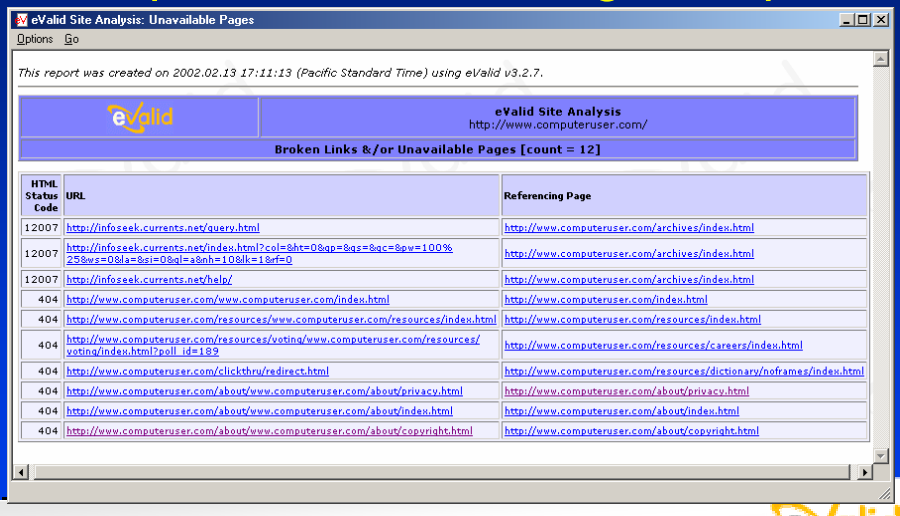
Filter Selection Menu



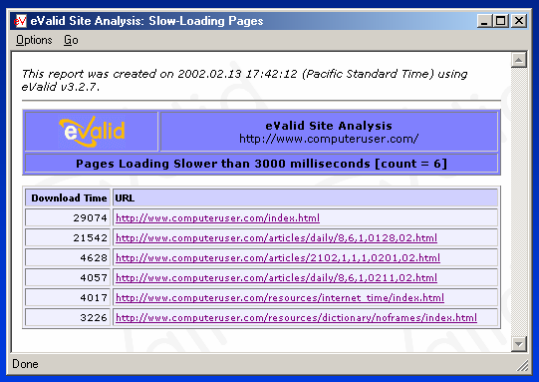
Site Analysis Summary Report



Sample Unavailable Pages Report

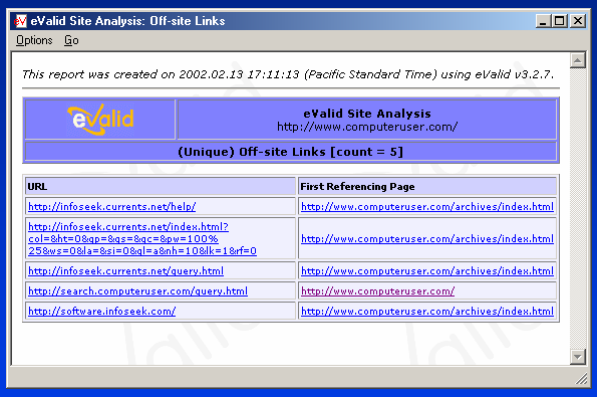


Sample Slow Loading Pages Report



Sample Too Old Pages Report

Sample Off Site Pages Report



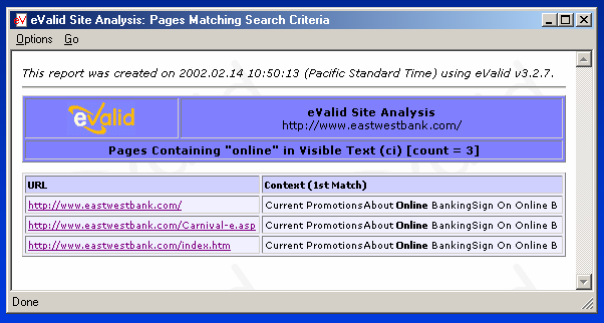
String Match Target Selection

- Complete HTML
- Visible Text
- META Tags Only
- URL Text Only

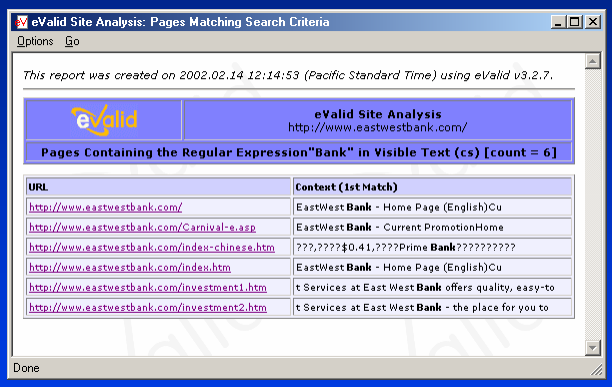
String Match Specification

- Base String
- Case Sensitivity
- Inverted Logic Sense (NOT)
- Regular Expression

Simple String Match Report



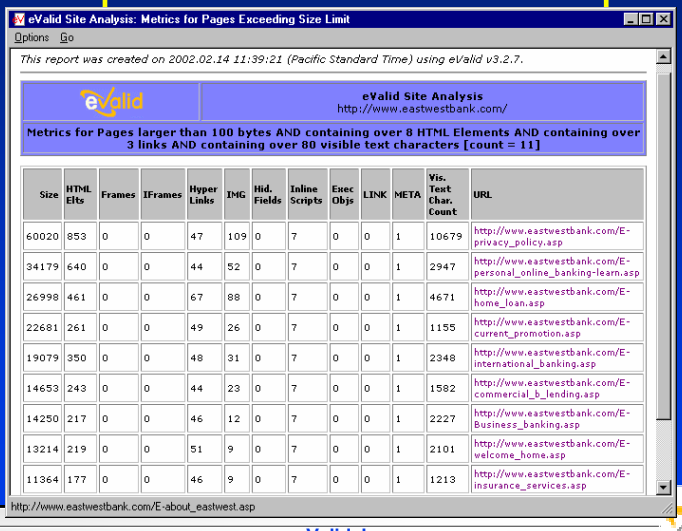
Regular Expression Match Report



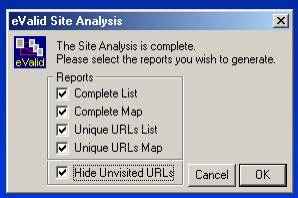
Pages Matching Metric Criteria

- 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

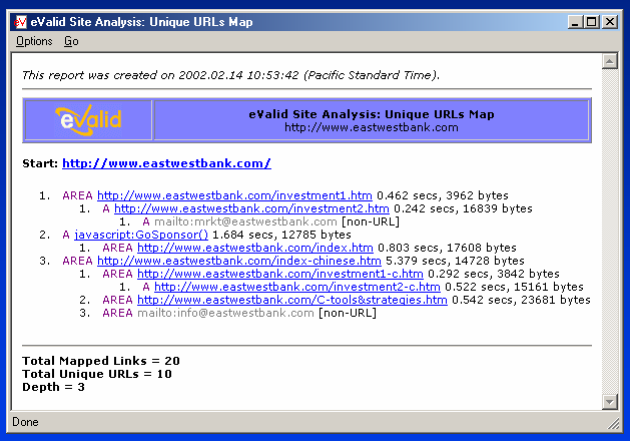


Site Analysis Report Selection

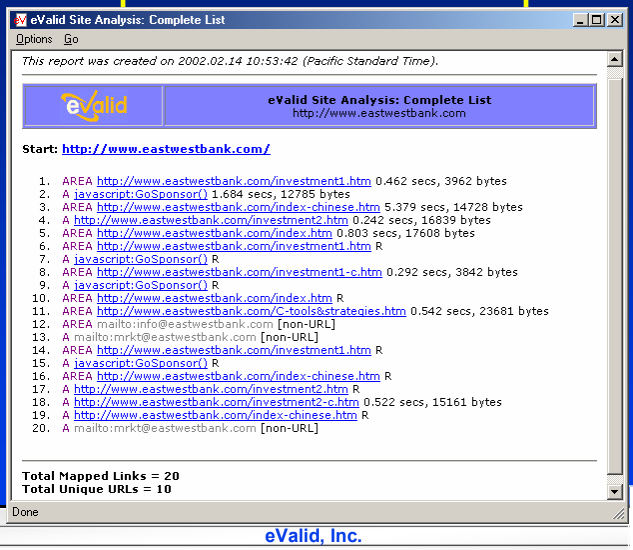
- Full SiteMap
- Unique SiteMap
- Full URL List
- Unique URL List
- Show/Don't Show Unvisited URLs

Sample Full SiteMap Report

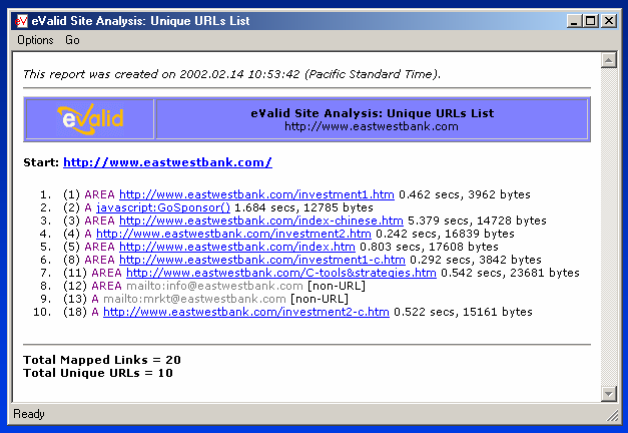
Sample Unique SiteMap Report



Sample Full URLs Report



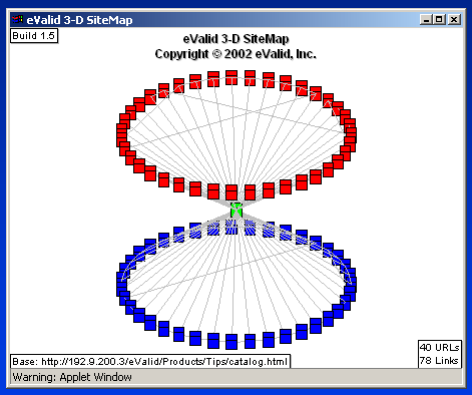
Sample Unique URLs Report



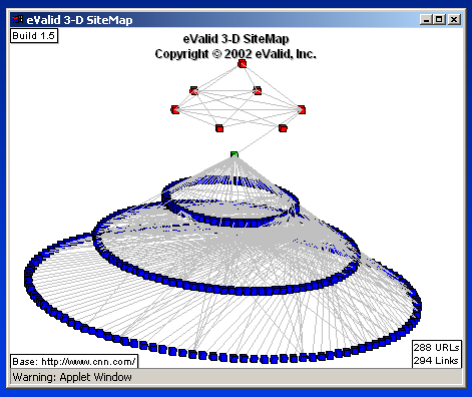
3D SiteMap Graphs

- Derived From Complete SiteMap Table
- Fully Scalable
- Fully Rotatable (Horizontal & Vertical)
- Dependence Annotation
- Redraw from Current Image

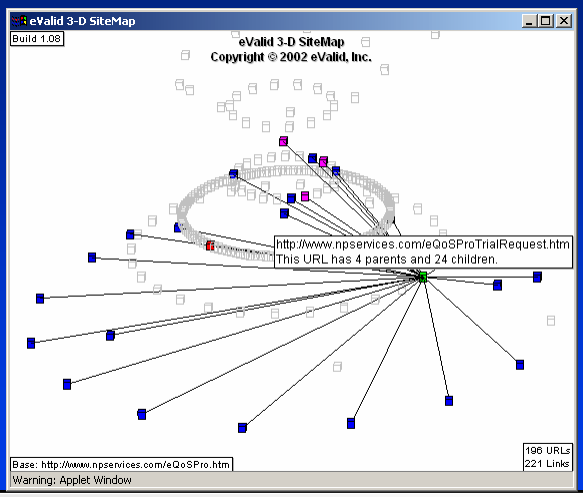
Simple 3D SiteMap



Complex 3D SiteMap

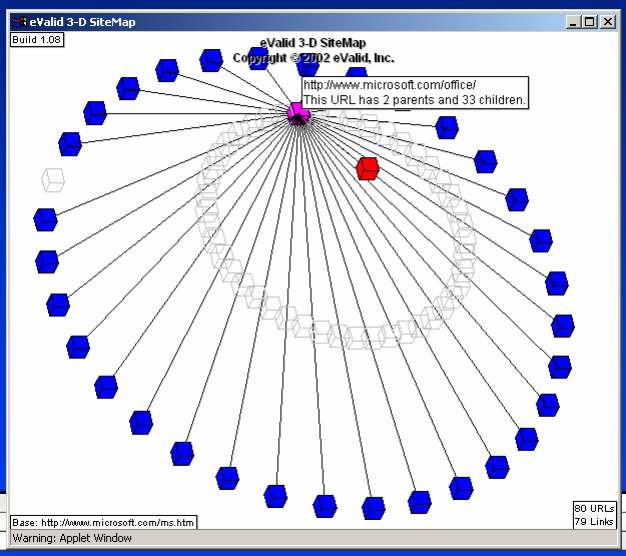


Dependency Display #1



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

QWE2002 Paper 8I

Mr. Scott Jefferies
(Starbase Corporation)

A Requirements-Based Approach to Delivering E-Business and Enterprise Applications

Key Points

- Minimize application failure and revenue loss and maintain loyal customers.
- Shorten development timeframe, decrease costs and improve application quality.
- Integrated requirements-based approach that enables project teams to optimize initiatives.

Presentation Abstract

This presentation will demonstrate to attendees how a requirements-based approach to delivering E-business and enterprise applications will minimize application failure, avoid revenue loss and maintain loyal customers. And with a proven process in place, organizations can make an even bigger impact on the development cycle, shortening the development timeframe, decreasing costs and significantly improving application quality.

About the Author

Currently a Regional Technical Manager with Starbase, Scott Jefferies has over 25 years' of experience in providing business and information management solutions to Fortune 1000 companies. He is experienced in the installation and implementation of automated software quality (ASQ) tools on a variety of platforms.

Prior to Starbase, Scott also served as Technology Engineer for Technology Builders, Inc. (TBI), where he was responsible for the implementation of both requirements management and automated testing tools, with a focus on the integration of those tools. Scott was a key player in establishing TBI 's requirements-based approach to delivering e-business and enterprise applications, which includes requirements management, test case design, test and defect management, and automated testing. In February 2001, TBI was acquired by Starbase, the leading provider of end-to-end collaboration products for both the development and management of content and code for eBusiness applications.

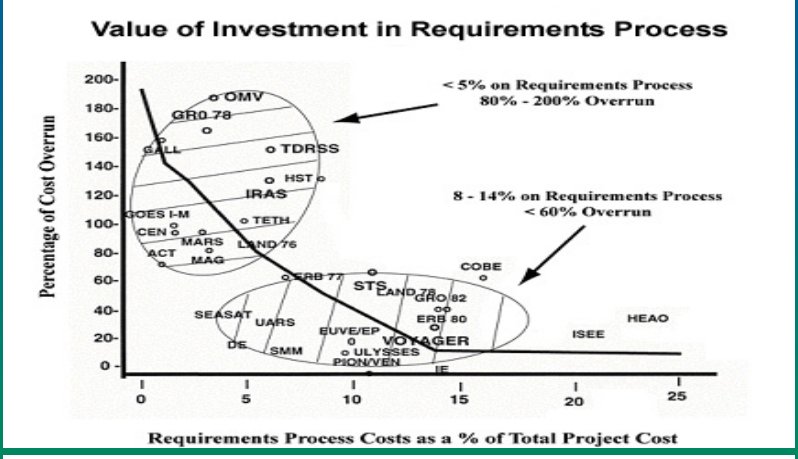
A Requirements-Based Approach to Delivering E-business and Enterprise Applications

Scott Jefferies
Technical Director
Starbase Corporation

Agenda

- Gathering and defining requirements
- Performing ambiguity reviews
- Managing requirements
- Designing test cases
- Defining test completion criteria
- Other steps in the process

Why A Requirements-Based Approach?



Data collected from 30 actual NASA projects which show that the probability of project success (as measured by meeting its Target Cost) is greatest when 8-14% of the Total Project Cost is invested in requirements activities. Source: Ivy Hooks

Gathering and Defining Requirements

- Who should be involved:
  - Business analysts
  - Users
  - Other project stakeholders

Gathering and Defining Requirements

- How to gather requirements:
  - User interviews
  - Brainstorming
  - Facilitated sessions
  - Project specification

5

Gathering and Defining Requirements

- What to gather:
  - The requirement itself
    - "the system shall…"
  - Attributes
    - Priority, need, precedence, relationships with other requirements
  - Supporting information
    - Graphics, object models, regulations

6

Performing Ambiguity Reviews

- Who should be involved:
  - Newest member of the team
  - Business analysts
  - Users
  - Other project stakeholders

starbase

Performing Ambiguity Reviews

- Why we should perform ambiguity reviews:
  - Eliminate ambiguities
  - Identify conflicts and logic errors
  - Reorganize requirements for clarity

starbase

Performing Ambiguity Reviews

- What we are looking for and how to find them:
  - Traceability and inconsistency errors
    - Requirement traces to nothing (orphans)
      - Use Traceability Matrix to identify
    - Terms used inconsistently

Performing Ambiguity Reviews

- What we are looking for and how to find them:
  - Imprecise terminology
    - Acronyms, company- or industry-specific jargon, non-explicit terms ("quickly," "user-friendly")
      - Use outside resource to help identify
      - Create project glossary to explain terms used

Imprecise Terminology Exercise

How many examples of imprecise terminology can you find in the following:

The ATM shall respond quickly and in a user-friendly manner to any user action, and print a TR when the transaction is completed.

11

Imprecise Terminology Exercise

How many examples of imprecise terminology can you find in the following:

The ATM shall respond quickly and in a user-friendly manner to any user action, and print a TR when the transaction is completed.

12

Performing Ambiguity Reviews

- What we are looking for and how to find them:
  - Ambiguities
    - "If the table is next to the chair, then move it."
      - Look for exceptions, use outside resource
  - Logical errors
    - "If A and B then C" … "If A and B then Not C"
      - Use cause-effect graphs, rearrange requirements

starbase

Performing Ambiguity Reviews

- What we are looking for and how to find them:
  - Undocumented assumptions
    - "The system shall perform the calculation in the usual way."
      - In-depth interview by business analyst
      - Make inferences

starbase

Performing Ambiguity Reviews

- **How** the **requirements are improved:**
  - **Become testable**
    - **Deterministic, unambiguous, complete, non-redundant, traceable, explicit and feasible**
  - **Become easier for developers to work with**
  - **Become easier to manage**

starbase

Transportation Device Example

- **Transport one person at a time**
- **Over hard flat surfaces**
- **At speeds not to exceed 20 miles per hour**
- **For distances up to 2 miles**
- **Using only person power for locomotion**
- **Personal comfort is not important**

???

starbase

Managing Requirements

- What is involved:
  - Manage changes (errors, new requirements and user requests)
    - Reduces scope creep due to unnecessary changes
  - Establish priorities
    - Focuses development on core set of requirements

starbase

Managing Requirements

- What is involved:
  - Assign responsibilities
    - Assists in communicating changes
  - Document rationale
    - Allows project team to understand why certain decisions were made or changes not made

starbase

Managing Requirements

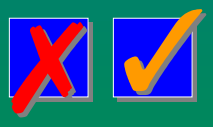
- What is involved:
  - Trace requirement relationships
    - Allows impact analysis for more informed decisions
  - Communicate changes
    - Allows entire project team to understand current project status and scope

Managing Requirements

- What is involved:
  - Establish baselines
    - Tracks scope creep and changes for management
  - Track requirement histories
    - Ensures audit trail

Designing Test Cases

- What factors should be considered:
  - Test cases should be based on requirements
  - Test data should be designed to provide maximum coverage with minimum number of tests



starbase

Designing Test Cases

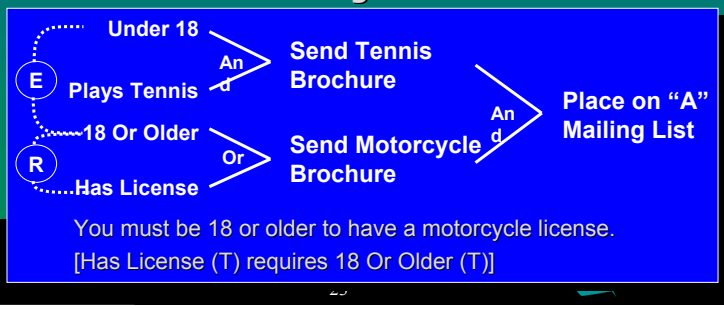
- What factors should be considered:
  - Use cause-effect graphing to clarify requirement relationships and testing needs
  - Group test cases into test sets for ease of execution/organization

starbase

Cause-Effect Graph Example

Criteria

- If the person is under 18 and plays tennis, then send them a tennis club brochure.
- If the person is 18 or older, or has a motorcycle license, then send them a motorcycle club brochure.
- If the person was sent both brochures, then put them on the "A" mailing list.



Designing Test Cases

- How testers are currently designing tests:
  - "Gut feel"
  - Live data
  - Brute force combinations

Designing Test Cases

- Why current methods fall short:
  - Too many tests, too little time
  - Varying skill levels and experience
  - Not enough coverage (functional or code) to ensure system integrity

starbase

Designing Test Cases

- The solution:
  - Use scientific methods to produce the minimum number of test cases that will validate all of the functional requirements
  - Result: 100% functionality coverage and 85%-90% code coverage on the first pass

starbase

Reviewing Test Cases

- Who should be involved:
  - Spec writer
  - User/domain experts
  - Developers
- Why this is important:
  - Minimizes misunderstandings

27

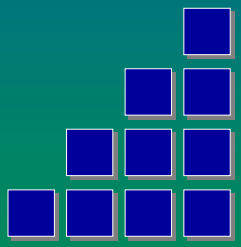
Defining Test Completion Criteria

- Why this is important:
  - Sets policy for when software will be considered for release
- What should be specified:
  - Which tests must have been performed
  - Which tests must have passed
  - How many iterations of the testing cycle need to be clean

28

Other Steps in the Process

- Build tests
- Execute tests and verify results
- Verify test and functional coverage
- Track defects
- Manage repositories



Summary

- Gather and define requirements
- Analyze them to eliminate ambiguities, conflicts and other errors
- Manage requirements and their evolution throughout the development cycle
- Design and build test cases based upon the requirements
- Review test cases with spec writer, user/domain experts, developers

Summary

- Define test completion criteria
- Execute tests and verify results
- Verify test coverage
- Track defects
- Manage the requirements, test, code and defect repositories

31

Questions?

QWE2002 Paper 9I

Mr. 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

Robert 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.

Vulnerabilities and Developing for the Net

Robert A. Martin
The MITRE Corporation

14 March 2002

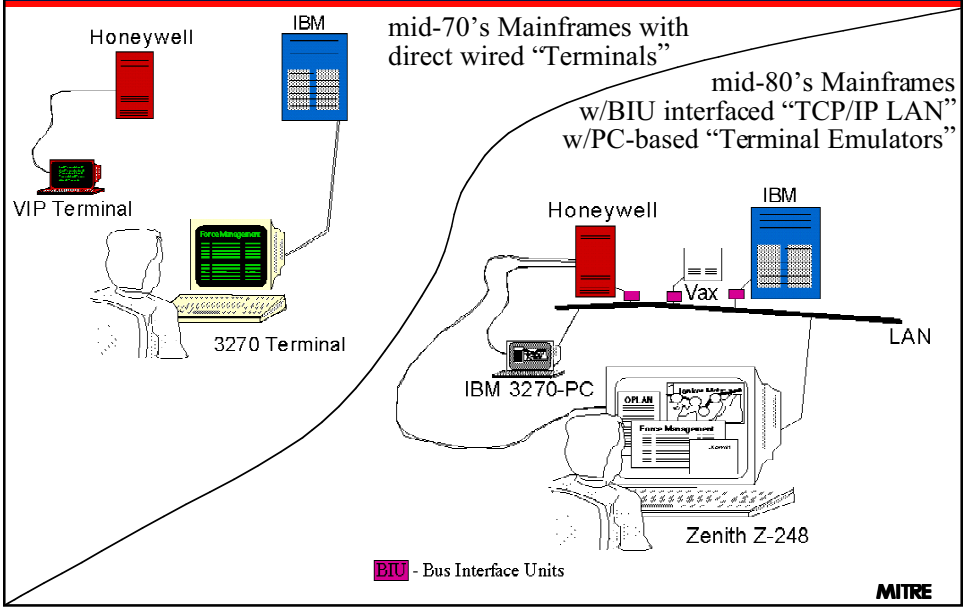
The views expressed in this presentation are those of the authors and do not necessarily reflect the policies or position of The MITRE Corporation.
Editorial graphics © 1996, 1997, 1998, 1999, 2000, 2001 Martin, used with permission. © 2001 The MITRE Corporation

MITRE

Outline

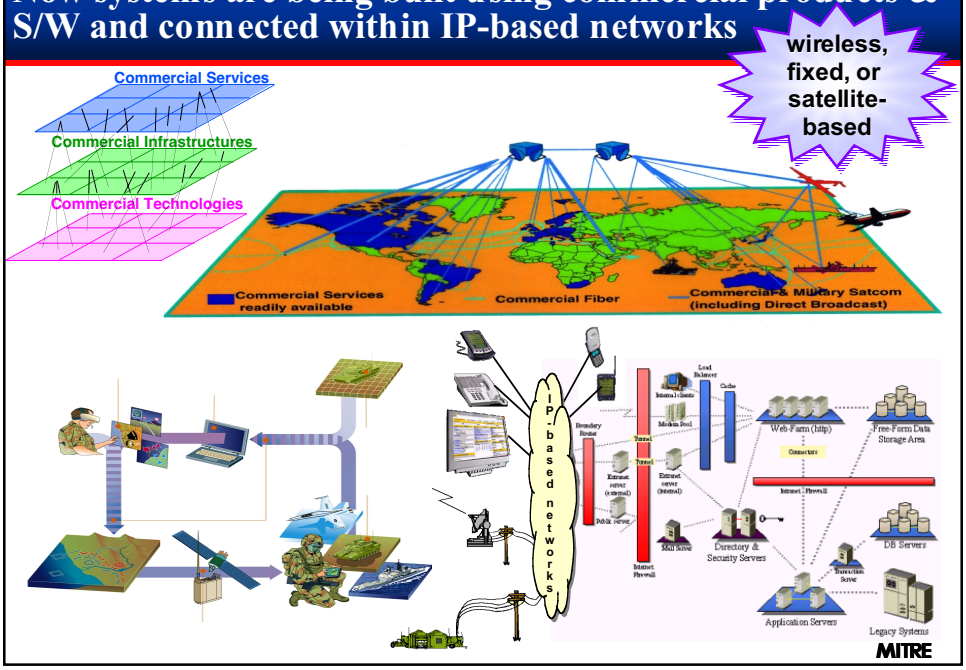
- **Background and Motivation**
- **Finding Out About Vulnerabilities**
- **The Problem and a Solution - CVE**
- **CVE Compatibility**
- **The CVE Process**
- **Summary**

DoD started w/stand-alone computers, terminals & custom S/W -- Then came PCs w/COTS S/W terminal emulators and TCP/IP LANs



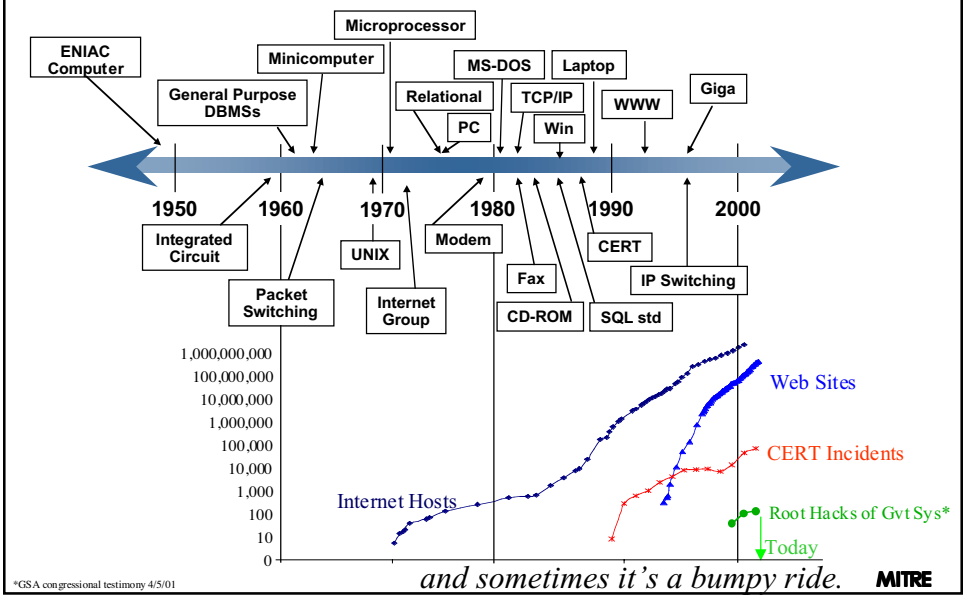
MITRE

Now systems are being built using commercial products & S/W and connected within IP-based networks



MITRE

The explosive growth of commercial technology and the Internet has taken us all for a ride…



*GSA congressional testimony 4/5/01

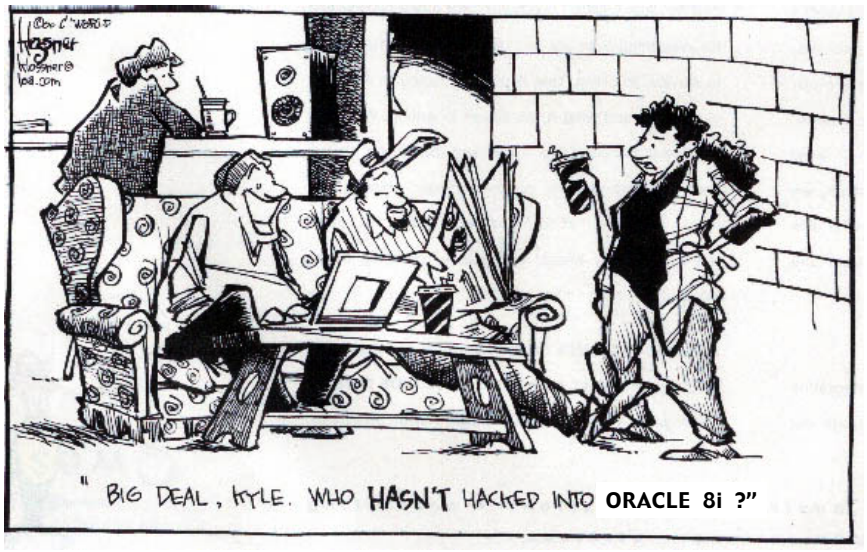
and sometimes it's a bumpy ride.

MITRE

Organization's Internet-visible "Faces" are being abused through Vulnerabilities in Commercial S/W

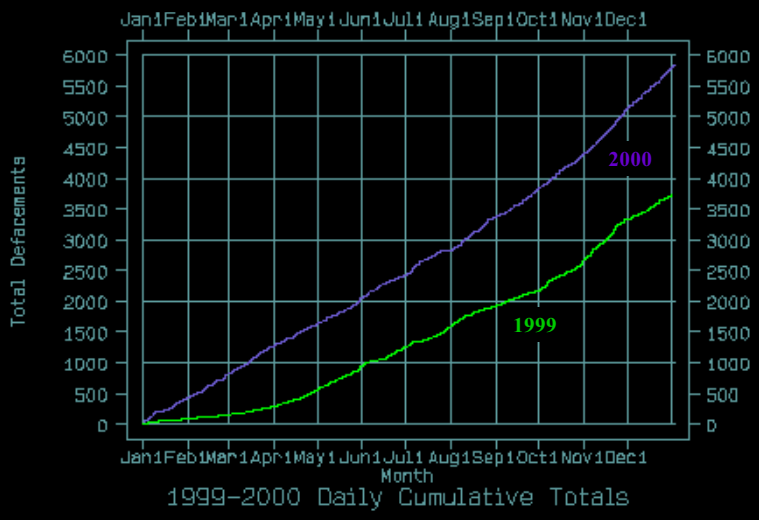
MITRE

The Wrong Publicity Can Be Bad...



MITRE

1999 vs. 2000 Daily Hack Cumulative Total Comparison



Source: Attrition.org

MITRE

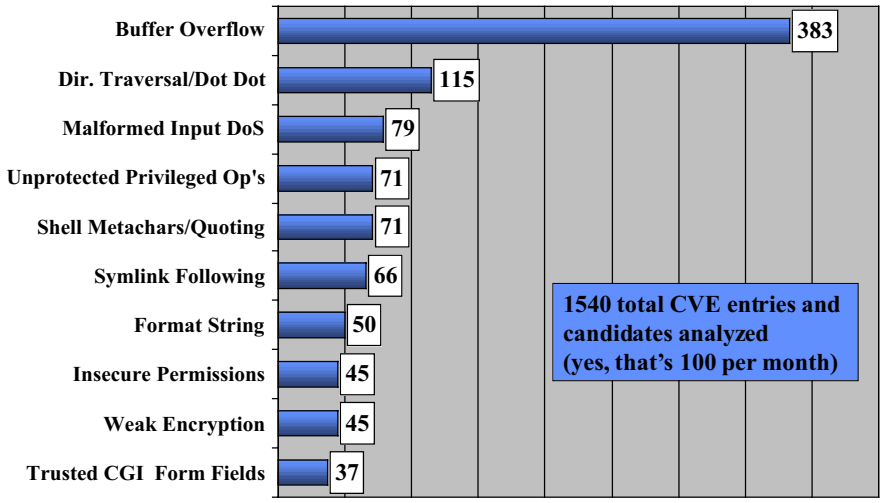
Software problems with security implications are referred to as Vulnerabilities or Exposures

- 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

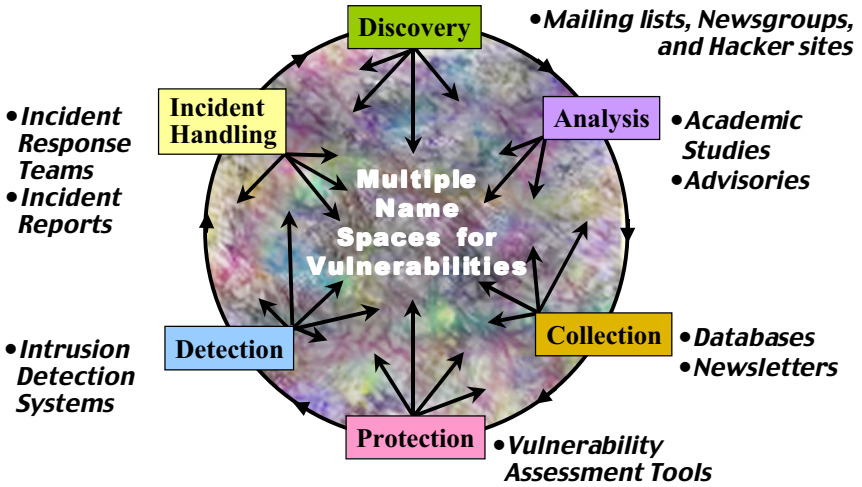
So how do you find out about commercial software vulnerabilities if the vendors aren't going to tell you?

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
Fyodor's Playhouse Online Vulnerability Database	hacker web site database	Insecure.Org
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
Cassandra	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
Nessus	freeware scanner	Renaud Deraison & Jordan Hrycaj
Prowler	IDS	AXENT Technologies
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.

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

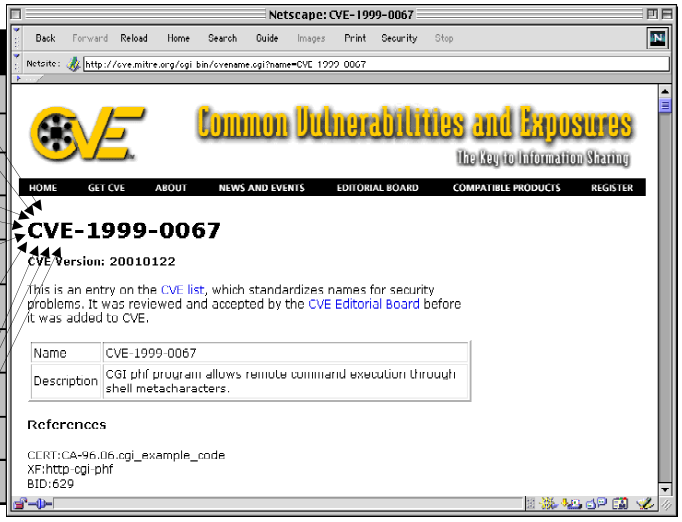
Outline

- 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

Organization
CERT
CyberSafe
ISS
AXENT
Bugtraq
BindView
Cisco
IBM ERS
CERIAS
NAI

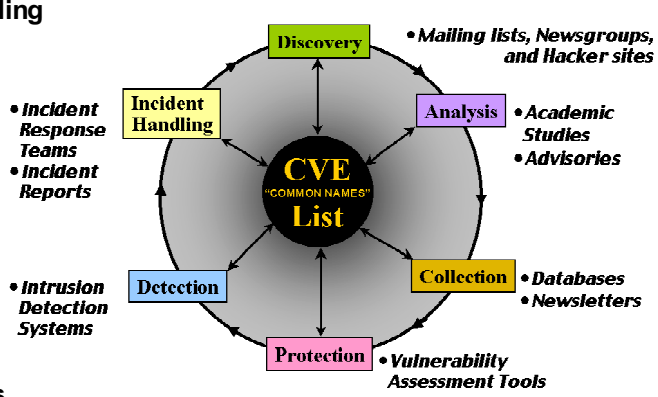


Along with the new rule, "Whoever finds it, gets a CVE name for it"

MITRE

The Vulnerability Information Sharing Process with CVE - - *"Whoever finds it, gets a CVE name for it"*

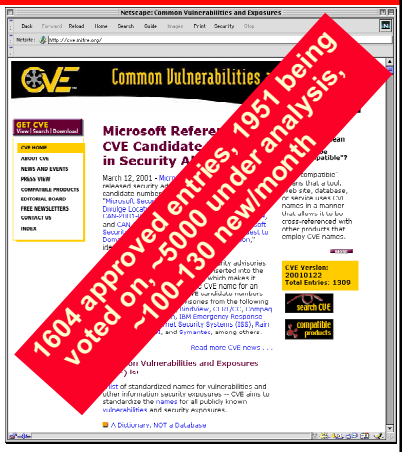
- Assigning a unique identifier to each problem
- Remaining independent of any particular perspective
  - Not just a developer's, researcher's, tester's, or analyst's view
- A community-wide effort via:
  - the CVE Editorial Board, the CVE Advisory Council, and the organizations adding CVE names into their tools, databases, web sites, & services
- Publicly open and shared
  - Will eventually list all publicly known security problems



MITRE

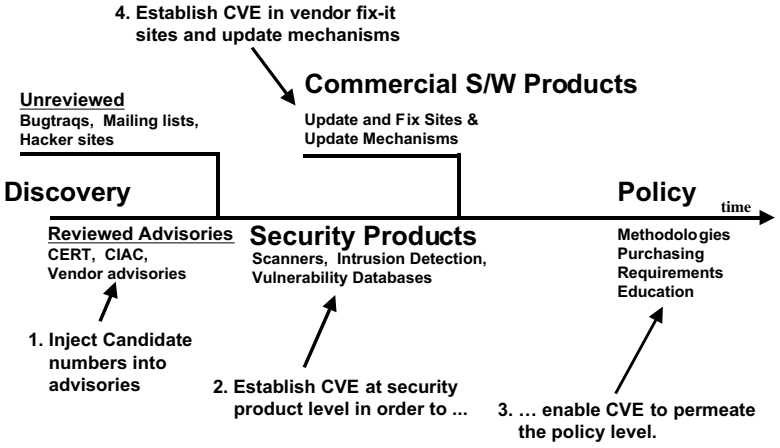
The Common Vulnerabilities and Exposures (CVE) Initiative

- 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]



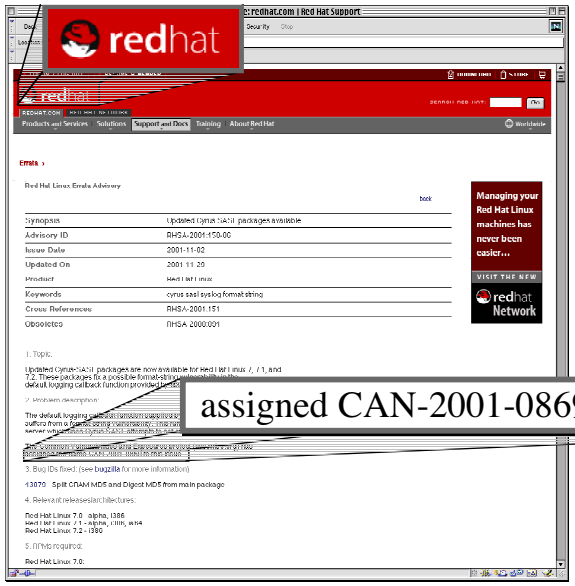
MITRE

The CVE Strategy



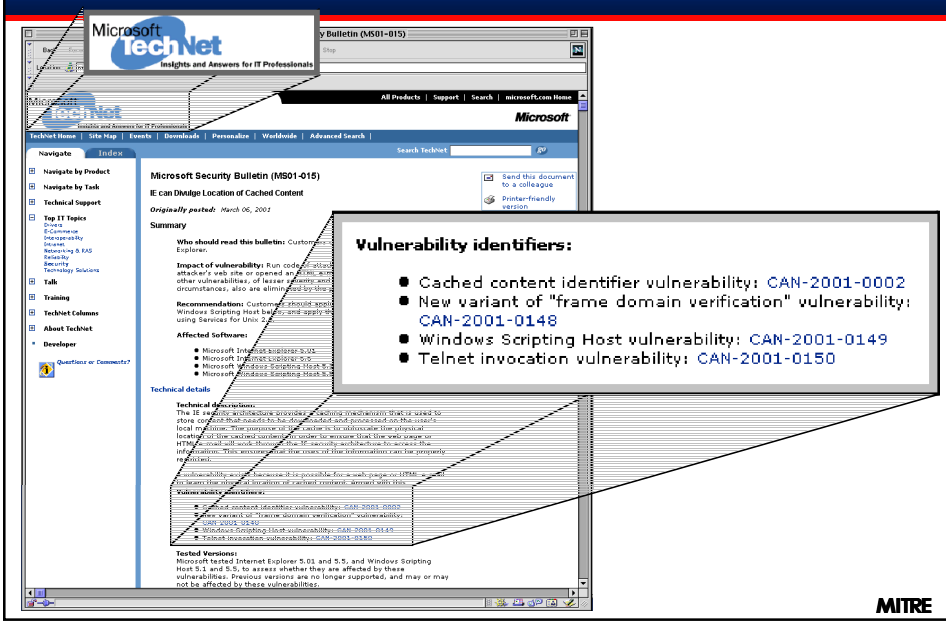
MITRE

The latest organization to start using CVE names in their alerts and advisories



MITRE

The company with the most alerts using CVE names



MITRE

Outline

- Background and Motivation
- Finding Out About Vulnerabilities
- The Problem and a Solution - CVE
- CVE Compatibility
- The CVE Process
- Summary

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

Organizations With Products That Use CVE
(as of 14 December 2001)

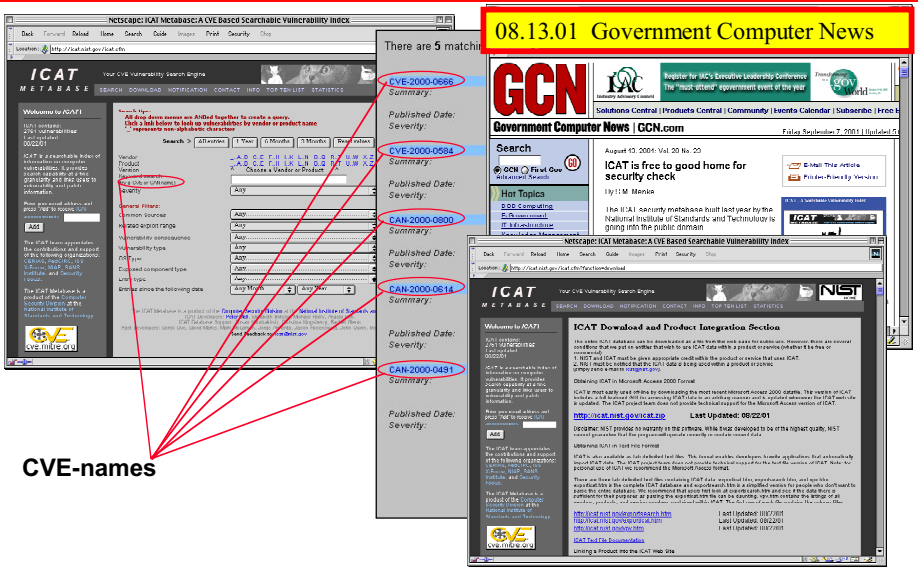
- **These (39) organizations have publicly declared that they are working on (65) CVE-compatible tools, databases, web sites, or services**

Advanced Research Corp
Alliance Qualité Logiciel
BindView Development
CERIAS/Purdue University
CERT Coordination Center
Cisco Systems
CS&S *(China National Computer Software & Technology Service Corporation)*
CyberSafe
CYRANO
UC Davis
Enterasys Networks *(bought Network Security Wizards)*
Entercept Security Technologies
Ernst & Young
Foundstone, Inc.
Harris Corporation
Intranode
Intrusion.com
Internet Security Systems
LURHQ Corporation
Max Vision Network Security/Whitehats
nCircle *(formerly Hiverworld)*
The Nessus Project
Network Security Systems
NIST
nSecure Software (P) LTD.
NTBugtraq
Penta Security Systems
PGP Security, NAI
Qualys
SANS
Security Focus, Inc.
SecurityWatch
spiDYNAMICS
Symantec
Tiger Testing
Tivoli Systems Inc.
Tsinghua UnisNet Technology, Ltd.
Venus Information Technology Inc.
World Wide Digital Security

Up-to-date list at http://cve.mitre.org/compatible

MITRE

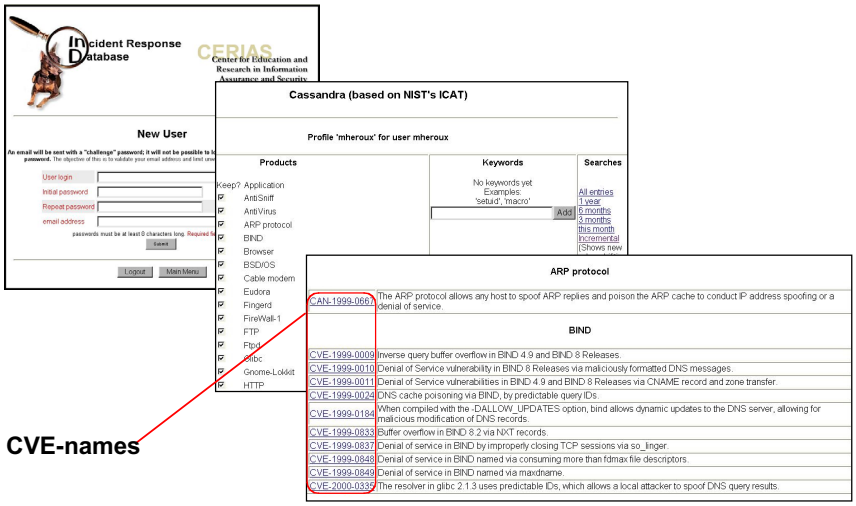
Examples of CVE-compatible items:
The ICAT Metabase



http://icat.nist.gov

MITRE

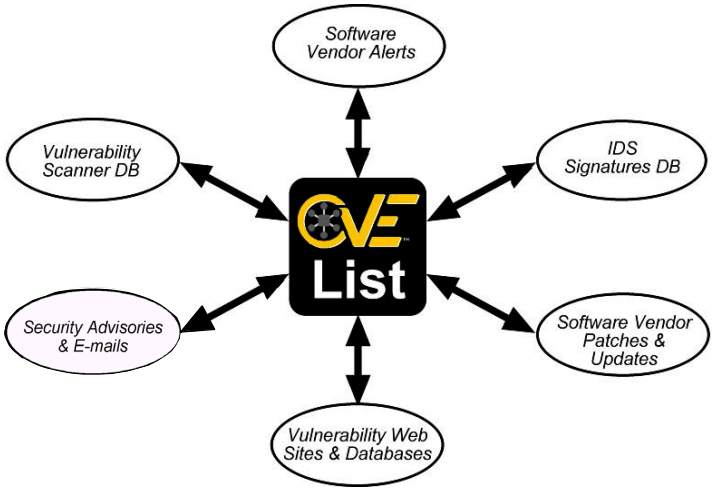
Examples continued: *Cassandra*



https://cassandra.cerias.purdue.edu

MITRE

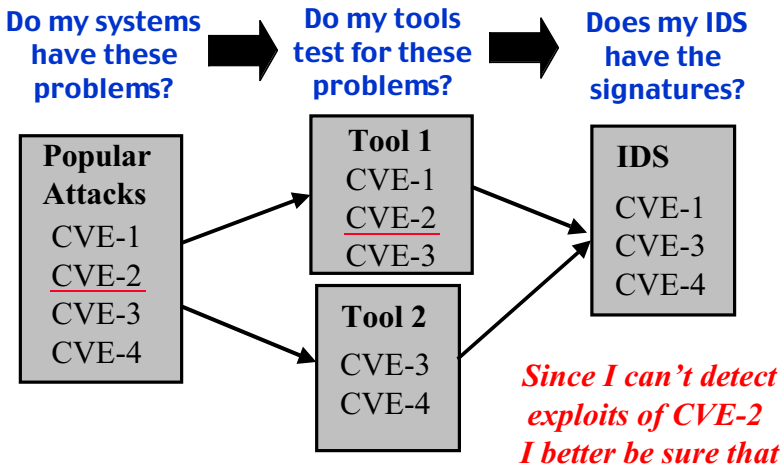
CVE compatibility provides a path for integrating information on Vulnerabilities and Exposures



CVE compatibility means that a tool or database can "speak CVE" and correlate data with other CVE-compatible products.

MITRE

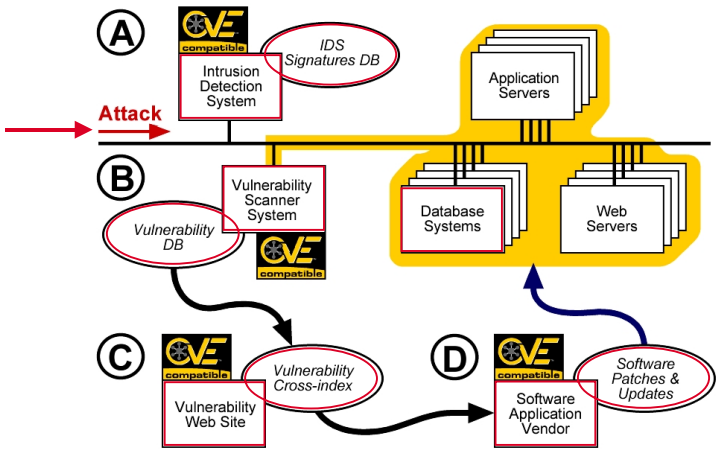
Example using CVE compatibility to go from Advisories to Vulnerability Scanners to IDSes



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



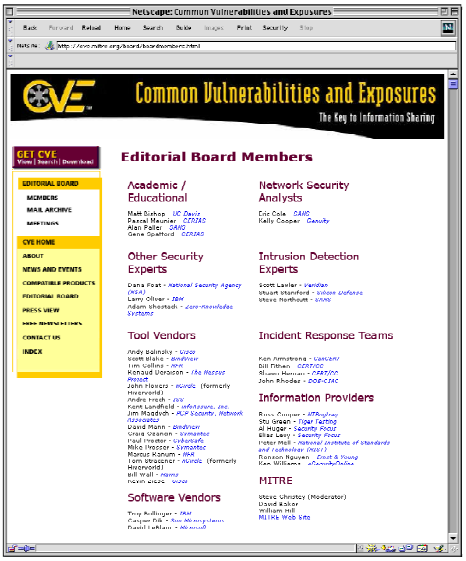
MITRE

Outline

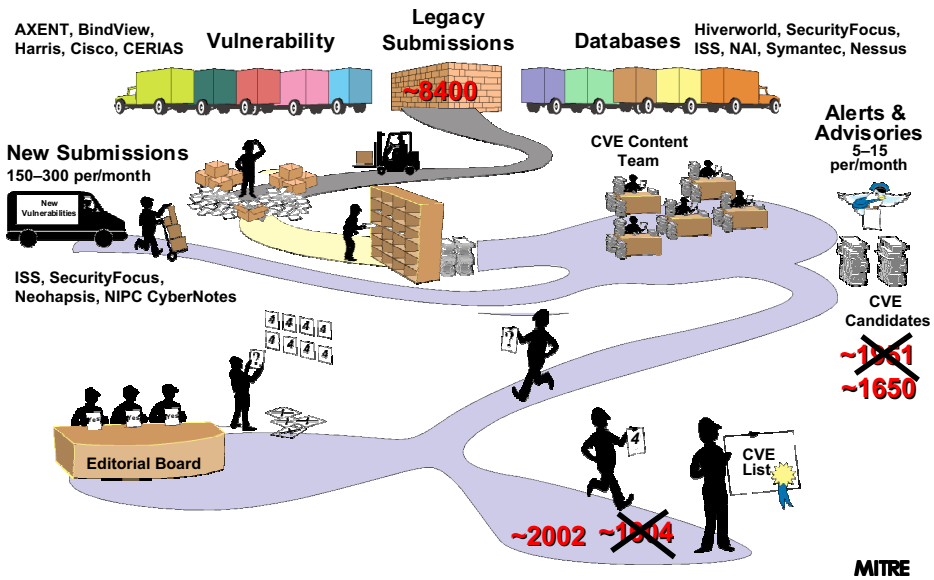
- Background and Motivation
- Finding Out About Vulnerabilities
- The Problem and a Solution - CVE
- CVE Compatibility
- The CVE Process
- Summary

CVE Editorial Board

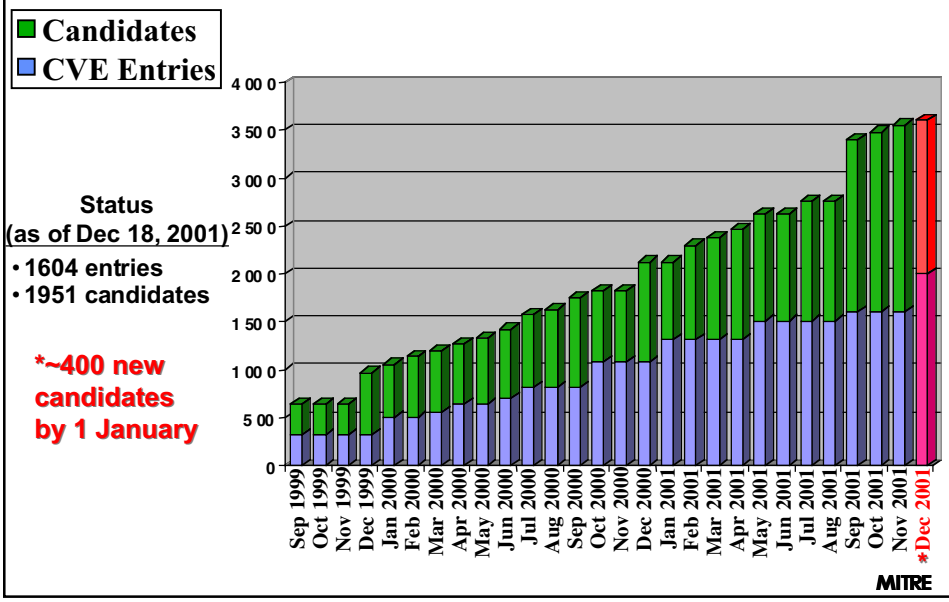
- Includes mostly technical representatives from 32 different organizations including researchers, tool vendors, response teams, and end users
- Reviews and approves CVE entries
- Discusses issues related to CVE maintenance
- Holds monthly meetings (face-to-face or phone)
- Maintains publicly viewable mailing list archives
  [cve.mitre.org/board/archives]



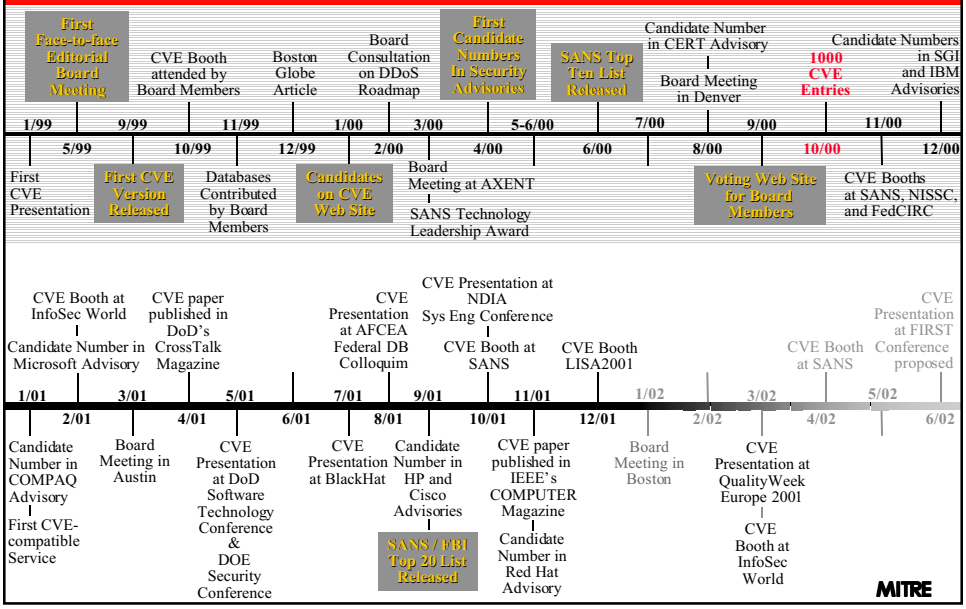
Where the CVE List comes from



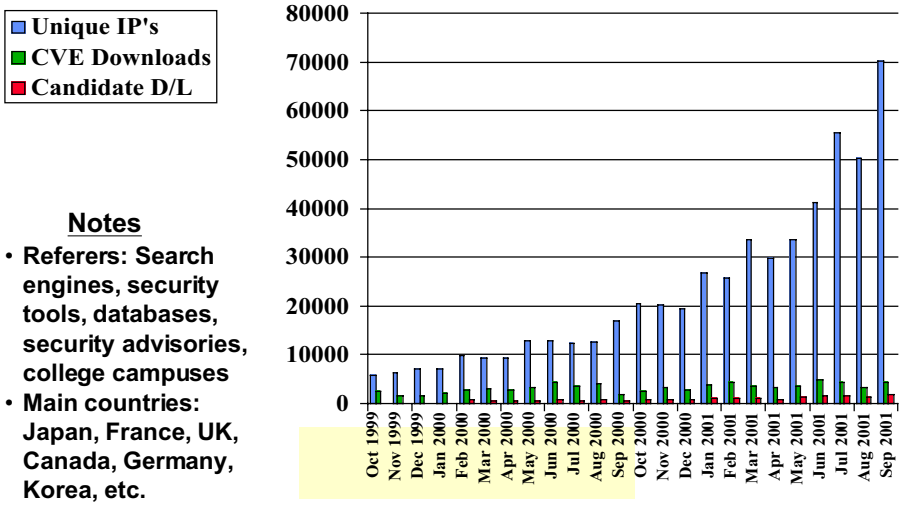
CVE Growth



Major CVE Milestones



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

Outline

- Background and Motivation
- Finding Out About Vulnerabilities
- The Problem and a Solution - CVE
- CVE Compatibility
- The CVE Process
- Summary

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(IIS 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)

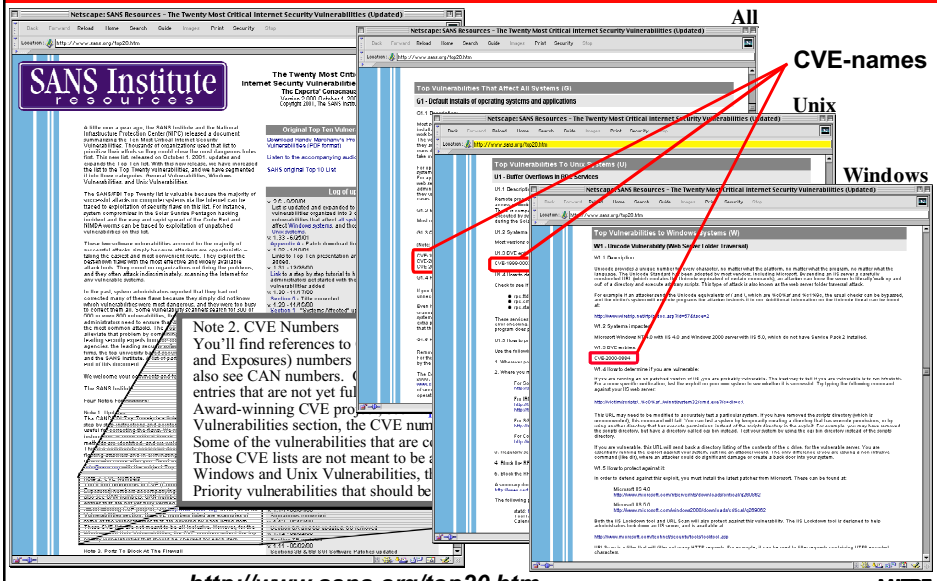
(see http://www.microsoft.com/technet/security/bulletin/ms99-013.asp)
Remedy:
Apply patch at : ftp://ftp.microsoft.com/bussys/iis/iis-public/fixes/usa/Viewcode-fix/

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

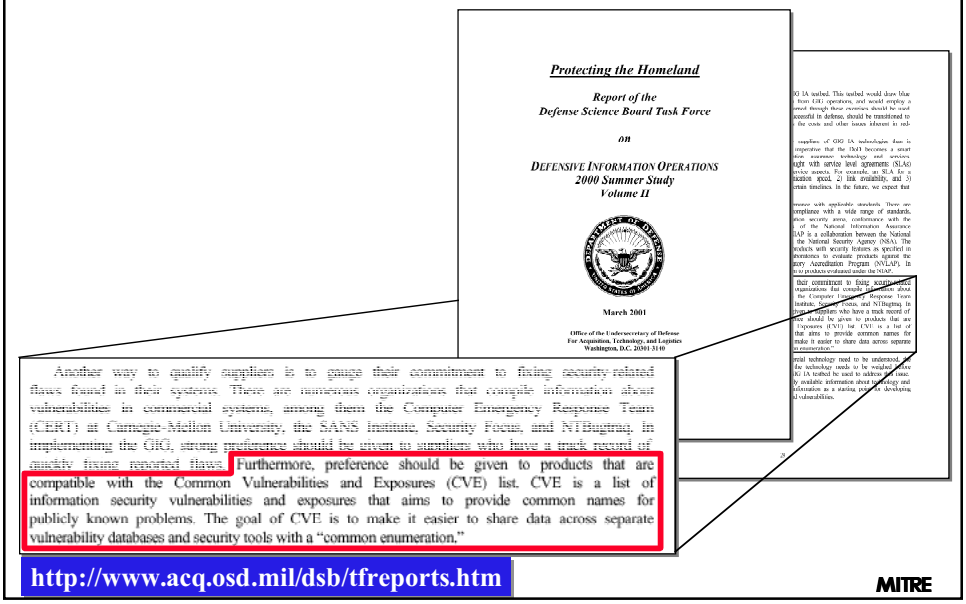
FBI/SANS Institute 2001 Top Twenty uses CVE names …yet another step down the policy road



http://www.sans.org/top20.htm

MITRE

Defense Science Board Report on Defensive Information Operations calls for CVE-compatible Products



Protecting the Homeland

Report of the
Defense Science Board Task Force

on

DEFENSIVE INFORMATION OPERATIONS
2000 Summer Study
Volume II

March 2001

Office of the Undersecretary of Defense
For Acquisition, Technology, and Logistics
Washington, D.C. 20301-3140

Another way to qualify suppliers is to gauge their commitment to fixing security related flaws found in their systems. There are numerous organizations that compile information about vulnerabilities in commercial systems, among them the Computer Emergency Response Team (CERT) at Carnegie Mellon University, the SANS Institute, Security Focus, and NTBugtraq. In implementing the GIG, strong preferences should be given to suppliers who have a track record of quickly fixing reported flaws. Furthermore, preference should be given to products that are compatible with the Common Vulnerabilities and Exposures (CVE) list. CVE is a list of information security vulnerabilities and exposures that aims to provide common names for publicly known problems. The goal of CVE is to make it easier to share data across separate vulnerability databases and security tools with a "common enumeration."

http://www.acq.osd.mil/dsb/tfreports.htm

MITRE

CVE Has Become Part of Product Comparisons…a step down the road of policy...

Vulnerability Scanner Features

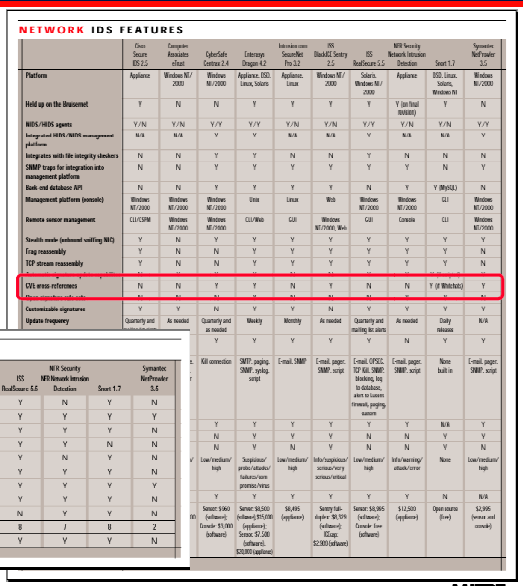
	Axent Technologies NetRecon 3.0 + SU7	BindView HackerShield	eEye Digital Security Retina	Internet Security Systems Internet Scanner	Nessus Security Scanner	Network Associates CyberCop Scanner	SARA	World Wide Digital Security SAINT
Price	Starts at $1,995	$19.95 per IP scanned	Starts at $1,145	Starts at $2,795	Free	$32 per node, $2,232 server	Free	Free (report generator starts at $100)
Platform	Windows NT	Windows NT	Windows NT	Windows NT Workstation	Unix	Windows NT	Unix	Unix
Built-in automatic signature update feature	● (download from Web)	●	●	●	● (download from Web)	●	○	○
Scans for host vulnerabilities	○	●	●	●	○	●	○	○
CVE cross-references	○	●	○	●	●	○	●	●
Automatic fixing of select vulnerabilities	○	●	●	○	○	●	○	○
Open source	○	○	○	○	●	○	●	●
Command-line automation	○	○	○	●	●	●	●	●
Integrates with a data-management suite	● (Enterprise Security Manager)	○	○	● (ISS SafeSuite)	○	● (Security Management Interface)	○	○
Capable of custom security checks	○	○	○	○	● (NASL)	● (CASL)	●	●

● Yes ○ No

Network Computing Article "Vulnerability Assessment Scanners" (1/8/2001)

MITRE

CVE Enables Detailed Product Comparisons



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 Ethernet	Y	N	N	Y	Y	Y	Y	Y (on Intel NMAWS)	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 NIDS/HIDS 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	GUI	Windows NT/2000
Remote sensor management	CLI/CSPM	Windows NT/9000	Windows NT/9000	CLI/Web	GUI	Windows NT/9000, Web	GUI	Console	CLI	Windows NT/9000
Stealth mode (inbound sniffing NIC)	Y	N	Y	Y	Y	Y	Y	Y	Y	Y
Frag reassembly	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-references	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 as needed	As needed	Quarterly and as needed	Weekly	Monthly	As needed	Quarterly and mailing list alerts	As needed	Daily releases	N/A

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
RDS	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 5 ISAPI	CAN-2001-0241	11	Y	Y	N	N	Y	Y	Y	N
Total (out of 9)			6	9	2	5	8	7	8	2
Detect attacks fragmented (Frag T8)			Y	Y	Y	Y	Y	Y	Y	N

Y = YES N = NO

Tables from Network Computing Article "To Catch a THIEF" (8/20/2001)

MITRE

The CVE Strategy: Where are we? (as of 14 December 2001)

4. Establish CVE in vendor fix-it sites and update mechanisms

• Adding CVE names broached with 13 groups.

Commercial S/W Products

Update and Fix Sites & Update Mechanisms

Unreviewed
Bugtraqs, Mailing lists, Hacker sites

Discovery

Policy

time

Reviewed Advisories
CERT, CIAC, Vendor advisories

Security Products
Scanners, Intrusion Detection, Vulnerability Databases

Methodologies
Purchasing
Requirements
Education

1. Inject CVE Names into advisories

2. Establish CVE at security product level in order to ...

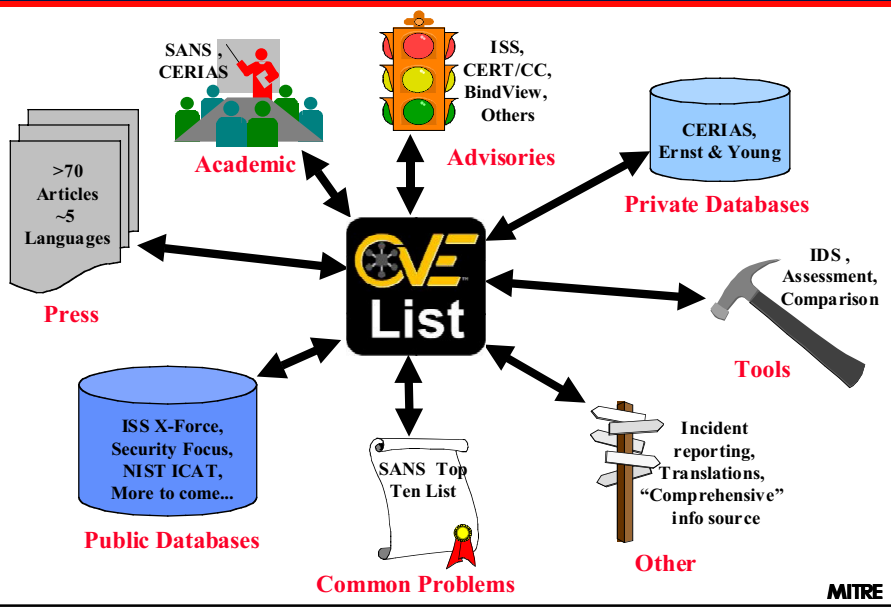
3. ... enable CVE to permeate the policy level.

So far, advisories from ISS X-Force, Rain Forest Puppy, BindView, Compaq, SGI, IBM, CERT/CC, Microsoft, HP, CISCO, and Red Hat have included CVE names.

- 2002 CVE Entries -- 1650 Candidates.
- 64 CVE-compatible products from 39 groups.
- 19 more from 14 others in "the works".

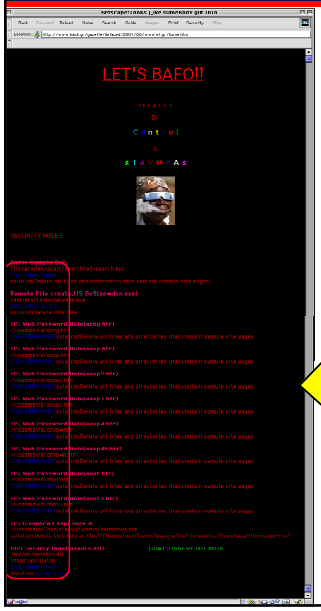
- SANS / FBI Top 20 uses CVE names
- Network Computing IDS & Scanner Comparisons included CVE
- DSB Report calls for CVE compatibility
- Network World IDS Comparison included CVE coverage

CVE is the center of many activities and efforts… …and it's still growing



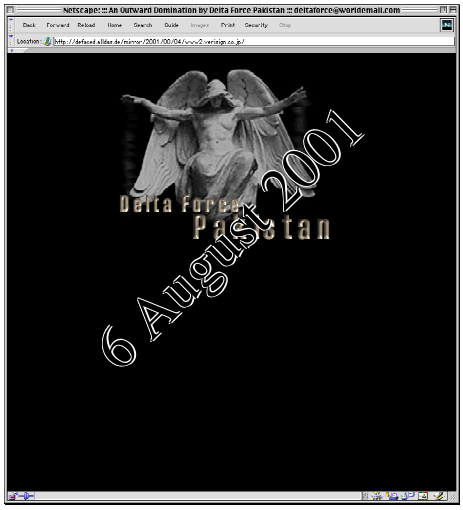
MITRE

CVE is even getting used by Hackers !



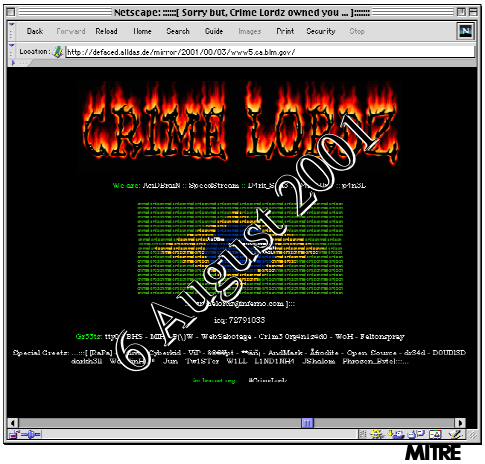
At least two hackers are now suppling CVE names for the vulnerabilities that they find in the sites they hack into.

And Yes, In Case You Wondered… …the Hacking Continues



VeriSign Inc. Japanese Site
http://www.verisign.co.jp

Bureau of Land Management in California
http://www.ca.blm.gov



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.

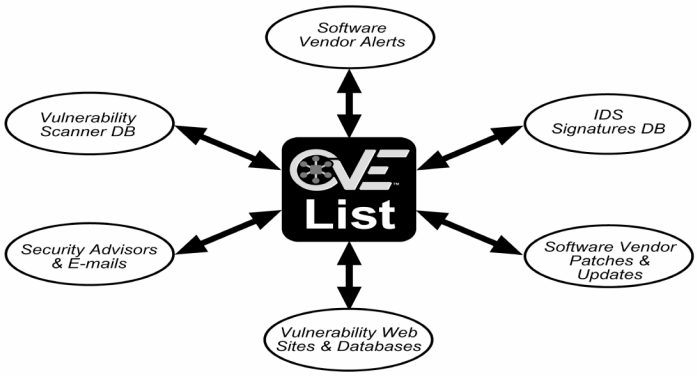


Figure 1: *CVE List as a Bridge*

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.

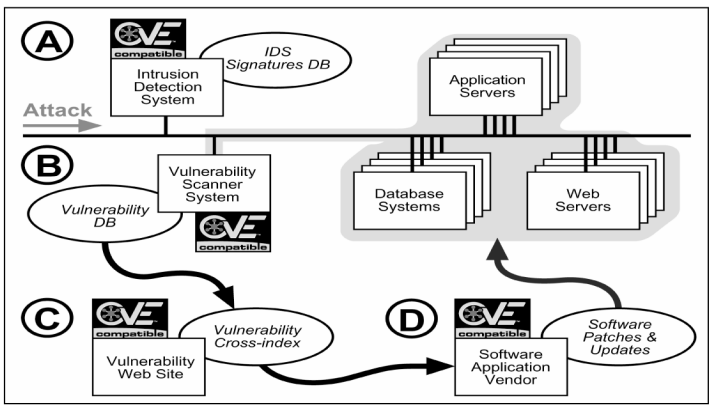
Improving the Process

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].

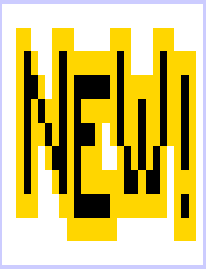
Did this article pique your interest?

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



QWE2002 Paper 10I

Mr. Simon J. Hardiman
(SQS S.A., Spain)

Multifaceted Internet Application Quality Validation Methodology

Key Points

- The facets of the SQ-MET Methodology.
- The quality testing capabilities of the SQ-MET Methodology.
- The SQ-MET Methodology in practice.

Presentation Abstract

The SQ-MET Methodology was developed to provide Internet developers and companies with a framework with which they can comprehensively test the quality of their internet applications. The methodology was formulated using a rigourous process of investigation, development and validation. Each stage of the methodology formulation process had to take into account and incorporate the diversity of the Internet world, in terms of the different application types, user types, etc..

Briefly, this presentation will focus on the SQ-MET development methods, the capabilities that the methodology offers those involved in Internet application development, the real advantages that companies and individuals can realise through the use of this methodology, and finally, the conclusions that can be drawn from this experience.

About the Author

Simon Hardiman

```
Born: 25th November 1971 in London

Education:

Classical Civilisation Graduate, Leeds University, England

Professional Experience:

Technology Consultant, working for SQS S.A., Spain
          -National and International R&D Project
           Management and Co-ordination
         -National and International R&D Project Development

Analyst / Programmer, working for Lloyds TSB Bank plc., England
         -Analyst and Programmer for the Lloyds and TSB CHAPS
         Sterling Environments
         (Critical 'A' Inter-Bank Payment Systems)

Configuration Manager, working for Lloyds TSB Bank plc., England
         -Configuration Manager for 4 Information Systems projects.
```

QWE Conference - Brussels
March , 2002

SQ-MET Methodology

"Multifaceted Internet Application Quality Validation Methodology"

Presented by:
Simon Hardiman
(SQS Consultant)

Introduction

Background

- Explosion of Internet use and development
- Assumption of ever more critical and complex functionality
- Very short development life-cycles - *quality can be an easy victim!*
- Therefore… strong emphasis must be placed on the importance of functional and performance quality, as well as the accuracy and quality of testing processes, is essential.

SQ-MET Objectives

- **High level objective**
  - *to develop & validate a methodology to enable Internet developers & companies to systematically validate the quality of their applications*
- To develop & validate a methodology to measure the quality of all *Internet application types and their characteristics*, regarding their respective requirements
- To produce a comprehensive *set of theoretical requirements that are easily applicable to, or modifiable for, any given application type*
- To produce *verifiable quality metrics* for the different types of Internet applications, and *procedures for their application and extraction*
- To produce a *highly applicable series of quality assurance test cases for Internet applications*

SQ-MET Approach

SQ-MET Methodology is a response to a perceived lack of quality assurance in the realm of Internet applications.

- SQ-MET Methodology is characterised by its *pragmatic approach and obtainable objectives*
- The development of the methodology involved *specialist companies with a vested interest* in the success of Internet application quality assurance
- *Rounded scope of the methodology*, covering application genres and their related sub-categories
- Special attention was given to building a comprehensive and readily *applicable sets of metrics by application and attribute type*.

Theoretical Requirements for Internet Applications

Threefold approach:

- Use of own company experience
- In-depth research of papers, tutorials & style guides on Internet application design & development
- Important & representative information sources on Internet application requirements were identified & interviewed, under the following headings:
  - application users
  - application developers
  - application owners

Result

- Extraction of well-rounded theoretical requirements for Internet applications.

Internet Application Genres

Based on that threefold approach, the requirements are defined and categorised per Internet application genre or type:

- **Informative**
- **Search Engine or Facility**
- **Forum**
- **Chat**
- **Job Marketing**
- **Download Facility**
- ***e*Commerce**
- ***e*Business**

Tangible Application Parameter Types

- **Certification** - the SQ-MET Methodology groups the tangible parameter types of Internet applications as follows:
  - **Functionality**
  - **Performance**
  - **Reliability**
  - **Maintainability**
  - **Usability**
  - **Security**

Metrics & Measurements

- The SQ-MET Methodology development involved in-depth research and analysis, resulting in *the establishment of a set of quality characteristics or attributes and their respective sub-characteristics*, regarding the different Internet application types
- These characteristics laid the foundation for the research into, and production of, a set of quality metrics for Internet application types
- Classification of the SQ-MET metrics are based upon the quality definition model of the **ISO/IEC 9126 (ISO92) standard**

Metrics & Measurements

- Research and analysis performed by a leading university in Internet application technologies
- Development of metrics were based upon the structure of the *six general quality characteristics of the ISO 9126 standard*:

  - **Functionality**
  - **Performance**
  - **Reliability**
  - **Maintainability**
  - **Usability**
  - **Mobility**

Study & Analysis of Internet Application Testing Tools

Reasons behind this task:

- Great number of Internet application testing tools available on the market - both as individual tools or as part of a package
- Each tool offers its own particular suite of services with varying ranges of quality
- Validation / coverage of many of the metrics and requirements is impossible without the use of automated testing tools
- Large volume of information available ➔ lack of understanding & need for information assimilation ➔ *how should SQS complement the SQ-MET Methodology with right tools for a given application type?*

➨ **Intention** - to filter and analyse what is available and make it more palatable, and facilitate companies with easier access to Internet application quality assurance.

Overview of Study & Analysis of Internet Testing Tools

- **Preliminary Evaluations:** over 100 available Web Testing Tools evaluated
  - **Result:** documented listings and tables of those tools were compiled, detailing the functional capability, quality, cost, compatibility, etc… of each tool
- **Secondary Evaluations:** each tool compared and contrasted with each of the defined SQ-MET quality requirements and metrics
  - **Result 1:** the requirements and metrics that each tool was able to satisfy or accomplish were recorded
  - **Result 2:** the *top five tools were selected* according to their ability to satisfy the highest quantity of Internet application requirements and metrics, as defined by the SQ-MET Methodology.

SQ-MET Methodology & Internet Testing Tools

Automated Testing v Requirements Coverage

- Post-evaluation analysis of the Web Testing Tools reveals that *no single tool available is able to satisfy more than 20% of the theoretical requirements defined*
- The *most effective method* for satisfying the majority of requirements *is to combine a number of individual tools*, according to company needs and priorities
- Tool combinations can provide approximately 70 - 75% of requirements coverage
- The remaining 25 - 30% can be covered manually,
  - e.g. through visual analysis

Validation of Theoretical Requirements

- **Objective:** to gauge the degree to which Internet Testing Tools could satisfy the theoretical requirements defined, and their performance with real Internet applications
- **Step 1:** for each of the top five Web Testing Tools selected, the respective tool manufacturer was invited to provide SQS with a detailed demo over a 1 - 2 day period
- **Step 2:** SQS then performed their own in-depth evaluation of each tool, both with the requirements and, with agreement, a number of web applications
  - **Result 1:** documented records of each tool's performance
  - **Result 2:** evaluation results were channelled into:
    - the definition & refinement of the Methodology Test Cases
    - the definition of procedures for validating the metrics

Defining & Refining Test Cases

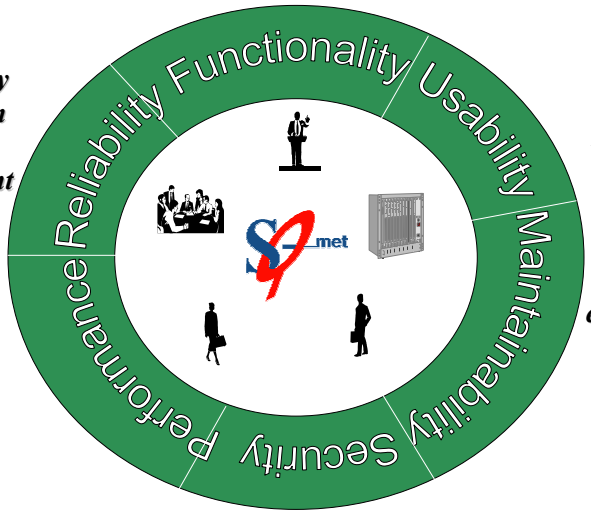
- SQS defined a series of test cases and test case groups for each specific requirement and requirement category
- SQS utilised their own in-house test case automation tool, **Test WorkFlow**, to successfully generate structured, highly classified and cross-referenced test cases
- Test cases were then distributed to, and reviewed by, each company contributing to the development of the SQ-MET Methodology
- Test cases were refined as applicable upon review, then re-submitted for review, only acquiring the status of final version upon agreement by all companies involved

Procedures for the Validation of Metrics

- Metrics are a pivotal ingredient of an accurate measurement of objectives versus success, however, one must know how to apply them properly and effectively
- With the metrics defined, the Methodology's development focused on the *identification and establishment of forms and formulas through which the metrics could be verifiably tested*
- From an automation point of view, the results of the tool evaluations proved to be very useful.

Application of the SQ-MET Methodology

SQ-MET Methodology facilitates an accurate measurement of quality within in these six functional areas of Internet applications



SQ-MET Methodology applied and validated in external company environments

Application of the SQ-MET Methodology

- SQS applied and trialled the SQ-MET Methodology in a number of large and small companies involved in web application development, according to the application types defined
- In each case, the Methodology was adapted to the company type, and their respective requirements
- *Approximately 40% of the theoretical Internet application requirements defined by the Methodology were, with slight modifications, applicable to all trial companies*
- Additionally, *approximately 60% of those theoretical requirements proved to be reusable*.

Market Study & Analysis Methods

- **Purpose** - to assess the potential impact of, and the business attraction to, the developing product(s)
  - i.e. the entirety or elements of the SQ-MET Methodology
- **Methods:**
  - interviews with companies
  - feedback from questionnaires
- The Methodology was packaged and marketed on three levels:
  - basic
  - intermediate
  - advanced
- The products adapted to fit the market interest

Success Stories

- The SQ-MET Methodology is a relatively new product on the market, so success stories are just beginning to be made
- However, as an example of its success so far:
  - SQS have sold the use of the Methodology to a large application development company with lengthy testing processes
  - Methodology applied within an internal *e*Commerce development environment
  - Use of the methodology has allowed them to halve the duration of their testing processes
  - Reduction in time and effort equates to reduction of costs
- SQS have established a info/helpdesk, which is now being used to assess the success of the Methodology

Presentation Paper for the Quality Week Europe Conference 2001

Subject:

Multifaceted Internet Application Quality Validation Methodology

SQ-MET Methodology

Author:

Simon James HARDIMAN (SQS Consultant)

Contents

1. Introduction 2
2. Theoretical Requirements for Internet Applications 2
3. Metrical Values 5
4. Study and Analysis of Internet Application Testing Tools 8
5. Validation of Theoretical Requirements 8
6. Defining and Refining Test Cases 9
7. Procedures for the Validation of Metrics 9
8. SQ-MET Methodology Application 10
9. Market Study and Analysis Methods 10
10. Success Stories 11

1. Introduction

As the strongest potential, if not actual, information channel or medium that exists, Internet use and development has exploded in recent years and will continue to do so. Within the foreseeable future, few can doubt that the Internet will be the defining instrument of communication for the whole planet.

The present Internet is playing host to a rapidly growing number of businesses, and the services and functionalities that they offer are increasingly becoming more complex and critical. The critical nature of these services and functionalities, combined with their complexity, place paramount importance on the following aspects of Internet applications:

- Functional quality
- Performance quality
- Quality and accuracy of testing, in terms of metrics and approach.

Indeed, the need for quality assurance is even greater when one considers that, in relation to normal IT environments, the development lifecycles of internet applications are extremely short, and thus quality is an easy victim.

In light of the need to guarantee the quality of Internet applications, SQS began developing the SQ-MET Methodology. The principal objective was to develop and validate a methodology which would provide Internet developers and companies with a framework with which they could comprehensively test and analyse the quality of their internet applications.

The Internet world, by its very nature, is a vast and rapidly changing entity. Therefore, in order to achieve its overall objective, the SQ-MET Methodology had to focus upon, cover and incorporate the diversity of Internet application types.

2. Theoretical Requirements for Internet Applications

The approach to this task was threefold:

- SQS, together with the assistance of specialist partners, drew on their own experience of working with Internet applications:-

- SQS's previous experience in the field of Web Application Testing;
- Experience of the assisting partners, whose business areas are dedicated to the design and development of websites;

- the partners assisting the development work jointly researched numerous papers, tutorials and style guides on Internet application design and development;

- the partners identified and interviewed a selection of highly relevant information sources on Internet application requirements, based upon the following groups – application users, developers and owners – as the best means of extracting rounded theoretical requirements for the Methodology.

Upon the basis of that threefold approach, the SQ-MET partners proceeded to define the theoretical generic requirements for Internet applications by categorising the different types of those applications, under the following headings and descriptions:

- **Informative Websites** – authentication free, these sites serve to transmit content information to the user, without offering search, input or interaction facilities.

- **Websites with a Search Engine or Facility** – these sites offer users the ability to locate existing information as they wish, with the search facility arranging the information according to intelligent search criteria.

- **Forums** – this category defines a part or the entirety of a website in which a user can publish structured information, generally by subject, and then within that, questions and replies, whilst maintaining a link with the previous subject consultation.

- **Chats** – this category defines a part or the entirety of a website in which each user may converse publicly with other users in "chat rooms", or privately one on one, or one with various.

- **Job Marketing** – this category defines a part or the entirety of a website in which each user candidate is able to introduce their CV, professional preferences, as well as search profiles which will allow them to select offers that match their preferences.

- **Download Areas** – this category defines a part or the entirety of a website that facilitates the downloading of files or images, whether they be HTTP or FTP, from a structured directory (this can be done locally).

- ***e*Commerce** – this category defines a part or the entirety of a website that permits financial transactions, whether they are between company and client (B2C), or between companies (B2B). Available functionality permits the user to acquire, bid for, and consult product offers, and request and pay for them on-line.

- ***e*Business** - this category defines a part or the entirety of a website that facilitates the execution of business processes or procedures via the Internet. Access to possible transactions is subject to user authentication, and in many cases, teleworking, remote working from delegations, and the fulfillment of orders, are also catered for. This category can be associated or not with *e*Commerce.

- **Certification** – regarding the exigible parameters of each type of application, SQ-MET grouped the different types as follows:

  - **Functionality**: capacity of the website to perform the tasks it was designed to perform

  - **Reliability**: capacity of the website to function without errors

  - **Usability**: focuses on the ease with which one can use, understand and learn the system

  - **Efficiency**: capacity of the system to proportion the required return on investment with the level of resources used and any given conditions – i.e. functions are performed quickly, even in the face of multiple, simultaneous access.

  - **Security**: capacity of the system to prevent unauthorised access, whether accidental or deliberate. The aim of this study is not to prevent attacks on server taking place, but rather to ensure that the there are sufficient mechanisms in place to avoid the data being accessed.

The above framework was adopted for the compilation and definition of the theoretical requirements for Internet applications, because it was felt that grouping application and parameter types was the best way of managing such a diverse subject.

SQ-MET contends that the most essential features of any type of web application are functionality and usability. The stress we have placed on functionality lies in the fact that an application simply makes no sense if it doesn't do what it was designed to do. Equally, whether users are trying to find out the time of flights from Paris to Rio de Janeiro, to purchase a CD or simply read about their favourite films, if they cannot easily access and obtain that information then it is very likely that they will turn to a competitors website to locate what they are looking for.

3. Metrical Values

The objectives of this phase of the development work were:

- to research and compile a comprehensive set of quality metrics for Web applications

- to produce a classification of those metrics, using the quality definition model of the ISO/IEC 9126 [ISO92] standard as a reference point

The specification and evaluation of the quality of a product requires the identification and selection of relevant quality characteristics or attributes and their respective sub-characteristics, as applicable. Using this method, a measurement criteria is established through which quality can be assessed.

The ISO 9162 standard states six general characteristics, each of which are themselves broken down into further sub-characteristics, and it is those sub-characteristics which possess attributes that can be measured in a verifiable manner.

This development stage was led by a leading university in Internet technologies. They employed this framework of quality characteristics as a guideline for the accomplishment of this task. The structure of the quality characteristics / sub-characteristics framework is:

- **Functionality** – the capacity of the system to provide functions that cover concrete requirements under specified conditions

- Adjustments: the degree to which the available functions adjust themselves to required tasks
- Accuracy: functions produce predicted / adjusted results or effects
- Interoperability: capacity to interact and integrate with other systems
- Conformity: conformity with rules, laws and standards
- Access security: ability to prevent unauthorised access, accidental or deliberate

- **Reliability** – capacity of a system to deliver return on investment, under given conditions and over a specified period of time
  - Maturity level: attributes related with the frequency of system failures
  - Fault tolerance: ability to maintain a performance / return on investment in the face of internal errors
  - Recoverable: capacity to recover performance / return on investment in the case of faults, as well as time and effort required to do it

- **Usability** – capacity of the system to be understood, learnt and used by the user, and which will attract the user, under certain use conditions
  - Understandable: the degree of effort required for the user to understand the structure, logic and applicability of the system
  - Ability to learn: dependent on the level of interaction, the degree to which the system adjusts itself to the intuitive expectations of the user
  - Operability: the ease with which the system can be operated and controlled
  - Attractiveness: how appealing is the system to the user?

- **Efficiency** – capacity of the system to produce return on investment, based on the resources used, under imposed conditions
  - Behaviour over time: response time, process time and yield indices per task
  - Behaviour with resources: quantity of resources used and the length of time those resources were used to complete a task
- **Maintainability** – the extent to which the system can be modified, including fixes, improvements, and changes in the system, whether they be to do with the environment, requirements or functional specifications
  - Analysable: effort required to diagnose deficiencies, the cause of errors, or to identify components that have been modified
  - Modifiable: effort required for modifications, error fixing, or environmentally related changes
  - Stability: risk of unpredictable consequences as a result of change
  - Ease of testing: effort required to test and validate a modified system
- **Mobility** – capacity to transfer a system from one environment to another
  - Adaptability: multi-environment system adaptiveness without the need to carry out extra tasks other than those anticipated
  - Installation ease: effort required to install a system in a particular environment
  - Co-existence: capacity to co-exist with other systems in a common resource-sharing environment
  - Relocating: capacity and effort required to deploy and use the system in place of other specified systems and their environments.

4. Study and Analysis of Internet Application Testing Tools

With the theoretical requirements defined, as well as the metrics, through which the satisfaction of the requirements could be analysed, the SQS turned their attention to the next stage in the development of the SQ-MET Methodology – the study and analysis of the available Internet applicaton testing tools.

Their are a great number of Internet application testing tools currently on the market. Each of these tools offers its own particular suite of services to potential buyers and companies. In many cases, the testing facilities they offer are very similar (with varying ranges of quality), and many cases not.

Specific tools and tool providers can be found under particular headings such as “Functionality Testing Tools”, and whilst many may cover specific areas of functionality, the combination of testing facilities they offer tends to vary greatly. Furthermore, most of the testing tool suites that providers offer are broken down into individual tools that one may buy independently or as part of a suite. In short, the realm of Internet Testing Tools is somewhat of a minefield, however, these testing tools play a very important role in ensuring an application’s quality.

It was for these two reasons that the SQ-MET chose to perform a study and analysis of these tools in relation to application type and requirements. On the one hand, SQ-MET aimed to filter and analyse the information available and turn it into a more palatable entity, and on the other hand, SQS was aiming to complement the SQ-MET Methodology with clear and concise testing tool matches (per application types and requirements).

5. Validation of Theoretical Requirements

The validation of Internet application requirements, as defined during the development of the SQ-MET Methodology, was accomplished jointly between SQS and a company involved in security technologies. The latter company, based upon their company expertise, was soley focused on aspects of Internet application security, whilst SQS focused on the other areas of importance (functionality, usability, efficiency, maintainability, reliability, etc…).

The idea of this task was to measure in profundity the degree to which the selected Internet Testing Tools could address the requirements that had been defined.

For each of the top five Web Testing Tools selected, the respective tool manufacturer was invited to provide SQS personnel with a detailed demonstration of their tool and its capabilities, over a 1 – 2 day period. This initial tool training enabled SQS personnel to perform in-depth evaluations of those particular tools, through which they were able to properly guage each tool's potential, in relation to the development of the SQ-MET Methodology.

The subsequent results from the evaluations served two important functions:

- They provided an educated understanding of the capabilities of each tool to complement and automate elements of the developing methodology

- They provided very useful input into the production and modification of the Methodology's test cases and the production of the procedures for the validation of the metrics.

6. Defining and Refining Test Cases

The definition of the SQ-MET Test Cases was performed by SQS, using one of their in-house tools, called Test WorkFlow. Test WorkFlow enables the creation of structured and highly classified and cross-referenced test cases. The test cases were designed and produced to cover all of the requirements, both by application type and their respective characteristics.

Once formulated, the test cases and test case groupings were distributed to the Methodology development partners for review. Modifications were made, or approval given, as was applicable in each case. Where modifications were made, that test case or grouping was then re-submitted for another review, until approval was given.

7. Procedures for the Validation of Metrics

Metrics are a pivotal ingredient of an accurate measurement of objectives versus success, however, one must know how to apply them properly and effectively.

This consideration gave rise to the SQ-MET Methodology's focus on procedures through which the metrics defined could be systematically applied and through which readily digestable results could be extracted. This development task was divided into two parts:

- The identification of forms and formulas through which the metrics could be verifiably tested

- The testing and establishment of those procedures as an integral part of the Methodology test cases.

8. SQ-MET Methodology Application

SQS applied and trialled the SQ-MET Methodology in a number of large and small companies involved in web application development, according to the application types defined. This external application and trialling provided SQS with vital feedback on all aspects of the Methodology, such as the aptness of the requirements and metrics formulated and compiled, right the way through to the effectiveness of the test cases, test groupings and the procedures for validating the metrics.

In each trial, the SQ-MET Methodology was adapted according to the subjects' company type and their respective requirements, thus demonstrating the Methodology's flexibility.

This application and trialling exercise produced the following summary figures:

- Approximately 40% of the theoretical Internet application requirements defined by the Methodology were, with slight modifications, applicable to all trial companies

- Approximately 60% of those theoretical requirements proved to be reusable.

9. Market Study and Analysis Methods

The intention of the market study and analysis was to assess the potential impact of, and the business attraction to, the developing Methodology, both as a product or service in its entirety or as a group of individual products/services that collectively form an overall product/service.

With a wide selection of contact companies already compiled, the marketing exercise was performed in two ways:

- Direct interviews with those companies

- Feedback resulting from questionnaires and publications

Based upon the elements that make up the SQ-MET Methodology and the functional application areas that they cover, the Methodology was packaged and marketed on three levels:

- Basic
- Intermediate
- Advanced

The resultant feedback from this exercise led to the products being adapted to fit the market interest.

10. Success Stories

The SQ-MET Methodology is a relatively new product/service on the market, so success stories are just beginning to be made.

However, an example of its success so far can be given:

- SQS have sold the use of the Methodology to a large application development company with lengthy testing processes
- Methodology applied within an internal eCommerce development environment
- Use of the methodology has allowed them to halve the duration of their testing processes
- Reduction in time and effort equates to reduction of costs

Additionally, SQS have established a info/helpdesk, which aside from its obvious function, will enable SQS to assess the success of the Methodology,

makes improvements or changes as necessary. SQS also views this methodology as a stepping stone or catalyst for future advancements.

QWE2002 Paper 11I

Dr. Klaus Quibeldey-Cirkel
(TLC GmbH)

Checklist for Web Site Quality Assurance

Key Points

- QA checklist with more than 200 categorized criteria, implemented as a MS Excel tool.
- Checklist criteria are based on long-term studies about web quality standards.
- Offered free of charge on an Open Source policy, i.e. may be used in multi-authorship.

Presentation Abstract

The paper offers a Microsoft Excel based QA checklist of more than 200 criteria, categorized and equipped with automatic graphic analysis, for determining what results in a good experience for the web site customer. The checklist considers industry best practices, generally accepted web standards, and has successfully been applied in several e-business projects.

About the Author

At the time of paper submission, Dr. Klaus Quibeldey-Cirkel was working as a senior system architect at the software company TLC GmbH located in Frankfurt, Germany. He has more than 7 years of industrial experience as a hardware and software engineer. Beginning in September 2001, he will be a professor of Computer Science and a faculty member of the University of Applied Sciences in Darmstadt, Germany. His research interests include object-oriented modeling, web system design, design patterns, and performance management. A detailed CV is available at www.quibeldey-cirkel.de

Checklist for Web Site Quality Assurance

Klaus Quibeldey-Cirkel

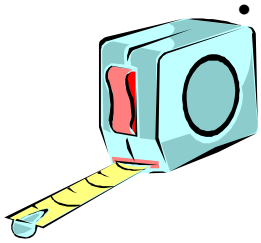
quibeldey-cirkel@acm.org

The Checklist Project

- Cooperation between Industry and University
  - www.bahn.de and www.uni-siegen.de
  - 250 Person Days
  - Fulfilling CMM 2nd Level
- Checklist Evaluated in Industrial INet Projects
  - Corporate Standard of TLC GmbH

Concepts & Foundations

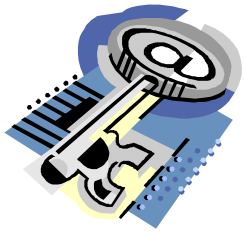
- Usability Engineering
  - What are Usability Factors?
  - What is Quality of Experience?
  - What are the Metrics and Methods?
  - What are Usability Standards?
  - What is Style, what Innovation?
  - What can be Measured?



- Usability Testing Tools
  - Checklists
  - HTML Validator
  - Link Checker
  - ?

3

Sources



- Berners-Lee, Tim, W3C: Style Guide for Online Hypertext www.w3.org/Provider/Style
- Cook, John: The Sevloid Guide to Web Design www.sev.com.au/webzone/design/guide.asp
- Flanders, Vincent: Web Pages that Suck.com www.webpagesthatsuck.com/index.html
- Friedman, Steven Morgan: Resources on Web Style www.westegg.com/unmaintained/badpages
- IBM Corp: Web Guidelines http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/572
- International Journal of Human Computers Studies: Special Issue, World Wide Web Usability http://ijhcs.open.ac.uk
- Isys Information Architects: Interface Hall of Shame www.iarchitect.com/mshame.htm
- Karp, Tony, TLC Systems Corp.: Art and the Zen of Web Sites www.tlc-systems.com/webtips.shtml
- Microsoft Corp.: Designing for the Web: Empirical Studies www.microsoft.com/usability/webconf.htm
- Nielsen, Jakob: useit.com www.useit.com
- SAP AG: SAP Design Guide www.sapdesignguild.org
- Siegel, David: Creating Killer Web Sites www.killersites.com/core.html
- Sullivan, Terry: All Things at Web www.pantos.org/atw
- SUN Microsystems: SUN's New Web Design www.sun.com/980113/sunonnet
- Tognazzini, Bruce: Homepage www.asktog.com
- Webmonkey http://hotwired.lycos.com/webmonkey/design/?tw=design
- Gartner's Web Evaluation Tool: www.gartner.com/ebusiness/website-ings
- Quality of Experience: www.qualityofexperience.org
- Yale University: Yale Web Style Guide http://info.med.yale.edu/caim/manual

4

Open Source Policy

- Public Domain

- Checklist‘s Homepage
  - www.quibeldey-cirkel.de
    (Button <Projekte>)

5

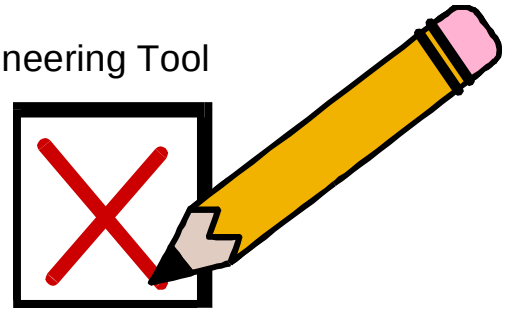
Structure of Checklist

- Main Categories
  - User Friendliness
  - Security
  - Standards
  - General Quality Assurance
  - Reliability and Performance
- Sub Categories
  - Technical
  - Psychological
  - Miscellaneous
- Priorities
  - Flexible Weighting

6

When and How to Use it

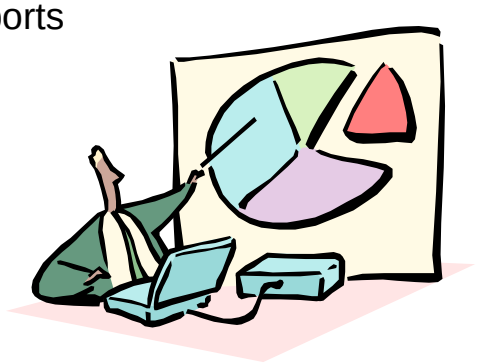
- As Early as Possible
  - at the Project's Kick-off Meeting
  - during Software Engineering
  - at Acceptance Testing
- As a Usability Engineering Tool
  - Excel Spreadsheet
  - Don't Use as Printout



7

Graphic Analysis

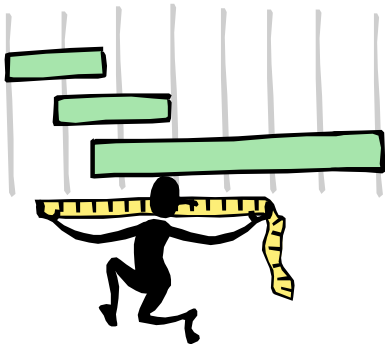
- Summary Report
- Category Reports



8

Tailoring

- Excluding Criteria
- Adding Criteria
- Adding Categories
- Changing Weighting



9

Checklist for Web Site Quality Assurance

- Summing-up

Use it!

Improve it!

Inform me!

10

Checklist for Web Site Quality Assurance

Klaus Quibeldey-Cirkel
Senior System Architect
TLC GmbH, Kleyerstr. 27, D-60326 Frankfurt
quibeldey-cirkel@acm.org

It has become trite to state that Quality Assurance (QA) is indispensable for any Web-enabled application, not only before deployment but also in the early stages of systems development. Certainly, QA is a central task of usability engineering [1]. However, there is no publicly available, comprehensive checklist that guides the developer, tester, and project lead. This causes QA personnel to rely on prior experience, intuition, personal preference, and ad hoc decisions. It is the intent of the author to provide the QA community with a solid source of checklist criteria for assuring Web site quality.

This paper introduces a Microsoft Excel based checklist of more than 200 criteria, categorized and equipped with automatic graphic analysis, for determining what results in a good experience for the Web site customer. The checklist is a product of a recent research and development project consisting of about 250 person days, conducted by the University of Siegen, Germany and the software company TLC GmbH of Deutsche Bahn. TLC GmbH has just re-launched one of Europe's most frequented travel portals, www.bahn.de. The checklist considers industry best practices, generally accepted Web conventions, and has successfully been applied in several e-business projects.

Concepts and Foundations

The experience that end users are having with business-to-consumer applications is the principal focus of usability engineering. The difference between pleasure and frustration results in revenue gained or lost.

Impatience with poor site performance is the most common reason end users prematurely terminate their e-business. It is estimated that slow performance loses e-business sites as much as $ 4.35 billion annually.[1] Web impatience can be measured; according to Zona Research, the waiting threshold for average users is at eight seconds. A Web page that takes any longer to load is at risk of losing its visitors. To illustrate the impact of performance, consider a report, again by Zona Research, that describes a Web site that reduced from 30 percent to 6-8 percent the rate at which visitors abandoned the site from the homepage. This was achieved by reducing the load time of the page by one second.

Besides performance issues, quality of experience is affected by a great deal of other Web usability factors, for a survey see, for example, the usability assessment model in [1]. According to this study, page layout, navigation, design consistency, information

[1] www.availability.com dated from 06/05/2001

content, customer service, reliability, and security are main areas where usability engineering should take place. Any checklist for Web site quality assurance has to address these areas in a comprehensive way.

It is the intent of the author of this paper to provide the QA community with a solid source of checklist criteria that take into account most of current usability factors. To assure legitimacy, all criteria are based on long-term studies about Web quality standards, present-day conventions, and best practices. They are quoted from recognized sources; see the links section at the end of the paper. Those criteria that can be automatically checked with tools like HTML validators or link checkers are annotated accordingly. Finally, the usefulness of the checklist has been validated in several e-business projects.

Open Source Policy

The checklist is considered public domain and is accessible on an Open Source policy. The tool may be deployed for use by any company free of charge. The Web site QA community should use it in multi-authorship as a living document that is deployed, maintained, and updated by any company, so long as any changes are made publicly available.

Download

A pointer to the URL of the checklist can be found at the author's homepage; see www.quibeldey-cirkel.de (button "Projekte"). The German version is already available for downloading; the English one will follow at the end of November 2001.

Structure

Currently, the checklist comprises more than 200 criteria to assure a minimum quality standard from the perspective of the end user. To manage this volume of individual criteria, we have divided them into five main categories (a typical question is attached to each category to illustrate its scope):

1. User Friendliness
   Is the navigation schema consistently implemented on the entire Web site?
2. Security
   Will the transfer of sensitive user data be encrypted?
3. Standards
   Have all pages been tested on different platforms and with different browser types and versions?
4. General Quality Assurance
   Have all internal and external links been checked?
5. Reliability and Performance
   Has the Web site been functionally tested under concurrent load?

Each main category is further divided into three sub categories:

- Technical
- Psychological
- Miscellaneous (legal, commercial, and other aspects)

When and How to Use the Checklist

It goes without saying that quality should not be tested "into" a Web application afterwards, i.e. for acceptance testing; quality is to be engineered from the outset. Therefore, it is recommended to apply the checklist at the early stages of application development. It is good practice to swear the project team to keeping to the quality standard addressed by the checklist already at the project's kick-off meeting.

All questions in the checklist can be answered yes or no. Those questions that are not applicable to the project can be checked as "irrelevant"; if so, they will not be considered for calculating the summary reports. Most criteria are annotated with comments and examples to avoid misunderstandings. In addition, a glossary of relevant terms is attached at the end of the list. To ease the readability of the checklist, you will find filtering options at the left side of the categories.

In principle, the checklist is self-documenting; you will find all necessary information for using and maintaining the checklist's criteria within the Excel spreadsheet.

Graphic Analysis

To make use of the automatic graphic reports, the checklist should, of course, be used electronically as an Excel spreadsheet and not as a printout. Both individual and summary reports are directly produced while checking the questions.

Tailoring

As mentioned before, individual criteria can be left out of account by checking the corresponding column. A second option for tailoring the list is changing the weighting of each criterion or category. All categories and criteria are grouped according to their priority. By default, P1 is the highest priority with a weighting factor of three, and P3 is the lowest priority with a weighting factor of one. All main categories are equally weighted by a factor of three. The calculation is done by multiplying the category's weight by the weight of the individual criterion.

Summing-up

Use it! Improve it! Inform me!

Links on Web Site Guidelines

- Berners-Lee, Tim, W3C: Style Guide for Online Hypertext
  www.w3.org/Provider/Style
- Cook, John: The Sevloid Guide to Web Design
  www.sev.com.au/webzone/design/guide.asp
- Flanders, Vincent: Web Pages that Suck.com
  www.webpagesthatsuck.com/index.html
- Friedman, Steven Morgan: Resources on Web Style
  www.westegg.com/unmaintained/badpages
- IBM Corp: Web Guidelines
  http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/572
- International Journal of Human Computers Studies:
  Special Issue, World Wide Web Usability
  http://ijhcs.open.ac.uk
- Isys Information Architects: Interface Hall of Shame
  www.iarchitect.com/mshame.htm
- Karp, Tony, TLC Systems Corp.: Art and the Zen of Web Sites
  www.tlc-systems.com/webtips.shtml
- Microsoft Corp.: Designing for the Web: Empirical Studies
  www.microsoft.com/usability/webconf.htm
- Nielsen, Jakob: useit.com
  www.useit.com
- SAP AG: SAP Design Guide
  www.sapdesignguild.org
- Siegel, David: Creating Killer Web Sites
  www.killersites.com/core.html
- Sullivan, Terry: All Things at Web
  www.pantos.org/atw
- SUN Microsystems: SUN's New Web Design
  www.sun.com/980113/sunonnet
- Tognazzini, Bruce: Homepage
  www.asktog.com
- Webmonkey
  http://hotwired.lycos.com/webmonkey/design/?tw=design
- Gartner's Web Evaluation Tool:
  www.gartner.com/ebusiness/website-ings
- Quality of Experience:
  www.qualityofexperience.org
- Yale University: Yale Web Style Guide
  http://info.med.yale.edu/caim/manual

[1] Becker, Shirley A.; Mottay, Florence E.:
A Global Perspective on Web Site Usability.
IEEE Software, Jan/Feb 2001, pp. 54-61

QWE2002 Paper 12I

Mr. Guillermo Pastor
(INAD, Spain) & Prof. Antonio de Amescua
(Carlos III University of Madrid, Spain)

Web Development: A New Quality Paradigm

Key Points

- Web quality.
- Hierarchical models for evaluation of quality.
- Quality control.

Presentation Abstract

Quality problems are more specific in web development. Currently there are no specific models for web products quality control. This paper presents an approach to this new paradigm of web development based on quality models.

About the Author

Dr. Antonio de Amescua
Ph. D. in Computer Science. He has been working as researcher in Polytechnic University of Madrid from 1983 to 1991. He worked parallely as software engineering in private and public companies. Now, he is a Full Professor in the Computer Science Department of the Carlos III University of Madrid. He has been researching in new software engineering methods, with results edited in international papers and congress. Also, he has written several books on software engineering. He was the research project leader for the development of the Information System Development Methodology for the Spanish Administration. He, usually, participates in projects sponsored by the European Commission.

Guillermo Pastor
Software engineer by the Polytechnic University of Madrid. From 1996 to 2000 he was a research assistant at the Artificial Intelligence Department of the Polytechnic University of Madrid, working in several Spanish and European research projects. He is currently developing his Ph. D. thesis on web quality at the Carlos III University of Madrid. Now, he is the Organization Quality Coordinator of INAD (INtelligent ADvisors) a Spanish consultancy company belonging to the ALTRAN Group.

Web Development
A New Quality Paradigm

Guillermo Pastor García
Quality Manager INAD
Antonio de Amescua
Professor – U.Carlos III

Web Development Characterization

PROCESS

- Rapid development
- Several emerging technologies
- Product-driven development
- Lack of specific process models

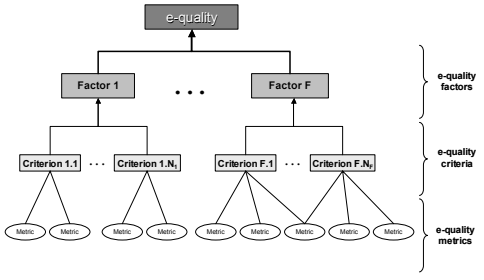
PRODUCTS

- High degree of usability
- Response / Service rapidity
- Error-free necessity
- Security restrictions
- High maintenance load

E-quality Model Introduction (I)

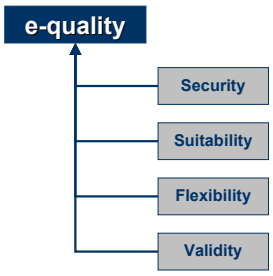
- E-quality is aimed with the global strategic point of view of the e-business
- Hierarchical model for evaluation
- Three levels of granularity
  - **Factors** are related with the tactical aspects to obtain an e-qualified web system
  - **Criteria** are the operative perception and design of the web
  - **Metrics** imply the realization of the operative planning

E-quality Model Introduction (II)



- **Hierarchic**
- **Weighted**
- **Objective**
- **Non intrusive**
- **Verifiable**
- **Several measures**

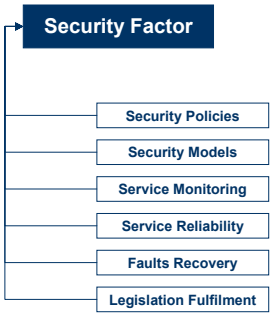
E-quality Factors



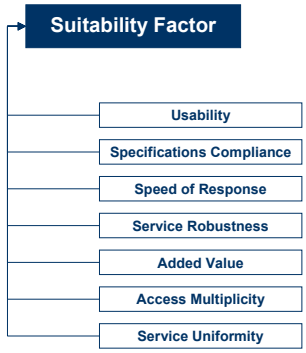
E-quality definition:

- ✓ The requirements of a given product and/or service are satisfied and they are achieved in a satisfactory way.
- ✓ The services offered have an optimal level of approval and satisfaction for its clients.

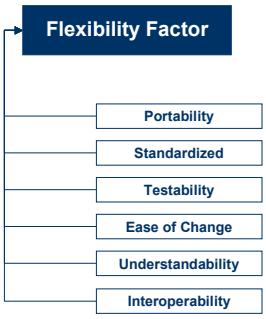
Security Criteria



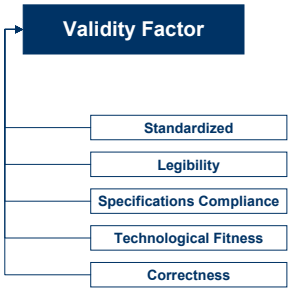
Suitability Criteria



Flexibility's Criteria

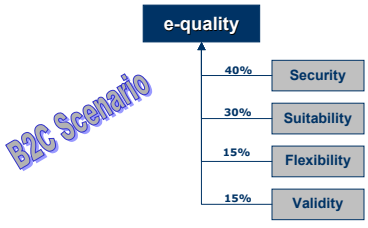


Validity Criteria



Tailorability

- Reference model for e-quality
- Instantiation for each site



B2C Scenario

Value Assignment for Factors

$$F_k = \sum_i \omega_i \times C_i$$

E-quality calculation rule

e-quality = $W_1 \times F_1 + W_2 \times F_2 + W_3 \times F_3 + W_4 \times F_4$

Validation & Conclusions

- "e-confianza" 2001 awards
  (INAD-CommerceNet Spain)
- Auditing of 39 e-business with a success matching in 100%
- Short term:
  - 2nd version of the metrics program
  - Web support system
  - Semi-automated toolset
- Medium-term: Spanish web market
- Long-term: Rest of the net

Contact information

Guillermo Pastor García
Quality Manager
INAD (INtelligent ADvisors)
C/ Albacete, 5 - 3ª planta
28027 Madrid (SPAIN)
Phone: +34 91 404 84 50
Fax: +34 91 403 94 36
E-mail: gpastor@inad.es

INAD INtelligent ADvisors Powering Your Business
http://www.inad.es

Antonio de Amescua
SecoProfessor
Carlos III University of Madrid
Avda. de la Universidad, 30
28911 Leganés (Madrid) Spain
Phone: +34 91 624 94 98
Fax: +34 91 624 94 30
E-mail: amescua@inf.uc3m.es

http://www.uc3m.es

WEB DEVELOPMENT: A NEW QUALITY PARADIGM

Guillermo Pastor García
Organization Quality Responsible
INAD (INtelligent ADvisors)
C/ Albacete, 5 - 3ª planta
28027 Madrid (SPAIN)
Phone: +34 91 404 84 50 Fax: +34 91 403 94 36
E-mail: gpastor@inad.es

Antonio de Amescua Seco
Professor
Carlos III University of Madrid
Avda. de la Universidad, 30
28911 Leganés (Madrid) Spain
Phone: +34 91 624 94 98 Fax: +34 91 624 94 30
E-mail: amescua@inf.uc3m.es

Abstract

Web systems suffer from specific problems and needs because their own nature. As well as with the classic software systems happened in the past, web development could be a handicap for the software engineering discipline if there is not an adoption of some basic rules. This paper describes a basic mechanism based on software quality that evaluates the e-quality of a web development and diagnose it.

Keywords: Web quality control, quality models, auditing

Introduction

Since the beginning, Internet and more concretely the world wide web has been characterized by the strong demand and the high necessity of technological solutions that support the set of ideas that emerge within this context.

This environment has been the key for the evolution of this new area of software engineering. And it also has been the main factor that has characterized the development process as well as the software products that result from it.

The next table summarizes the main characteristics of such environment from the double dimension that has been analyzed: processes and products.

Web Development Characterization	
Processes	**Products**
▪ Rapid development (mostly less than three months) ▪ Several emerging technologies ▪ Product-driven development ▪ Lack of specific process models	▪ High degree of usability (web users) ▪ Response / Service rapidity ▪ Error-free necessity ▪ Security restrictions ▪ High maintenance load (specially perfective one)

As it can be observed, web systems are highly specific ones. They are constrained by their own nature. Some of the restrictions related to the development process have been partially solved, mainly by the development of new and specific process models by the adaptation of classical ones [1]. However, the rest of restrictions related to the web development process in several cases they force to sacrifice quality parameters in order

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

to maintain the scheduling. This exchange between quality and time is the main cause of the problems that can be found in the web products [2].

Web products present several deficiencies and lacks. In fact, lot of web sites find some barriers to maintain an acceptable degree of quality factors like usability, security and correctness [3]. In several cases the evolution of a given web site (i.e. its maintenance) is clearly determined by a poor definition and planning during its development. This lack of maintenance planning is generally associated with the adoption of technological solutions with a low flexibility and scalability factors [4]. This situation avoids the products to evolve as fast as the market do.

Several market studies [3] reflect this scenario and the conclusions are always similar: Most of the cases, when a user detects a lack of some of the premises of the table, he uses to leave the site and in a high percent of the cases he never come back with the decrease of traffic and revenues. Typical problems and their consequences are showed in a cause-effect relation:

- Errors in the products, since broken links to operative faults, security holes, etc.
- Lacks in the services, as a result of the errors in the products
- Unsatisfied users and inefficient politics for fidelization
- Lack of return of investment and so problems with the perdurability of the e-business
- Need of big investments in information technologies, in order to maintain and evolve the software systems of the site

All the related problems are a clear symptom of a lack of investment in politics and mechanisms to assure the quality in web development (e-quality). This basic requisite is not managed with sufficient rigor mainly by the strong restrictions of time associated to this kind of software projects.

This paper presents a solution to this situation. It is based on the definition of a formal mechanism for quality control over the web products. A toolset argued in a model for the evaluation of e-quality and a metrics program that instantiates such model and the characteristics of the web development paradigm.

The e-quality model built is characterized by a set of attributes: Hierarchic, weighted, objective, non intrusive, verifiable and based on multiple metrics. Each of these attributes make the reached solution as a valid mechanism to control the quality parameters of web products as well as a proper formalism to manage the multiple e-business models due its taylorability.

The paper is organized following the next contents and chapters:

- E-quality evaluation model architecture: This chapter will describe the organization, elements and mechanism underlying the model
- E-quality factors and criteria: It will define and explain the components of the evaluation model, in order to clarify the scientific approach.
- Model taylorability: This chapter will specify the scenarios approach, the way to obtain it and its functioning, for the multiple e-business models.

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

- Validation of the model: The final chapter will relate with the experimental platform that has been designed to prove the e-quality model. The results of the theoretical approach given in previous chapters will be put in practice following the steps designed here.

As it was showed above and according to Kuhn's terminology [5], web development owns all the characteristics and symptoms of a paradigm: It is a scientific fact universally recognized that provides to the scientific community with a set of problems and solution patterns during a time period. Due to the specific area with this paper deals we can afford that web development is a new quality paradigm.

E-quality model architecture

The approach to evaluate the quality of a web development that is given in this paper is related with the software quality control techniques [6] and measurements programs [7]. The final result is a hierarchical model of e-quality and a measurement program obtained through the Goal/Question/Metric technique [8].

The aim of such model is to measure the quality degree of a web system from the point of view of the final user or client. This approach is valid for any kind of web system and it fits in every kind of e-business model like B2B and B2C ones.

First of all we must define the concept underlying the term of e-quality. In such sense, for us, e-quality is the capability or facility that a given web site owns so it is true that:

- The requirements of a given product and/or service are satisfied and they are achieved in a correct and satisfactory way (fitness for purpose [9]).
- The products and/or services offered by the web site have an optimal level of approval and satisfaction for its clients.

E-quality, as a quality concept, is a multidimensional attribute [6] for a given product and/or service. It will be necessary to decompose it in a set of sub attributes, in order to simplify its evaluation and to obtain a whole perception of it. Using this approach we can get a set of conclusions of which aspects and elements of the web system would be susceptible of modification and improvement in order to increase the global degree of e-quality.

We have chosen the hierarchical models approach because they allow us to get a multiple visibility of the web system since the strategic to the operative points of view. [10].

The e-quality model that results from our study is mainly characterized by the following attributes:

- **Hierarchic**, that is, every attribute that conforms the e-quality concept contributes to the evaluation of its predecessor and it takes its value from its descendants that are sub attributes of itself.
- **Weighted**, each node of the model contributes to the final evaluation of e-quality through a relative weight associated to its importance within the e-business model that it is currently evaluated. As we will show in this paper, such characteristic is the main factor that confirms the solution as a taylorable one.

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

- **Objective**, the measures, factors and criteria that supports the evaluation model are defined in order to facilitate an evaluation of the e-quality concept in a impartial and independent way. The assessment could be repeated in different times and by different people obtaining the same values and results.
- **Non intrusive**, the audit process applied for each web system doesn't interfere in the organization neither in its software systems. The model collects a set of metrics over the whole system but this process of examination doesn't modify or affect to the web service and its operability.
- **Verifiable**, every measure or valuation could be validated and repeated in order to certify the final result of the evaluation. The audit process as well as the assessment is perfectly defined and normalized so it could be verified in each of its steps.
- **Several measures**, each attribute or sub attribute shall be evaluated through a set of different measures and parameters in order to increase the objectivity and completeness of the final valuation of e-quality, so the trustworthiness of the assessment will be guaranteed.

Deeply in the architecture of the model, e-quality is divided up to three levels of detail: factors, criteria and metrics. Each of these sub attributes follow a hierarchical organization that corresponds with the next figure.

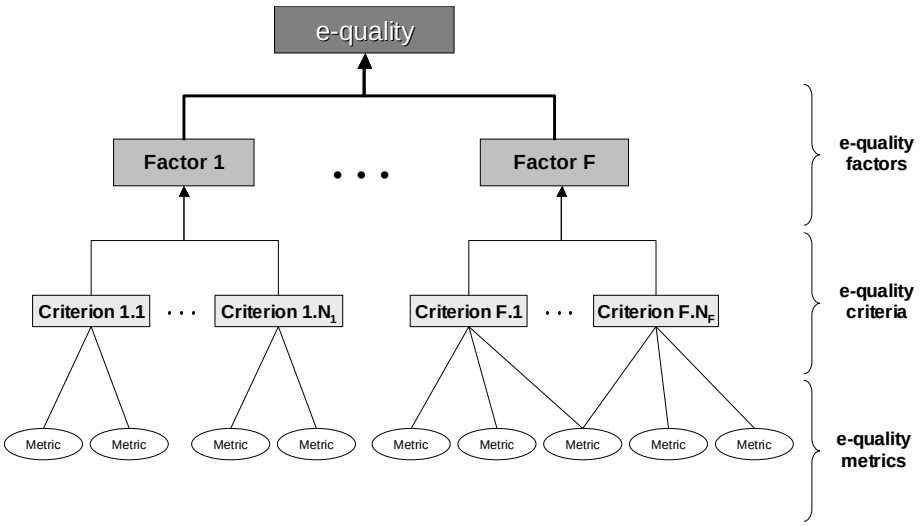


Figure 1 General architecture of the e-quality model

As it was mentioned at the beginning of this chapter, every level of detail is associated with a different point of view as we explain:

- E-quality is aimed with the global strategic point of view of the e-business
- Factors are related with the tactical aspects to obtain an e-qualified web system
- Criteria are the operative perception and design of the web

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

- Metrics suppose the realization of the operative planning

This set of perspectives will be more comprehensive when we instantiate the elements of the model in the next chapter. As a result, we have obtained a whole model that in its first version is composed by four major factors of e-quality, twenty four criteria and about two hundred metrics and parameters for the evaluation of criteria.

E-quality factors and criteria

In the previous chapter we have seen the general architecture of the model. As it was mentioned, e-quality concept is decomposed in four factors for its evaluation. Each of these factors is also divided into a set of twenty four criteria that takes their value from a set of metrics [11].

In this section we will present the instantiation of the model, that is the concrete factors and criteria that grouped bring the concept of e-quality its whole meaning. Next figure represents the first division of e-quality that it is associated with its strategic view.

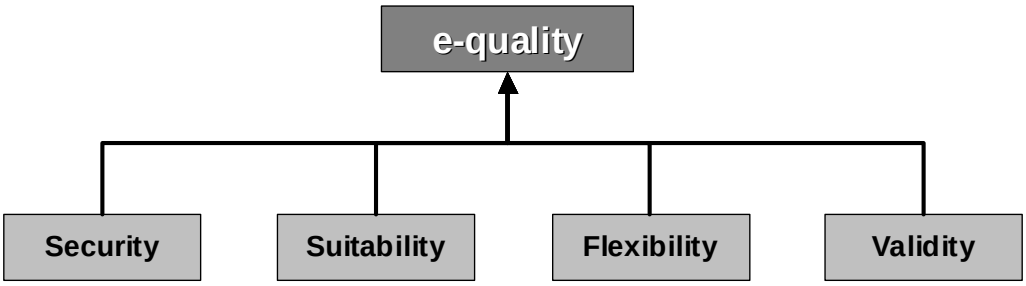


Figure 2 E-quality factors of the model

As a result a more concrete definition of e-quality can be done. A given web system is e-qualified if and only if it is a secure, suitable, flexible and valid one. This definition, as well as the instantiation of factors, matches with the characterization of the web development that was made at the beginning of the paper.

In the rest of the section we will define every of these factors as well as the concrete criteria that are behind them.

Security factor

This factor could be the most known but not the best managed in every web system. There are lot of companies and methods to evaluate the security offered by a web site to its users and clients [12] but security rules are related to several parameters since the technological ones to those associated with law and the whole set of criteria accomplish the concept of web security.

So in order to have a clear idea of what security means within the context of this paper, we define security as the degree of assurance that a given web site offers to its clients and users in terms of privacy of the sensible information managed, the integrity of the operations and services provided, and the fulfillment of the current legislation.

In such terms, security must be decomposed in a set of criteria that cover the whole meaning underlying this concept.

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

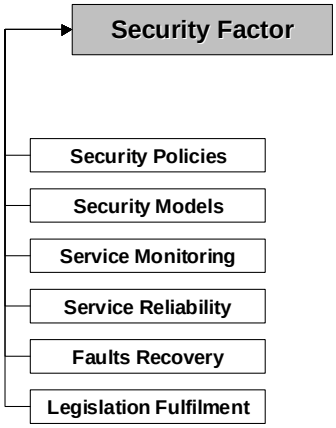


Figure 3 Security Criteria

Each of the criteria identified in figure 3 are detailed now:

1) Security Policies: This criterion is related with the evaluation of the rules, infrastructures, resources and elements that improve the security of the systems that support the web site. It is also evaluated the protection from unauthorized accesses to the web systems from inside and/or outside the organization.

   This criterion is associated with the means and internal processes that the owner of the site has established in order to avoid attacks and unforeseen situations that could affect the web system and the information managed.

2) Security Models: It is related with evaluation of the services security. That is the degree of guarantee of the interaction process between the web system and the user. In that sense, this criterion valuate the protocols, technologies and infrastructures (logical and physical ones) that assures the integrity of the users transactions and operations which are one of the most known security potholes.

3) Service Monitoring: To know what exactly happens in the web site is a proper policy of security. So, this criterion validate the existence and correctness of mechanisms that support the monitoring and supervising of the services, daily operations and user behavior, like logs analysis, datamining systems, etc.

   Using such mechanisms and tools we can ask for any inconvenience in the service and we can register and classify the whole operations which can be very useful when they are economic ones.

4) Service Reliability: A known weakness in web security occurs when the service falls down or suspends its operation due to a system failure leaving accessible it main components to a foreign access. To check how robust is a web site we must know the failures rate and we must to evaluate those parameters that assures the correct operation.

5) Faults Recovery: Security may also be associated with the ability of a system for keep the services, information and data stored in a stable mode when a system failure occurs. This criterion is also related with the service reliability one. So when we evaluate this criterion we must observe the existence of back-up

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

policies, redundancy systems (logical and physical) and a whole inspection of the degree of logical robustness. Analysis of historical data is also recommended.

6) Legislation Fulfilment: Several countries and regions have its own legislation over the treatment of digital sensible information (personal, finance, ideological data, etc). This legislation offers a security guarantee for the final users and must be followed by the organization that supports the web site .

Suitability factor

This factor allows the measuring of those parameters relate to the acceptance of the web service provided. Then it is concerned with the use, supply and completeness of the services and the ease of access to them.

A web system is suitable when it is usable, it verifies the expectations, it has a good time of response, it adds value to the user operation, and it is accessible to the whole community in the same conditions. We can see de detailed decomposition of this e-quality factor in the next figure.

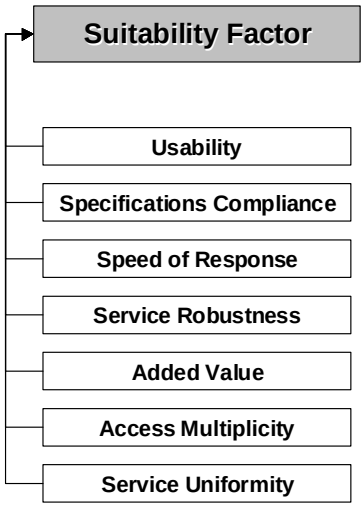


Figure 4 Suitability criteria

Further explanation of these criteria is given below:

1) Usability: One of the main criteria that involves the suitability is the ease of use or usability, since it is a measure and a study of the intuitiveness, processes of service and contents management and intelligibility, that the web system offers to its clients and users [13]. It involves the study user interface design and the execution of tests that simulate the users behavior.

   Mainly, the elements of study are the following:

   - Simplicity of the web site utilization (balanced maps, length of services, etc.)
   - User friendly (number of items in each page, net information, complexity of the contents, etc.)
   - Use learning capabilities (online help, examples, demos, support online, etc.)

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

- Operations dependability (confirmations, ticketing, etc.)
- Intuitiveness (location systems, contextual help, menus, etc.)
- Use of design standards (style guides, contents organization, contents management, etc.)

This criterion is clearly related with the evaluation of the front-end subsystem.

2) Specifications Compliance: Quality in general has a clear relation with the final user of the product/service in terms of offer and demand [14]. In such sense, every web system must have a complete specification of the services and capabilities that offers to its users and clients and must respect them.

   So, this criterion must measure the degree of compliance with the specifications of the web site/system since its functionality to its quality of service. This may imply the study and inspection of the documentation and code of the front-end as well as the back-end, and it also may imply the execution of test that verify the specifications of the site.

3) Speed of Response: Perhaps this criterion is one of the most important with the most influence in the satisfaction of the user. Most of the times an user goes to a web site and the response it is slow he/she never come back.

   But to objectively measure the speed of a given web system we must take into account several aspects and weight them into the final result:

   - Processing power of the servers, its redundancy and the COTS used.
   - Bandwidth used by every peer and the type of connection (ADSL, Modem, LAN, etc).
   - Time of the day when the connection is made as well as the network load and server load.
   - Size of the information downloaded/uploaded since it influences in the speed.
   - Type of processing for the pages: server pages as ASP, JSP, CGI versus client pages as Javascript, Java, etc.
   - Number of users accessing the site and/or the concrete service.
   - Processing power of the users not only hardware but software elements and its versions.

   In such sense, the tests must be done taking all these aspects into account and then the measures gathered must be sufficiently weighted in order to get a meaningful degree for the speed of response.

4) Service Robustness: This criterion has been described into the security factor but here it has a significant impact into the suitability of the web services and in the final user satisfaction. We can collect metrics like failures statistics for each user, broken links, scripts faults, failures rate, time of inoperability, etc.

5) Added Value: Beside the specifications or the main objective of a given web site the organization that supports it can offer several services and products that

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

conforms an added value packet for the final user. These are policies that try to get the fidelity of the client / user by giving him/her a set of products without moving away from the web site. For instance, some added value services could be the following if they are "outside" the basic specification:

- Support services such call centers, software downloads, etc.
- Web mail and web space or hosting
- Personalization facilities
- Others

6) Access Multiplicity: Internet is the common way to interact with a web site/system but not the only available. Currently we are attending to the raise of several technologies that increase the interoperability of the users (WAP, GPRS, SMS, Pocket PCs, television, etc) . So this criterion is clearly concerned with the existence in a given web system of alternatives to interact with the final user and the degree of service that offers through these means.

   There is an important difference with the added value, because in this case the only change is in the means and technologies used not in the final services provided.

7) Service Uniformity: Uniformity simplifies the operability of a web system, but there are several parameters that are involved in this concept. Mainly the uniformity is related with the universality of Internet a global communication and interaction framework without frontiers. So an e-qualified web system must take this concept into account in order to satisfy its clients and users. We can classify the uniformity in various ways that must be evaluated within the context of this model:

   - Geographic dispersion: The service must be supplied in the same way and without distinctions for every person and in every place. This parameter would be affected by logistics and it would be defined by the scope of the organization so its absence could not decrease the level of uniformity.
   - Multiculturalism: Several persons from different cultures and countries can reach the web system, so it is important to take it into account and to provide solutions that allow the interaction within a specified cultural framework. A basic mechanism would be the language choice facility.
   - Platform independence: There are a wide variety of platforms to access Internet not only hardware specifications and model but software browsers, clients and operative systems. To obtain a uniformity (i.e. suitable) web system must consider the standardization for such variety of platforms. This could involve an investment in the development and maintenance of the web system (mainly in the front-end subsystem) but it assures that the user will not reject the site in its first visit as in many cases occur. To really consider this parameter will be necessary to have statistics about the visitors in order to consider the development of alternatives compatibles.

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

- Technological standardization: As well as the client platforms uniformity has been treated there are several aspects of a web must be standardized in the side of the server. The use of more or less extended technologies and products will increase the suitability of the web because they will reach to every potential client. The following examples could guide us to get a more standard web system:
  - In documents and reports dispatching it is more standard to use Postscript files against PDF documents. PDF against Rich Text Format. And Rich Text Format against Microsoft Word files.
  - The use of incrusted elements such vectorial graphs or Flash presentations could be a good choice if we offer the user the plug-in.

  Anyway, when a different or non-standard solution will be offered it is really important to supply to the user of alternatives ways to visualize them such direct links to HTML pages or to skip the items. On the other hand such approach allow us to increase, in many cases, the speed of response that has a heavier weight within the suitability factor.

Flexibility factor

This factor is related with the ability of the web system to change and evolve as well as the e-business do. In such sense we can talk about flexibility as the evolve-ability or maintainability of a web system, that are a well-known quality factors in software engineering [4]. As mentioned at the introduction of this paper, the degree of change and evolution of web systems is one of its most acute characteristics.

The flexibility is concerned with the technical aspects of the web solution that has been implemented as well as with the way that development was carried out. So we think in flexibility (i.e. maintainability) in terms of ease to improve the capabilities of the system (perfective maintenance), the ease to adapt the web system to a new technological environment (adaptative maintenance), the ease to correct errors and failures in the software systems (corrective maintenance) and the ease to transform and to restructure the software in order to increase its quality and e-quality degree [15].

On the other hand, to speak about maintenance implies the concept of change management, validation and verification activities and the understanding of the software currently operational.

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

For such reasons, we can determine that the underlying criteria of the flexibility factor are those showed in the following figure:

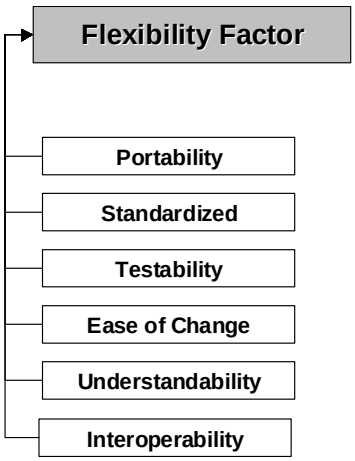


Figure 5 Flexibility criteria

To prevent the future changes of a software system (i.e. a web system) must be undertaken and planned during its development in order to decrease their impact over the maintenance phase and to carry on an evolution process instead a revolution one. The prevention actions will improve the final quality observed in new releases of the web system and will increase the speed of the change process.

As we can deduce, flexibility is related with the study of the back-end as well as the front-end and it implies the examination and detailed analysis of the code and designs of both software systems. This reason gives to the flexibility factor a software inspection nature and so several techniques such RTF, walkthroughs or similar, would be needed to evaluate it.

Last paragraph also alerts us for the need of a good estimation of resources to carry on these flexibility analysis in a proper way.

1) Portability: This criterion aims with the ease of a system or component to migrate from on environment (hardware or software) to another. Every web technology scores in portability. In such sense, a detailed benchmark of technologies and platforms must be available in order to determine the exact degree of portability that a given web system has.

   This could be the most significant element that influences the portability of a software system but not the only one. In fact, there is another parameter that would increase or decrease this e-quality criterion. Such parameter is the number and nature of the COTS that are interrelated with our web system. By defect, as well as the number of COTS grows, the effort to migrate (portability) also is increased. Such COTS analysis could be made with the rest of subsystems or another web projects which ones ours interact..

2) Standardized: The existence of a standardized development and maintenance process a priori implies that the future changes over the products will be

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

effortless than when such standardization is not present [4]. As more extensive a process model is settled higher flexibility is reached.

So those organizations or providers that develop web system could be evaluated as their internal processes are defined. For instance an ISO 9000 certification is an "official" guarantee that there is an standardized process.

But not only the existence of a normalized process is enough to valuate the flexibility. A deeper study of the broadening and the intensity of such process abroad the Internet factory must be made to get an idea of the level of maturity of such standardization.

3) Testability: It is commonly defined as the ease to test over a system. It is mainly concerned with the existence of strategies and test planning previously applied over the system, that is, the degree of knowledge about the test activity.

   To have a defined and detailed test process allows us to believe that the check of future changes and releases will improve the degree of quality of the web system because of the easiness to validate that the new system will fit with its specifications (i.e. the certification factor will be assured).

   The tests activities are possibly the most critical in the software system evolution phase and so the flexibility factor will be affected by the way they are made.

4) Ease of change: As well as portability is related with the platforms migration (adaptative maintenance), the ease of change is related with the rest of changes that a web system could suffer:

   - To include or to modify the capabilities of the web system.
   - To correct errors and defects detected in the exploitation phase.
   - Preventive actions to increase the degree of e-quality.

   In order to evaluate this criterion detailed inspections of the architecture and components design must be undertaken, as well as code inspections over both the back and front-end subsystems.

5) Understandability: The first task that must be accomplished when a software system evolves is the understanding of its architecture and design. This understanding task will consume several resources if the elements of the system are not well documented and clearly implemented. In most of the cases the understanding will involve the inspection of the code in order to define the best way to include changes.

   Due to this the flexibility factor is clearly determined by the ease to understand the whole system and its components, the quality of the technical documentation and the cleanliness of the code.

   For instance, a understandable code must follow a guide of naming components and artifacts in a standard way, for each of the pieces of the system.

6) Interoperability: Usually a web system interacts with several external and heterogeneous systems. The interoperability allows us to measure the simplicity of such interaction. This criterion influences in the final flexibility factor because,

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

as portability did it, a standard or commonly extended protocol of interaction will simplify the change of a system or component.

Think in the interoperation between a script and a database system. If the interface protocol used is SQL a future change in the client or in the server will not affect to the data access sentences.

On the other hand, if the interoperability between two systems is a in-house solution changes in one of the two members of the relationship will suppose a change in the routines that realize the interface, in most of the cases.

This situation is more acute when we are talking about a B2B schema. Usually such web systems need to interrelate applications of two different organizations and the connectivity is a critical and strategic premise. In cases like this the use of opened solutions, like XML and XSP are preferred for the flexibility of the environment.

Validity factor

This fourth factor measures the correctness and accuracy of the technological solution that supports the web site, both in the back-end and front-end.

At this point of the evaluation, a global inspection and valuation of the decisions taken along the life cycle will be done. For such objective, we decompose the validity factor in five criteria as in next figure.

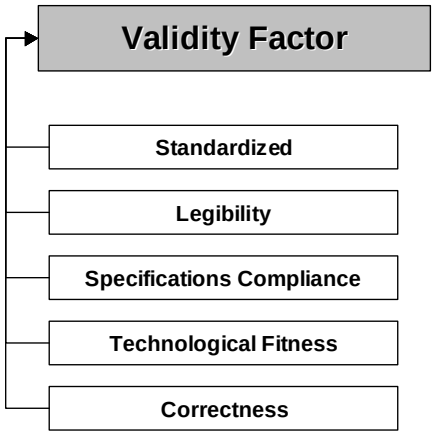


Figure 5 Validity criteria

1) Standardized: This criterion also was included in the flexibility factor that was studied before, but at this point we will emphasize in the test phase and its suitability to a defined standard.

   The evaluation will focus in checking that the web system has been tested enough and that there is a commitment with the validation plan and the defined test process and tools.

   At the end a physical audit will be done in order to check if the standards and project planning has been followed through an inspection of the whole development artifacts and actions taken.

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

2) Legibility: This criterion makes an detailed analysis of the documents and code of the web system in order to evaluate the degree of significance and meaning that they transmit to the work team that must evolve the system.

   So, several metrics can be used in order to evaluate this criterion: For instance, during the analysis of the code we can measure the density of comments and we can weight it by a measure of its usefulness. On the other hand, to evaluate the legibility of the documents contents we can apply some linguistic metrics and also can be weighted by the criticism of the information within the document and along the project.

3) Specifications compliance: In the functional and capability sense, we can affirm that the degree of validity of a web system is related to the degree of fulfillment and achievement of the technical specifications. This criterion also has been evaluated for the suitability factor, but now it is time to confirm how many of the technical specifications (not only service ones) are actually implemented.

   A strict evaluation will examine the whole set of specifications: quality attributes, restrictions, interface requirements, functional requirements, etc. These must be compiled in the Requirements Specification Document. To check that if a given requirement has been fulfilled we will need to trace it from the analysis phase to the implementation one.

4) Technological fitness: To decide that a given technological platform is the best for a given site will require to contrast the problem definition, the technical requirements (specially quality attributes like maintainability, security, etc) and the benchmarking of the final platform. For instance, ISAPI will fit better than a solution based on ASP if the security is a critical requisite of the web system.

5) Correctness: We have evaluated that the functional capabilities are implemented in a standardized and understandable way, but we must also check if they work and they do it in a proper and expected way. This is the concept of validation. We will not apply the test plan again but we can evaluate the test activities that were carried out during the development.

   The whole test activity is summarized in the validation planning, in the defect forms and in the configuration management archive. A simple inspection of these informations can give us a global idea of the correctness.

   In order to improve the evaluation, some test will be applied specially those one that were critical under the development and maintenance. Other parameters like the coverage of the test cases, its rigor and the defects management will help in such evaluation.

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

Model taylorability

One of the main characteristics of the evaluation model is its taylorability. This feature is related with the ability of the model to be adapted for any kind of e-business and any case of web site.

The architecture of the model allows to easily adapt it for a specific site through a mechanism that we call scenarios.

The scenario mechanism is based in a assignment of weight for each of the branches of the model. The e-quality model was decomposed in four factors and each of these factors was also decomposed in a total number of twenty-four criteria.

For instance, security factor takes its value from the combination of the values of its six criterion using the following standard equation:

$$_k = \sum_i \omega_i \times C_i$$

Where F_k is the K-th factor, C_i is the i-th criterion of F_k and ω_k is the associated weight of C_i within the factor F_k.

To assign a value for the e-quality of a given site we use the following equation:

$$\text{e-quality} = W_1 \text{ x } F_1 + W_2 \text{ x } F_2 + W_3 \text{ x } F_3 + W_4 \text{ x } F_4$$

To define a scenario for a web site, we only have to assign numerical values to the weights ω_k and W_1 , by taking into account the specific characteristics and requirements of the web site.

We have defined several generic scenarios for the most usual e-business schemas (B2B, B2C, generic, etc) like the showed in the figure[1].

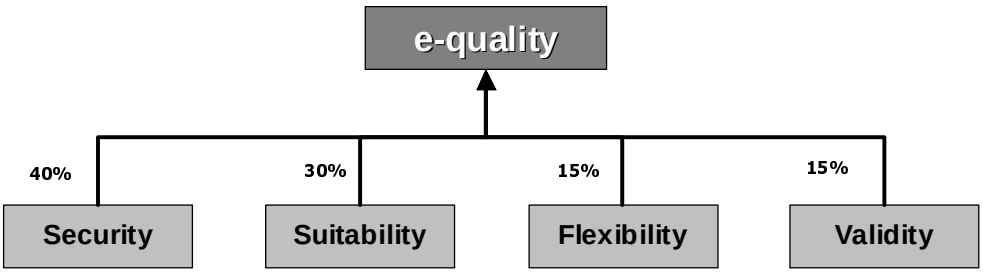


figure 1 Scenario example - B2C schema -

Anyway, this set of generic scenarios is a starting point. Our approach needs from a previous analysis of the site and the business sector where it is included in order to identify the exact values for each of the twenty-eight weights.

[1] Due to confidentiality reasons we cannot detail the full values of the weights

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

Validation of the model

Currently we are developing the second version of the metrics program in order to increase the objectiveness and the number of quantity measures. We hope that for the beginning of the new year we will finish this second version of the model.

With the current version, INAD have participated as the audit team for the "e-confianza" awards 2001, sponsored by CommerceNet Spain [16]. This first attempt to the validation of the model was carried out over thirty-nine e-business companies of different nature (e-brokers, e-shops, marketplaces, e-health, e-finance, telecommunications, etc).

The main results of this first experimental platform have been very valuable because of the relevance of the companies audited and for the suitability of the results obtained. The evaluations made fit with the needs of the companies and when the results were reported to them there was a common agreement that the evaluations matched with the perceived e-quality in the 100% of the cases.

Currently, the e-quality evaluation is an INAD project. A working team of four consultants are working in the depuration of the model and in the building of a software toolset that supports the metrics gathering as well as the analysis and reporting facilities in order to semi-automate the whole process of auditing.

For the next version of the model, we expect that a bigger number of companies will be evaluated and in future papers we will present the results of our research.

References

[1] Jacobson, I. *The unified software development process*. Addison Wesley 1999.

[2] Jarvis, A. *Inroads to software quality.* Prentice-Hall, 1997.

[3] ACNielsen. *Primer Estudio Sobre el Comercio Electrónico en España.* CommerceNet / ACNielsen, 2001.

[4] Pigoski, T.M. *Practical software maintenance.* Wiley computer publishing. John Wiley & Sons, 1997.

[5] Kuhn, T.S. *La estructura de las revoluciones científicas*. Fondo de cultura económica, Madrid, 1975.

[6] Gillies, Alan C. *Software Quality.Theory and Management*. Ed. Chapman & Hall. London, 1992

[7] Offen, R. J. & R. Jeffery. *Establishing Software Measurement Programs*. IEEE Software (March/April 1997), p. 45 - 53.

[8] Basili, V.R. *Software Modeling and Measurement: The Goal/Question/Metric Paradigm*, UMIACS-TR-92-96, September 1992.

[9] Dunn, R.H. *Software Quality. Concepts and Plans.* Prentice-Hall, 1990.

[10] Aedo, I. et al. *Cuadernos de Informática.* Chapter 1. Federación Española de Sociedades de Informática, 1999.

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

[11] Pastor, G. et al. *Modelo de Evaluación y Certificación de Parámetros de E-confianza.* INAD technical report, INAD-CN/TR/MECPE, 2001.

[12] Web reference: http://www.verisign.com/consulting/index.html. Verisign, Inc.

[13] Nielsen, J. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, Indianapolis, 2000.

[14] Pressman, R.S. *Ingeniería del software. Un enfoque práctico.* McGraw Hill, 1993.

[15] *IEEE Standard on Software Maintenance*. IEEE Std 1219, 1993.

[16] Web reference: http://www.commercenet.org/premio2001/ . CommerceNet Español

© Copyright 2001 Intelligent Advisors (INAD). All Rights Reserved.

QWE2002 Paper 1M

Mr. Stale Amland & Martin Pol
(Amland Consulting, Polteq IT Services B.V.)

Test Process Improvement -- Theory and Practice

Key Points

- How to improve your test process?
- How to prepare for and execute a TPI assessment?
- Pitfalls and problems to look out for in the improvement process -lessons learned.

Presentation Abstract

The presentation will focus on the theory and the practice of test process improvement. Part one of the presentation will focus on the theory, based on the TPI® model: Part two of the presentation will focus on the case study: The background of the organization will be briefly explained and experiences from the assessment process will be discussed, including interviews and obtainment of management buy in. The outline and the findings in the assessment report will be presented, like the current situation, the short and long term recommendations and the proposed implementation plan. Pros and Cons of the assessment process will be discussed as well. Further, this part of the presentation will focus on the experiences from the implementation of the recommendations. This will include a description of the actual improvement activities and the prioritization, summarizing what did work and what went wrong.

About the Author

Stale Amland was born in 1961. He has a Master of Science from the Norwegian Institute of Technology, 1985, Industrial Economics. He has been employed by software houses and consulting companies and done consulting in a wide range of businesses, including financial industry, telecommunication and transportation. Stale is a frequent presenter at conferences in Europe and USA. He received the "Best Paper Award" at EuroSTAR 1999.

Martin Pol, with more than 25 years of experiences in structured testing is the (co-)author of many publications and books on structured testing in different languages. He is the CEO and a senior consultant for Polteq International Testing Services B.V. and as an advisor connected to the R&D department of IQUIP Informatica B.V.. He was involved in the development of the structured testing approach TMap and the creation of TPI. Also because he's working in the testing practice every day Martin is a highly regarded lecturer and speaker at conferences throughout the world. He twice chaired the EuroSTAR conference and recently he received the European Testing Excellence Award for his contribution to the field of testing. TPI® and TMap® are registered trademarks of IQUIP Informatica B.V.

Experiences from a TPI® assessment at a large Telecommunication Company

Ståle Amland & Martin Pol

Presentation Outline

- Test Process Improvement
- TPI® in practice
  - Introduction
  - Current situation
    - trends & TPI®-matrix
  - Recommendations
    - short term and long term steps to take
- Summary

TPI®

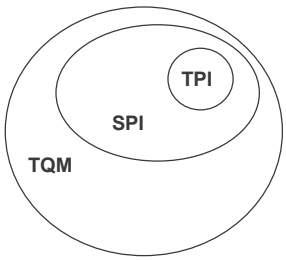
is a registered trademark of IQUIP Informatica B.V.

Amland
Consulting

What is Test Process Improvement?

"Continuous improvement of the quality and the efficiency of the test process, related to the output of the total software process"

- Quality — Insight, Coverage, Control, Timeliness
- Costs — Cheaper
- Lead time — Faster



Amland
Consulting

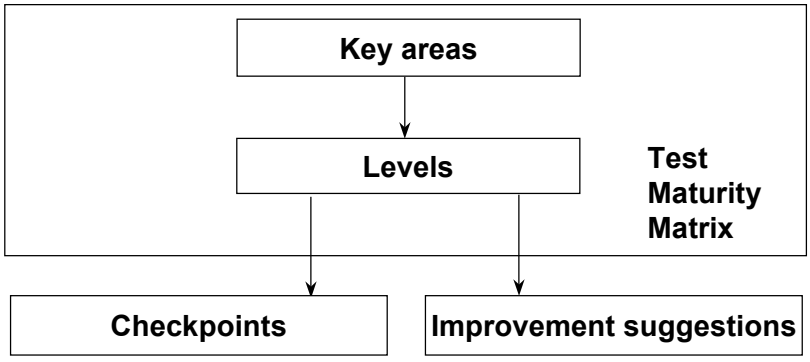
© IQUIP Informatica B.V.

The TPI® model

- Reference Guide
- Process of change

Amland
Consulting

The TPI® model



Amland
Consulting

© IQUIP Informatica B.V.

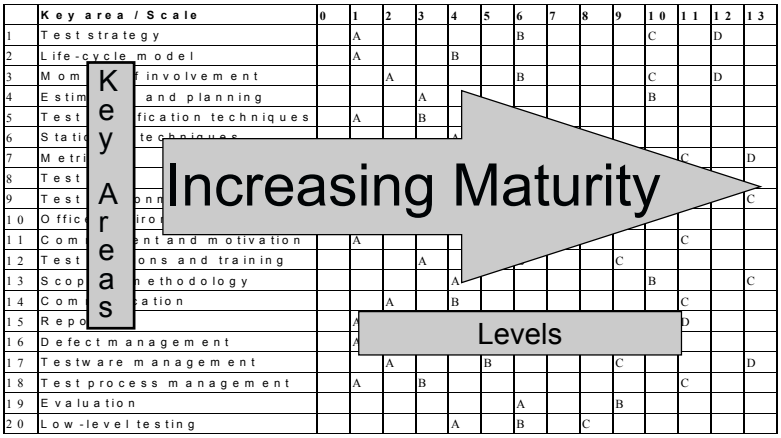
20 Key areas

- Test strategy
- Life-cycle model
- Moment of involvement
- Estimating and planning
- Test specification techniques
- Static test techniques
- Metrics
- Test automation
- Test environment
- Office environment
- Commitment and motivation
- Testing functions and training
- Scope of methodology
- Communication
- Reporting
- Defect management
- Testware management
- Test process management
- Evaluation
- Low-level testing

Amland Consulting

© IQUIP Informatica B.V.

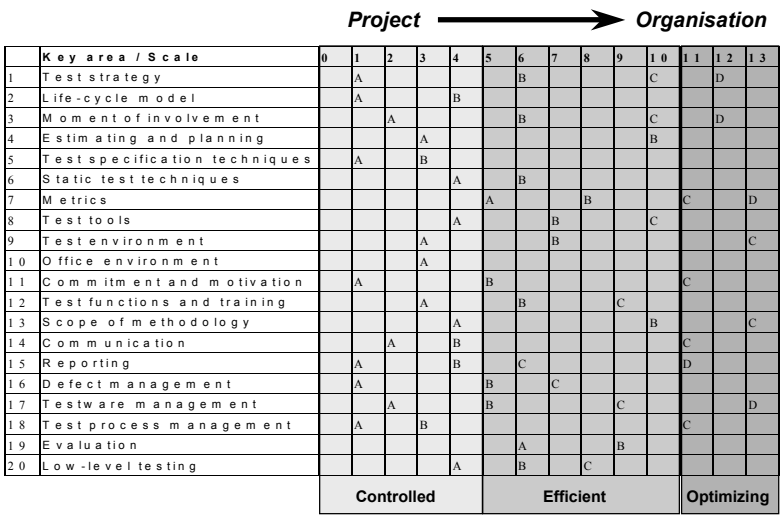
Test Maturity Matrix



	Key area / Scale	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	Test strategy		A					B				C		D	
2	Life-cycle model		A			B									
3	Mom involvement			A				B				C		D	
4	Estim and planning				A							B			
5	Test fication techniques		A		B										
6	Stati techniques					A									
7	Metri												C		D
8	Test														
9	Test onm														C
10	Offic iro														
11	Com ent and motivation		A										C		
12	Test ons and training				A						C				
13	Scop ethodology					A						B			C
14	Com cation			A		B							C		
15	Repo		A										D		
16	Defect management		A												
17	Testware management			A			B				C				D
18	Test process management		A		B								C		
19	Evaluation							A			B				
20	Low-level testing					A		B		C					

Amland Consulting

TPI Maturity Categories



	Key area / Scale	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	Test strategy		A					B				C		D	
2	Life-cycle model		A			B									
3	Moment of involvement			A				B				C		D	
4	Estimating and planning				A							B			
5	Test specification techniques		A		B										
6	Static test techniques					A		B							
7	Metrics						A			B			C		D
8	Test tools					A			B			C			
9	Test environment				A				B						C
10	Office environment				A										
11	Commitment and motivation		A				B						C		
12	Test functions and training				A			B			C				
13	Scope of methodology					A						B			C
14	Communication			A		B							C		
15	Reporting		A			B		C					D		
16	Defect management		A				B		C						
17	Testware management			A			B				C				D
18	Test process management		A		B								C		
19	Evaluation							A			B				
20	Low-level testing					A		B		C					
		Controlled					Efficient						Optimizing		

Amland
Consulting

Background

- New and challenging technology
  - GPRS and UMTS
- Grew from 2 up to 200 people in 2 years
- Many organisational changes
- Outmost time-to-market pressure

Amland
Consulting

Assignment & Approach

- Assess current test process
  - study documentation
  - interview
- Recommend improvement steps
  - analyse and propose
- Create rough implementation plan
- Present results in report and presentation

Amland
Consulting

Objectives

- Reduce project delivery time (time-to-market)
- Improve the quality (structure) of the test process
- Reduce time and resource consumption in testing

Amland
Consulting

Guidelines

- Rational Unified Process (RUP) as foundation
- Etc.

Amland
Consulting

Trends (strong areas)

- Motivated staff, optimistic, great team spirit
- Test script review activities, early detection
- Timely involvement
- Test automation experiences
- Defect administration, analysis and tracking

Amland
Consulting

Trends (areas of concern/challenges)

- Time pressure
- Time pressure
- Time pressure
- Time pressure
- Time pressure
- Time pressure
- Time pressure
- Time pressure
- Time pressure
- Time pressure

No time to learn, document, teach or improve!

Amland Consulting

Trends (areas of concern/challenges)

- Time pressure
- Current Test Method - very limited
  - only high-level overview
- No clear process and job descriptions
- No master test plan, no risk based testing strategy
- Unit Test and Unit Integration Test carried out poorly
- No or informal techniques
  - test strategy definition
  - test effort estimation
  - inspections and reviews
  - test case specification

Amland Consulting

Trends (areas of concern/challenges)

- Unstable test environments
- Poor foundation for test management
  - time pressure
  - underdeveloped process
  - lack of supporting tools and metrics
  - continuous reorganisation
- Not enough resources for test automation support
- No appropriate training and coaching

Amland Consulting

Current situation in a Test Maturiry Matrix

Scale / Key area	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Test strategy		A					B				C		D	
Life-cycle model		A			B									
Moment of involvement			A				B				C		D	
Estimating and planning				A							B			
Test specification techniques		A		B										
Static test techniques					A		B							
Metrics						A			B			C		D
Test tools					A			B			C			
Test environment				A				B						C
Office environment				A										
Commitment and motivation		A				B						C		
Test functions and training				A			B			C				
Scope of methodology					A						B			C
Communication			A		B							C		
Reporting		A			B		C					D		
Defect management		A				B		C						
Testware management			A			B				C				D
Test process management		A		B								C		
Evaluation							A			B				
Low-level testing					A		B		C					

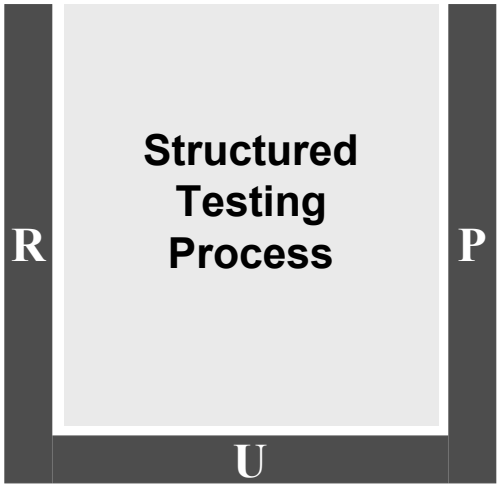
Amland Consulting

Recommendations

- Structured Testing Process
- Test Management
- Test Support
- Improve towards “Controlled TPI-level”
- Continuous improvement

Amland
Consulting

Integrating structured testing



RUP =
Rational Unified Process

Amland
Consulting

Integrating structured testing

R

Structured Testing Process
- Test Life Cycle
- Risk-Based Strategy

Techniques
- Testing techniques

Infrastructure & automation
- Test Automation support

Organisation
- Test Management
- Testware Management
- Test Support (centre)

P

U

Rational the e-development company

RUP =
Rational Unified Process

Amland
Consulting

Short term goals in a Test Maturiry Matrix

Scale / Key area	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Test strategy		A					B				C		D	
Life-cycle model		A			B									
Moment of involvement			A				B				C		D	
Estimating and planning				A							B			
Test specification techniques		A		B										
Static test techniques					A		B							
Metrics						A			B			C		D
Test tools					A			B			C			
Test environment				A				B						C
Office environment				A										
Commitment and motivation		A				B						C		
Test functions and training				A			B			C				
Scope of methodology					A						B			C
Communication			A		B							C		
Reporting		A			B		C					D		
Defect management		A				B		C						
Testware management			A			B				C				D
Test process management		A		B								C		
Evaluation							A			B				
Low-level testing					A		B		C					

Amland
Consulting

Long term goals in a Test Maturiry Matrix

Scale / Key area	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Test strategy		A					B				C		D	
Life-cycle model		A			B									
Moment of involvement			A				B				C		D	
Estimating and planning				A							B			
Test specification techniques		A		B										
Static test techniques					A		B							
Metrics						A			B			C		D
Test tools					A			B			C			
Test environment				A				B						C
Office environment				A										
Commitment and motivation		A				B						C		
Test functions and training				A			B			C				
Scope of methodology					A						B			C
Communication			A		B							C		
Reporting		A			B		C					D		
Defect management		A				B		C						
Testware management			A			B				C				D
Test process management		A		B								C		
Evaluation							A			B				
Low-level testing					A		B		C					

Amland Consulting

Planning the recommendation actions

Recommendations

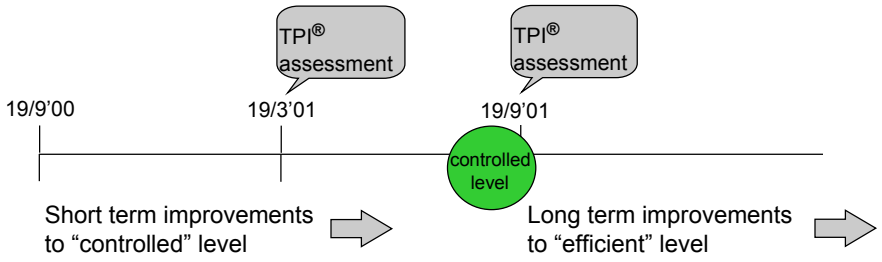
Structured Testing Process
Test Management
Test Support
Improve towards
"Controlled TPI-level"
Continuous improvement

Actions

- Create and install Test Support Unit and support facilities
- Create and install test life-cycle, techniques, etc.
- Create and/or acquire and facilitate training
- Measure, test quality, product quality, test process improvement process
- Dedicate (2 part-time) employees to continuous improvement
- Carry out a TPI® assessment every 6 months.

Amland Consulting

Rough implementation schedule



Amland Consulting

TPI® assessment report

- Used for creating management awareness
- Included in corporate strategy plan
- Prerequisite reading for all test manager prior to planning new projects
- Version 1.0 in September 2000
  - Updated version 2.0 end of January 2001
  - New update September 2001

Amland Consulting

Initial benefits

- Management awareness/buy in to test process improvement
- Strong support for a separate test support organisation (test centre / test factory)
- Focus on the test process and measurement
- Main benefit:
  - External people expressing with strength what other people internally had said repeatedly

Amland Consulting

Test Support Organisation

- Currently 55 staff
- Career plans and mentoring Program
- Separate group for test automation support
- Responsible for:
  - Defining the master test strategy
  - Planning, Preparation and Execution of System Test
  - Acquiring material and organising training
  - Test methodology and test process

Amland Consulting

Metrics

- Measuring by TPI®-level
- New focus areas defined per quarter
- Accurate project measurement possible

- Elapsed time for Use Case Test: 12% reduction.
- System Test: 5% reduction in test execution time
- More defects detected!

- Final Metrics available after program completion: Mid 2002

Amland Consulting

Progress

- ✓ Create and install Test Support Unit and support facilities
- ✓ Create and install test life-cycle, techniques, etc.
- ✓ Create and/or acquire and facilitate training
- ✓ Measure, test quality, product quality, test process improvement process
- ✓ Dedicate (2 part-time) employees to continuous improvement
- ✓ Carry out a TPI® assessment every 6 months.

Amland Consulting

Pros and Cons of TPI® assessment

- Negative impacts? Absolutely non!
- Resistance? Not known!
- Interview process:
  - open and out-spoken organisation
- *"Bottom-up"* process
  - hard work to get management buy-in
- The TPI® report has been a source for motivation and inspiration to continuously improve testing.

Amland Consulting

Did they really improve?

YES……

- Better control
  - improved structure and test management
- Improved testing
  - reduced time
  - better quality
- Improved training and support

NO…..…

- Set-back because of external influence
  - new priorities (schedule and scope)
  - resources shifted around

Amland Consulting

Why TPI®?

- Provides a reference model for test process improvement
- Practical – tell you “how” and “why”
- Based on ”real world” experiences
- Guidelines for prioritisation
- Independent of test methodology and standard
- Works very well with Software Process Improvement

Amland Consulting

References

Koomen, Tim and Pol, Martin;
”Test Process Improvement:
“A Practical Step-By-Step Guide to Structured Testing”,
Addison-Wesley Pub Co 1999;
ISBN: 0201596245

Amland Consulting

Ståle Amland
Amland Consulting

Hulda Garborgsv. 2
N-4020 STAVANGER
Norway

e-mail:
stale@amland.no

visit:
www.amland.no

Martin Pol
Polteq IT Services B.V.

Amstelboulevard 154
1096 HL AMSTERDAM
The Netherlands

e-mail:
info@polteq.com

visit:
www.polteq.com

Amland
Consulting

QWE2002 Paper 2M

Mr. Geir Kjetil Hanssen, Mr. Hans Westerheim
(SINTEF, Norway) & Mr. Tor Stalhane
(NTNU, Norway)

Improving the Software Estimation Process

Key Points

- Estimation should be done from a Work Breakdown Structure (WBS).
- The changing environment should be handeled by risk assessment and control.
- Investment analysis is needed in order to handel uncertainties.

Presentation Abstract

Most of what has been published on software cost estimation has focused on finding the right formula for combing some size measure and some project factors and derive a cost figure. We have analysed in detail the estimation process of four projects in two Norwegian companies and were inspired to think anew on this question. In our opinin, the main reasons why cost estimation fails has two sources: Missing or changing information and lack of understading impotant relationships which makes it difficult to make sensible trade-offs.

About the Author

Mr. Geir Kjetil Hanssen is a Research Scientist in SINTEF Telecom and Informatics. He is a M.Sc. (Cand.Scient) from the Norwegian University of Science and Technology (1996). He works within the computer science area covering various aspects of information technology. System architecture is one of his focus areas, and he has experience in systems design and systems engineering methodology and processes. Other focus areas are transport telematics and process improvement where he assists several Norwegian software companies in improving their internal software development processes. . He has experience from the IT industry, both as a systems engineer and developer and as a project leader.

Mr. Hans Westerheim works as a scientific researcher in SINTEF Telecom and Informatics. He is educated in informatics from the Reginal College in Nord-Trondelag (www.hint.no) and the Norwegian University of Science and Technology (www.idi.ntnu.no). He is also educated as a economist in transportation from the Regional College in Molde (www.himolde.no). His focus area is process improvement, software quality and information architecture within the area of transportation. A special field of interest is the organisational and human issues when developing and implementing information systems. He has worked as a project manager in The Norwegian State Railways' computer departement, now known as Arrive (www.arrive.no). He has also worked as a project manager in a consultant company implementing CRM-systems.

Mr. Tor Stålhane: Born 1944 in Skien Norway. MSc from the the Technical University of Norway in 1969. Worked for SINTEF from 1969 to 1984 with compiler development. PhD in statistics 1988. Worked for SINTEF 1988 to 2000 on process imperovent and software safety and reliability. Professor of Computer Science at the Norwegian University of Science and Technology from 2000.

Quality Week Europe 2001

The Many Facets of Software Estimation

Professor Tor Stålhane Norwegian University of Technology and Science

MSc Hans Westerheim

MSc Geir Kjetil Hansen
SINTEF Telecom and Informatics

Focus

How to improve the software estimation process?

Traditionally, software estimation is about developing a cost figure by combining:

- a size measure like LOC or function points etc.
- some project factors

We claim that the search for the perfect formulae is not enough!

Background

PROFIT

A Norwegian SPI research program (1999-2001) for SME software companies

Focus on i.a. improving the software estimation process

The results are based on experiences from four projects at two companies

Our observations

We performed a PMA on the four projects:

"What are your problems with software estimation?"

The answers can be summed up as follows:

Missing information – we have the right formula but lack some vital bits of information

Changing information – the information was OK when we did the estimation, but what was true then is no longer so.

Lack of understanding – we know that something changes, but we do not know how it will affect the project costs.

The customer's representative has a low level of project understanding.

The good estimation process

Three kernel questions:

- What shall we do – based on the available specification?
- Who shall estimate the costs and who shall do the job?
- How are we going to do the job – choice of tools, methods and process?

Missing and changing info.

Who will do the job?

- Seniors have problems when they estimate the effort needed on behalf of inexperienced personnel
- May not know who will do the job at the time when the estimates are beeing developed
- Even if we do know this we have no guarantee that the planned personnel actually do the job

If we don't know who will do the job, how do we evaluate the need for training, courses etc.?

Missing and changing info. (cont.)

Which package?

- A WBS-based estimation gives good results
- Each work package must be on the same level
- The main problem is not that some work packages are underestimated but that they are forgotten!

In general terms, the main reasons for missing packages in the WBS structure are:

- Missing or incomplete application knowledge
- Misunderstanding customer requirements
- Incomplete knowledge of platform or operating environment

PROFIT
Process Improvement for IT Industry

Missing and chugging info. (cont.)

The time effect

- Long-lasting projects may experience that the platform – hardware or software – changes and impose new requirements on the project
- A change of personnel at the customer's side can cause the requirements to be interpreted in a new way

Other customer related changes to take into consideration:

- The change of customer contact person.
- The customer's requirements and expectations may change during the course of the project.

PROFIT
Process Improvement for IT Industry

Some solutions (1)

Estimation process improvement

- Software project estimation is a process, and like all other processes it can be improved
- We recommend PMA
  - Short lead time
  - Needs few resources
- PMA results are fairly subjective and must be utilized thereafter
  - Direct implementation in checklists, or:
  - Further investigation using GQM

PROFIT
Process Improvement
for IT Industry

Some solutions (2)

Other ways of improving the estimation process:

- Calibrate the estimators.
- Use the Delphi method (for instance the broadband version described by B. Boehm)
- Let each estimator present their estimates for the other estimators in this project for discussion
- Keep a record of all activities in an 'activity record' for later use.
- Learn from other companies

PROFIT
Process Improvement
for IT Industry

Some solutions (3)

Estimation = uncertainty = risk

The risks must be handled!

We recommend two techniques:

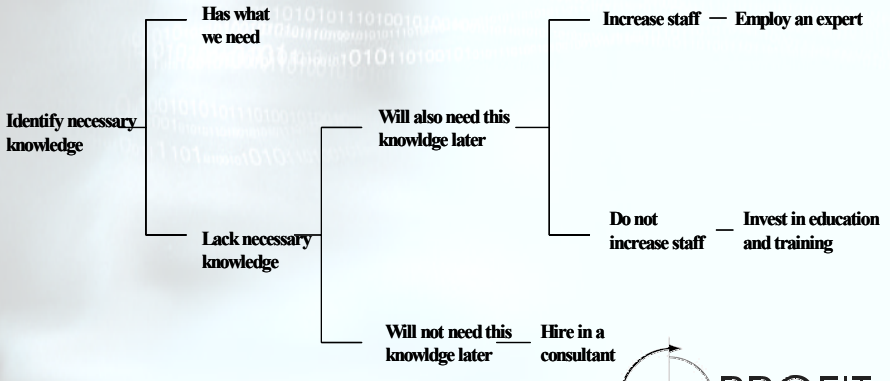
- Event trees
- Failure Mode Effect Analysis (FMEA)

PROFIT
Process Improvement
for IT Industry

Risk assessment - Event Trees

Shows main consequences connected in cause-consequence chains.

Visualise the decisions and their consequences, makings it possible to discuss alternative solutions and give feedback.



PROFIT
Process Improvement
for IT Industry

Risk assessment - Failure Mode Effect Analysis

FMEA identifies the tasks and indicate how they can fail and the related probability and consequences of a possible failure.

Task	Problem	Effect	P	C	Risk	Actions
Testing	Incomplete test plan	Errors in software to customer	5	10	50	Use requirements trace matrix
	Missing testing competence		1	10	10	Ignore
	Missing application competence		1	10	10	Ignore
Design	Missing customer liaison	Incomplete design	5	1	5	Ignore
	Missing application competence		1	5	5	Ignore
	Incomplete requirements		10	10	100	Build mock up prototype to elicit requirements Negotiate for a pre-project

PROFIT Process Improvement for IT Industry

Investment analysis (1)

Finding the 'right' estimate is only half the game. Just as important is how to use the estimate in the bidding process

In software estimation, the question of investment comes up in three contexts:

- Knowledge needed for the new project. Usually the available choices will be training, buy or hire.
- Methods and tools needed. This will also have influence on what we can use of our available knowledge.
- Our future position in the market. This question is particularly relevant when we need to quote a price that is below our most realistic estimate.

PROFIT Process Improvement for IT Industry

Investment analysis (2)

An investment analysis includes both expenditures and incomes. The latter is most difficult to see, but will usually be one of the following:

- Savings – enables us to use less resources on future projects.
- Higher productivity – enables us to do more with fewer resources. This can be realised as a better profit margin or as a lower price, which again will improve our position in the market with regard to our competitors.
- Better quality - enables us to raise our price. This will have the same effects as higher productivity mentioned in the previous bullet point.

PROFIT
Process Improvement for IT Industry

Investment analysis (3)

An example:

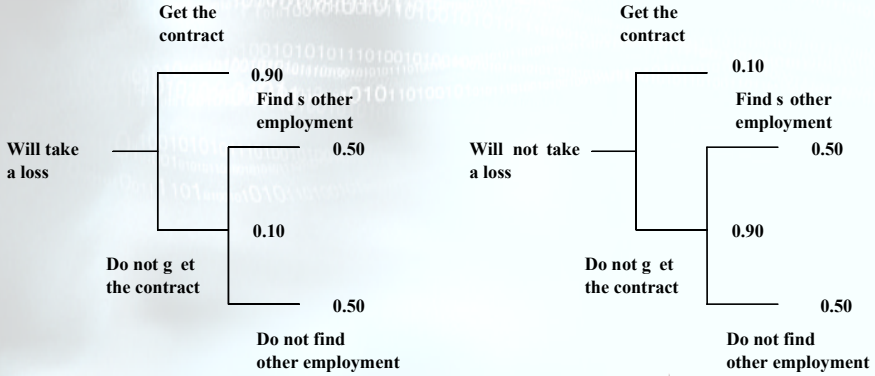
- PRO is the profit after n years
- r is the interest rate
- NPV (net present value) = $PRO/(1+r)^n$

- We are bidding on a contract and need to take a loss of NOK 100.000 in order to get it. We are 90% sure that we will get two new contracts as a result of this (one three years and one four years from now). Both contracts will give a profit of 100.000.
- Given an interest rate of 10%, these two contracts will have a NPV of $0.9 * 100\,000\,[1 / (1 + 0.10)^3 + (1 + 0.10)^4]$ which is approximately NOK 129 100.
- Taking the first contract is a sound investment.

PROFIT
Process Improvement for IT Industry

Investment analysis (4)

Investment analysis can be combined with event trees to give a good overview over the possibilities and at least some of their consequences. An example:

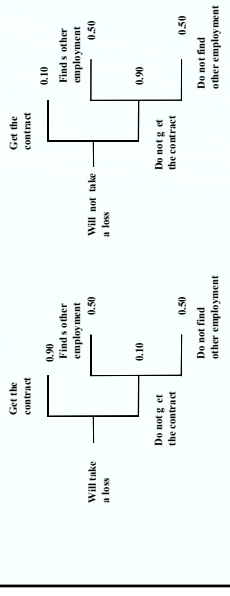


PROFIT
Process Improvement for IT Industry

Investment analysis (5)

Situation	Probability	Loss / Income kNOK	Future income - NPV - kNOK	Expected result kNOK
Get the contract	H = 0.90	-100	150	45.0
Do not get the contract – find other employment	L*M = 0.10*0.50	50	100	7.5
Do not get the contract – do not find other employment	L*M = 0.10*0.50	-320	-	-16.0
Expected result of investment				36.5

Situation	Probability	Loss / Income kNOK	Future income - NPV - kNOK	Expected result kNOK
Get the contract	L = 0.10	200	150	35.0
Do not get the contract – find other employment	H*M = 0.90*0.50	50	100	67.5
Do not get the contract – do not find other employment	H*M = 0.90*0.50	-320	-	-144.0
Expected result of investment				-41.5



Measurements and experiments

To estimate we need to know and understand which environment factors influences what and how

For SMEs the following is important

- Most of the projects that supplied the data used to estimate the effects of the COCOMO cost drivers stem from large military projects done in the US. Their world is rather different from the world as seen from SMEs.
- Software development has changed quite a lot during the last years. Incremental development and extreme programming are just two of the new methods that should lead us to think anew when it comes to environmental factors and their influence.
- Development of web applications differs from development of a traditional software system in many respects.

Experiments can help us to find the environment factors that influence the estimate

We can use methods like PMA and/or GQM to collect data that can be used to identify possible solutions

PROFIT
Process Improvement for IT Industry

The problem with good estimates

- Realistic estimates have a tendency to be higher than the unrealistic ones (the right price > "the right price")
- There is always somebody who is willing to give a lower bid, just to survive to the next pay-day
- In the .COM-business new companies are mushrooming. They have fresh money and are willing to take a risk.
- Strategy:
  - Make a realistic estimate.
  - Find out what the "right" price is.
  - Consider how to make up for the difference.

PROFIT
Process Improvement for IT Industry

Conclusions

The search for *the* formulae is interesting but it is not enough – it might not even be the most important part

Missing and changing information and lack of understanding are the main reasons for bad estimates

Estimation is all about finding the right information with the right quality

When we miss some of the information we need, we need a risk assessment

Contract competition may be a case of investment
It's wise to build a knowledge base to continuously improve the estimation process

Estimation is a process, as all other processes.
It can be improved!

Improving the Software Estimation Process

Professor Tor Stålhane (NTNU)
MSc Geir Kjetil Hanssen (SINTEF Telecom and Informatics)
MSc Hans Westerheim (SINTEF Telecom and Informatics)

Introduction

Most of what has been published on software cost estimation has focused on finding the right formula for combing some size measure – like lines of code or function points – and some project factors and derive a cost figure. Our work in the PROFIT program, where we have analysed in detail the estimation process of four projects in two Norwegian companies has, however, inspired us to think anew on this question. Our results are presented in this paper – not as the final word on software cost estimation, bus as the start of a new way of looking at this important problem.

The rest of this paper is organized as follows: Firs we describe a set of observations that we use to identify what we consider to be the important problems. After this, each of the main problems is discussed in some detail. Finally, we suggest a set of methods that in our opinion will remedy the most serious problems, before we discuss the curious problem that good estimates can be a mixed blessing.

Our observations

Our observations stem from trying to improve the estimation process for two Norwegian SMEs. Instead of starting by collecting large amounts of data and performing more or less well-founded statistical analyses, we decided to stay true to the TQM philosophy and ask the companies "What are your problems". We obtained answers to this all-important question by performing PMAs (Post Mortem Analysis) [1] on four of their projects and analysing the results.

When we summed up the answers and observations from the four aforementioned projects, we found that the estimate of a software project can be wrong for the following reasons:

- Missing information – we have the right formula but lack some vital bits of information
- Changing information – the information was OK when we did the estimation, but what was true then is no longer so.
- Lack of understanding – we know that something changes, but we do not know how it will affect the project costs.
- The customer's representative has a low level of project understanding.

These points are where we should direct our effort if we want to get realistic estimates. Thus, we need a process that handles missing and changing information in a sound manner, which mostly boils down to risk assessment and risk control.

A good estimation process should be built around three kernel questions:

- What shall we do – based on the available specification?
- Who shall estimate the costs and who shall do the job?
- How are we going to do the job – choice of tools, methods and process?

The rest of this paper will discuss each of these problem areas and what we should do about them

Missing and changing information

Who will do the job

The estimation of project costs is mostly done by senior personnel. We observed, however, that senior personnel and other personnel with much experience and knowledge had problems when estimating the effort needed when people with less experience should do the work. In addition, at the time of estimation the companies often had no idea who should do the work.

The problem is enlarged by the fact that even if we knew who would do the job when we do the estimation, we have no guaranty what so ever that this person will actually do the job. One of the common complaints from project leaders is that management move people rather freely between projects without consideration for plans, estimates and previous assumptions and agreements. The reason for this is often a crisis in another project, with a higher priority. In addition, a company cannot allocate personnel to each project they bid on since they will only get some of the projects.

Another problem stemming from the fact that we do not know who will do what, is the need for training and courses. Too often, a company has just handed a job to an inexperienced developer and told him to do it – one way or another. The belief that a developer will learn any programming language or any tool or method on the fly, and with no consequences for project costs and lead-time is flattering but unfortunately totally unfounded.

In some cases, taking away personnel from a project due to crises elsewhere can have quite dramatic consequences. This happens for instance when the person is one of the key developers in a project. The work that should have been done by this key person is not done which again will delay the whole project. In the best case, the rest of the project personnel will do nothing. If we are not that lucky they will do something – just to keep occupied. This will consume resources while producing nothing of value to the customer.

The problem is often due to management's dislike for prioritising. The main problem here seems to be that they do not like to give something lower priority, which is what they will have to do if they shall increase the priority of something else.

Which package

From what we have seen in the PROFIT program, the WBS based estimation is the best way to use all information and experience that a company has available. A good WBS structure – broken down to the right level of details – is important. We have observed that the main problem with this way of estimation is not that we underestimate several work packages but that several work packages are forgotten in the process. Missing activities leads to incomplete plans and a continuous need to improvise during the project.

In the dot com world, a project is no longer just software development. Many projects will now include one or more graphical design packages that are essential for the success of the product that we are developing.

From our exper4eince, these activities are among those that are often forgotten in the project's WBS structure – often due lacking to developer understanding of what web system design really is.

In general terms, the main reasons for missing packages in the WBS structure are:
- Missing or incomplete application knowledge.
- Misunderstanding customer requirements
- Incomplete knowledge of platform or operating environment.

A large part of any project cost is testing and error correction. If we miss here, we do not just underestimate the testing part of the project. Our experience has shown that we will also underestimate the duration of the project and that important costs like administration will also be underestimated.

The time effect

According to both our own observations and those of others – see for instance [8] - two common reasons for problems pertaining to customer relations are that:
- Some projects take a long time and during this time the platform – hardware or software – changes and impose new requirements on the project if the final product shall be useful.
- Change of personnel at the customer's site. Since a requirements specification is never cast in concrete, but always leaves room for interpretation, a change of personnel at the customer's side will cause the requirements to be interpreted in a new way.

Other possible changes that we need to take into consideration during estimation are:
- The change of customer contact person. This will usually carry an extra overhead due to bringing the new person up to speed, agreeing on ways to communicate and learning to know one another.
- The customer's requirements and expectations may change during the course of the project.

Lack of understanding

There is a large number of trade-offs that need to be done both during project estimation and during the actual project work. However, we often lack the understanding needed to make reasonable trade-offs. A typical example is the following:

In order to make good estimates, we need to break the job down into small work packages. However, we have observed that this fragmentation of the work has several bad effects:
- Many small packages – and thus too many details – may lead to overestimation.
- Many small activities often leads to less work being done on the design for each part, thus making geographical closeness between the developers a sine qua non for the project.
- Many small activities lead to many reviews, which create coordination problems. This will increase the project overhead.

During the PMAs and the GQM planning session [7], several of the developers also had other important experiences:
- When the level of details in a specification is low, the success of the project and its probability of finishing within estimated time and cost depends critically on the developers' domain knowledge.

- The estimator's experience in estimating and the amount of time available for the estimation are important. The importance of these factors increase with the complexity of the system to be estimated.
- The assumption of independence between the costs for the projects task is not only a simplification – it is a gross misunderstanding. This was shown quite clearly in one of the PMAs where the participants told us "because we had more errors than expected, the cost of testing and reviews increased. In addition, we got a longer project and thus, the administrative costs also increased". Clearly, correlations between the costs of the packages are important.

The identification of the problems is, however, just the first step in the cure. What is needed is a clear, quantitative understanding of the problem in order to do reasonable tradeoffs during estimation and after the project has started. We can gain important knowledge here by using our observations to build hypotheses that should be checked by collecting and analysing data.

Some solutions

The solutions to all the above problems will touch several important areas, which we will discuss in the sections below. First we will look at the estimation process and how it can be improved. Then we discuss the learning aspects and the importance of handling customer feedback. Finally we look into the two important activities risk control and investment analysis, before having a look at the need for measurements, case studies and experiments.

How to estimate

There are several possible estimation processes, ranging from the simplest – just asking, "How much will this project cost" – to rather advanced simulation models. One of the more common methods is to let the person responsible for the estimation or the person responsible for doing the job give the maximum, minimum and most probable value for the number of resources – for instance person days - needed. He can then use the PERT formula to estimate the resources needed and the standard deviation of each estimate. Both resources needed and the standard deviation for each activity is accumulated to give estimates for the whole project [2].

There are at least two problems with this approach:

- The people who estimate the maximum and minimum values tend to overestimate the minimum value and underestimate the maximum value. Instead of the usually assumed 99% interval, experiments have shown that this interval more often is approximately 60% [10]. This will have an impact on our estimation of the standard deviation and thus on the risk we take.
- The PERT formulas assume that all tasks are independent. If this is not the case, the estimated uncertainty of our cost estimates may be way off target. During one of the PMAs the effect of correlation was emphasised quite strongly: "Since activity Y needed much more work than expected, we also had to increase the resources for activities V, W and Z".

Estimation process improvement

Software project estimation and control is a process and like any other process it can be understood, documented, assessed and improved. The most efficient way to improve the estimation process in the short term is in our opinion to conduct a focused PMA on the estimation and project control processes to identify problems and propose solutions. The main

reason for this is that the PMA will give us a large amount of information in a short time and by using a rather small amount of resources. The method does, however, pose a problem: all information that we can obtain in this way is qualitative and subjective. During a PMA session, two types of problems will surface – each with their own solutions:

- Problems that can be solved by using checklists. The only thing we need to do is to collect the problem descriptions and insert them into a checklist. There are three types of information that goes into a checklist: things we must remember to do, things we must remember to avoid and risks we need to consider. In addition, we must make sure that the checklists are
    - Addressing topics that are valid for all projects or all projects of a certain type.
    - Used during estimation
    - Updated each time we have finished a project.
- Problems that need more understanding before we can solve them. These are the cases where the PMA is not enough. Thus, we need to use a GQM process [7] on each identified problem in order to get a clearer understanding of the relationships between the factors involved.

There are also several other ways to improve the estimation process. Some of the more important are:

- Calibrate the estimators. This is done by letting the estimators estimate the resources needed for several tasks where we know the right results. By analysing their estimates we can find their biases and standard deviations. These two numbers can then be used to correct their new estimates. .
- Use the Delphi method – either the real Delphi method as described by [3] or the broadband version described by B. Boehm in [2]. In both cases both the uncertainty *and* the dependencies between activities need to be included. The necessary formulas and a small example are included in appendix A.
- Let each estimator present their estimates for the other estimators in this project. Each estimate is discussed with focus on assumptions, possible risks and whether the estimate should be changed.
- We have observed that we often miss out activities during estimation and planning. Thus, it is important to keep a record of all activities – both those in the original plan and those that we had to add to the project later. This activity archive can be used during estimation and plan review by asking for each of the activities in the archive: “Could this be relevant here”. This will increase the chance of including all relevant activities in the project plan and estimates.

A simple improvement, at least in the short term, is to assign a reserve budget – also called contingency budget – to each project or project activity. In order to do this, we need to know how the differences between costs and estimates are distributed. For one company, we found that the deviations from the correct value for all estimates were normally distributed with mean (m) equal to … and a standard deviation (SD) equal to … Using these two values the company could improve their planning by giving each project a contingency budget of $(m + \alpha*SD)\%$, where the factor α reflected their willingness to accept the risk of cost overruns.

Whichever solution we choose, it is important that the estimates, their reasons, assumptions and risk are brought out into the open so that it is possible to review and discuss them.

Learning from others

Since it is a waste of resources to do all mistakes our self, most companies will profit greatly by learning from each other. There is no point in keeping methods to make good estimates a secret – the more who do it, the better. The companies' willingness to share experiences and information shows that they are well aware of this fact. In the PROFIT program, several companies meet regularly in order to report their own experiences, discuss problems and solutions pertaining to estimation and being informed by researches about what is going on in the estimation scene around the world, both in academia and in the industry. Some of the questions that have been discussed in this forum are:

- The problem with giving the customer an early, "back-of–the-envelope" estimate and how to control the expectations this creates.
- How do we estimate resources needed for non-software project work – for instance graphic design in web applications?
- Models are often used to express requirements and design – e.g. use cases and early class diagrams. How can we use this information to improve the estimates?

We have observed that testing is one of the activities that are often underestimated. In order to make a realistic estimate for this part of the project cost, we need to plan the testing process. In order to do this we need a realistic estimate of the system's error contents. For this reason, a company should build up tables of the relationship between error density and some important product and environment factors.

Customer feedback

A good estimation process is important but we also need to have a good process during the actual project. If the estimates are based on invalid or incomplete assumptions concerning the project process, they will not be all that useful. One of the problems we have observed is the handling of customer feedback. In this brave new world of dot com companies, many customers do not have software development - or even project organisation – competence. If we can get *one* contact person at the customer's site, who can make decisions and who understand what software development really is, we are in a good position to finish on time and within cost limits. On the other hand, if this is not the case we will waste large amounts of resources on:

- Long, pointless or inefficient meetings due to lack of common understanding and common goals.
- Sorting out and responding to large amounts of uncoordinated feedback.
- Doing part of the customers coordination work for him. This will for instance happen when the customer has little or no control over his own process.

Risk assessment

During estimation we have to make several decision. These decisions are concerned with the future project and their outcome is thus uncertain. Uncertainty always implies risks and these risks needs to be handled - one way or another. In our opinion, two techniques will be especially useful for risk assessment when we are estimating a development project – FMEA (Failure Mode Effect Analysis) and ET (Event Trees). We will present a short version of these two methods below.

Event Trees

Event trees [5] will be of great help when we want to get an overview of the alternatives available for a certain activity during estimation. The notation, originally taken from the area of system's safety and control, has several important features that make it useful during estimation:

- It shows all main consequences and how they are connected in cause-consequence chains.
- It makes all the decisions and their consequences available for all to see. This makes it possible to discuss alternative solutions and give feedback.

An example – shown in figure 1 – gives a good idea of how it can be used.

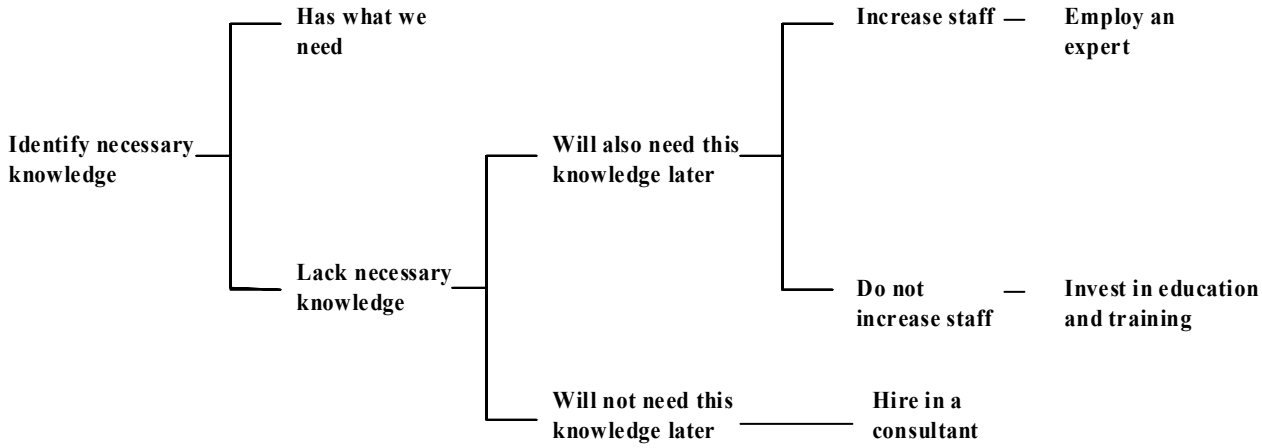


Figure 1 Event Tree for "Identify necessary knowledge"

From the diagram in figure 1 we see for instance that one of the important decisions we need to make is whether we will reuse a piece of knowledge later or not.

Failure Mode and Effect Analysis

Just like the event tree, the Failure Mode and Effect Analysis (FMEA) is also adapted from the field of safety analysis [4]. Its use in project estimation and control is the same as for safety – look at the units, components or what ever one by one, and try to understand how it can fail and the probability and consequences of a possible failure. The results are documented in an FMEA table. This table can have several layouts but the one below is fairly common.

Task	Problem	Effect	P	C	Risk	Actions

The columns marked Task, Problem, Effect and Actions are filled in as follows:

- Task – an activity in the project that we have estimated and planned – i.e. one that can be identified in the project plan.
- Problem – something bad that can happen, that will cause the cost or lead time to rise above the cost or lead time estimated for this task.

- Effect – the effect of the problem if it occurs. This is later used to find an appropriate value for C – see below. It is, however, important to describe it in some details here so that we can arrive at a common understanding of the problem's nature and its consequences.
- Actions. We have two types of actions – those needed if the problem occurs and those needed to prevent it from happening. The last category is most important during planning - where we still have enough freedom to change the process, the system's architecture or our choice of tools and methods.

The three columns marked P, C and Risk are important for grading and ranking the potential problems. They are filled in as follows:

- P – the probability of problem occurrence. It will usually not be possible to attach numerical values to the probability as such. We will use a probability score, for instance P = 1 for low probability, P = 5 for medium probability and P = 10 for high probability. It is possible to use all the values in between also, but it is unreasonable to require people to have the knowledge that enables them to separate for instance P = 5 from P = 6.
- C – the consequence of the problem, given that it has occurred. We can use the same scoring scheme here as for the probabilities – for instance C = 1 for small consequences, C = 5 for medium consequences and C = 10 for grave consequences.
- Risk – the risk, which is given as P * C.

When we have filled in the table and computed all the risk indices, we can rank the problems according to their risk value. The table below shows an example of how this might work out.

Task	Problem	Effect	P	C	Risk	Actions
Testing	Incomplete test plan	Errors in software to customer	5	10	50	Use requirements trace matrix
	Missing testing competence		1	10	10	Ignore
	Missing application competence		1	10	10	Ignore
Design	Missing customer liaison	Incomplete design	5	1	5	Ignore
	Missing application competence		1	5	5	Ignore
	Incomplete requirements		10	10	100	Build mock up prototype to elicit requirements Negotiate for a pre-project

The main question is now – how do we know which risks and events to consider during an FMEA or an ET? The answer is "From experiences". If we perform a PMA after each project we will build up a library of potential problems, each related to a special type of activity or to the project at large. For each problem in this library we can, for each activity, ask the important question: "Can this happen here?"

It is often difficult to get people to consider possible problems. They tend to react in the following way: "Yes, it is possible – No, it won't happen here". If we can point to the fact that it has already happened once, it is much easier to get people to consider the possibility that it could happen again.

Investment analysis

Up till now we have focused on how to get our estimate as close as possible to the "right" value. However, the estimate is only one of several pieces of information that is important for the bidding process. The view that either the estimate is correct and we make money or we underestimate and loose money is simplistic. Both winning and loosing the contract has several ramifications and not all of them are obvious from the start. In order to get a good overview of the available choices, the company should do an investment analysis. This analysis will enable the company to act in a rational way and will prevent it from rash decisions that it might later regret.

In software estimation, the question of investment comes up in three contexts:

- Knowledge needed for the new project. Usually the available choices will be training, buy or hire.
- Methods and tools needed. This will also have influence on what we can use of our available knowledge.
- Our future position in the market. This question is particularly relevant when we need to quote a price that is below our most realistic estimate.

It is, however, just not a question of making money or not on a contract. Just as important is the question "If we do not get this contract, what will the persons who we at the present have allocated to the project do instead". An example of how this question influences the decision is shown in the example in figure 2 below.

In order to do a sensible investment analysis, we need to have a clear understanding of the incomes, the expenditures and the rate of interest involved. However, while the expenditures are highly visible to everyone, some of the incomes often require a little more work before they are appreciated. Usually, the incomes or benefits can be categorized as one of the following:

- Savings – the investments enables us to use less resources on future projects.
- Higher productivity – the investment enables us to do more with fewer resources. This can be realized as a better profit margin or as a lower price, which again will improve our position in the market with regard to our competitors.
- Better quality - which will enable us to raise our price. This will have the same effects as higher productivity mentioned in the previous bullet point.

Let us use the question of new knowledge as an example. The sensible way to handle this problem is first to answer the following question: "Will we need this knowledge again later, in future projects?" If the answer is "No" we should bring in a consultant. If the answer is "Yes", then we should consider training or new employment. Training will be an investment for the company and should be handled as such, not as an extra cost for the project.

A small example shows how it can be used. If we get a profit denoted PRO after n years and the interest rate is r, then the current value – NPV or net present value – can be computed as $PRO / (1 + r)^n$. Let us assume that we are bidding on a project and find that we need to make a loss of NOK 100 000 in order to get the contract. We are almost sure – 90% say – that we will

get two new contracts as a result of this, one three years from now and another one four years from now. Both contracts will generate a profit of NOK 100 000. If we use a 10% interest rate, then these two contracts will have NPV of $0.9 * 100\,000\,[1 / (1 + 0.10)^3 + (1 + 0.10)^4]$ which is approximately NOK 129 100. Thus, taking the first contract at a loss might be a sound decision. Note, however, that if we think the chance of getting these two contracts is only fifty – fifty, the NPV is reduced to NOK 71 700 and we would take an expected loss of NOK 28 300.

Investment analysis can be combined with event trees to give a good overview over the possibilities and at least some of their consequences. An example for an estimation and bidding case is shown in the next diagram.

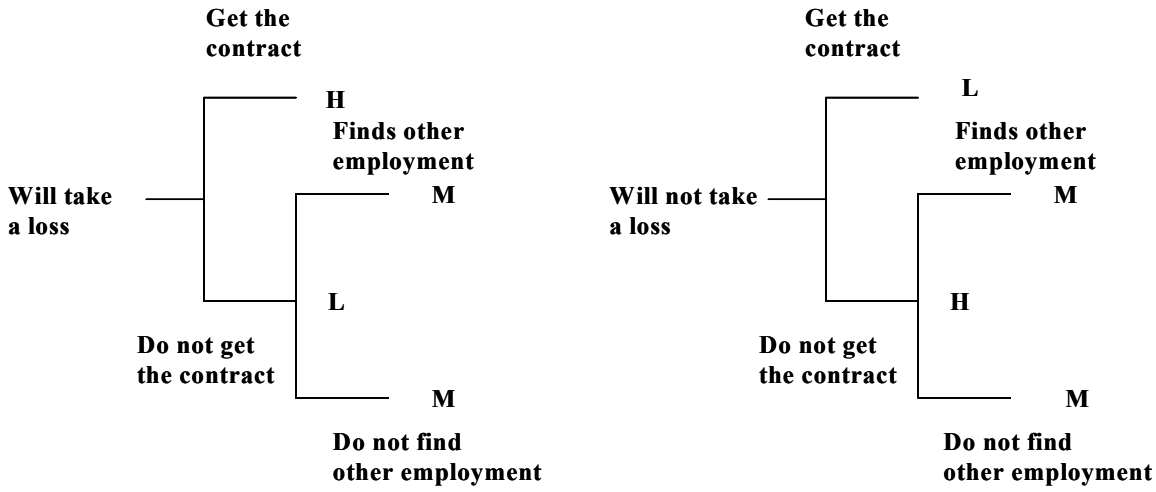


Figure 2 Event tree for contract bidding

In order to use this event tree as a basis for decisions, we need to know:

- The expected loss – investment - and the NPV of the future income if we accept a loss - bid to win - and get the project.
- The expected income generated by alternative activities for the personnel that we originally had planned to use in the proposed project.
- The losses we get if no alternative income can be generated for this personnel.
- The profit we get if we get the project.

We can sum up our estimates in the following tables:

Situation	Probability	Loss / Income kNOK	Future income - NPV - kNOK	Expected result kNOK
Get the contract	H = 0.90	-100	150	45.0
Do not get the contract – find other employment	L*M = 0.10*0.50	50	100	7.5
Do not get the contract – do not find other employment	L*M = 0.10*0.50	-320	-	-16.0
Expected result of investment				36.5

Table 1 Will take a loss – bid to win

Situation	Probability	Loss / Income kNOK	Future income - NPV - kNOK	Expected result kNOK
Get the contract	L = 0.10	200	150	35.0
Do not get the contract – find other employment	H*M = 0.90*0.50	50	100	67.5
Do not get the contract – do not find other employment	H*M = 0.90*0.50	-320	-	-144.0
Expected result of investment				-41.5

Table 2 Will not take a loss - realistic prizing

We see that with the values used in the example, the expected result if we are willing to make a loss on the first contract, is kNOK 36.5 while the realistic prizing will give us kNOK –42.0. The main reason for this result is the large loss stemming from having developers that are not working on any useful task. This example shows a rather common conclusion and is the main reason why companies take project that they know they will loose money on.

The event tree shows only the economic aspects of the situation. Many other factors must also be considered – we need for instance in alternative C to consider how soon we can get people back to work and the loss of expertise if we fire them.

The need for measurements and experiments

The question of what influences what and how strongly, is essential if we want to include the project's environment and the effect of their changes. Some work has been done here earlier – for instance the work on COCOMO I and II by Barry Boehm [2, 6] and by the FESMA organization [9] – but much remains to be done. For our purpose, the most important thing needed is the identification and estimation of relevant cost drivers for SMEs. The following points need to be considered:

- Most of the projects that supplied the data used to estimate the effects of the COCOMO cost drivers stem from large military projects done in the US. Their world is rather different from the world as seen from SMEs.
- The way we do software development has changed quite a lot during the last years. Methods like incremental development and extreme programming are just two of the new methods that should lead us to think anew when it comes to environmental factors and their influence.
- Development of web applications differs from development of a traditional software system in many respects. It would be surprising if these differences did not show in a sound estimation model.

Most experimentation in cost estimation will have to be conducted as quasi experiments. Quasi experiments are performed by collecting data from several projects and then using these data as we would otherwise have used results from real experiments. Many companies already have data that can help them to increase their understanding and thus to improve their estimation process. One important dataset comes out of using a WBS based estimation process. If the company follows up on the use of resources for each work package, they can see which packets are overestimated, underestimated and on target respectively. If the company uses a consistent set of activities for all projects, we can obtain answers to important questions like:

- Is there any type of packages that are worse or better than the others?

- What kind of packages are not included in the original WBS but had to be brought in later?

Based on these – and other pieces of information – it is possible to identify problems that occur often and some characteristics pertaining to these problems. This can be the first step in improving the estimation process. In order to identify possible solutions, several options are available. The final choice is a matter of time and cost:

- A focused PMA to get as much information as possible on why the problem occurs.
- RCA (Root Cause Analysis) to identify the problem's root causes [4].
- GQM to get a better understanding of the nature of the problem.

The estimations process can be improved based on the results of the chosen process. In any case, the effect of the improvement activities should be controlled by one or more metrics, preferably chosen based on a GQM process.

The problem with good estimates

One should think that when we get a good estimation process, everybody would be happy. Unfortunately, this is not the case. Realistic estimates have a tendency to be higher than the unrealistic ones and the company run a much larger chance of loosing when bidding on projects. There is always somebody who is willing to give a lower bid, just to survive to the next payday. Even if their customers are aware that the offer is unrealistic they accept it – at least if the developing company is able to carry the extra costs caused by cost overruns without going broke. Just as Gresham observed in the field of economy, bad estimates drive out the good ones. The customers' strategy seems to be to follow up on the contract and insist that it is fulfilled. All the customer will loose is some calendar time.

If the development company has a weak economy, the problem is different. In this case, the customer runs the risk of facing two possibilities – either they give the development company the extra money or the development company goes broke and the customer gets nothing.

The problem is increased by the fact that new companies are mushrooming. Starting with fresh money, they are always ready to take a risk. As a consequence of all this, their price does no reflect their productivity, but their willingness to take risks or their belief that they will get the extra money one way or another.

The strategy of the companies with a realistic cost estimation process should be to do the following:

- Make a realistic estimate.
- Find out what the "right" price is.
- Consider how they can make up for the difference.

Since the difference between the right price and the "right price" really is an investment, it should be treated like any other investment done by the company. Thus, the big questions are: "When and how will I get it back?" and "Is the interest rate better than alternative investments?" The project plan should nevertheless always be based on the best, most realistic estimates.

Conclusions

We have looked at the many aspects of software project cost estimation. In our opinion, the search for *the* formulae is interesting but it is not enough – it might not even be the most important part.

This paper starts out underlining the fact that missing and changing information and lack of understanding are the main reasons for bad estimates. Thus, estimation is all about finding the right information with the right quality. The source for this vital knowledge is in many cases experiences and the best source of such knowledge is in each project participant's mind. The best techniques for getting to this information are PMA and GQM, which let us tap the information from those that have the experience. But sometimes we just don't have the information we need to produce the estimates we need, there can be many causes for this. We might for instance never have done such a task before or key personnel might have left the company. In such cases we need to assess the risk. We identified Event Trees and FMEA as useful techniques in such cases.

Establishing the estimates themselves and possible assess the risks is, however, only half the cure, the information must be utilised. In a simple case it is natural just to consider a single project and present the realistic estimates to a possible customer but the reality is often more complex than that. In many cases a contract could bee seen on as an investment. A contract with a short-time loss can turn out to be a good investment over time. This must always be a natural part of the software estimation process.

Last, but certainly not least, companies need to build up an experience database for such things as often missed work packages, problems pertaining to estimation, dos and don’ts for the estimation process etc.

References

[1] T. Stålhane, T. Dingsøyr, G.K. Hanssen and N.B. Moe: PMA – An Assessment of two Approaches.

[2] B. Boehm, Software Engineering Economics. Prentice Hall Inc., 1981

[3] H.A. Linstone and M. Turoff: The Delphi Method – Techniques and Applications, Addison – Wesley Publishing Company, Inc., 1975

[4] D. Straker: A Toolbook for Quality Improvement and Problem Solving, Prentice Hall International (UK) Ltd., 1995

[5] N.G. Leveson, Safeware – System safety and computers. Addison – Wesley Publishing Company, 1995

[6] B. Boehm, Software Cost Estimation with COCOMO II, Prentice Hall, 2000.

[7] van Solingen, Rini, and Berghout, Egon. The Goal/Question/Metric Method, McGraw Hill, 1999

[8] M.R. Vigder and A.W. Kark, Software Cost Estimation and Control, National Research Council of Canada, NRC No. 37116.

[9] P. Foreslius Situation Analysis for Software Projects – the Key Input for SPI, Presented at EuroSPI’99, Pori, Finland, 25 – 27 October, 1999. Note that the presentation is not printed in the proceedings but is only available in power point files.

[10] M. Alpert and H. Raiffa, A progress report on the training of probability assessors in Judgement under uncertainty: heuristics and biases, H. Kahneman, P. Slovic and A. Tversky (editors). Cambridge University Press, Cambridge, UK, 1982.

Appendix A

We start out with a dataset {X}. If we denote the standard deviation of component *j* by σ_j we can write the variance of the sum of all components in the set as:

$$Var(\sum_j X_j) = \sum_j \sigma_j^2 + \sum_{j,k} \sigma_k \sigma_j \rho_{j,k}$$

When the correlation coefficient ρ is zero, we are back to the usual PERT formula. In the use of expert judgement, it is common practice to use the correlation coefficient value of 0.7 for strong correlation.

Let us now assume that we have used the Delphi method and arrived at the following estimates:

Id	Best estimate	Max estimate	Min estimate	Correlates with	Expected value	Standard deviation
A	7	12	5	C	7.5	1.2
B	5	7	3	-	5.0	0.7
C	9	12	8	A	9.3	0.7
D	3	7	1	-	3.3	1.0

By using the usual PERT formulas we get estimated effort to be 25.1 and its standard deviation is 1.8. If we, however, take the correlation between activities A and C into consideration, the standard deviation will increase to 2.0.

QWE2002 Paper 3M

Mr. Jose M. Xexeo, Dr. Ana Regina, Mr. Alvaro Alves, & Dr. Jose Blaschek
(COPPE/Universidade Federal do Rio de Janeiro, Brazil)

Information Systems as an Instrument for Quality Programs

Key Points

- Methodology.
- Case study.
- Information system and quality

Presentation Abstract

Our paper present a project developed by COPPE/Universidade Federal do Rio de Janeiro and the Funda o Bahiana de Cardiologia. It is an article about the development and the validation of a methodology that identifies non-functional requirements for an information system that guarantees the access to quality indicators related to the implementation of quality programs in health institutions.

About the Author

Jose A.M. Xex o is a candidate for a DSc degree at the Universidade Federal do Brasil. He is a telecommunications engineer with a MSc title on systems engineering, both obtained at the Institute Militar de Engenharia.

Information Systems as Instruments for Quality Programs

José A. M. Xexéo[1] , Ana Regina C. da Rocha[1] ,

Álvaro Rabelo Alves Júnior[2,3] J. R. Blaschek[4]

[1] System and Computer Engineering Program, Federal University of Rio de Janeiro, Brazil.

[2] Federal University of Bahia, Brazil

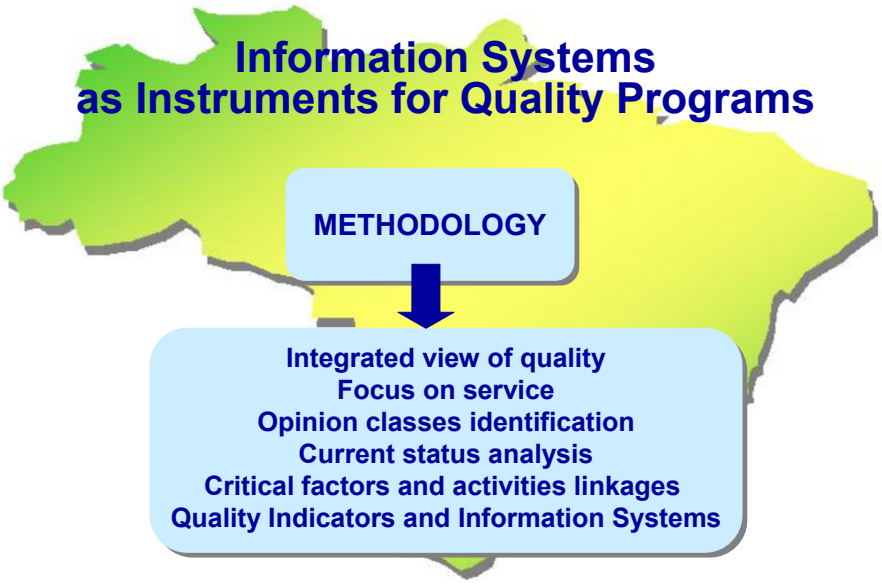
[3] Bahiana Cardiology Foundation, Bahia, Brazil

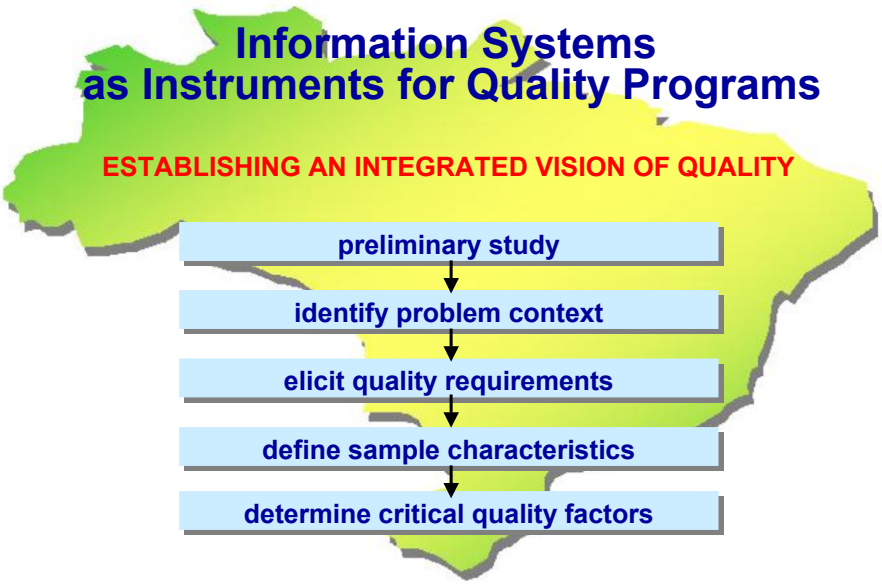
[4] Pontifical Catholic University of Rio de Janeiro, Brazil

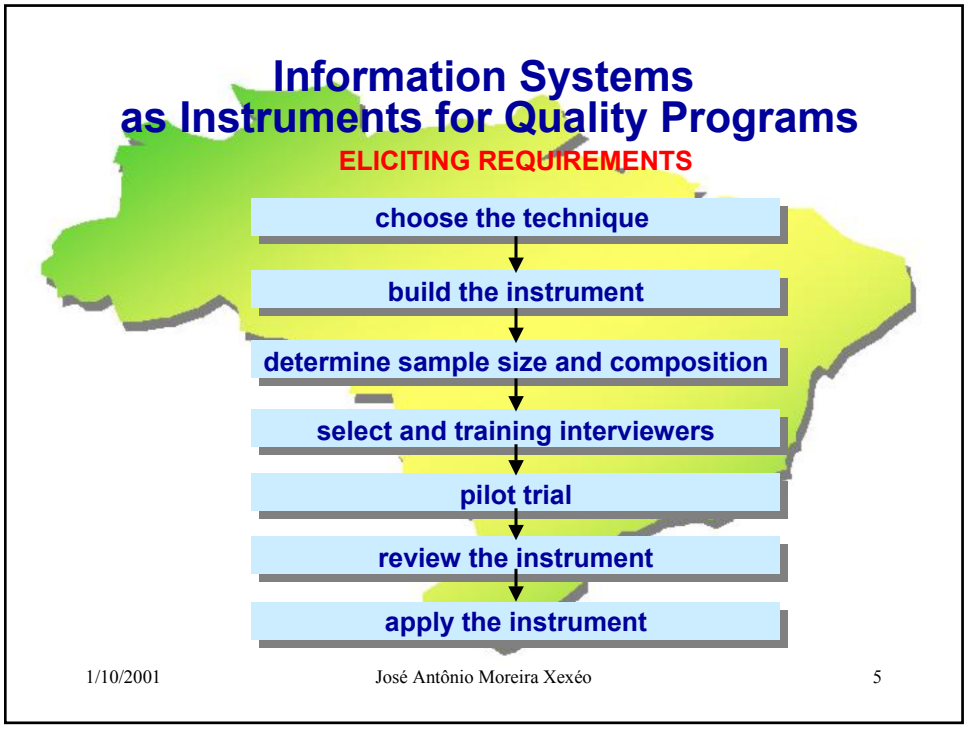
Information Systems as Instruments for Quality Programs

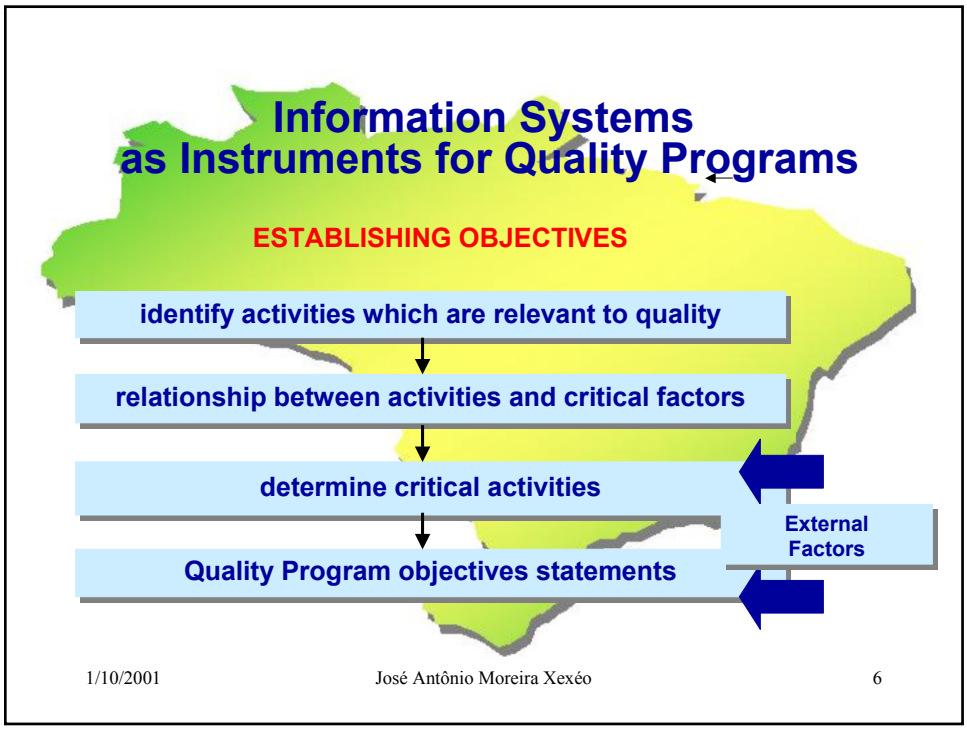
INFORMATION SYSTEMS → QUALITY PROGRAMS → HOSPITALS

Tamworth Hospital (Fahey, 1992)
Barnaby Hospital (Chesney, 1993)
Naval Medical Center (Appleman, 1995)
Chesshire Medical Center (Roland, 1997)
Overlook Hospital (Stratton, 1998)
Brigham and Women´s Hospital (Bates, 1999)
Haukeland University Hospital (Enchaug, 2000)


Information Systems
as Instruments for Quality Programs
METHODOLOGY
Integrated view of quality
Focus on service
Opinion classes identification
Current status analysis
Critical factors and activities linkages
Quality Indicators and Information Systems

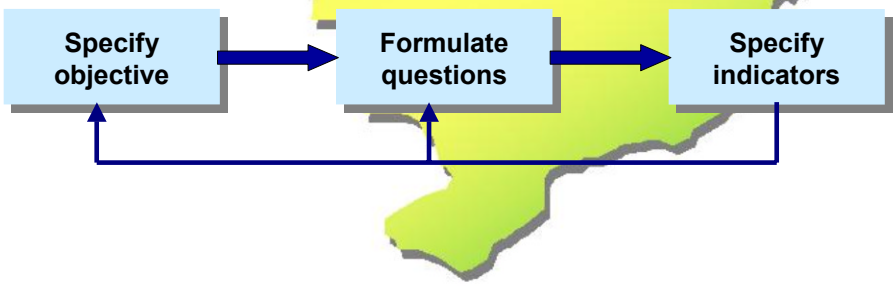

Information Systems
as Instruments for Quality Programs
ESTABLISHING AN INTEGRATED VISION OF QUALITY
preliminary study
identify problem context
elicit quality requirements
define sample characteristics
determine critical quality factors


Information Systems
as Instruments for Quality Programs
ELICITING REQUIREMENTS
choose the technique
build the instrument
determine sample size and composition
select and training interviewers
pilot trial
review the instrument
apply the instrument
1/10/2001
José Antônio Moreira Xexéo
5


Information Systems
as Instruments for Quality Programs
ESTABLISHING OBJECTIVES
identify activities which are relevant to quality
relationship between activities and critical factors
determine critical activities
External Factors
Quality Program objectives statements
1/10/2001
José Antônio Moreira Xexéo
6

Information Systems as Instruments for Quality Programs

THE GQM CYCLE USED TO IDENTIFY QUALITY INDICATORS



Information Systems as Instruments for Quality Programs

ESTABLISH REQUIREMENTS TO INFORMATION SYSTEMS

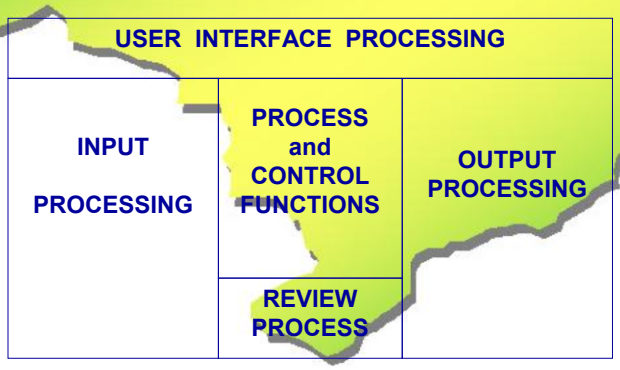
specifying identified indicators

- indicator name
- indicator acronym
- calculation formula
- required data
- data origin
- collection mode
- collection period
- aggregation period
- presentation
- goal

Information Systems as Instruments for Quality Programs

ESTABLISH REQUIREMENTS TO INFORMATION SYSTEMS

specifying Information System architecture



Information Systems as Instruments for Quality Programs

THE CASE STUDY

The survey

Importance: 384 interviews

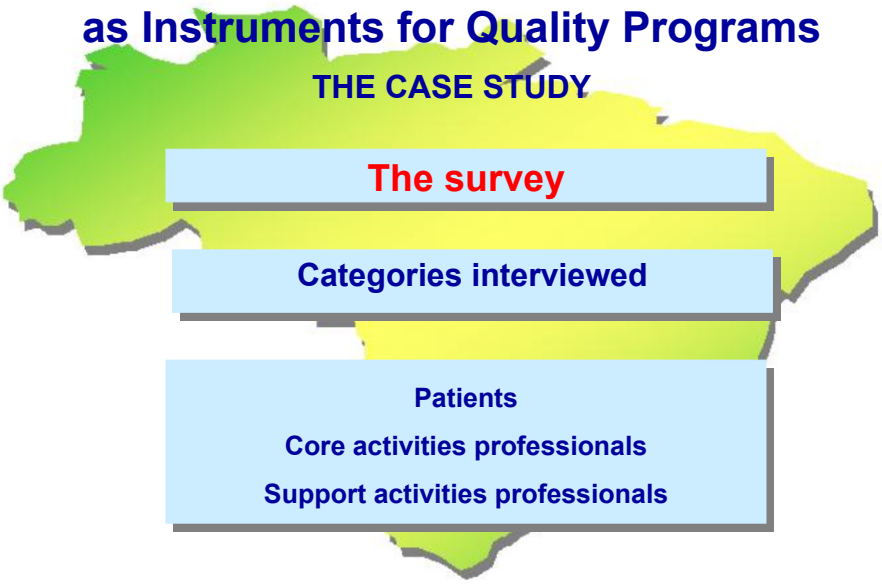
Perception: 384 interviews

47 quality factors

5 point likert scale

Information Systems as Instruments for Quality Programs

THE CASE STUDY



The survey

Categories interviewed

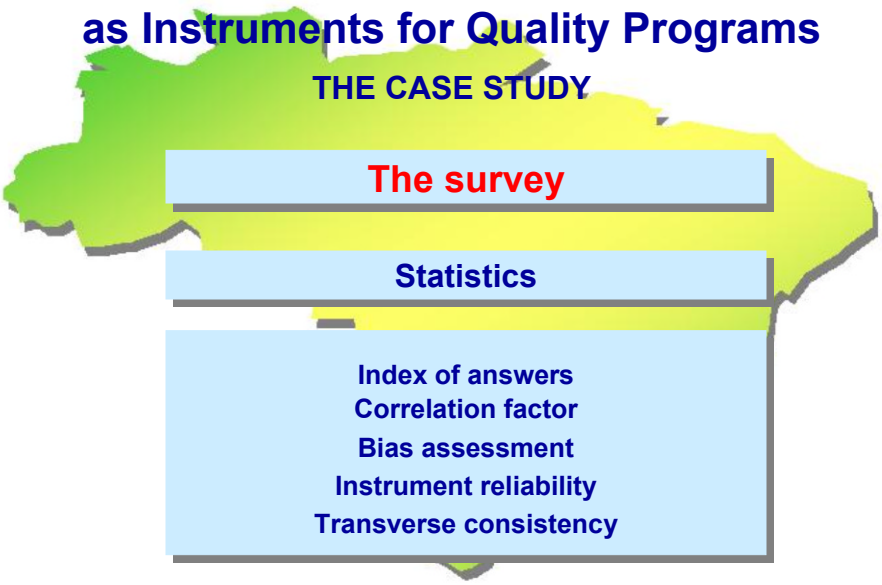
Patients

Core activities professionals

Support activities professionals

Information Systems as Instruments for Quality Programs

THE CASE STUDY



The survey

Statistics

Index of answers
Correlation factor
Bias assessment
Instrument reliability
Transverse consistency

Information Systems as Instruments for Quality Programs

THE CASE STUDY

Critical quality factors

Importance >= average (4.15)
Perception < average (4.09)

resources and access
promptness
24-hour availability

punctuality
facilities to set appointments

Information Systems as Instruments for Quality Programs

THE CASE STUDY

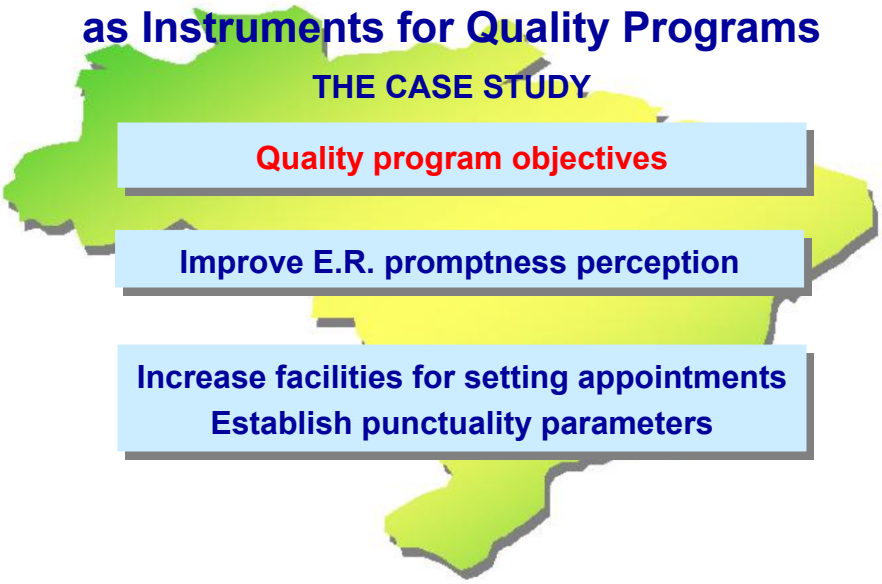
Critical activities

patient admission
nursing operations management
physician involvement in decision-making
in-service education and training

quality operational management
involving people with quality
schedule for continuity

Information Systems as Instruments for Quality Programs

THE CASE STUDY



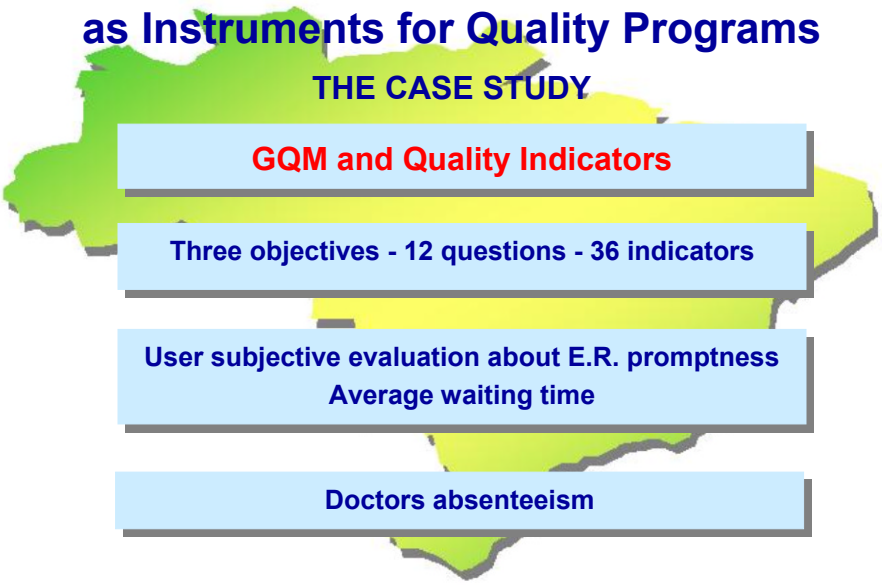
Quality program objectives

Improve E.R. promptness perception

Increase facilities for setting appointments
Establish punctuality parameters

Information Systems as Instruments for Quality Programs

THE CASE STUDY



GQM and Quality Indicators

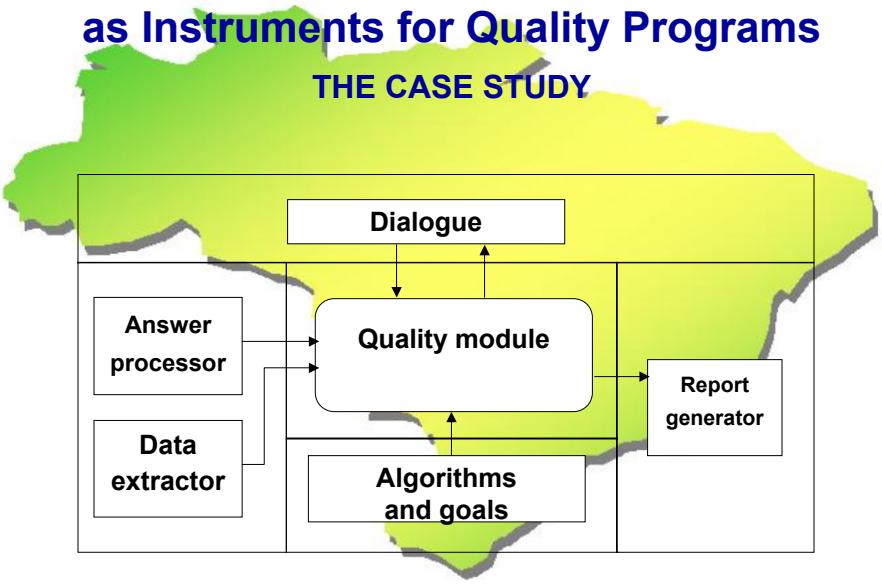
Three objectives - 12 questions - 36 indicators

User subjective evaluation about E.R. promptness
Average waiting time

Doctors absenteeism

Information Systems as Instruments for Quality Programs

THE CASE STUDY



Information Systems as Instruments for Quality Programs

THE CASE STUDY

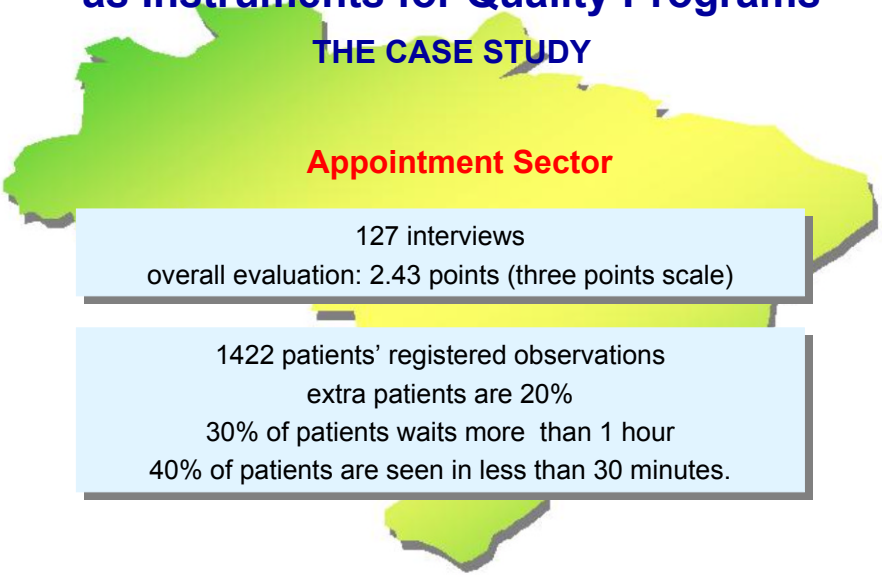
Emergency Care

434 interviews
overall evaluation: 2.1 points (three points scale)

494 patients' registered observations
50% of patients go back home
40% are referred to other institutions.
50% of patients waits more than 1 hour
29% of patients are seen in less than 30 minutes.

Information Systems as Instruments for Quality Programs

THE CASE STUDY



Appointment Sector

127 interviews
overall evaluation: 2.43 points (three points scale)

1422 patients' registered observations
extra patients are 20%
30% of patients waits more than 1 hour
40% of patients are seen in less than 30 minutes.

Information Systems as an Instrument for Quality Programs

José A. M. Xexéo[1], Ana Regina C. da Rocha[1], Álvaro Rabelo Alves Júnior[2,3] J. R. Blaschek[4]

[1] Programa de Engenharia de Sistemas e Computação COPPE, UFRJ – RJ, Rio de Janeiro, Brasil.
[2] Universidade Federal da Bahia, Salvador, Bahia, Brasil.
[3] Fundação Bahiana de Cardiologia, Salvador, Bahia, Brasil.
[4] Pontifícia Universidade Católica do Rio de Janeiro, RJ, Rio de Janeiro, Brasil

Abstract

In the current context of the economy there is a general consensus that the assimilation, dissemination and application of quality management techniques and the intensive use of information systems (IS) are critical factors to the success and the survival of organizations.

In this scenario there is no exception for institutions providing health services. Studies in the hospital segment also establish this interdependence between quality, information and competitivity. Corporate information analysis generate trends and establishes cause and effect relationships to support decision-making at the various organizational levels. Therefore, information is an agent of change, quality is information intensive and the IS play a crucial role in the process.

There are several reports on literature about attempts to implement quality programs in hospitals. Most of these experiences have adapted procedures related to methodologies well proven at manufacturing plants. This, nevertheless, seems to fail in meeting the specificity of the service area. A methodology is, therefore, required to guide managers in the implementation of quality programs in health organizations supported by information systems.

This article proposes a methodology and presents the case study that has validated it. The objective of proposed methodology is to guide the implementation of Information Systems supported Quality Programs in health care organizations. The validation was carried out at Fundação Bahiana de Cardiologia, Cardiology and Cardiovascular Surgery Unit, at the Federal University of Bahia.

(Key-Words: Information, Healthcare, Quality, Methodology, Indicators, Information systems)

1. INTRODUCTION

Fewer barriers to international trade and increased access to technology and to the markets have changed quality into a critical success factor. ISO 9000:2000 standard explicitly justifies quality management systems as instruments to achieve customer satisfaction. The ability to manage information as a strategic resource is also linked to the success of the business and leads organizations to invest in Information Systems development and updating.

Organization doing business in the health segment are not immune to these influences. The quality and the productivity of patient care are one of the main focus of investments (Richardson and Gurtner, 1999). When the subject is information, the importance is the same; the practice of medicine and information management are intertwined (Shortliffe, 1990, Berndt *et al* 2000 and Plummer, 2001).

The search for quality supported by Information Systems for Quality Programs is, then, a natural outcome of this environment. Bates (1999) argues that Information Systems are low cost alternatives to increase the quality of provided care. Thatcher and Oliver

(2001) refer to hospitals in which the Information Systems help to improve the quality of care provided to patients.

This article proposes a methodology, and presents the case study that has validated it. The objective of proposed methodology is to guide the implementation of Information Systems supported Quality Programs in health care organizations. The validation was carried out at Fundação Bahiana de Cardiologia, Cardiology and Cardiovascular Surgery Unit, at the Federal University of Bahia.

2. QUALITY PROGRAMS IN HOSPITALS

Several authors report experiences of quality program implementation at hospitals, among whom are Fahey (1992), *Tamworth Base Hospital*; Materna (1992), *Community Memorial Hospital*; Matherly (1992), *Blount Memorial Hospital*; Dasch (1995), *Naval Hospital Orlando*; Burney (1994), *Virginia Beach Surgery Center*; Chesney (1993), *Barnaby Hospital;* Appleman (1995), *Naval Medical Center*; Shaw (1995), *Strong Memorial Hospital*; Roland (1997), *Chesshire Medical Center*; Hart (1997), *Leicester General Hospital*; Valdivia e Crowe (1997), *Truman Memorial Hospital*; Stratton (1998), *Overlook Hospital*; Bates (1999), *Brigham and Women's Hospital;* Enchaug (2000), *Haukeland University Hospital*; Potter *et al* (1994); Huq (1996); Brashier (1996); Nabitz and Walburg (2000); Lim and Tang (2000) and Lim *et al* (1999).

The analysis of these experiences reveals the following facts:

a. identified objectives reflect the shift of focus occurring as years go by. Fahey (1992) and Matherly (1992) detect control and cost reduction objectives. Roland (1997), Stratton (1998), Hart (1997) and Valdivia and Crowe (1997) are concerned with the focus on process and client satisfaction. Bates (1999) refers to the use of Information Systems to measure and improve quality. Enchaug (2000) has furthered the establishment of doctor-patient partnerships.

b. among other, the following issues are deemed important to achieve success: top management involvement (Godfrey, 1992, Potter, 1994, Huq, 1996, Roland, 1997 and Sanders, 1997); faculties involvement (Godfrey, 1992, Huq, 1996 and Brashier, 1996); minimizing reaction to change (Materna, 1992, Matherly, 1992, Shaw, 1995 and Sanders, 1997); development of a culture of concern with the patient (Materna, 1992, Chesney, 1993, Burney, 1994, Huq, 1996); establishment of clear objectives (Shaw, 1995 and Brashier, 1996); making the program part of organizational culture (Brashier, 1996, Sanders, 1997 and Roland, 1997); training (Gopalakrishnan 1992, Dasch, 1995, Shaw, 1995, Brashier, 1996, Roland, 1997, Sanders, 1997 and Enchaug, 2000); measuring (Godfrey, 1992, Brashier, 1996, Huq, 1996 and Lim *et al*, 1999); survey user satisfaction (Brashier, 1996); conduct self-evaluations (Nabitz and Walburg, 2000).

c. the improvement targets represent top management vision. The other stakeholders do not make a significant contribution to establish desired quality standards. Every one is interviewed and stated expectations are recorded. Yet, the interpreter is the one who listens - the company. The organization decides about the user needs. This behavior impairs the make-up of a consistent vision of quality shared by the majority of employees and clients of the organizations.

d. there are few occurrences of IS as a component of an evaluation system. Bates (1999) relates the integration of the IS of a hospital to a system of quality measurement and improvement.

e. there is no emphasis on the characteristics of measures. Finison (1992) established that measures in the health area must reflect an explicit relationship between clients demands and key process variables.

f. The dimensions of service quality are not referenced for the implementation of quality programs. Where used, the view of services occurs in an evaluation context, using *SERVQUAL*[1] resources or adapting it for a specific application.

g. Most experiences have adapted procedures related to methodologies well proven at manufacturing plants. It reflects the influence from Feigenbaum, Crosby, Deming, Juran, Ishikawa and other researchers. This, nevertheless, seems to fail in meeting the specificity of the service area. Davis (1993) advances the hypothesis that faults in quality programs implemented this way are a consequence of the process itself. Sanders (1997) records that quality concepts and tools must not be transferred from manufacturing to health care.

h. Brashier (1996) has a proposal for the hospital area. He detects that TQM programs lacks focus and insists that patient satisfaction requires specific solutions. Lim and Tang (2000) propose a model for TQM implementation in health care, but limit the research to patients expectations, which results in the established quality view being biased.

3. THE METHODOLOGY

3.1 INTRODUCTION

The analysis of experiences of quality program implementation in health organizations have led to the identification of determinants for the definition of the methodology, namely: (i) build an integrated view of quality; (ii) focus on the service provision view; (iii) identify and analyze opinion classes in the surveyed universe; (iv) emphasize the analysis of the current status of the organization; (v) link quality factors to activities and establish priorities; (vi) establish indicators and use information systems for indicators follow-up; (vii) establish a comprehensive methodology, but foreseeing a methodic and gradual implementation; (viii) focus on success factors and barriers to program implementation.

3.2 METHODOLOGY PROCEDURES

The proposed methodology comprises stages interacting through feedback links which are characteristic of continuous improvement processes, with the ultimate purpose of specifying requirements to enable the Information Systems of organizations to support Quality Programs (figure 3.1).

➢ _____

[1] Instrument to evaluate service provisions, designed by Parasuraman *et al (1985)* , and since its dissemination has become a broadly used technology to manage and measure service quality.

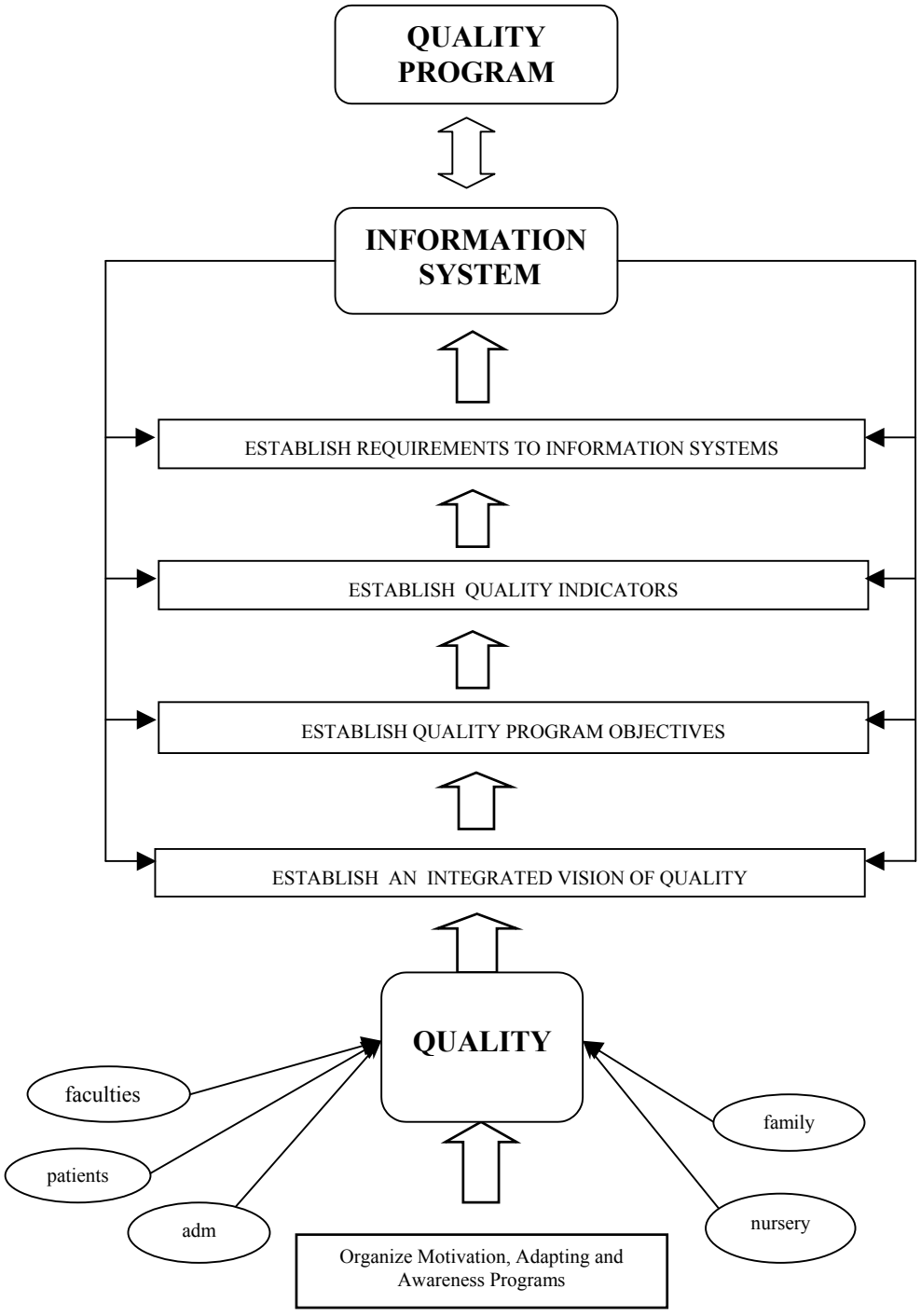


Figure 3.1 the methodology

PHASE 1. ORGANIZE MOTIVATION, ADAPTING AND AWARENESS PROGRAMS

The objective of this first phase is to obtain the commitment from the organization top management. There are several ways of achieving it; an example is the establishment of an 5S Program, because of its ability to mobilize and easy implementation.

PHASE 2. ESTABLISH AN INTEGRATED VISION OF QUALITY

This five stage phase, starts with a study of the environment, and its final product is the set of critical quality factors for the considered universe of individuals. The integration is ensured by the representativeness adopted in the process of eliciting requirements (figure 3.2).

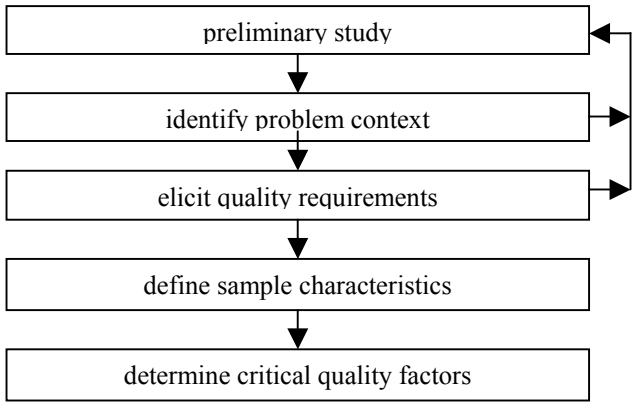


Figure 3.2 – establishing an integrated vision of quality

1st stage: preliminary study

The purpose of the preliminary study is to examine the universe in which the organization operates and similar situations, to enable the process to be developed to capitalize on positive factors and neutralize negative factors identified in this study.

2nd stage: identify problem context

The purpose is to study and define the context, with a clear and precise statement of the problem. The Software System Methodology (SSM) approach, proposed by Checkland (1981), the JAD (Joint Application Design) workshops and the non structured interviews are adequate techniques. The final choice of the technique depends on the size of the company, level of pre-exiting knowledge and of organization specific objectives.

3rd stage: elicit quality requirements

The objective of this stage is to survey the views of several user categories to integrate these categories in a solution that meets collective priority interests. A seven step process is proposed (figure 3.3).

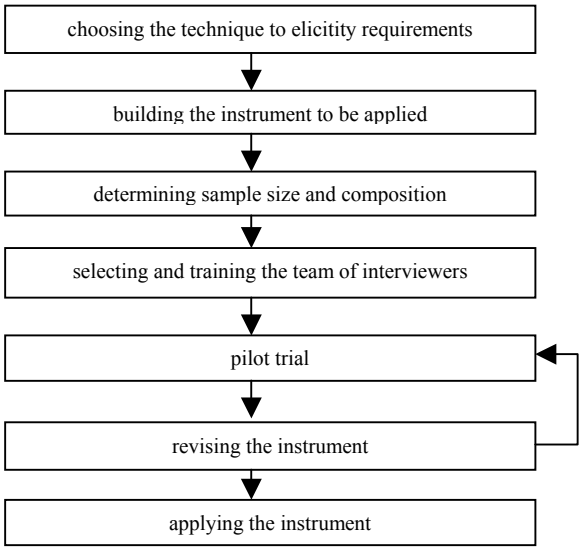


Figure 3.3– eliciting quality requirements

Step 1 – choosing the technique to elicit quality requirements.

In this step the technique to elicit requirements is selected. The desired scope of the survey determines the use of the interview technique with questionnaire. For the context of health institutions, the SERVQUAL was chosen, with two changes. Measurement of the importance instead of expectation and identification of quality based on perception and not by the difference between expectation and perception.

Step 2 – building the instrument to be applied.

In this step of the methodology the instrument to be used in the survey is built. The technique takes an instrument traditionally used for the approached problem and adapts it in the context of the new problem. The reference instrument is the *SERVQUAL*, proposed by Parasuraman *et al* (1988). The adaptation is carried out through interviews with representatives of the various categories of users.

Step 3 – determining sample size and composition

In this step, the total number of interviews to be carried out is determined and the way in which this number will be segmented across the categories selected to form the sample is defined. In this methodology, samples of all categories involved in the process should be interviewed.

Step 4 – selecting and training the team of interviewers

In this step interviewers are selected and trained to use the instrument. Training deals with care in the approach and with the several ways of explaining and exemplifying the content of used questionnaire.

Step 5 – pilot trial

The pilot trial simulates the eliciting of quality requirements working with a restricted group. The adequacy, clarity, simplicity and accuracy of the questions are verified, as well as the sufficiency of questionnaire scope and the level interviewer training.

Step 6 – revising the instrument

This step makes the changes indicated by previous step. These two steps are repeated in a feedback loop, until process and instrument are deemed to be ready to be applied.

Step 7 – applying the instrument

This step comprises the interviews. Two questionnaires are required to determine the critical quality factors. The first one, intended to rate the priorities of patients, core activity professionals and professionals working on support activities regarding evaluated functional quality factors. The second one, targets the evaluation of the perception of the same agents about these same factors, during or after service provision.

4ª stage: define sample characteristics.

The purpose of this stage is to define sample characteristics and the opinion of its various segments about quality. The final report must submit demographic data on interviewees, discuss the results of the processing of interviewee answers to the questionnaire and state the statistical tests carried out to evaluate the degree of results reliability.

5ª stage: determine critical quality factors.

In this stage critical quality factors are determined. Critical quality factors are those quality factors that simultaneously meet two requirements (i) have the average of the ratings, that have been assigned to them by interviewees in the importance questionnaire, equal or above the general average of the importance questionnaire (the general average is calculated adding the averages of each factor and dividing the result by the number of factors); (ii) have the average of ratings, assigned to them by the interviewees in the perception questionnaire, below the general average of the perception questionnaire.

PHASE 3. ESTABLISH QUALITY PROGRAM OBJECTIVES

In this stage, the relationships between quality factors and institution activities are determined. From these relationships and from the results of the previous phase, critical activities are identified and, consequently, the Quality Program objectives. This phase is influenced by factors like financial restrictions, organizational policies, etc (figure 3.4).

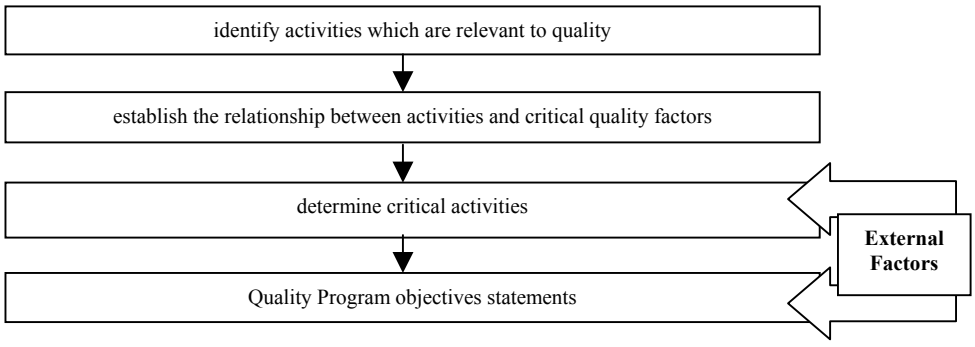


Figure 3.4 – establishing Quality Program objectives

1st stage: identify activities which are relevant to quality.

The organization activities which are significant to quality are identified through the following procedures: (i) based on the survey, on the literature, prepare an initial list of activities which are relevant to the quality programs implemented in institutions from the same segment; (ii) prepare and validate the final list of activities in interviews with professionals who are familiar with institution procedures and routines.

2nd stage: establish the relationship between activities and critical quality factors.

In this stage, an ordering matrix is constructed to establish a relationship between activities and critical quality factors through the following procedures: (i) choice of participants of this stage among organization staff; (ii) build two similar matrixes for each evaluator, the relationship matrix and the ordering matrix, with n lines and $m+1$ columns, where n is the quantity of quality factors and m the quantity of activities; (iii) match matrix lines and columns to quality factors and the activities, respectively, leaving the first column blank; (iv) in the first column enter the averages achieved by quality factors in the importance questionnaire; (v) through each evaluator, assign to each cell in the relationship matrix the value corresponding to the degree of relationship between the activity (column) and the corresponding quality factor (line); (vi) calculate the average of evaluations for each matrix cell; (vii) for each cell, multiply the value of found average by the value of the corresponding quality factor (same line); (viii) enter the result in the corresponding ordering matrix cells.

3rd stage: determine critical activities

This stage is conducted by quality project team and the purpose is to determine critical activities from the ordering matrix. The most immediate solution is simple ordering, based on the result of the sum of the cells in each column. Nevertheless, there are organizational factors which may point to another solution. Thus, the first step is to establish criteria for the selection of critical activities.

4a stage: establish Quality Program objectives

In this stage, Quality Program objectives are established from the critical quality factors and associated critical activities. This decision, nevertheless, depends also on Program scope, and on financial restrictions, human resources and even on the physical space available. These objectives, should, therefore, result from decisions taken at meetings with organization management in which, a formal document should be drafted – the Action Plan – which constitutes the base of the Quality Program to be implemented.

PHASE 4. ESTABLISH QUALITY INDICATORS

This phase to be carried out by project team, based on the Action Plan, has the purpose of defining a set of quality indicators capable of measuring up to what point Quality program objectives are met. These indicators should be used to follow-up the results as time goes by, through continuous improvement goals.

To identify indicators, the *Goal Question Metric* (GQM) is proposed. This is a software engineering method developed to support the identification of what is required to follow-up the achievement of objectives (Solingen and Berghout, 1999). GQM has three levels. A conceptual level, to specify objectives, the operating level, to design the statement of questions which will quantitatively characterize the objectives, and the quantitative level, in which the questions are associated to indicators, that is, metrics (figure 3.5).

The objective in this stage is to identify quality indicators, determining the name and the acronym of the indicator and calculation formula.

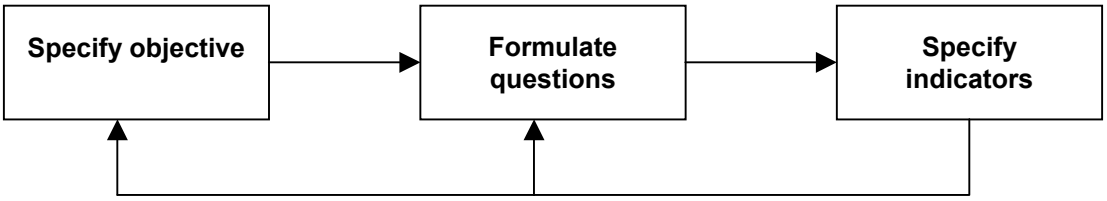


Figure 3.5 - The cycle used to identify quality indicators

PHASE 5. ESTABLISH REQUIREMENTS TO INFORMATION SYSTEMS

In this phase, the requirements enabling managers to follow-up the Quality Program launched by the Action Plan are identified.

These requirements comprise two documents, a set of specifications of quality indicators and a suggestion of an architecture for the information system.

So, there are three stages: establishing procedures for data collection to calculate quality indicators; completing the specification of quality indicators and specifying Information System architecture.

1[a] stage: establishing the procedures to collect data to calculate the indicators

Through the steps described below, this stage establishes the procedures to collect data to calculate the indicators. Indicators may be calculated based on data obtained from several sources: opinion polls, observation or organization information systems.

Step 1: classifying indicator data

In this stage, indicators are analyzed regarding the origin of data required to calculate them. To that end, it is necessary to (i) identify data required to calculate each indicator, (ii) determine where and how to get data; (iii) build a table containing, for each factor, its acronym, the data required to calculate it and data source.

Step 2: establishing procedures for data collection.

This step establishes procedures for data collection, taking into account the origins identified in the previous step.

For data collected by opinion polls, pertinent instruments must be prepared, sample have to be defined, interviewers ought be selected and trained and the pilot trial also must occur. The product will be the final tool ready to be applied.

For data obtained from observation a tool have to be prepared to record observed information and the person who will be in charge of observations must be selected.

For data obtained through organization information systems hose, which are already available in the systems, must be identified and also which developments must be introduced in the information systems to enable collecting new data will be identified.

2[nd] stage: specifying identified indicators.

The objective of this stage is to complete the specification of quality indicators, determining the specification standard items, which still have not been determined. The specification contains 10 items. Four of these items are determined in the previous stage (indicator name, indicator acronym, required data and calculation formula), data origin and collection mode are available in the results of the preceding stage, the four remaining ones

(collection period, aggregation period, presentation and goal) should be suggested by project team.

3rd stage: specifying Information System architecture.

In this stage, an Information System architecture is suggested. This methodology uses the representation scheme proposed by Hatley *et al* (2000), and suggested by Pressman (2001). It is a hierarchical model and accepts successive refinements. The figure 3.6 depicts the representation scheme for the architecture in question, used at all detailing levels. This standard represents the relationship between system elements, distributing them in five processing regions. Because of its simplicity this scheme was selected.

This methodology presents the architecture at its highest representation level, the context diagram, which establishes the information boundary between the system being implemented and the environment in which it will operate.

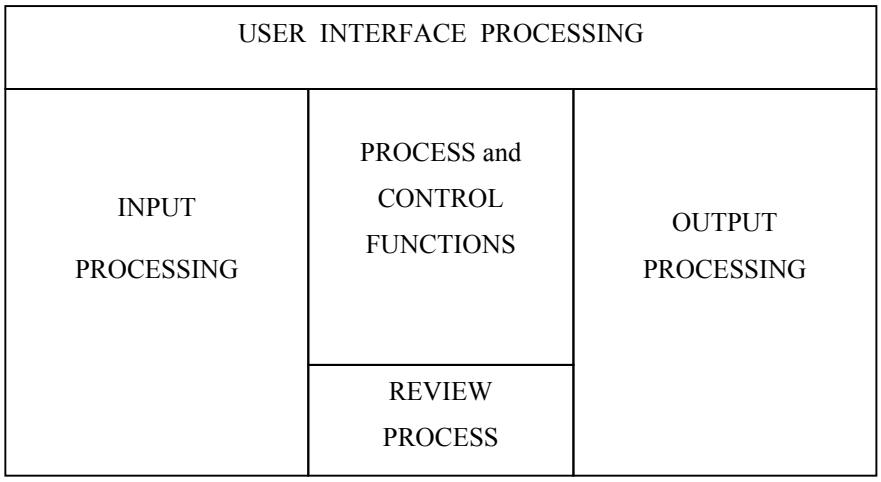


Figure 3. 6 - the architecture framework (PRESSMAN, 2001)

The central area, which communicates with all other areas, is the area destined to the representation of the system to be implemented (a rectangle with rounded corners). At the input processing area rectangles with rounded corners also should be drawn representing the external entities generating information to the system. For example, a translator of opinion polls results. Similarly, at the output processing area the external entities, which receive information from the system, are represented; for example a report generator. The area of user interface processing is appropriate to establish the type of mechanism for interaction with the user, for example, a dialogue sub-system. Review process, means, for example, optimizing the algorithms used to calculate indicators or a review of established goals.

4. THE CASE STUDY

Starting from the second phase, this section reports the case study carried out in the Cardiology and Cardiovascular Surgery Unit at Fundação Bahiana de Cardiologia (UCCV/FBC).

4.1 ESTABLISH AN INTEGRATED VISION OF QUALITY

The first stage in the second phase – preliminary study- is the analysis, presented in the introduction section, of cases reported in the literature about the implementation of quality programs by health organizations. As a result of the second-stage – identify problem context - the President of UCCV/FBC decided to conduct an opinion poll to check the degree of satisfaction with the services rendered by UCCV/FBC in which, all patients and professional categories would be heard, aiming at guiding an Action Plan for continuous service improvement and increased satisfaction of UCCV/FBC users. The third stage comprising seven steps is described below.

The first step chose *SERVQUAL* as the tool to be used to elicit quality requirements, introducing two changes: measure significance instead of expectation and identification of quality by perception.

The second step built the tool to be applied – adapting the SERVQUAL. The relevance of access and security aspects (Babakus and Mangold, 1992, Youssef *et al,* 1996, Conway and Willcocks, 1997, Andaleeb, 1998 and Van Der Bij, 1999) led to the segregation of these two dimensions from the five original ones to built seven dimensions. Initial questions were taken from the original Parasuraman (1988) proposal. Transposition to the hospital environment was guided by the works of Babakus and Mangold (1992), Vandamme and Leunis (1992), Youssef *et al* (1996), and by UCCV/FBC suggestions and structure. The questionnaire has more questions than the original one because of the complexity of the services rendered by UCCV/FBC, which include emergency care and out-patient and in-patient services. Several questions were broken down and eventually a total of 47 items were defined, as shown in the table 4.1 prepared for the importance questionnaire. Only positive worded statements were used, and a five points Likert scale was adopted. The first versions were tested with 16 people and some questions were reformulated.

In step three a sample of 384 interviews for each one of the questionnaires was calculated. The calculations for segmentation chose as reference the movement of patients during a week at the several UCCV/FBC sectors. To select the patients and professionals to be interviewed lots were drawn – without replacement.

In step four, interviewers were selected among students from the Universities existing in Salvador. Two training meetings were scheduled, in which an interview was simulated with each interviewer and each question was analyzed. Each interviewer's first interview was conducted under supervision.

The pilot trial adjusted the process in the fifth step. FBC Chairman and Head of UCCV was interviewed as well as the Head Physician of the Nuclear Medicine Department, FBC Head-Nurse, the Administrative Manager, three employees and twelve patients. The results were used in step six – revising the instrument – in which several questions were changed.

Step seven concludes stage three and deals with tool application. Questionnaires were applied in two different moments and in different groups. The full working hours schedule at UCCV/FBC were covered.

Regarding demographic data, in the fourth stage it was observed that the number of women (87,5%) exceeds the number of men, that the average women's age (50.9) is similar to men's age (50.2), and that 48% of the sample is within the 40 to 60 years range. The results of quality factors processing are presented in the table 4.1 that shows the values of global average for each of the questionnaires (importance = 4.15 and perception = 4.09)

Table 4.1 – quality factors

Factor	It is important that	Importance	Perception
P1		4.77	4.43
P2	all the physical facilities are permanently sanitized.	4.74	4.37
P3	number of professionals is sufficient to meet the demand.	4.57	3.76
P4	facilities and equipment is sufficient to meet the demand.	4.40	4.02
P5	the physical facilities are comfortable.	3.61	4.00
P6	doctor's offices and complementary exams rooms have sufficient resources to support patients' consults and exams.	4.12	4.10
P7	emergency facilities have sufficient resources and easy access to support service.	4.70	3.08
P8	in-patient facilities have the necessary resources to support service.	4.29	4.06
P9	signalization of the corridors helps patients' circulation.	3.62	3.52
P10	in-patient meal is conforming to best nutritional practices.	4.25	4.16
P11	the equipment is up-to-date.	3.70	4.44
P12	the equipment has high level of availability.	4.65	4.45
P13	professionals are satisfied with their working and payment conditions.	4.48	3.26
P14	professionals have opportunity for professional updating.	4.57	4.01
P15	support facilities for professionals are comfortable.	3.94	3.11
P16	parking facilities are compatible with the demand.	3.30	2.59
P17	hospital distributes folders that explain the services that are provided.	3.20	2.60
P18	compromises take upon with the patients are accomplished	4.46	4.31
P19	=services are provided at the right time.	4.21	3.81
P20	in-patient support services are timely provided.	4.15	4.14
P21	the consult period is sufficient to establish the patient status.	4.15	4.30
P22	services are carried out right the first time.	3.37	4.40
P23	staffs' attitude instills confidence in patients.	4.50	4.61
P24	information provided to patients is correct.	4.20	4.42
P25	records about patients are error free.	4.34	4.49
P26	billing is accurate.	3.56	4.14
P27	professionals give individual attention to patients.	3.73	2.95
P28	professionals always are courteous and respectful with patients.	4.49	4.49
P29	professionals have sincere interest in solving patients' problems.	4.37	4.45
P30	impatient requests are promptly satisfied.	4.10	4.26
P31	professionals from the institution guide patients to the sites were they will be assist.	3.44	4.17
P32	in-patient support services are promptly provided.	4.24	4.24
P33	the staff has the knowledge to answer patients' questions.	4.47	4.56
P34	emergency care is promptly.	4.69	3.47
P35	schedule for continuity is provided.	4.17	4.28
P36	professionals are easily accessible.	3.71	4.05
P37	facilities are easily accessible.	3.40	4.20
P38	records and relationship with patients are classified as confidential.	3.57	4.54
P39	equipments are securely operated.	4.57	4.57
P40	nuclear medicine and radiology facilities have controlled access.	4.13	4.61
P41	emergency has 24-hour availability.	4.71	3.77
P42	are enough facilities to set appointments with doctors and for complementary exams.	4.32	3.81
P43	patients are listened about their feeling concerning the approach given to their illness.	4.28	4.50
P44	patients are kept informed about their illness and its treatment.	4.27	4.54
P45	patients are informed about the procedures and complementary exams in schedule.	4.21	4.52
P46	patients are immediately informed about their exams results.	4.02	4.27
P47	treatment instructions are written	3.77	4.33
average		4.15	4.09

and the average for each quality factor, regarding importance and perception. Closing the fourth stage, the following statistical tests were performed:

In the first test – correlation between the sets of interviewers' answers to the questionnaires – it was assessed that the absolute value of the correlation coefficient for the pair (perception, difference between importance and perception) exceeds 0.7, and also exceeds the correlation coefficients of other pairs (importance, difference between importance and perception) and (perception, importance). Aside from the approximation of expectation by importance, the results were favorable.

In the second test - verification of factor response indexes – the importance questionnaire indexes were high (over 98%). This did not occur with the perception questionnaire. As not all of the factors applied to all groups, the interviewees did not answer questions about the services they did not use. Even though, just the P15 factor was left with a response index below 25%.

The objective of the third test - bias assessment – was carried out through the analysis of variance (one-factor experiment using Excel 2000, alpha = 0.05), with the first 100 interviews and with the last 100 interviews, for both questionnaires. The result showed that, in each questionnaire, the values of the variations within treatments and between treatments are of the same order for both sets of interviews, and no evidence of bias was detected.

The fourth test – tool reliability analysis – evaluated the level of internal consistency of the questionnaires using Cronbach alpha coefficient. The results for the complete questionnaires were satisfactory; the coefficient stayed above the lower limit (0.7) for both questionnaires. Nevertheless, the security and access dimension that had been segregated from the original ones did not stand, presenting low coefficients, maybe because they were formed by few factors.

The objective of the transverse consistency test – fifth test – was to check the consistency of results obtained when all interviewees were considered, with the results obtained when the several categories represented in the sample and the classes identified by clustering technique were individually considered. The initial procedure is to identify classes of opinion, using the clustering technique in the considered universe of the questionnaire being examined (Statgraphics Plus 5.0 supporting k-means method combined with the square Euclidean distance metric applied to six classes). The second step is to determine the composition of these classes relatively to three categories: patients, core activity professionals and professionals working on support activities. In the third step the ordering of quality factors obtained when considering the set of all interviewees is compared to the orderings obtained when considering separately the six identified classes, the three former categories, and UCCV/FBC's wards.

The results of the two questionnaires were analyzed separately. For the importance questionnaire, the existing differences do not avoid the uniformity of the composition of the classes relatively to the three interviewed categories (table 4.2) and of the existence of significant transversal consistency (table 4.3). Factors P1, P2, P41, P34 and P7, related to hygiene and the Emergency Room, are the most important factors to UCCV/FBC community of patients and professionals (table 4.3). In the perception questionnaire, it can be observed that the uniformity of classes is maintained at the same level (table 4.4), but the transversal consistency does not show the same intensity obtained in the importance questionnaire (table 4.5).

The results of these statistical tests show that there are no indications that would prevent the acceptance of questionnaires and carried out survey as statistically valid. There is also no indication of relevant discrepancies between the various categories and user classes justifying conciliation procedures.

Table 4.2 – classes' composition for the importance questionnaire

CLASSES of OPINION	CORE	SUPPORT	PATIENTS	TOTAL
C1	2 (7%)	0	48 (14%)	50 (13%)
C2	10 (35%)	21 (91%)	55 (17%)	86 (22%)
C3	0	0	15 (4%)	15 (4%)
C4	9 (31%)	1 (4.5%)	76 (23%)	86 (21%)
C5	3 (10%)	1 (4.5%)	69 (21%)	73 (19%)
C6	5 (17%)	0	69 (21%)	74 (20%)
TOTAL	29	23	331	384

note:
CORE: core activities professionals
SUPPORT: support activities professionals

Table 4.3 - pertinence of the ten quality factors with highest levels of importance of each identified class, UCCV/FBC sectors and general patients and professionals' categories relatively to the set of the ten quality factors identified for the UCCV/FBC as a whole is examined.

Quality factors	C1	C2	C3	C4	C5	C6	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(a)	(b)	(c)	Total of occurrences
P1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	17 (100%)
P2	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	16 (94%)
P41	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	16 (94%)
P34	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	17 (100%)
P7	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x		15 (88%)
P12	x	x	x		x	x		x	x	x	x	x	x	x	x	x	x	15 (88%)
P3					x	x		x	x	x	x	x	x		x			9 (53%)
P39	x	x			x			x	x	x	x	x		x	x		x	11 (65%)
P14			x		x	x			x	x	x		x	x	x			9 (53%)
P23	x	x										x	x	x	x	x	x	8 (47%)
Total of occurrences	8	8	7	3	9	8	5	8	9	9	9	9	9	8	10	7	7	

note:
(a) patients (b) core activities professionals (c) support activities professionals
(1) hemodynamics (2) arrhythmia (3) offices (4) ecographic (5) ergometric (6) nuclear medicine (7)emergency (8)in-patients

Table 4.4 – classes' composition for the perception questionnaire

classes of opinion	CORE	SUPPORT	PATIENTS	TOTAL
C1	16 (56%)	6 (29%)	23 (7%)	45 (12%)
C2	0	1 (4.5%)	87 (26%)	88 (23.2%)
C3	4 (14%)	6 (29%)	45 (14%)	55 (14%)
C4	0	2 (9%)	58 (17%)	60 (15%)
C5	5 (16%)	5 (24%)	77 (23%)	87 (22.8%)
C6	4 (14%)	1 (4.5%)	44 (13%)	49 (13%)
TOTAL	29	21	334	384

note:
CORE: core activities professionals
SUPPORT: support activities professionals

Table 4.5 - pertinence of the ten quality factors with highest levels of perception of each identified class, UCCV/FBC sectors and general patients and professionals' categories relatively to the set of the ten quality factors identified for the UCCV/FBC as a whole is examined.

Quality factors	C1	C2	C3	C4	C5	C6	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(a)	(b)	(c)	Total of occurrences
P40	x				x	x		x	x	x	x	x	x		x	x	x	12 (71%)
P23	x	x	x		x		x		x	x		x	x	x	x	x		12 (71%)
P39	x	x	x		x	x	x	x	x			x	x		x	x	x	13 (76%)
P33	x		x	x	x		x			x		x	x	x	x	x		11 (65%)
P38	x	x			x	x			x	x			x		x	x		9 (53%)
P45		x	x	x	x	x		x	x		x	x			x		x	11 (65%)
P44		x	x	x	x		x				x	x			x		x	9 (53%)
P25		x		x	x		x		x	x		x	x		x			9 (53%)
P28	x	x		x						x	x			x	x		x	8 (47%)
P43		x	x	x	x				x	x								6 (35%)
Total of occurrences	6	8	6	6	9	4	5	3	7	7	4	7	6	3	9	5	5	

note:
(a) patients (b) core activities professionals (c) support activities professionals
(1) hemodynamics (2) arrhythmia (3) offices (4) ecographic (5) ergometric (6)nuclear medicine (7)emergency (8)in-patients

The fifth stage - determining critical quality factors - concludes the second phase of the methodology. Critical quality factors are those factors with a importance (IMP) level equal to or above the 4.15 average and, simultaneously, a perception (PERC) level below the 4.09 average. An analysis of table 4.1 has led to the identification of ten factors with this property, which can be separated in four groups, namely:

Group of factors related to Emergency Care at UCCV/FBC:

- P7: emergency care facilities have sufficient resources and easy access to support service (IMP=4.70, PERC=3.08);
- P34: emergency care is promptly (IMP=4.69, PERC=3.47)
- P41: emergency has 24-hour availability (IMP=4.71, PERC=3.77)

Group of factors related to appointments and exams

- P19: services are provided at the right time (IMP=4.21, PERC=3.81)
- P42: is easy to set appointments with doctors and for complementary exams (IMP=4.32, PERC=3.81)

Group of factors related to infrastructure

- P3: number of professionals is sufficient to meet the demand (IMP=4.57, PERC=3.76)
- P4: facilities and equipment is sufficient to meet the demand (IMP=4.40, PERC=4.02)
- P8: impatient facilities have sufficient resources to support service (IMP=4.29, PERC=4.06)

Group of factors related to professional interests

- P13: professionals are satisfied with their working and payment conditions. (IMP=4.48, PERC=3.26)
- P14: professional have opportunities for professional updating (IMP=4.57, PERC=4.01)

The consistency of this result is checked examining the figure 4.1, which records the pertinence of critical quality factors of each sector relatively to the set of critical factors identified for the UCCV/FBC as a whole is examined.

UCCV/FBC sectors	Quality critical factors									
	P3	P4	P7	P8	P13	P14	P19	P34	P41	P42
Hemodynamics			x				x	x		
Arrhythmia	x		x		x		x	x	x	x
Offices	x	x	x		x		x	x	x	x
Ecographic	x	x	x	x	x	x	x	x	x	x
Ergometric	x		x		x		x	x		x
Nuclear Medicine	x	x	x	x		x	x	x	x	
Emergency	x	x	x		x	x				
In-patients	x				x	x		x		x

Figure 4.1 - pertinence of critical quality factors of each sector relatively to the set of critical factors identified for the UCCV/FBC as a whole is examined

4.2 ESTABLISH THE QUALITY PROGRAM OBJECTIVES

1st stage: identification of activities, which are relevant to the quality program

The initial list was built based on Lim et al (1999), Naveh (1998), Ahire (1996), Thiagarajan and Zairi (1998), Black (1996), Anderson, (1997), Capon (1995), Forza (1995), Camilleri and Callaghan (1998). The Head Physician of the Nuclear Medicine sector, FBC Head Nurse and the administrative Manager were interviewed to evaluate, add to or remove activities from this list and give it its final format. The final list comprises 24 activities as physician involvement in decision-making, patient education, schedule for continuity, advance medical research, survey of patients, benchmarking.

2nd stage: establishment of the relationship between activities and critical quality factors.

The Head Physician of the Nuclear Medicine sector, FBC Head Nurse and the administrative Manager were selected to establish this relationship following the steps defined in the methodology. To assign the degree of relationship between the quality factor (line) and the corresponding activity (column) the values one, three and nine were used to indicate a week, medium and strong relationship, respectively.

3rd stage: determining critical activities

The objective of this stage is to order the activities and select those that are critical. The results are presented in the tables 4.6, 4.7 and 4.8.

The criteria for ordering were established in a meeting with FBC Chairman and Head of UCCV. First adopted criteria was to respect the principle of limiting project action and focusing on the two first sets, emergency and appointments. The second criteria was identifying the first three activities which, are simultaneously among the ten first activities in the specific set and in the set in which all critical factors are considered.

This intersection of criteria identified A9, A10, A12 and A20 for emergency activities. For appointments activities A1, A11 and A14 were identified.

Table 4.6 – first ten ordered activities when all critical quality factors are considered

	Quality activity	pts
A14	quality operational management	401
A11	involving people with quality	382
A1	schedule for continuity	377
A4	clinical treatment activities	372
A20	patient admission	359
A5	benchmarking	359
A9	in-service education and training	356
A3	clinical diagnosis activities	353
A10	physician involvement in decision-making	353
A12	nursing operations management	350

Table 4.7 - first five ordered activities when only emergency related critical quality factors are considered

Quality activity		
A20	patient admission	129
A12	nursing operations management	129
A10	physician involvement in decision-making	129
A9	in-service education and training	129
A14	quality operational management	119

Table 4.8 - first five ordered activities when only appointment related critical quality factors are considered

	quality activity	pts
A1	schedule for continuity	86
A14	quality operational management	86
A11	involving people with quality	77
A15	patient scheduling	76
A20	patient admission	76

4th stage – establish quality program objectives.

Quality objectives are determined through the analysis of survey results and the design of an Action Plan.

This plan resulted from a series of meetings with the FBC Chairman and Head of UCCV, which, considering the impossibility of physical enlargement, budget limitations and urgent need to provide better service under specific aspects, decided to act in the reception activity of the Emergency Room and on the follow-up activities and quality operating management of the appointments and complementary exams sector.

The following objectives were defined:

- Improve perception of the readiness of emergency care service by improving patient reception;
- Increase facilities for setting of appointments and scheduling complementary exams;
- Investigate punctuality indexes in the conduction of appointments;

To achieve these objectives the following procedures were implemented at the Emergency Room:

- Receive emergency patients – in two work shifts - with a nursing student in charge of buffering the arrival to the E.R.;
- Collect with these students data related to the times of arrival, treatment and referral, besides final destination of each patient;
- Simultaneously, with another group of students, conduct a satisfaction survey, using a questionnaire, after treatment.

At the appointment scheduling service:

- Receive appointment, in two working shifts, with a group of students to record data associated to the punctuality of doctors and patients;
- Simultaneously, with another group of students, receive appointment and complementary exam patients, after care, at an office close to Drs. Offices to schedule requested complementary exams;
- 30 days after procedure implementation, conduct a satisfaction survey for 15 days, using a questionnaire.

4.3 ESTABLISH QUALITY INDICATORS

The purpose of this stage was to follow-up the Action Plan using specific indicators. Two sets of indicators were established, one for the Emergency Care sector and the other for Appointment and Complementary Exams sector. So, there are two stages.

1st stage: establish a set of quality indicators for the Emergency Care sector

This stage established a set of quality indicators for the Emergency Care sector to follow up the actions and results of the Action Plan. The Goal Question Metric (GQM) method was used.

As it is defined by GQM an objective was established: improve the rates of patient satisfaction perception relatively to the quality of Emergency Room (E.R) services provided.

This objective worked in the GQM cycle produced four questions and 17 indicators. Three of these questions and seven related indicators are pointed out as examples, excluded the calculation formulas.

Question 1: Which is the level of promptness?
Indicator 1.1: User subjective evaluation about promptness in the E.R
Indicator 1.2: Average waiting time in the E.R
Question 2: Which is the availability of services?
Indicator 2.1: Number of assistances provided in the E.R
Indicator 2.2: Distribution of patients relatively to time spent in the E.R
Indicator 2.3: Distribution of patient destination in the E.R
Question 3: What is the assistance capacity?
Indicator 3.1: Doctors absenteeism in the E.R
Indicator 3.2: Unavailability of critical equipment in the E.R

2nd stage: establish a set of quality indicators for the Appointments and Complementary Exams Sector

This stage established a set of quality indicators for the Appointments and Complementary Exams Sector with the same purpose of the first stage. The technique used was also the GQM. However, instead of one, two objectives were defined.

The first objective was to reduce awaiting time for appointments with physicians. The GQM cycle produced five questions and 11 indicators. Two of these questions and two related indicators are pointed out as examples, excluded the calculation formulas.

Question 4: Which are the characteristics of the demand met?
Indicator 4.1: Percentage of extra services
Question 5: Which are the patients' attributes?
Indicator 5.1: Percentage of late patients

The second objective was to make easier for patients to set appointments with physicians. Three questions and eight indicators were produced. Two questions and three related indicator are examples, excluded the calculation formulas.

Question 6: Which are the characteristics of the agenda?

Indicator 6.1: Number of cancellations *versus* number of confirmed appointments

Indicator 6.2: Number of confirmed appointments *versus* capacity of the agenda

Question 7: Which are the patients' attributes?

Indicator 7.1: Loyalty of first appointment patients.

4.4 ESTABLISH REQUIREMENTS TO INFORMATION SYSTEMS

In this phase, the requirements to Information Systems that enables managers to follow-up the Quality Program launched by the Action Plan were identified.

1st stage: establishing the procedures to collect data to calculate the indicators

This stage established the procedures to obtain the required data for indicators calculation through the following three steps.

Step 1 – classifying indicator data

The indicators established within the previous phase were analyzed in order to determine the data required to calculate them and to identify were or how this data should be collected. The figure 4.2 point out some examples.

Indicator	Data	Origin
User subjective evaluation about promptness in the E.R	Subjective evaluation	Opinion poll in the E.R.
Average waiting time in the E.R	Arrival time of each patient Admission time of each patient	Observation in the E.R.
Distribution of patients relatively to time spent in the E.R	Admission time of each patient Liberation time of each patient	Observation in the E.R
Distribution of patient destination in the E.R	Destination of each patient	Observation in the E.R
Percentage of extra services	Number of extra patients	Observation in the appointments sector
	Number of services provided	Information system
Percentage of late patients	Arrival time of each patient	Observation in the appointments sector
	Appointment time of each patient	Information system
Number of cancellations versus number of confirmed appointments	Number of cancellations Number of confirmed appointments	Information system

Figure 4. 2 – required data for calculation and origin of indicators

Step 2 – establishing procedures for data collection

The sample comprises only patients. Every one who came to the E.R or to the appointment and complementary exam sector was potentially part of the sample. The team of interviewers was trained in three meetings, in which the way to approach users and to complete questionnaires were discussed.

Two questionnaires were designed to be used in the opinion poll. One for the E.R. and the other for the Appointments Sector. These questionnaires were evaluated at a meeting with FBC Chairman and Head of UCCV. For being simpler, the forms in which observations would be recorded were designed by project team.

The E.R. questionnaire addressed items related to promptness, facilities access, 24-hour availability, resources availability, and overall value judgment of the services offered in the E.R. The questionnaire distributed to Appointment Sector patients' appraised items related to waiting time, appointments and also overall value judgment.

The following date was collected in the form used in the E.R.: date, name or identity card, times of arrival, service provided or patients desisting and final destination. The form used by the Appointments Sector recorded: date, name, number of service order and times of arrival, appointment and time patient was admitted for appointment.

Regarding the information system there are several data available to calculate indicators.

The pilot trails was conducted with the team of interviewers and led to small changes to questionnaires. Content evaluation was a simple process because there was previous experience on more than 900 interviews with questionnaires with a very similar content. Final approval was obtained at a meeting with FBC Chairman and Head of UCCV.

2nd stage specifying identified indicators

This stage completed the specification of quality indicators, determining the items of the specification standard (figure 4.3), which still had not been determined.

Three of these items were determined during the prior phase (the name of the indicator, its acronym and the calculation formula). Data origin and collect mode are results of the previous stage. The last three items (collect period, aggregation period and goal) must be suggested by the project team. The final product of the stage is a document containing the specifications of all indicators. Figure 4.3 is an example of specification standard.

3a stage: Specify the Information System architecture.

This stage has the purpose of specifying an architecture for the Information System. This methodology uses the framework proposed in HATLEY *et al* (2000) and suggested by PRESSMAN (2001).

The framework proposed has a top-down hierarchy, which uses flow context diagram at the highest level. The flow context diagram shows the system embedded in its environment (figure 4.4) and has the following elements:

- one architecture module representing the system under development – the quality module;
- four terminators (one for each external item with which the system must communicate) – the answer processor, the data extractor, the report generator and the dialogue system;
- all required flows.

<table>
<tr><td colspan="2">Indicator
Percentage of late patients</td><td>Acronym
Late_pat</td></tr>
<tr><td>Required Data:
Arrival time of each patient
Appointment time of each patient</td><td>Data origin:
Appointments sector
Appointments sector.</td><td>Collect mode:
Observation
Observation</td></tr>
<tr><td colspan="2">Collect period:
Daily</td><td>Aggregation period:
One week</td></tr>
<tr><td colspan="2">Presentation mode:
Graphic</td><td>Goal:
Maximum of 10%</td></tr>
<tr><td colspan="3">FORMULA
Late_pat = $\frac{\sum(\text{arrival time - appointment time})^*}{\text{number of observations}}$
*Only if arrival time > appointr</td></tr>
</table>

Figure 4.3 – indicator specification standard

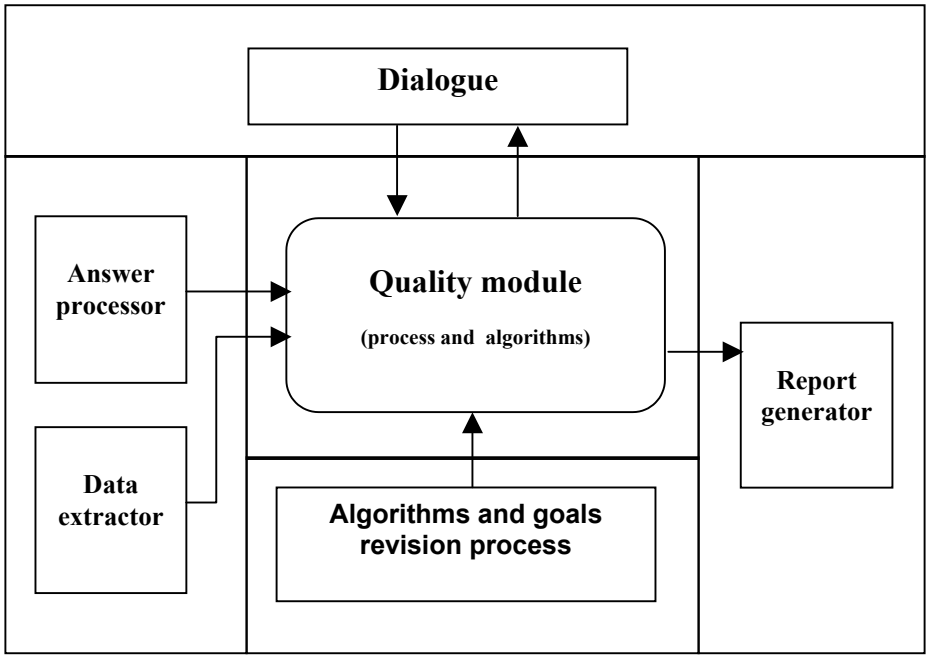


Figure 4.4 – the proposed architecture

4.5 ACTION PLAN - RESULTS

4.5.1 Emergency Care

The opinion poll was conducted with 434 interviews, resulting in a global evaluation, of which there was no previous record, of 2.1 points (three points scale), a classification corresponding to a "good" perception of the service.

The comparative table 4.9 shows a marked improvement of all surveyed factors with evidence that the way patients are received in the Emergency Room sector influences patient perception of service quality.

Table 4.9 - comparative data

Factor		Level	
		After the Action Plan	Before the Action Plan
P7a	Evaluation on access to E.R. facilities	4.3	3.08
P7b	Evaluation on resources availability	4.3	3.08
P34	Evaluation on promptness to provide service	4.1	3.47
P41	Evaluation on 24-hour availability	4.1	3.77

The analysis of data collected during the observation of 494 patients' receptions in the E.R revealed:

- The time patients remain in the E.R ranges around 90 minutes for 60% of patients. Just 15% stayed more than 3 hours.
- About 50% of patients go back home. Nevertheless, 40% of those who come to the E.R are referred to other institutions.
- For 50% of patients waiting time exceeds 1 hour and just 29% of patients are seen in less than 30 minutes.

4.5.2 Appointments

The opinion poll was carried out with 127 interviews, resulting in a global evaluation, for which there was no previous record, of 2.43 points (three points scale); a classification corresponding to a "upper good" perception of the service.

The comparative table 4.10 records a small improvement in almost all surveyed factors, confirming previous diagnosis. Both factors deemed to be critical showed the already expected behavior (P19 and P42). The factor that measures punctuality (P19) remained unchanged because punctuality was only the target of a survey and not an improvement target. The factor that measures perception of the facility to schedule appointments and exams (P42) has increased, possibly because of the decision to maintain, during the period of experience, an advanced station to schedule appointments and exams close to Doctors offices.

There are indications, therefore, that the action of scheduling appointments close to the offices has influenced the perception of service quality.

Table 4.10 - satisfaction level with the service provided by appointment scheduling system

	Factor	Level	
		After Action Plan	Before Action Plan
P3	The number of professionals is sufficient to meet the demand.	3.69	3.76
P6	The doctor's offices and complementary exams rooms have sufficient resources to support patients' consults and exams.	4.35	4.10
P12	The equipment has high level of availability.	4.69	4.45
P19	**The services are provided at the right time.**	**3.80**	**3.81**
P21	The consult period is sufficient to establish the patient status.	4.57	4.30
P23	The staffs' attitude instills confidence in patients.	4.76	4.61
P24	The information provided to patients is correct.	4.65	4.42
P28	The professionals always are courteous and respectful with patients.	4.63	4.49
P33	The staff has the knowledge to answer patients' questions.	4.67	4.56
P42	**It is easy to set appointments with doctors and for complementary exams.**	**4.07**	**3.80**
P44	The patients are kept informed about their illness and its treatment.	4.82	4.54
P45	The patients are informed about the procedures and complementary exams in schedule.	4.83	4.52
Mean		4.46	4.28

The analysis of data collected during the observation of 1422 patients' receptions in the appointment sector reveled:

- Just 40% of patients are seen with less than 30 minutes period of waiting and almost 30% of patients wait for more than 1 hour.
- The rate of unscheduled extra patients reaches a 20% average, which may account for the waiting time.

5. CONCLUSION

This paper discusses service quality, focusing on the heath services area. The priority assigned to service quality and user satisfaction can be easily evidenced in the new version of the ISO 9000:2000 standard which establishes that user measurement of user satisfaction is a requirement for an ISO 9000 certification.

Information and quality are seen as strategic organization management tools. The methodologies available to link these two sources of competitive advantages, particularly in health care organizations, do not cover the various aspects which should be emphasized to achieve user satisfaction.

This paper presents a methodology and its validation to identify the expectations and perception of quality by the social body of an organization with the purpose of defining the objectives of a Quality Program.

Methodology design has taken into account the following determining factors:

(1) build an integrated view of quality. This factor is ensured by the use of samples representing all social groups within examined universe and by the way in which critical quality factors are determined.

(2) value the view of service rendering. This factor is ensured by the use of techniques which focus on the behavior both of people and organizations at the times in which service is rendered.

(3) associate quality factors to organizational activities and establish priorities. This step is also quite clear in the methodology, and its purpose is to make priorities a consequence of an interaction between users and the organization.

(4) Identify indicators and establish follow-up procedures, emphasizing the use of organization Information Systems.

Another strength of the methodology is the study of context. In studied reports in the first stage of phase two the approach focus, methodological characteristics and detected objectives, main results, success factors, barriers and evaluation processes were identified.

It also became quite clear that the methodology supports decision making as it shows organization critical factors and links with organization activities.

The validation occurred through a case study developed in two phases: an opinion poll and an action plan. During the opinion poll 768 patients and health professionals were interviewed about the relative importance of 47 quality factors and about their perception of the quality level of the same quality factors. The survey results identified critical quality factors and critical activities that were the basis for the formulation of the action plan.

The action plan accomplished a satisfaction survey and an observation activity for the E.R and also for the appointment sector.

The satisfaction survey and the observation activity conducted 434 interviews and observed 494 patients' receptions in the E.R revealing a "good" overall evaluation for the perception of the service and a marked improvement of all surveyed factors with evidence that the way patients are received in the Emergency Room sector influences patient perception of service quality.

At the appointment sector the opinion poll was carried out with 127 interviews, resulting in a overall evaluation of an "upper good" perception of the service. The observation activity over 1422 patients detected a small improvement in almost all surveyed factors, and a positive increasing on the factor that measures perception of easiness to schedule appointments and exams, possibly because of the decision to maintain, during the period of experience, an advanced station to schedule appointments and exams close to Doctors offices.

Bibliography

Ahire, S. L., Golhar, D.Y., Waller, M. A., "Development and Validation of TQM Implementation Constructs", *Decision Sciences*, v. 27, n. 1, *pp.* 23-56, 1996.

Andaleeb, S.S., "Determinants of customer satisfaction with hospitals: a managerial model", *The International Journal of Health Care Quality Assurance* , v. 11, n. 6, 1998.

Anderson, J. C., "Clearing the Way for Physicians' Use of Clinical Information Systems", *Communications of the ACM* , v. 40, n. 8, *pp.* 83-90, August 1997.

Appleman, K., Large, K., "Navy Hospital Fights Diseases With a Quality Team", *Quality Progress*, *pp.* 47-49, April 1995.

Babakus, E. , Mangold, G., "Adapting the Servqual Scale to Hospital Services: An Empirical Investigation", *Health Service Research* , v. 26, n. 2, 1992.

Bates, D., Pappius, E., Kuperman, G. *et al*, "Using Information Systems to Measure and Improve Quality", *International Journal of Medical Informatics, v.* 53, *pp.* 115-124, 1999.

Berndt, D., Hu, P.J., Wey, C., "Introduction to the Minitrack: Databases, Data Warehousing and Data Mining in Health Care", In: *Proceedings of he 33rd Hawaii International Conference on System Sciences*, 2000.

Black, S. A., Porter, L.G., "Identification of the Critical Factors of TQM", *Decision Sciences* , v.27, n. 1, *pp.* 1-21, 1996.

Brashier, L. *et al*, "Implementation of TQM/CQI in the health-care industry: A comprehensive model", *Benchmarking for Quality Management & Technology* , v. 3, n. 2, 1996.

Burney, R., "TQM in a Surgery Center", *Quality Progress, pp.* 97-100, January 1994.

Camilleri, D., Callaghan, M., "Comparing public and private hospital care service quality", *International Journal of Heath Care Assurance , v.* 11, n. 4, 1998.

Capon, N., Kaye, M.M., Wood, M., "Measuring the success of a TQM programme", *International Journal of Quality and Reliability Management* , v.12, n.8, *pp.* 8-22, 1995.

Checkland, P.B., *Systems Thinking, Systems Practice.* Chichester, John Wiley and Sons, 1981.

Chesney, E., Dickenson, J., Lawrence, A., Talmanis, C., "Improving Health Care on a Tight Budget", *Quality Progress*, *pp.* 25-28, April 93.

Cortada, J., *TQM for IS Management.* New York, McGraw-Hill , 1995.

Dasch, Martha L., "Hospital Sets New Standard as Closure Approaches: Quality is Continuous", *Quality Progress*, *pp.* 45-480, October 1995.

Davis, G.B., Olson, M.H, *Management Information Systems: Conceptual Foundations, Structure and Development*, 2nd, Singapore, McGraw-Hill Book Company, 1985.

Donabedian, A. *Explorations in Quality Assessment and Monitoring.* Ann Arbor, Health Administration Press, 1980.

Enchaug, I.H., "Patient participation requires a change of attitude in health care", *International Journal of Health Care Quality Assurance* , v.13, n. 4., *pp.* 178-181, 2000.

Fahey, P. P., Ryan, S. "Quality Begins and Ends with Data", *Quality Progress*, *pp.* 75-79, April 1992.

Feigenbaum, A.V., *Total Quality Control,* NY, McGraw Hill, 1983.

Finison, L. J., "What are Good Health Care Measurements?" *Quality Progress*, *pp.* 41-42, April 1992.

Forza, C., "The impact of information systems on quality performance: an empirical study", *International Journal of Operations & Production Management* , v. 15, n. 6, *pp.* 69-83, 1995.

Godfrey, A. B., Berwick, D. M., Roessner, J., "Can Quality Management Really Work in Health Care?", *Quality Progress*, *pp.* 23-27, April 1992.

Gopalakrishnan, K. N., McIntyre, B. E., "Hurdles to Quality Health Care", *Quality Progress*, *pp.* 93-95, April 1992.

Gronroos, C., "Service Quality: The Six Criteria of Good Perceived Service Quality", *Review of Business,* St. John's University, v. 9, n. 3, 1988.

Hart, M., "Monitoring quality in the British health service – a case study and a theoretical critique", *International Journal of Health Care Quality Assurance , v.* 10, n. 7, 1997.

Hatley, D., Hruschka, P., Pirbhai, I., *Process for System Architecture and Requirements Engineering,* New York, Dorset House, 2000.

Huq, Z., "A TQM evaluation framework for hospitals", *International Journal of Quality an Reliability Management , v.* 13, n. 6, *pp.* 59-76, 1996.

Juran, J.M., *Quality Control Handbook.* NY, McGraw-Hill, 1988.

Lim, P. C., Tang, N. K. H., Jackson, P. M., "An innovative framework for health care performance measurement", *Managing Service Quality , v.* 9, n. 6, 1999.

Lim, P.C., Tang, N.K.H., "The development of a model for total quality healthcare", *Managing Service Quality* , v.10, n.2, *pp.* 103-111, 2000.

Materna, S., Rothe, K., "A Canadian Hospital Implements Continuous Quality Improvement", *Quality Progress*, *pp.* 89-91, April 1992.

Matherly, L. L., Lasater, H. A., "Implementing TQM in a Hospital", *Quality Progress, pp.* 81-84, 1992.

Nabitz, U.W., Walgurg, J.A., "Addicted to qualify – winning the Dutch Quality Award based on the EFQM Model, *International Journal of Health Care Quality Assurance v.*13, n.6, *pp.*259-265, 2000.

Naveh, E., Erez, M., Zonnenshain, A., "Developing a TQM Implementation Model", *Quality Progress, pp.* 55-59, February 1998.

Parasuraman, A., Zeithaml, V., E Berry, L., SERVQUAL: "A Multiple-Item Scale for Measuring Consumer Perceptions of Service Quality", *Journal of Retailing,* Spring, *pp.* 12-40, 1988.

Plummer, A.A., "Information Systems Methodology for Building Theory in Health Informatics: The Argument for as Structured Approach to Case Study Research", *Proceedings of the 34th Hawaii International Conference on System Sciences,* Hawaii, 2001.

Potter, C., Morgan, P., Thompson, A., "Continuous Quality Improvement in an Acute Hospital: A Report of an Action Research Project in Three Hospital Departments", *International Journal of Health Care Quality Assurance* , v. 7, n. 1, 1994.

Pressman, R. S., *Software Engineering: A Practioner's Approach*, 5th ed, New York, McGraw-Hill Book Companies, Inc., 2001.

Richardson, M.L., Gurtner, W.H., "Contemporary organizational srategies for enhancing value in health care", *International Journal of Health Care Quality* Assurance, v. 12, n. 5, pp. 183-189, 1999.

Roland, C., *et al.,* "Insights into Improving Organization Performance", *Qualiy Progress, pp.* 82-85, March 1997.

Sanders, Nada R., "Health Care Organizations Can Learn From the Experiences of Others", *Quality Progress*, *pp.* 47-49, February 97.

Shaw, D., *et al*., "Learning from Mistakes", *Quality Progress*, *pp.* 45-48, June 1995.

Shortliffe, E.H., Perreault, L.E.. In: Shortliffe, E.H., Perreault, L.E. (eds), *Medical Informatics: Computer Applications in Health Care.* New York, Addison-Wesley, 1990.

Solingen, R., Berghout, E., *The Goal Question Metric Method,* London, McGraw-Hill Book Company, 1999.

Stratton, B., Overlook Hospital Emergency Department: Meeting the Competition with Quality", *Quality Progress, pp.* 41-43, October 1998.

Thatcher, M., Oliver,J.R., "The Impact of Information Technology on Quality Improvement, Productivity and Profits: An Analytical Model of a Monopolist",

Proceedings of the 34th Hawaii International Conference on System Sciences, Hawaii, 2001.

Thiagarajan, T., Zairi, M., "An empirical analysis of critical factors of TQM", Benchmarking for Quality Management & Technology, v 5 nr 4, *pp.* 291-303, 1998.

Valdivia, M.T.R., Crowe, T.J., "Achieving hospital operating objectives in the light of patient preferences", *International Journal of Heath Care Assurance , v.* 10, n. 5, *pp.* 208-212, 1997.

Van Der Bij, J.D., Vissers, J.M.H., "Monitoring health care process: a framework for performance indicators", *International Journal of Heath Care Assurance , v.* 12, n. 5, *pp.* 214-221, 1999.

Vandamme, R., Leunis, J., "Development of a Multiple-item Scale for Measuring Hospital Service Quality", *International Journal of Service Industry Management , v.* 4, n. 3, *pp.* 30-49, 1993.

Wiederhold, G. *et al*, "Hospital Information Systems", In: Shortliffe, E.H., Perreault, L.E (eds), chapter 7, *Medical Informatics: Computer Applications in Health Care,* Addison-Wesley, 1990.

Youssef, F.N., Nel, D., Bovaird, T., "Health care quality in NHS hospitals", *International Journal of Health Care Quality Assurance*, v. 9, n. 1, 1996.

QWE2002 Paper 4M

Mr. Ton Dekkers & Mr. Mario van Os
(IQUIP Informatica BV)

Quality Radar: Getting Grip on Customer Expectations

Key Points

- Indentifying non-functional requirements.
- Valuing and specifying quality aspects.
- Product and process improvement.

Presentation Abstract

Achieving Customer Satisfaction by getting grip on expectations of customer by identifying and valuing the implicit quality aspects and specifying them in concrete terms. This improves building and testing of the information system required by the customer. The Quality Radar is a tool and an approach that supports the part quality aspects of the overall opportunity-based approach Quality Tailor-Made (QTM).

About the Author

Ton Dekkers has been working as a practitioner and manager within the area of software quality for a great number of years. Within this area he specialises in estimating, risk analysis and QA in projects (QTM in practise). He is a regular speaker both at national and international conferences and a trainer in software estimating, risk management and QTM. Ton Dekkers is quality consultant in the division Software Control of IQUIP Informatica B.V. He is member of the NESMA workgroups FPA in Maintenance and FPA New Technology and a member of Programme Committee of Quality Connection Conference (Helsinki, June 2002).

Mario van Os is quality manager in the division Software Control of IQUIP Informatica B.V. After his study Mathematics & Computer Science at the Technical University in Eindhoven he has followed the path from systems developer, project manager, consultant to quality manager. As a consultant he often advised the management of various companies about their information strategy. Since 1993 he specialised in the field of quality management, namely quality assurance, risk management, reviewing, inspections, validation and verification, software process improvements, CMM and QTM.

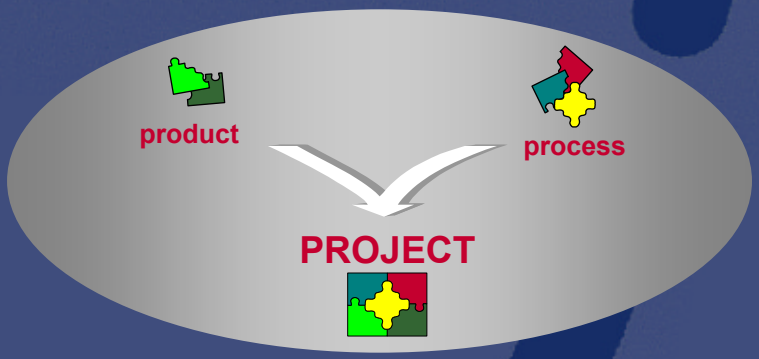
Quality Radar

Getting Grip on Customer Expectation

Ton Dekkers
Mario van Os

March 2002
QWE 2002

IQUIP



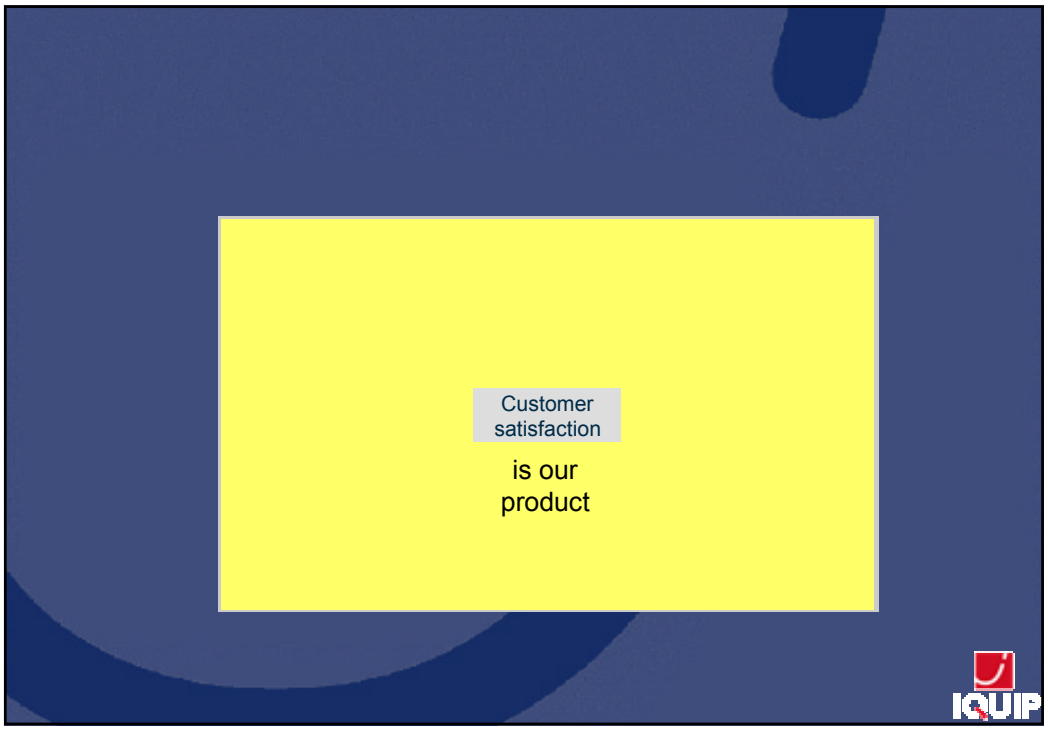
Quality Tailor-Made

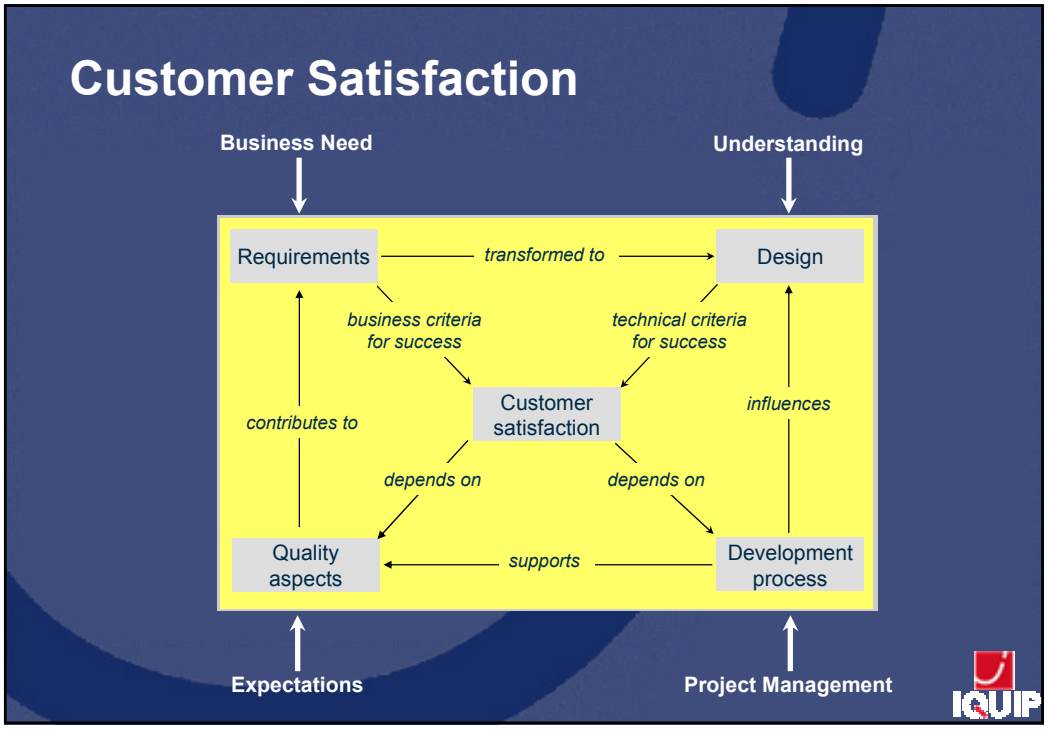
A practical approach
to optimise return on investment in IT-projects

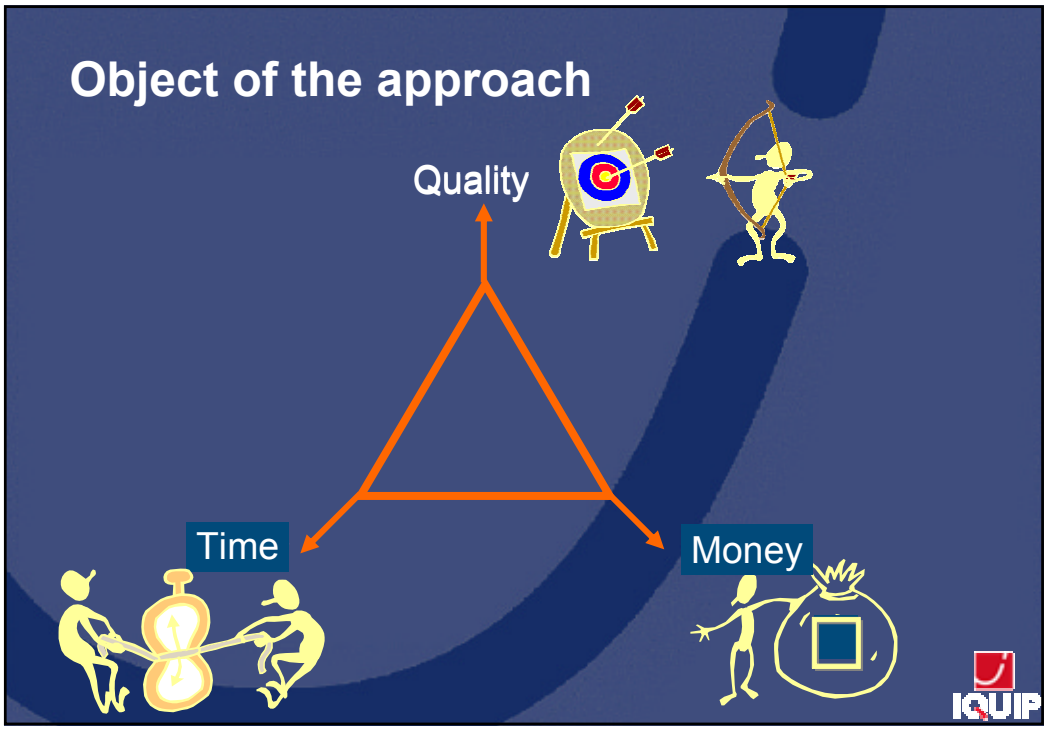
IQUIP

IQUIP Informatica B.V.

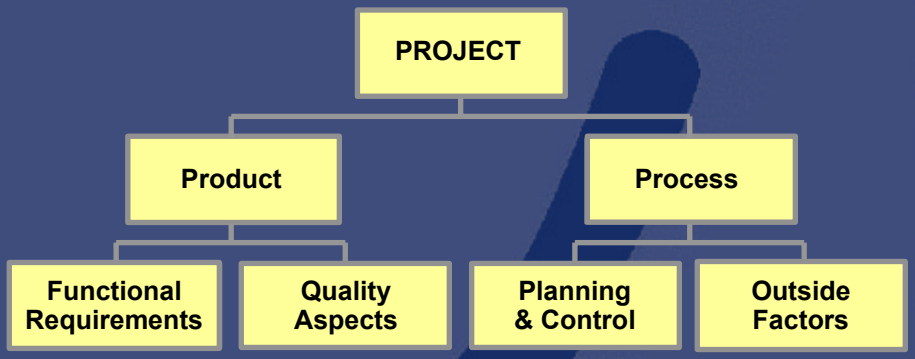
- Founded in 1972
- Full-time staff: 1,500
- Cap Gemini Ernst & Young SA, 100% shareholder
- Business units
  - E-Business (5 regional units)
  - Application Service Management (ASM)
  - Integration (Int)
  - Software Control (SC) - 450 employees
    - » Quality Assurance [100]
    - » Testing [350)
- Services
  - implementation of projects
  - posting of specialists




Customer Satisfaction
Business Need
Understanding
Requirements
transformed to
Design
business criteria for success
technical criteria for success
Customer satisfaction
contributes to
influences
depends on
depends on
Quality aspects
supports
Development process
Expectations
Project Management
IQUIP


Object of the approach
Quality
Time
Money
IQUIP

Attention Areas



Five steps to control

What do we have to do?

- What are the project deliverables / what are the conditions?
- In which order (priority)?
- Make expectations concrete (the project team and the user)

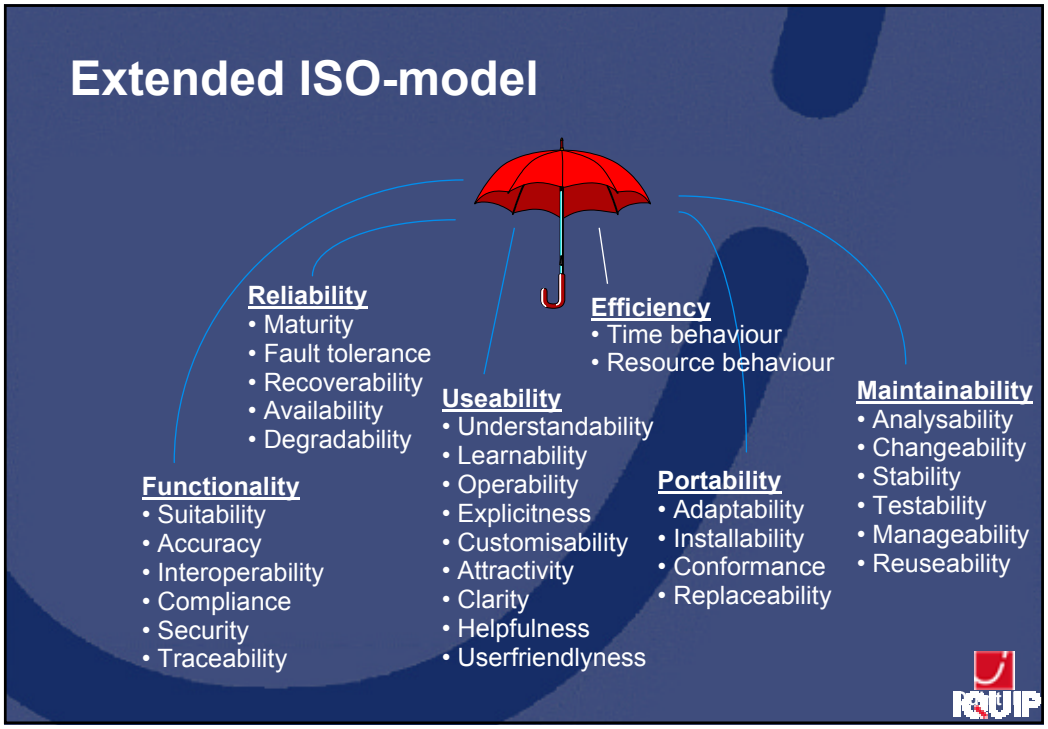
What makes this successful?

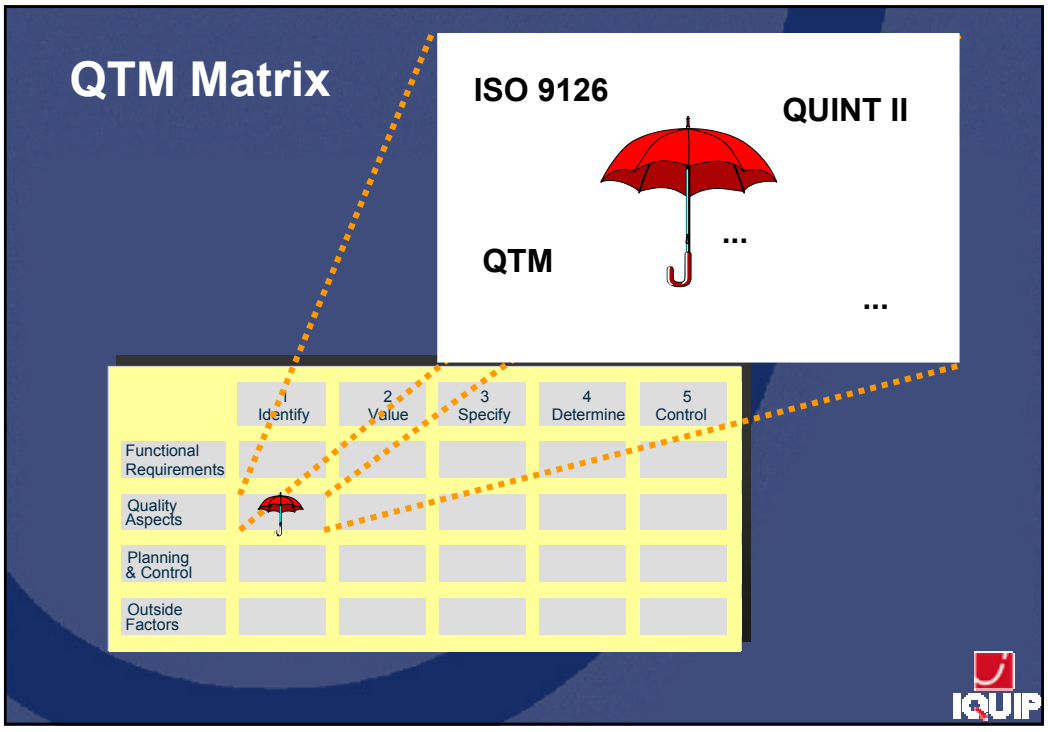
- Which risks are seen by the project team?
- Which measures are necessary?

QTM Matrix

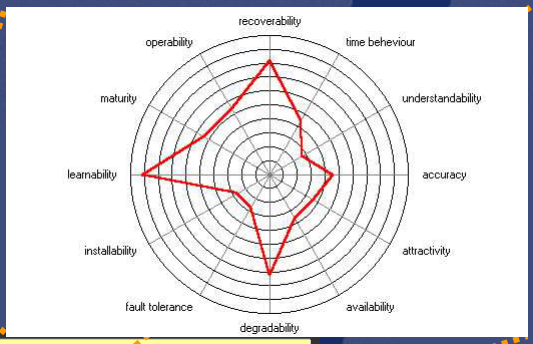
	1 Identify	2 Value	3 Specify	4 Determine	5 Control
Functional Requirements					
Quality Aspects					
Planning & Control					
Outside Factors					

Which properties?


Extended ISO-model
Reliability
• Maturity
• Fault tolerance
• Recoverability
• Availability
• Degradability
Efficiency
• Time behaviour
• Resource behaviour
Functionality
• Suitability
• Accuracy
• Interoperability
• Compliance
• Security
• Traceability
Useability
• Understandability
• Learnability
• Operability
• Explicitness
• Customisability
• Attractivity
• Clarity
• Helpfulness
• Userfriendlyness
Portability
• Adaptability
• Installability
• Conformance
• Replaceability
Maintainability
• Analysability
• Changeability
• Stability
• Testability
• Manageability
• Reuseability
IQUIP

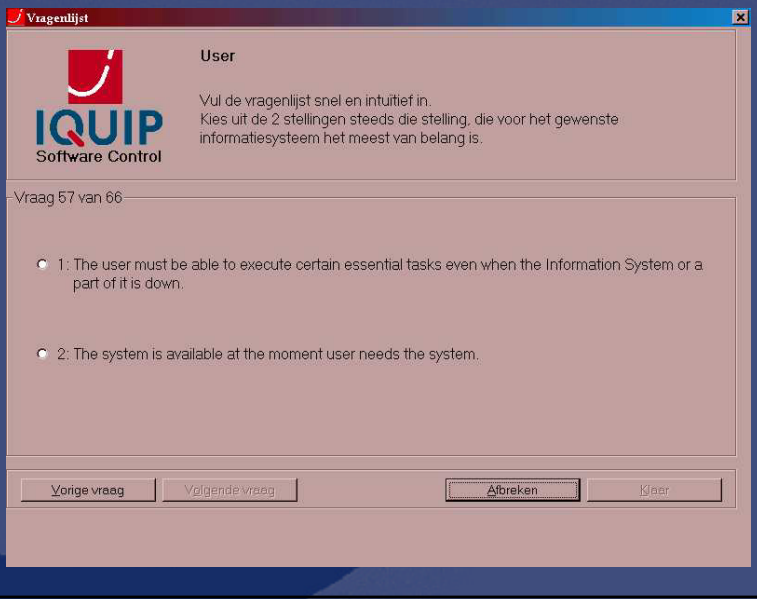

QTM Matrix
ISO 9126
QUINT II
QTM
...
...
1 Identify
2 Value
3 Specify
4 Determine
5 Control
Functional Requirements
Quality Aspects
Planning & Control
Outside Factors
IQUIP

QTM Matrix

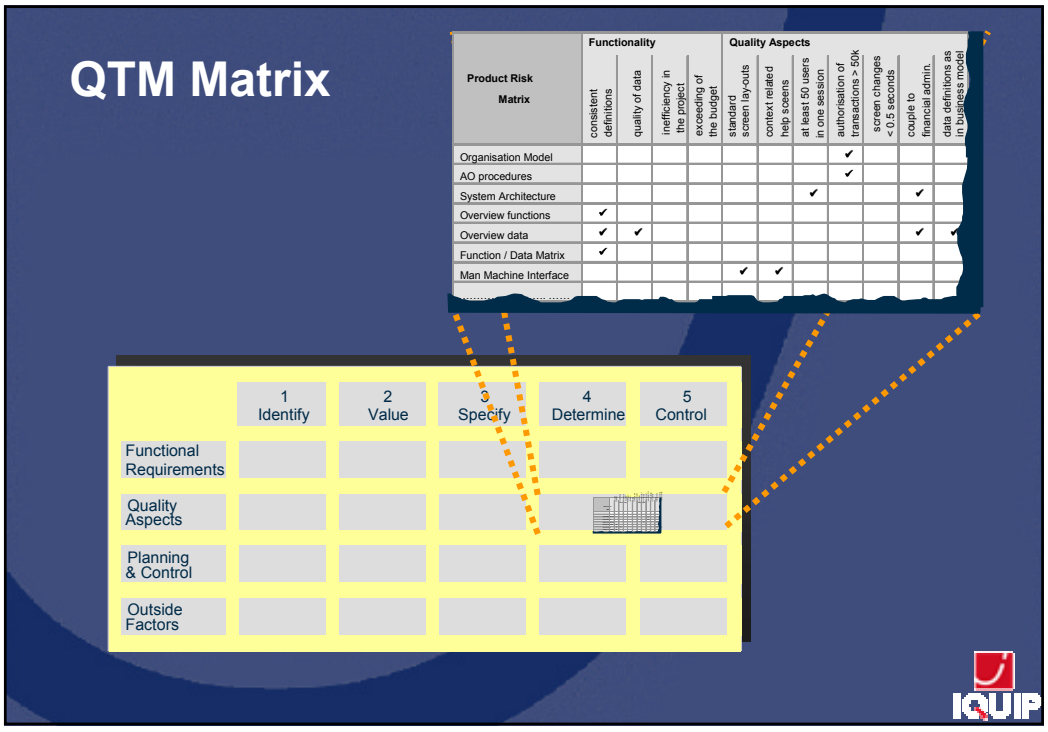


	1 Identify	2 Value	3 Specify	4 Determine	5 Control
Functional Requirements					
Quality Aspects					
Planning & Control					
Outside Factors					

Make choices - Value



Product Risk Matrix
Product Risk Matrix
Functionality
Quality Aspects
consistent definitions
quality of data
inefficiency in the project
exceeding of the budget
standard screen lay-outs
context related help sceens
at least 50 users in one session
authorisation of transactions > 50k
screen changes < 0.5 seconds
couple to financial admin.
data definitions as in business model
Organisation Model
AO procedures
System Architecture
Overview functions
Overview data
Function / Data Matrix
Man Machine Interface
IQUIP


QTM Matrix
Product Risk Matrix
Functionality
Quality Aspects
1 Identify
2 Value
3 Specify
4 Determine
5 Control
Functional Requirements
Quality Aspects
Planning & Control
Outside Factors
IQUIP

Project strategy

DEFINE MEASURES

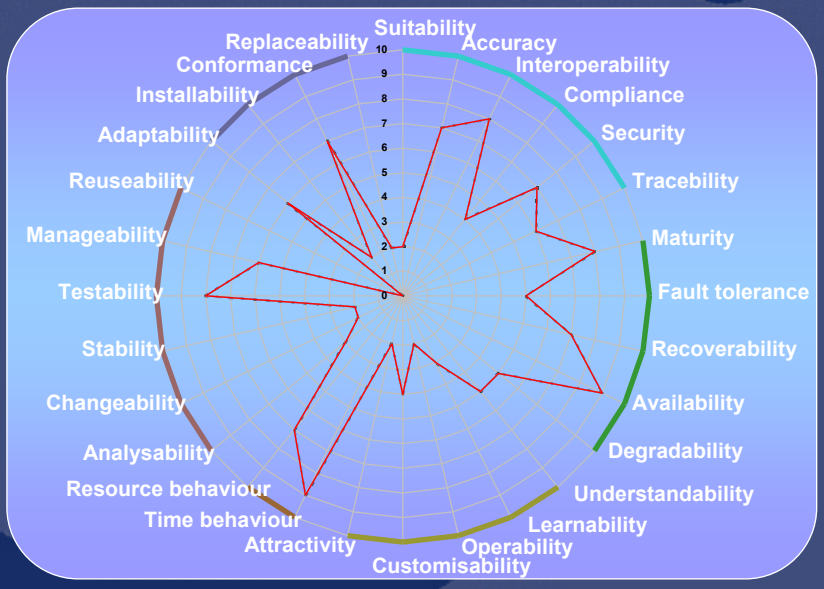
- Adjust project approach
- Preventive / detective
- Project leader makes an ‘all-in package’

TUNE MEASURES

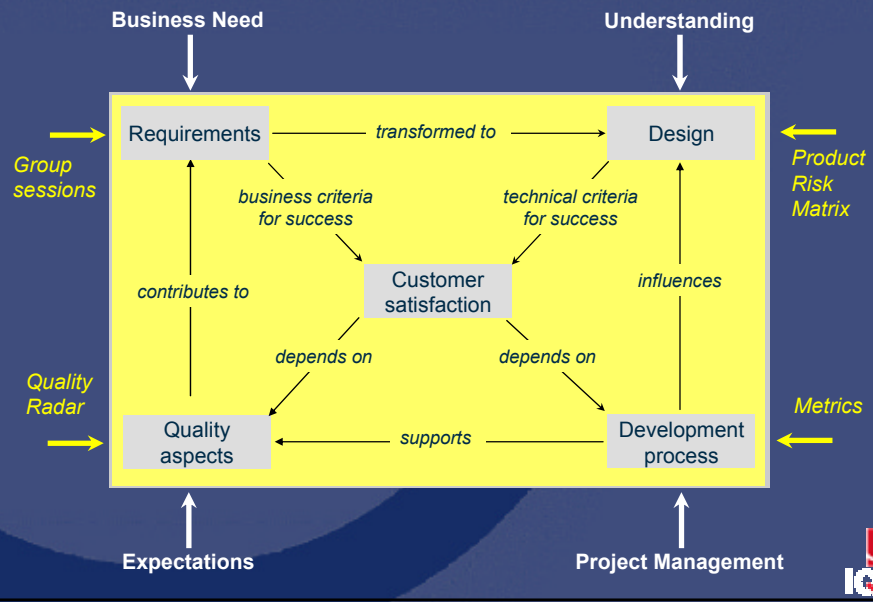
- Discuss risks
- Available budget
- Wanted “coverage”
- Determine impact of changes

Quality plan:
standards
check & test activities
techniques
settlement findings

Quality Radar Demonstration



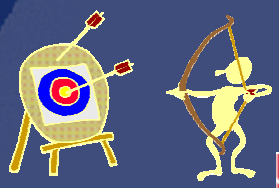
Customer Satisfaction



Quality Tailor-Made

Return (on investment) to the Customer

- A structured Approach to find the Balance between Quality, Time and Money
- Clarity on explicit AND implicit requirements
- Measures directly related to 'project deliverables'
- Supports communication (with all stakeholders)
- Certainty on project operations
- **Grip on Customer Expectations**



Ton Dekkers
Quality Management Consultant
dekkerto@iquip.nl
Estimating & Control
www.qualitytailor-made.com
Mario van Os
Quality Management Consultant
osmario@iquip.nl
Expertise QMap
IQUIP

Quality Radar: Getting grip on customer expectations

Ton Dekkers, Mario van Os
IQUIP Informatica B.V., Goudsbloemvallei 50,
5237 MK DEN BOSCH, Netherlands
a.j.e.dekkers@iquip.nl;qtm@interdependent.nl
m.h.m.van.os@iquip.nl

Abstract. *The success of an Information System is strongly related to how functionality is implemented. Not only by looking at what the Information System should do (the reflection of the business needs) but also to way the Information System reveals it to the customer. At the same time the Information System should be available at the right time and for a fair price. Customer Satisfaction is achieved when you get "value for your money". This is only possible when there is a right balance between Quality, Time and Money. To achieve this we use Quality Tailor-Made. Quality Tailor-Made is a risk-based approach to improve product as well as process in a project. In this paper we focus on the non-functional expectations. Using the Quality Radar you get grip on the elusive, implicit, quality expectations of the customer. In the presentation we will show how the Quality Radar (tool) is used in practice.*

1. Introduction

Before we can go into detail on the Quality Radar, you need to understand more of Quality Tailor-Made (QTM) [1, 2, 3] and the background of the approach.

When an information system is developed, the customer's interest is leading. The business needs written down in user requirements are the basis for the Information System (IS) wanted by the customer. A fully accepted information system is not just an implementation of the functional user requirements, it should also meet the quality requirements and all other (implicit) expectations. In order to achieve this, all people concerned has to be aware of the fact that they have to provide all necessary information to the developers. On the other hand the developers should be aware that it is difficult to think of all the aspects that could be important for the system. So the challenge of the developers is to help getting all the requirements clear and to assure that from the technical point of view the users get the right solution for their needs. The developer has to keep in mind: the solution is not a highly sophisticated state-of-the-art technical product, but a product fit for purpose that meets all the expectations of the user. Furthermore the product has to fit within the time and money constraints. The user has to be aware to set the scope of the system or the project in such a way that it is possible to develop a system within budget. Here you are; the balance between Quality, Time and Money. When the balance is right, we will get customer satisfaction.

2. Customer Satisfaction

We use a model to understand from which customer satisfaction can be assessed. The model expresses the customer satisfaction using four key related components:

- Business requirements (arisen from new or changed business strategy, the "hard" requirements).
- Quality aspects (the explicit and implicit expectations of system's behaviour, the "soft" requirements).
- Functional design (transformation of the requirements including quality levels, development criteria and constraints).
- Development process (process capability and people competence).

©2001 Ton Dekkers - Mario van Os, IQUIP Informatica B.V.

This model represents optimal result if all its components maximise their contribution to customer satisfaction. For example, customer satisfaction is achieved when the product (information system) matches the needs. Only this is not possible when the requirements are not transformed accurately into the design. The design not only reflects the projection of the requirement on the development environment. It also directly influences the development process, the project organisation and competence of team members involved. The right process execution will deliver trustworthy product quality.

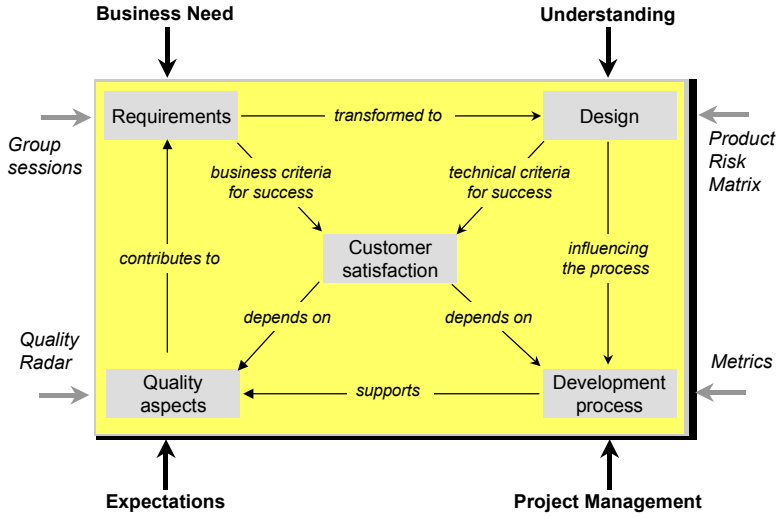


Fig. 1. The Customer Satisfaction Model

The left triangle (Requirements, Customer satisfaction and Quality aspects) reflects the product area. The customer will not be satisfied when the IS does not offer the functionality agreed upon in the functional requirements and is not functioning according expectations.

The right triangle (Design, Customer satisfaction, and Development process) shows the process area. The customer will also not be satisfied when it is not produced well; he expects the information system to be delivered in time and within budget. The customer wants value for money and kept agreements.

Software Control has looked for an approach to be able to manage all the aspects mentioned before and during the project. Quality Tailor-Made is developed based on experience in Quality Assurance in a great number of projects. For us, Quality Tailor-Made is the approach to maximise customer satisfaction. It helps you getting the answer on customer's issues. What do we need to do to satisfy the customer? How can we improve the quality of the product? What is the optimal process to create the product? The message of QTM is: control system development projects not just from the point of view of time or money, but choose an approach which keeps both the explicit and the implicit requirements of the customer in mind. Achieve goals on both the product and process: use the opportunities and control the risks.

3. Quality Tailor-Made

QTM is no new development method; QTM does not commit to replace existing tools or standards. No, it is just a practical approach that helps to find out what's important to do and where you must think off in a project.

3.1. Attention Areas

Looking at the customer satisfaction model it is obvious the risks can be split into an area related to the deliverable product and an area related to the process to get to the product. Analysing the triangles in the model we can recognise two sub-areas in the product area: the functionality and the quality aspects. In the process area also two sub-areas can be distinguished: issues in relation to internal planning and control of the project organisation and those in relation to the outside factors.

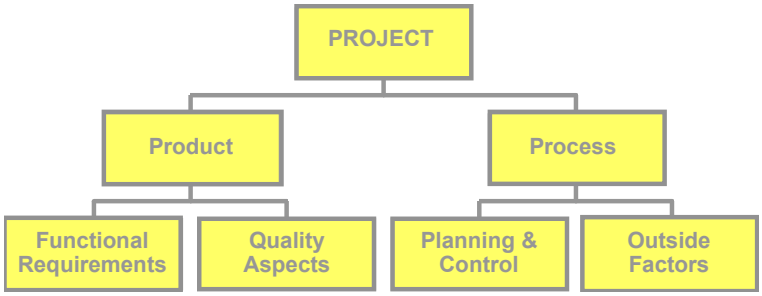


Fig. 2. The four Attention Areas

Briefly, the attention in respect of the product fixes upon; build the right system. The attention in respect of the process fixes upon; build the system right. To get customer satisfaction you need to take care of both, product and process.

When you analyse the activities of managing a project, you can categorise these activities in five steps. These steps are aimed at optimising results related to the product and activities related to the product development process. These five steps will be considered for all four attention areas; Functional Requirements, Quality Aspects, Planning & Control and Outside factors.

Fig. 3. The five Steps in a Project for both Product and Process

The first step is to *identify*: what do we really want? In this step we will try to identify all possible aspects within an attention area. When we *value* the objectives, a selection will be made from all aspects that are relevant for this specific project. There is always a hierarchy in the properties. From properties that are considered to be important, the requirements must be *specified* in detail. For instance, if response time of the system is important, criteria for response time must be written down, and quantitative aspects like the percentage tolerance must be set.

There are still two more steps to go. Once the requirements have been made explicit, we have to *determine* which risks there are. And finally the risks must be *controlled*: measures should be taken.

Summary: in five steps the needs will be defined, valued and worked out in requirements (demands and wishes), after which the risks will be mapped and measures taken to manage these risks. These steps have to be taken for each attention area.

©2001 Ton Dekkers - Mario van Os, IQUIP Informatica B.V.

3.2. QTM matrix

The QTM matrix is created when we match the four attention areas (Functional Requirements, Quality Aspects, Planning & Control and Outside factors) and the five steps (Identify, Value, Specify, Determine and Control):

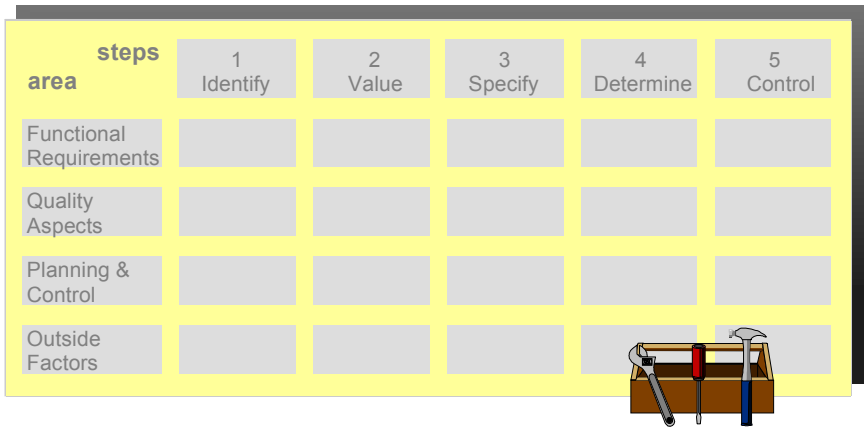


Fig. 4. The QTM Matrix, in rows the four Attention Areas and in columns the five Steps.

For a project carried out using this method, each cell of the matrix contains specified actions. It is possible that a certain cell is not filled up, but in that case there is a well thought out reason: you choose consciously to take no action, you are aware that you take a risk.

Looking at how projects are executed: they are all focused on controlling the process in terms of time and budget. Of course, in these methods Functional Requirements and Quality Aspects are also important, but the main goal is still: finish the project within time and budget. The quality (of the deliverables) plays second fiddle. In practice the system delivered (on time) often does offer most of the functionality required. The product is (not completely) fit for purpose and does not meet the expected quality. Conclusion: the customer is not satisfied.

QTM will help you to find the right balance between attention for process and product. The QTM matrix addresses the relevant issues regarding customer satisfaction in a structured way to ensure optimal result in each area. Take care you get the demands and wishes well mapped, determine the risks and the consequences of those risks. When you know the consequences, you can decide how much time and money you want to spend on controlling the impact of the risks.

In chapter 4 we will go into detail on the five steps in the area Quality Aspects.

3.3. Plans

The Product Risk Matrix that will be explained in the chapter 4 represents the outcome of addressing all cells in the QTM Matrix. Based on the Product Risk Matrix the project approach will be adjusted. Briefly, after defining the definitive list of deliverables the strategy must be defined, while the measures must be specified in a quality plan and/or a risk plan:

- Agree upon product standards: requirements need to be proven in at least one deliverable.
- Schedule reviews, checks and test activities;
- Decide on the techniques to be used, dependent upon the products to be reviewed, the manner in which those products will be reviewed and which techniques fit best.

The Product Risk Matrix is showing also a total overview of the risks and the relation to (intermediate) deliverables: at which moment, by whom and until what extend the checks will be executed, on which (intermediate) product, and against which costs? Decisions can be made, together with all participants on the basis of a cost - benefit analysis: do we take this measure or not? It is the responsibility of the project leader to decide if it is more efficient to combine test activities, to add additional products and so on.

The development process usually starts with drawing a project-plan. In this plan the goals for the project are set and the milestones in the process. The project leader identifies the issues that are important for the project, e.g. development platform, organisation, schedule, resources, security-plan, quality-plan, etc. With the customers the issues are valued and specified. The most important items in this area for customers are Time and Money: when is it finished and what will it cost? The availability of metrics helps us to estimate effort. Effort can be expressed in time or in money. We can do this on account of the correlation between functional size and effort [6, 7]. Another main variable of effort is the development platform and the available tools. The collective noun for all effort-related variables is productivity attributes.

After that the project leader should perform a risk analysis (determine) and define the appropriate measures to attend to the risks (control). At the end, all of this should be reflected in the project-plan.

Let's go and see how this approach works in the area of the Quality Aspects.

4. Quality Aspects

When discussing Quality Aspects all "other" properties of the information system are included. Aspects that can be seen as the Human factors of the system. The right attention to the quality aspects makes the information system more than just fit for purpose. Getting grip on these expectations will have a considerable influence on customer satisfaction.

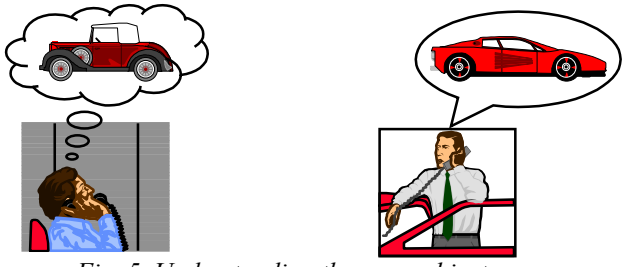


Fig. 5. Understanding the same object.

Although it is an often-used example, it helps to get things clear. The primary set of requirements for a car relates to functionality expected; the car should bring you from one place to another safe and dry. When you select a car, you assume the primary functionality is present so you will choose based on a secondary set of requirements. The purpose of the car fixes one's thoughts on the type of car; e.g. a limousine, a MPV, a 4-wheel-drive or a sportscar.

And at end the feeling you get by looking at the car and even sometimes the status of car makes you decide. And that's the reason why we have such a great variety of cars, the number brands, the various styling, innumerable types and in addition still a lot of accessories.

In IT it is challenging getting clear what the system should do, it is already difficult to define the primary set of requirements. Even more challenging is getting grip on what customers really expect of the way the IS should reveal its functionality. If you do not pay attention to the non-functional quality demands before you start developing, you will be confronted with these demands in Acceptance Testing or even worse you will get unsatisfied customers or you loose business. Identifying these quality aspects as early as possible will avoid both annoyance and additional costs. QTM tackles this issue in the same way as it does with functionality by passing through all five steps.

4.1. Identify Quality Aspects

It is very difficult to start identifying quality aspects without any reference. With functionality you can fall back on your business but identifying quality is another story. Fortunately there are already international standards that identify (non-functional) quality attributes. Examples are ISO-9126 [4], Quint II [5] and TMap® [1][6]. These standards give you a list of attributes (and definitions) that helps you to define and to decide about quality. So the fist activity in this QTM matrix cell is to decide on your set of quality attributes.

Trouble starts when the set of quality aspects is laid down. In practise we found out that it very difficult to relate your expectations about quality to abstract attributes defined in such lists. So we looked at other areas where they have the same problems. In recruitment they make a profile on keywords (personal attributes). In psychological tests these keywords are masked in propositions. When the participant values the propositions, he or she indirectly values the keywords. In difficult decision making processes they use similar techniques to help decision-makers. In mathematics you will find argumentation based on comparison. One of theories we have found is the Analytical Hierarchy Process [8]. This theory uses pare-wise comparison as a starting point. Valuing all possible combinations of decision object gives as result a score for each object. This score you can use in your process. By replacing the objects by propositions, you can score very easily your quality attributes. The use of propositions and the pare-wise comparison are the basis of the Quality Radar.

To make life easier the quality attributes are “translated” to propositions that refer to the environment the customer understands. Of course you can use the propositions from earlier projects or from other companies. The ability to understand the quality aspects will improve.

In practice we experienced that there is a wide differentiation in outcome between the participants. To control discussion of the outcome we started to use roles. A role relates to the interest, the focus and responsibility in the project. To get grip on expectation it is necessary to identify for each project the relevant roles. At least following roles should be identified:

- the business: how important is the function for the business process?
- the user: which functions are important for operational tasks?
- the administrator: which functions are necessary to manage and maintain the system?

Introducing roles has also another advantage. To do a full pair-wise comparison of order `n` attributes, it requires `n.n-1)/2` comparisons. Referring to TMap® (17 attributes) this will require 136 comparisons. Although filling in the questionnaire does not take that much time, the number of comparisons should be reduced. We experienced that not all attributes are relevant when you take into account the role of the participant in a project. So based on practice in the organisation you relate a subset of the quality attributes to a defined role. Using a subset the number of comparisons will definitely decrease and helps to come to consensus with the participants in a role.

[1] TMap® Test Management approach is a registered trademark of IQUIP Informatica B.V.

As you can see, in principle any (standard) list of quality aspects can be used as a reference. Using an international standard offers the possibility of comparing the propositions or interchanging propositions. The way a project is organised, the size of a project or the phase of the project is not that relevant, the approach can be used always and everywhere.

4.2. Value Quality Aspects

In this step the representatives of the roles defined in step 1, make the comparisons. He or she must value the quality aspects by choosing between two propositions shown in a questionnaire. Each question in the questionnaire consists of two propositions from which one must be selected. Deciding with the required information system in mind, the participant indicates the relative importance of the "other" properties of the system. The output of the questionnaire will be shown in the Quality Radar. The fig. 6 shows the result of a questionnaire using Quint II as a reference.

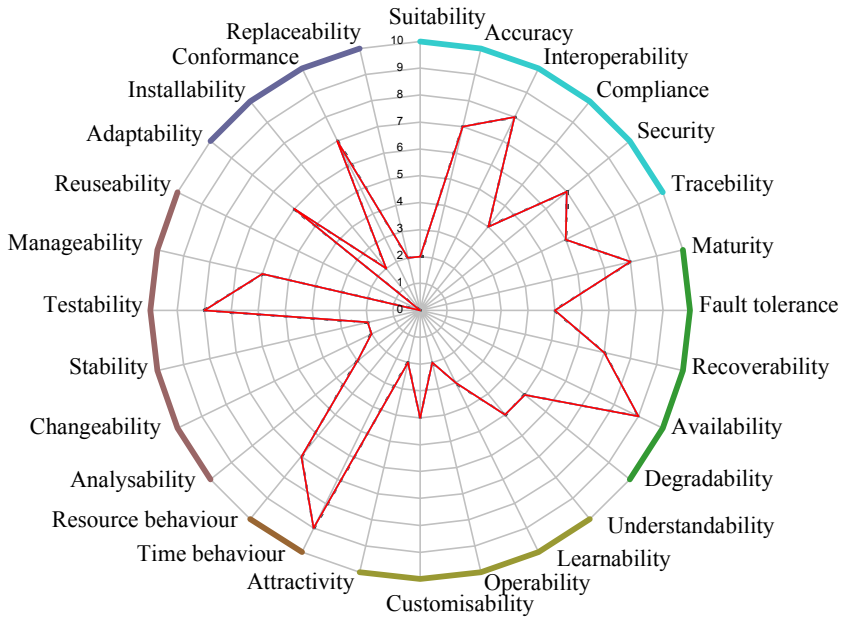


Fig. 6. This plot shows a possible result from a Quality Radar session. In the plot you read that Availability is more important than Stability.

You can make plots like these on individual level, group level and on overall level. These images indicate which quality aspects are important and that some of the quality aspects are more important than other aspects. It will probably also show differences in perception between participants in a specific role and differences between roles. Here you have the basis for a solid discussion. It helps to create consensus about the "soft requirements" (the human factors) of the product.

The Quality Radar proves to be a powerful communication tool: it helps to get grip on expectations of the various parties and leads to greater involvement of all participants.

In the appendix "How to use the Quality Radar" [9] we will describe the process in detail and refer to the use of the Quality Radar Tool [10].

4.3. Specify Quality Aspects

The (most) important quality aspects must be specified. If "Availability" is considered to be important, the customer must define concrete requirements, e.g.: "the information system must be operational five days per week, eight hours a day". The properties must be defined in such a manner that during the project, it can be checked whether the IS will meet the specified requirements.

By "Operability" we mean the customers want an easy to handle information system. This requires for example uniform screen layouts, not only within the system, but also common with other existing systems.

4.4. Determine Risks

When the properties are known in detail, the developer has a good understanding of what should be built. Further, he has to think about what kind of risks he runs in developing the system? Is the requirement a certain transaction should be completed within two seconds feasible? Is it easy to accomplish, or is it a problem - due to the way the transaction is defined, the performance of the network, etc.? If it is a problem, measures must be taken to meet the specified requirements.

4.5. Control Risks

The project already passed the steps: Identify, Value, Specify and Determine Risks for both functionality and quality aspects. So we known what we want, how we want it and what could possible mix up things. In this step we define the measures to insure we will meet the expectations and get customer satisfaction.

Essential in QTM is the balance of attention towards product and process. At this time the link is made to the process. Mostly measures require actions and activities are the building blocks of the process. This we will do by linking the (intermediate) deliverables of the process to the requirements of the product. By deliverables of the project we mean the documents, models and programs defined that are needed to control the process in order to produce software. Although this list of deliverables depends on a development method and the implementation of this method in the organisation, it is always possible to draw up a list of deliverables. This list does not contain only documents or coding; for instance a "prototype" is also a kind of intermediate product.

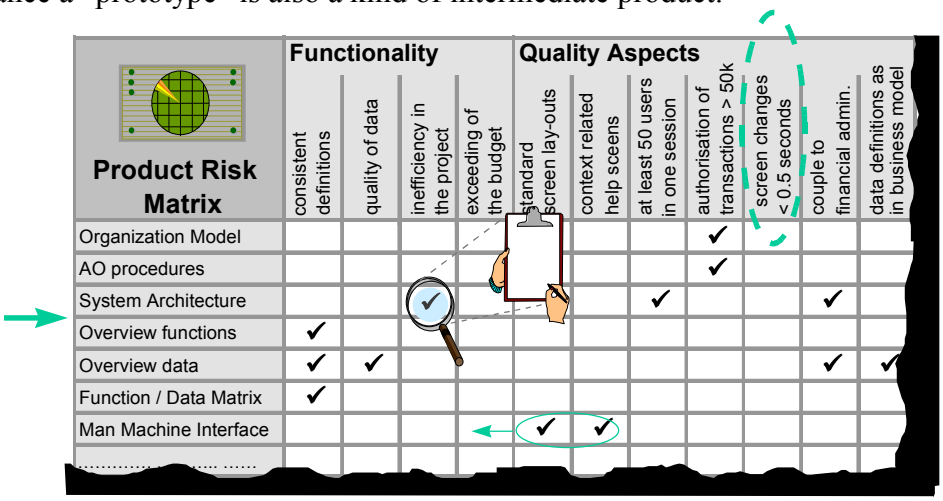


Fig. 7. The Product Risk Matrix. The marks in the cells indicate in which product (rows) a determined risks (columns) need to be checked.

In the Product Risk Matrix the columns reflect the possible product risk areas (Functional Requirements and Quality Aspects) and the rows, chronologically represent the deliverables of the project. In the Product Risk Matrix you link the risks directly to deliverables. So it easier to define concrete measures or to position the measures and the results will be visible on the right moment.

For the attention points the concrete requirements are known. To assure that the product will meet the requirements we have to define some (control) measures. In the Product Risk Matrix we mark (√) the points in which (intermediate) deliverable we can check that the requirement is met. By these marks the requirements are linked to the deliverables of the project. The earlier certainty is given in the project the better. With the Product Risk Matrix you can tune the measures, actions take place at the appropriate time: optimising activities and controlling risks. Mind that products like "Functional System Design" are too generic; the matrix works better when products comprise smaller logical units, such as function structure, data model, user interface and a description of the administrative organisation. Only the deliverables that describe the software will be included in the matrix (system documentation), not the process documentation (like plans, phase reports, etc.).

The Product Risk Matrix is the basis for the specification of product related measures. In the matrix concrete measurable requirements are linked directly to physical project deliverables. The (intermediate) deliverables should be used as early as possible in the project to ensure specified requirements of the end product will be met. It is up to the project leader to decide whether the standard project approach handles all determined risks and if so, on the right time. When not, he or she can take now appropriate measures based. Through the direct link to both product properties and process deliverables, all measures will improve the quality of product and process.

5. Getting grip on customer expectations

In the paper we described Quality Tailor-Made. The approach helps to take the relevant measures at the appropriate time based on the product risk matrix. This is possible because of attention for both product and process. The Quality Radar that proved to be very useful in risky area of the elusive expectations related to the product. QTM also helps you the keep up with requirements and to control risks, deliverables and activities. Focussing on product quality not only helps achieving product quality but also helps to keep the project within budget and on schedule. It also saves time and money later because the number of failures will decrease.

With Quality Tailor-Made we will **get grip on customer expectations**.

6. References

[1] KoM, Boeters, Andre, Noorman, Bert, ”Kwaliteit op Maat”, Kluwers Bedrijfswetenschappen, Deventer, ISBN 90-267-2579-5, 1997.

[2] Dekkers, Ton, “Maximizing Customer Satisfaction”, in ESCOM/SCOPE conference proceedings, London, April 2001. (available: http://www.qualitytailor-made.com)

[3] Dekkers, Ton, “Quality Tailor-Made”, in Quality Week Europe conference proceedings, Brussels, November 2000. (available: http://www.qualitytailor-made.com)

[4] ISO/IEC 9126, Information Technology - Software product quality, characteristics and metrics, International Organization of Standardization, 1997-1998

[5] Quint II, Zeist, B. van, Hendriks, P., Paulussen, R., Trienekens, J., ”Kwaliteit van softwareproducten”, Kluwers Bedrijfswetenschappen, Deventer, ISBN 90-267-2430-6.

[6] TMap®, Pol, Martin, Veenendaal, Erik van, ”Structured Testing of Information Systems”, Kluwer, ISBN 90-267-2910-3, 1998.

[7] IFPUG (International Function Point User Group), “Function Point Counting Practices”, IFPUG

[8] Saaty, T.L., The Analytic Hierarchy Process, McGraw-Hill, New York, 1980.

[9] Competence Centre Quality Assurance in Projects, ”Manual Quality Aspects Assessment”, IQUIP Informatica B.V., 2001

[10] IQUIP Informatica B.V., “KwaliteitsRadar”, versie 2.08, IQUIP Informatica B.V., 2001

APPENDIX

7. How to use the Quality Radar

7.1. Introduction

IQUIP Informatica B.V. developed a tool to support the use of the Quality Radar [10]. The recent introduced version 2.0.8 is developed using QTM. The consultants that used older versions participated in group sessions to identify, value and specify both the functional requirements and the quality aspects. The project-plan was finalised when was decided on the measures to control the determined risks. Project Control was by the Vakgroep Kwaliteitszorg in Projecten (Competence Centre Quality Assurance in Projects).

The tool is very useful and speeds up the process but is not necessary. With help of a word processor and spreadsheet program, identical results can be achieved. In the next paragraph the process of getting grip on the quality aspects is described. The same process applies when the tool is not available.

7.2. Quality Radar Process

7.2.1. Decide on a reference

Before deciding on a reference, one should have notion of what the organisation indicates as quality, the used development process and business importance of the Information System. Although it is the intention to have a reference that has an optimal fit to the specific situation, it is practical and preferable to choose an existing standard like ISO 9126, QTM or TMap®. In this step will be decided which reference will be used. The decision is made in consultation with the customer, the project manager and the Quality Assurance employee.

7.2.2. Set roles and Quality Attributes

As soon as the reference is determined, the roles that are important within the project and the organisation must be identified. In many cases the distinction is made between user, business and administration. It is, however, possible that other roles will be distinguished. In some cases, other parties like Controlling or Accounting Department define additional requirements. All roles need to be linked to a set of the Quality Attributes that is of interest to the roles concerned.

7.2.3. Formulate propositions

The Quality Radar contains a number of standard propositions related to Quality Attributes. After deciding on the reference and the defining of the roles, one needs to check the availability of Quality Attributes and the related propositions. In order to get the best possible understanding with the terminology as used in the actual situation the available propositions should be judged on usability. The propositions that do not match the situation in the project need to be reformulated. When no propositions are available for new or additional Quality Attributes fitting propositions should be formulated.

7.2.4. Set up of the Quality Radar [tool]

Before the Quality Radar tool can be used to value the Quality Aspects, it needs to be set up first. The information gathered in the previous steps will be put in the Quality Radar. For this, also see the user manual of the Quality Radar. It is obvious that the Quality Radar needs to be installed first.

7.2.5. Identify participants

Before the questionnaires can be completed the participants needs to be identified. In consultation with the project manager and representatives of the identified roles the individual participants will be selected. At least one representative for each role defined in the previous step should participate. The number of participants depends on budget and added value.

7.2.6. To invite participants and reserve a location.

The project manager or the facilitating Quality Assurance employee will personally inform each participant, sent the invitations and reserve a location. The location should be equipped with a computer with the Quality Radar installed and set up for the project. The questionnaires for the participants need to available in print.

7.2.7. Explanation of the Quality Radar and completing questionnaires

The Quality Assurance employee will inform the participants about Quality Attributes and Quality Aspects. Furthermore, the Quality Assurance employee will explain the intention of Quality Radar and will make an appointment with the participant to discuss the results later. Next to this individual appointment, the results will also be discussed in a group session (by role and overall). In case of a great number of participants, it is also possible to send the questionnaires and give an introduction for example by telephone, letter or e-mail. It is preferable to approach the most important stakeholders personally. First an interview and then completing the questionnaire.

The goal of the completion of the questionnaire is to value the Quality Attributes. Making a choice out of two propositions does the valuation. Every proposition is related to a Quality Attribute. By choosing one position per question, the participant indicates which Quality Attribute is (relatively) more important to the system. Completing the questionnaire takes about 30 minutes and is individually.

7.2.8. Processing scores completed questionnaire

A Quality Attribute employee processes the completed scores. The result of the processing of the results from the completed questionnaires in the tool, is a graphic (radar plot) that is also indicated as the Quality Radar. The picture shows the relative importance of the Quality Attributes indicated by a participant.

7.2.9. Feedback on the results of the questionnaires.

The results of the completed questionnaires will be discussed with each participant. The feedback session is to check whether the obtained results reflect the perception of the participant. Discrepancies between the results and perception are discussed and when necessary processed in the documentation. It is also possible to do the personal feedback at the beginning of the group session (by role or overall). In case time allows, direct feedback with a laptop after completion of the questionnaire, is also possible.

7.2.10. Preparation of the group session

Based on the results of the questionnaires and the described Quality Aspects, the strategy for the group session is determined and the session will be prepared. A location is reserved and the participants will be invited.

7.2.11. The Group session (presentation of the results, specification of Quality Aspects)

The results of the questionnaires will be presented. During the presentation, the relative assessment of the different Quality Attributes and the differences in appreciation of the individual Quality Attributes, will be discussed. It is important to get information on the background of the different appreciation.

The goal of this discussion is to accomplish consensus whether the highly valued Quality Attributes indeed represent the most relevant attributes of the Information System. Another goal is to determine whether the set of Quality Attributes fits the project. The outcome of the discussion is minute by a Quality Assurance employee.

The outcome of the discussion can be:

- the defined Quality Attributes meet the needs;
- additional Quality Attributes need to be added to the set;
- irrelevant Quality Attributes will be removed from the set.

The discussion of the initial valuing of the Quality Attributes also makes clear which Quality Aspects needs supplementation. At the same the owner of the different requirements is indicated.

Based on the selected Quality Attributes and the initial valuing of the Quality Attributes, the participants complete the specification of Quality Aspects in small groups. The specification is presented to all participants during the session. It is possible to organise a separate session for this. Techniques like “brown paper session” or Metaplan® can be used to control the discussions in a group session.

7.2.12. Overview Quality Aspects

The Quality Assurance employee records the Quality Aspects so that a list of requirements per Quality Attributes arises. This overview is distributed to the project manager and the involved project leader(s).

7.2.13. Review Quality Aspects

The project leader(s) and the test co-ordinator(s) estimate the consequences (costs, time) and judge the feasibility of the different requirements. When necessary, they consult specialists, participants and the project manager for further tuning and to get to a definite set of requirements addressing the Quality Aspects.

7.2.14. Formulising Quality Aspects

The definite specification of the Quality Aspects will be included in the project files and distributed to the participants.

7.2.15. Assigning Quality Aspects

The specified Quality Aspects, when applicable, will be assigned as “risks” in the Product Risk Matrix. Then the Quality Aspects can be linked to the deliverable (“products”) in this matrix. The matrix shows the relations between Quality Aspects and the products in the process and by this also the moment when requirements can be checked. This is input for the project plan made by the project manager.

QWE2002 Paper 5M

Mr. Geert Pinxten & Mr. Jens Pas
(I2B)

Variable Test Strategy: Learn To Only Do What You Need To Do

Key Points

- Key elements of a test strategy.
- Pragmatic in your approach to testing.
- What is important in your approach.
- Bringing pragmatism under control.

Presentation Abstract

There is a lot of information available on how to do proper and professional software testing. Yet, professional structured software testing is hardly adopted by software engineers, and certainly not institutionalised. We all know that we start of with a high ambition and motivation and end after one year with nothing more than a defect tracker and a set of unstructured automated tests. Why?

The answer lies in the pitfall of pragmatism. When people address testing issues they tend to call the available information 'too theoretically' or to much of a challenge. Next they pick the 'easy' parts and start implementing them. Examples are the selection of a tool, the automation of the defect tracking process or test execution. We all know that 1) these parts are not easy and 2) they don't work if you don't set your test strategy foundations first. Pragmatism is in many cases an acceptable euphemism for 'not doing what was proposed or even asked'.

When it comes to implementing a test strategy we need to implement those elements that will contribute most fast to software quality with the least effort. In other words, we need to determine the value adding drivers and make sure we steer upon them. For this purpose we developed a framework, called VTS (Variable Test Strategy) that identifies the value added drivers and their underlying parameters.

About the Author

Geert Pinxten is Managing Partner of I2B and responsible for the practice Software Testing and Quality Assurance. Geert Pinxten is a speaker and track chairman on many Software Engineering and Quality Conferences (ICSTEST, EUROSTAR, Quality week, …) He performed several test audits for large scale organisations active in the world of business software development and embedded systems. During these audits he refined the applied audit method. Geert acts as a test manager for different large testing projects. Geert Pinxten was as Development Co-ordinator for a Belgian consultancy company specialised in testing responsible for all internal development activities. These development activities ranged from developing new training packages to the development of new products. Geert Pinxten is an engineer specialised in electronics and information technology.

Idea to Business

I2B

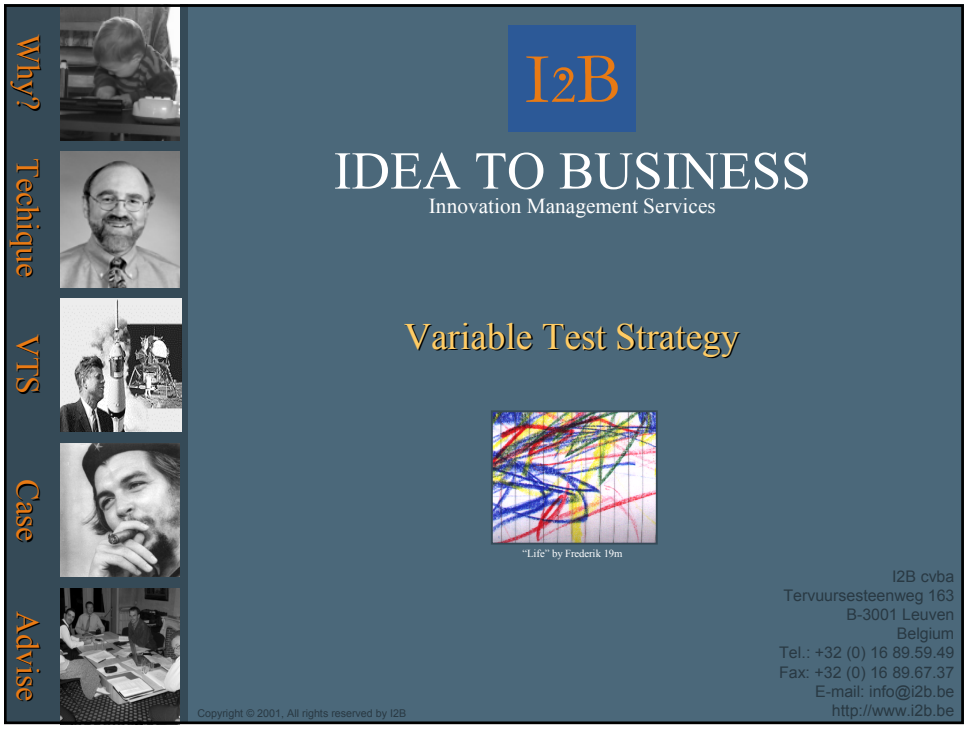
Consultancy for those who change the world

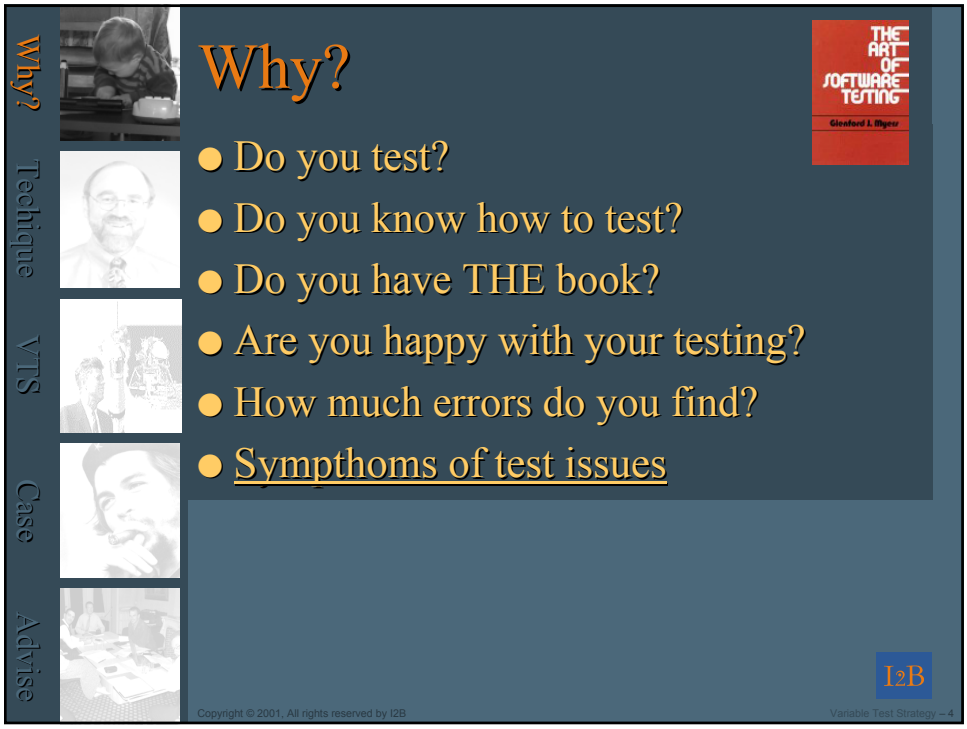
http://www.i2b.be

I2B

The cyclist approaches and sees the the dog

The dog in the street notices the the cyclist


Why?
Technique
VTS
Case
Advise
I2B
IDEA TO BUSINESS
Innovation Management Services
Variable Test Strategy
"Life" by Frederik 19m
I2B cvba
Tervuursesteenweg 163
B-3001 Leuven
Belgium
Tel.: +32 (0) 16 89.59.49
Fax: +32 (0) 16 89.67.37
E-mail: info@i2b.be
http://www.i2b.be
Copyright © 2001, All rights reserved by I2B


Why?
Technique
VTS
Case
Advise
Why?
THE ART OF SOFTWARE TESTING
Glenford J. Myers
Do you test?
Do you know how to test?
Do you have THE book?
Are you happy with your testing?
How much errors do you find?
Sympthoms of test issues
I2B
Copyright © 2001, All rights reserved by I2B
Variable Test Strategy – 4

Sympthoms of test issues

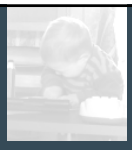
Why? Technique VTS Case Advise I2B

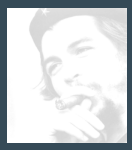
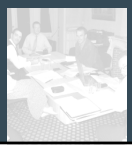
- No dedicated testers
- No test strategy
- Test plan not up to date (or not even made)
- Little defects found
- No test designs
- No test tools
- Automated testing poor (a lot of maintenance)
- Defect tracking not done or poor
- No repair time!
- Testing gets squeezed
- Defects are not repaired
- Testers seem hostile

Goto Start
Back
Goto End

Copyright © 2001 All rights reserved by I2B

Why?
Technique
VTS
Case
Advise

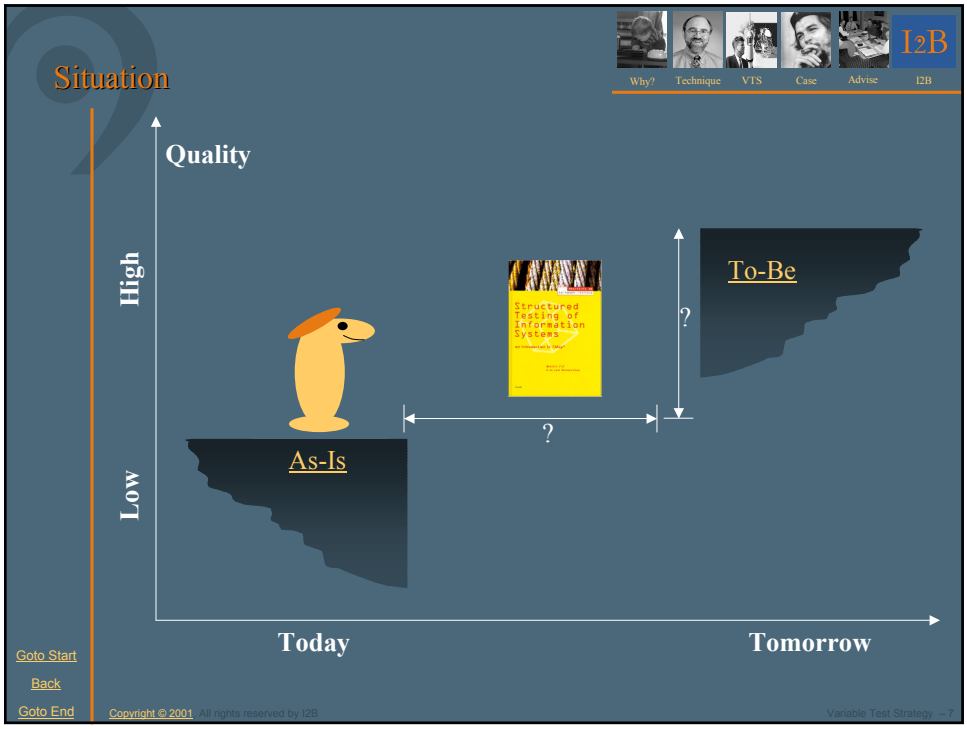


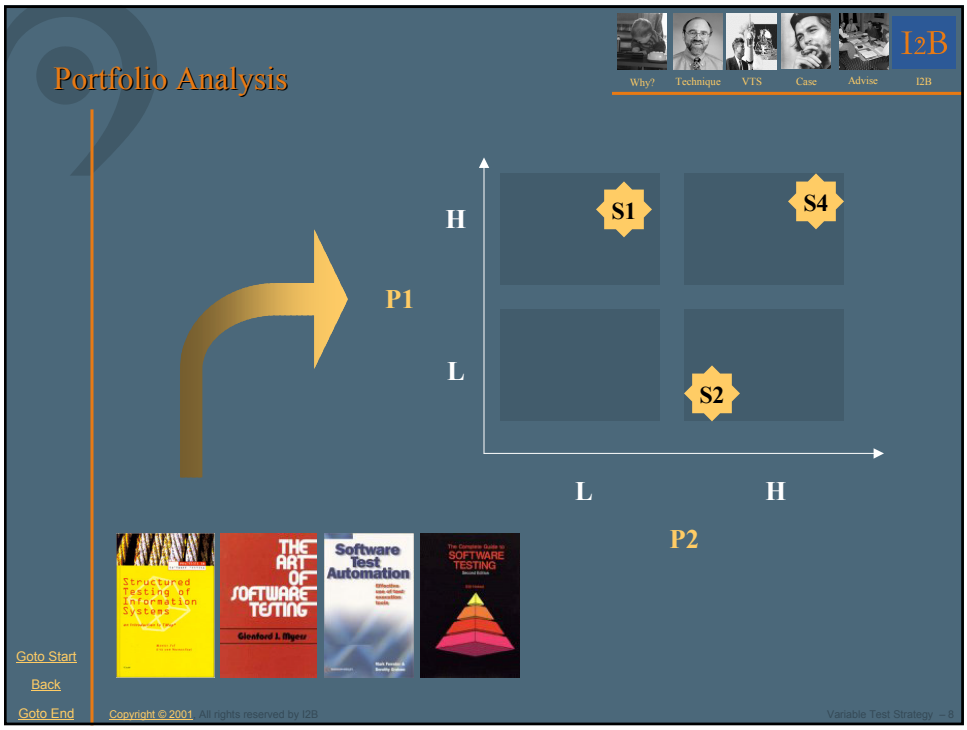



Technique

- Situation
- Portfolio Analysis
- Services Triangle
- Value Chain

Copyright © 2001, All rights reserved by I2B


Situation
Why?
Technique
VTS
Case
Advise
I2B
Quality
High
Low
To-Be
As-Is
?
?
Today
Tomorrow
Goto Start
Back
Goto End
Copyright © 2001 All rights reserved by I2B
Variable Test Strategy – 7

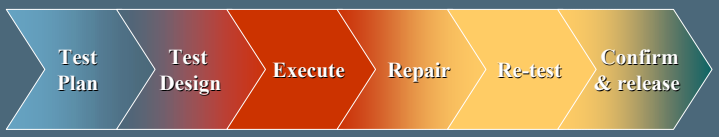

Portfolio Analysis
Why?
Technique
VTS
Case
Advise
I2B
P1
H
L
S1
S4
S2
L
H
P2
Goto Start
Back
Goto End
Copyright © 2001 All rights reserved by I2B
Variable Test Strategy – 8

Services Triangle

Goto Start
Back
Goto End

Copyright © 2001 All rights reserved by I2B

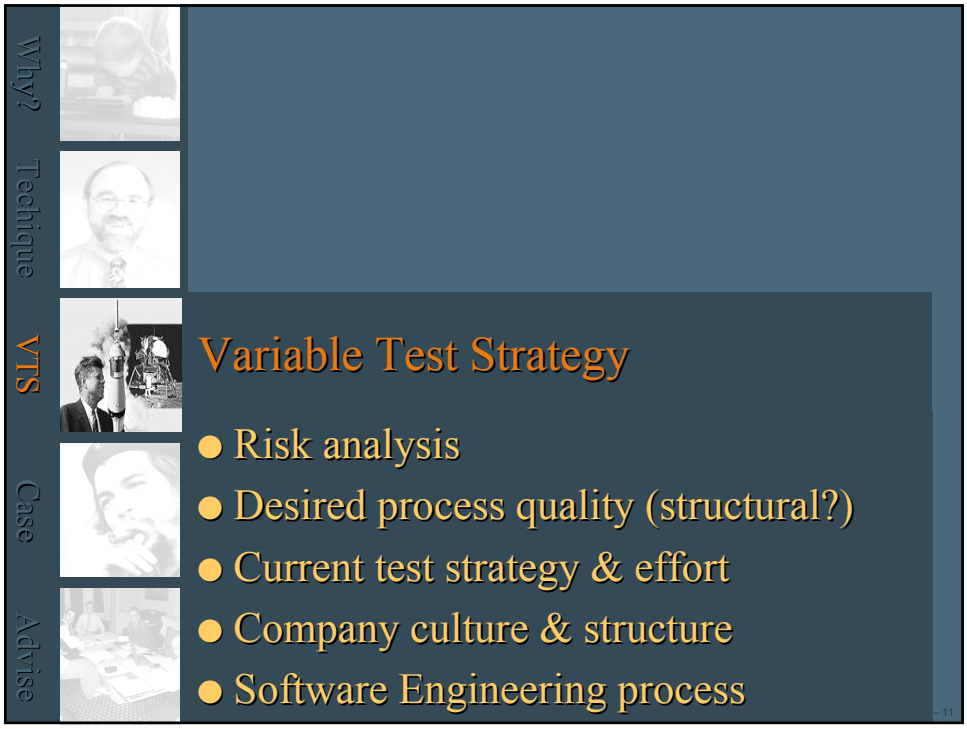
Value Chain



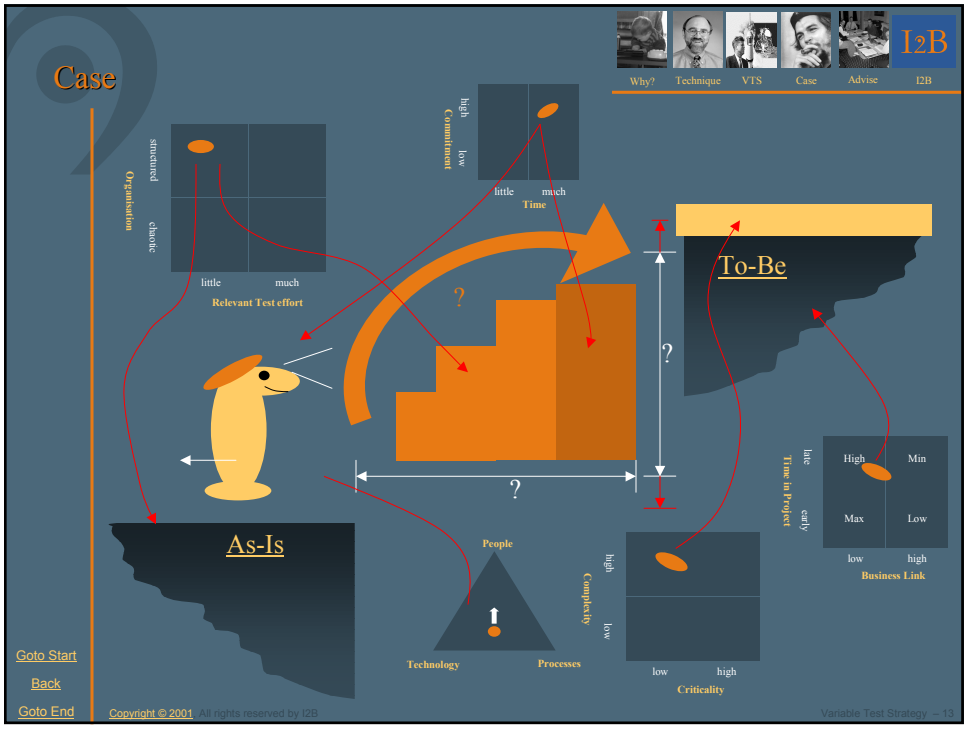
- Value drivers
- Measure throughput
- Localise bottleneck and remove

Goto Start
Back
Goto End

Copyright © 2001 All rights reserved by I2B


Why?
Technique
VTS
Case
Advise
Variable Test Strategy
Risk analysis
Desired process quality (structural?)
Current test strategy & effort
Company culture & structure
Software Engineering process
– 11

Why?
Technique
VTS
Case
Advise
Case
I2B
Copyright © 2001, All rights reserved by I2B
Variable Test Strategy – 12



Idea to Business

I2B

Consultancy for those who change the world

http://www.i2b.be

QWE2002 Paper 6M

Mr. Michael J. Hillelsohn
(Software Performance Systems)

Using SPICE as an Internal Software Engineering Process Improvement Tool

Key Points

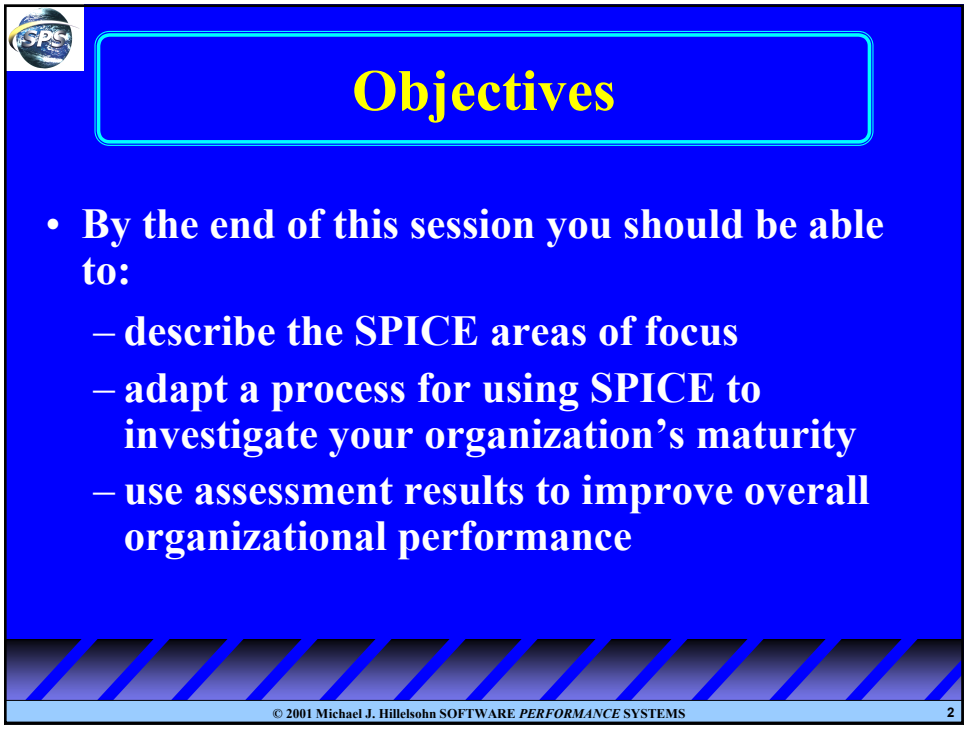
- Using a framework like SPICE facilitates improving quality of products and services.
- The process of performing software engineering assessments inherently improves quality.
- The journey is more important than the certification.

Presentation Abstract

This presentation will provide a brief overview of the SPICE framework and then present the case study of applying the framework, in a limited fashion, to obtain a "snapshot in time" of the organizations process maturity. The organization decided on using the framework because senior management saw a large benefit in improving overall organizational performance and commitment to quality. Well discuss the criticality of senior management support to the success of the effort. The presentation will emphasize the importance of achieving cultural changes that over the long term will show quantifiable improvements in internal processes that will enhance the ability of the organization to satisfy its mission. Only a relevant subset of process areas and key features were assessed because the priority was to identify opportunities for performance engineering interventions rather then determining certification status. The results did establish a meaningful baseline against portions of the SPICE framework so that the organization can show measured improvement in the future. After the assessment was completed the findings were consolidated and prioritized, and action plans for addressing the highest priority items were formulated after the results of the assessment were presented to senior management. The quality improvements that were initiated, addressed underlying infrastructure issues that were revealed during the assessment, instead of focusing more narrowly on gaps that would improve scores in specific SPICE areas and features. The SPICE based interviews proved to be an invaluable tool in revealing the process improvement opportunities that most impact the organizations performance and responsiveness to its customers.

About the Author

Michael J. Hillelsohn is a Director, Product Assurance at Software Performance Systems (SPS) in Arlington, VA. SPS builds secure e-commerce, case management, and network solutions for government and industry clients. Mr. Hillelsohn is a certified quality professional with more than thirty years of experience doing development, management and performance improvement in software and systems development environments. His multi-disciplinary approach combines quality systems and training expertise to improve the performance of organizations and individuals. Mr. Hillelsohns process-oriented, performance engineering methods facilitate adoption of external frameworks (CMM, ISO, SPICE, Baldrige) to improve the quality of organizational products and services.



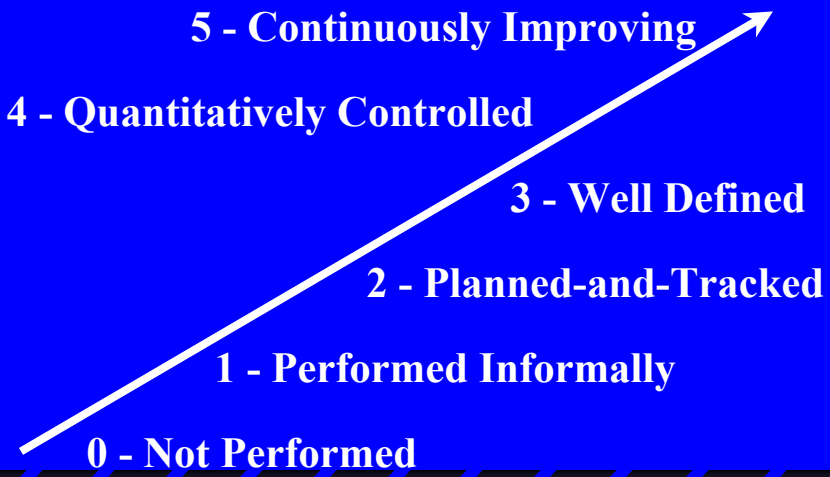
© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com

SPICE Structure

- Process Category
  - Customer Supplier
  - Engineering
  - Project
  - Organization
  - Support
- Processes
- Base Practices
- Capability Levels
- Common Features
- Generic Practices

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

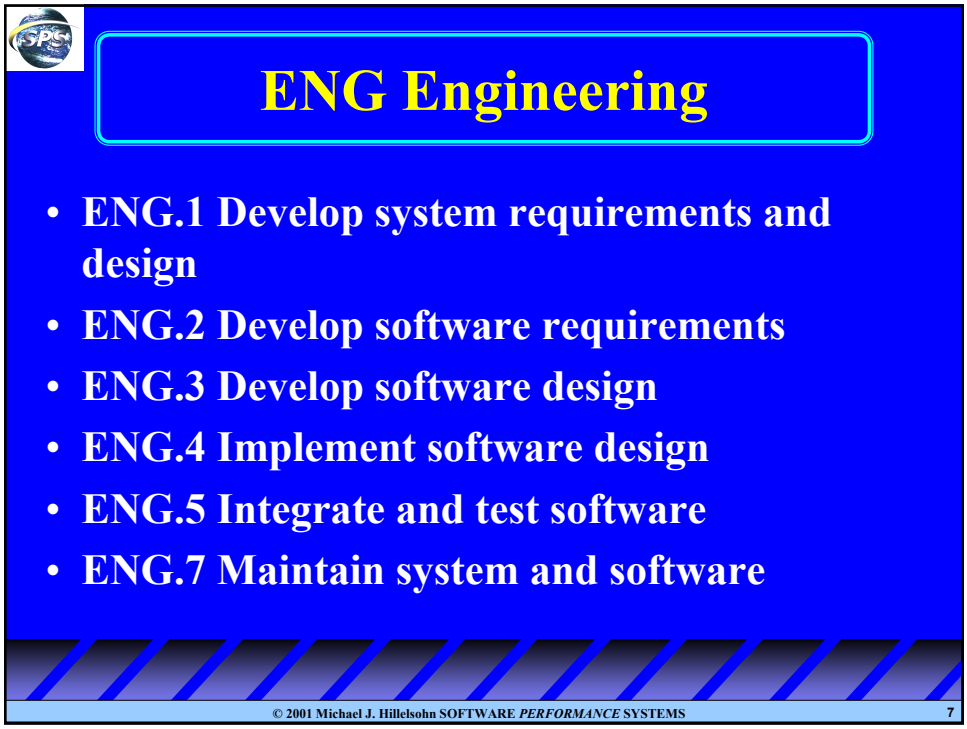
Capability Levels



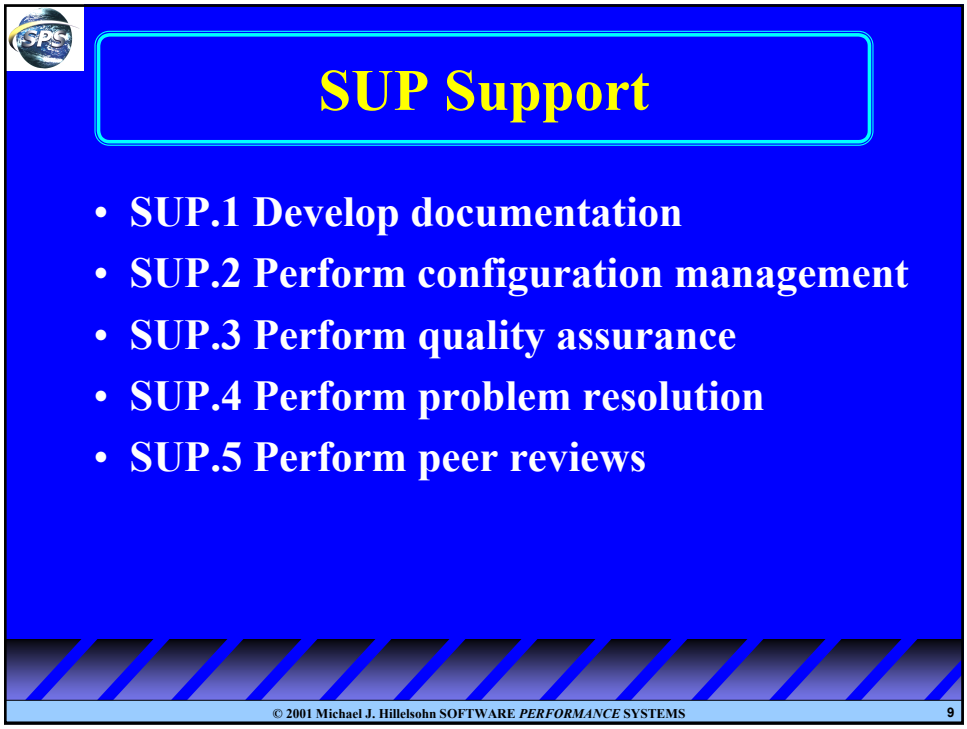
© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com

© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com



© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com



© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com

PRO.5 Manage quality

- PRO.5.1 Establish quality goals
- PRO.5.2 Define quality metrics
- PRO.5.3 Define quality activities
- PRO.5.4 Perform quality activities
- PRO.5.5 Assess quality
- PRO.5.6 Take corrective action

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

SUP.3 Perform quality assurance

- SUP.3.1 Select project standards
- SUP.3.2 Review software engineering activities
- SUP.3.3 Audit work products
- SUP.3.4 Report results
- SUP.4.5 Handle deviations

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com

ENG.5 Integrate and test software

- ENG.5.1 Determine regression test strategy
- ENG.5.2 Build aggregates of software units
- ENG.5.3 Develop tests for aggregates
- ENG.5.4 Test software aggregates
- ENG.5.5 Develop tests for software
- ENG.5.6 Test integrated software

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

The Assessment Process

- Plan the assessment logistics:
  - identify the sponsor(s)
  - identify the participants
  - identify resources
  - schedule data collection
  - schedule findings report/action plan
- Notify participants
- Develop data collection instrument
- Validate data collection method
- Conduct interviews/review artifacts
- Analyze and report results

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com

Process Areas Assessed

- PRO - Project
- SUP - Support
- ORG - Organization
- CUS.8 - Assess Customer Satisfaction

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Data Collection Form

Process Area	Process	Questions
CUS.8 Assess customer satisfaction		
	CUS.8.1 Determine customer satisfaction level.	
		Is the end user of your products or services the same as your customer?
		How do you determine whether the products that you deliver satisfy the needs of your customers? End users?
		Do you analyze field performance data on your delivered products?
		Do you conduct customer satsifaction surveys, user group meetings or use some other mechanism to gather information from your customers? What do you do?
	CUS.8.2 Compare with competitors. N/A	
	CUS.8.3 Communicate customer satisfaction.	
		How do you communicate customer satisfaction data throughout the organization?

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com

Data Collection Form X-Axis

- Evidence/Citation
- Comments
- Process Performance
- Performance Management
- Work Product
- Process Definition
- Process Resources
- Process Measures
- Process Control
- Process Change Management
- Continuous Improvement

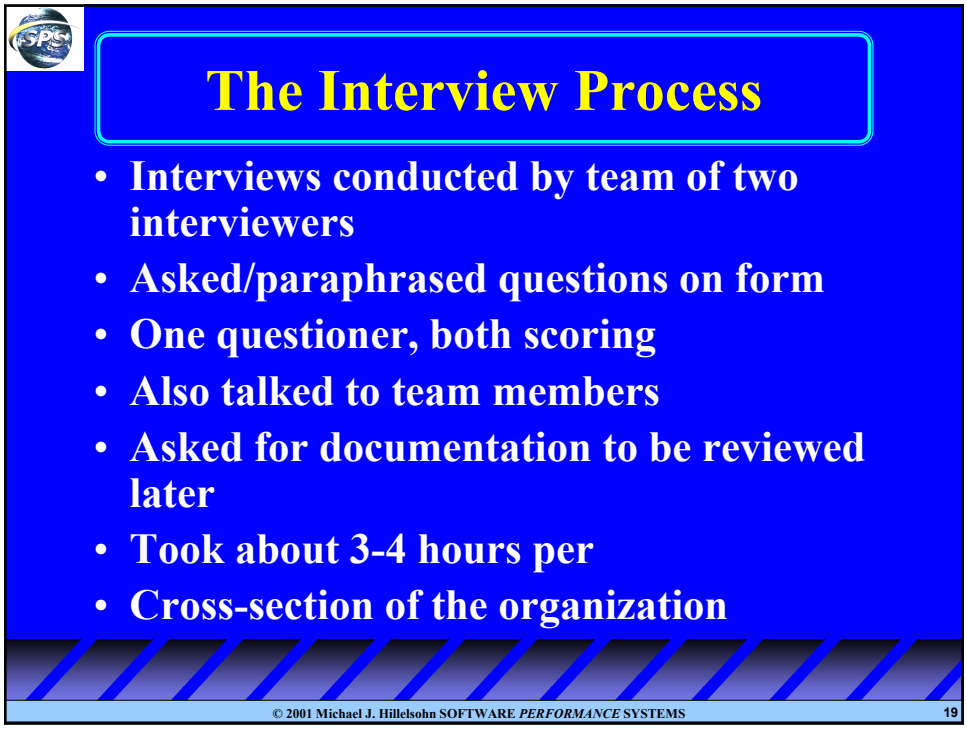
© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

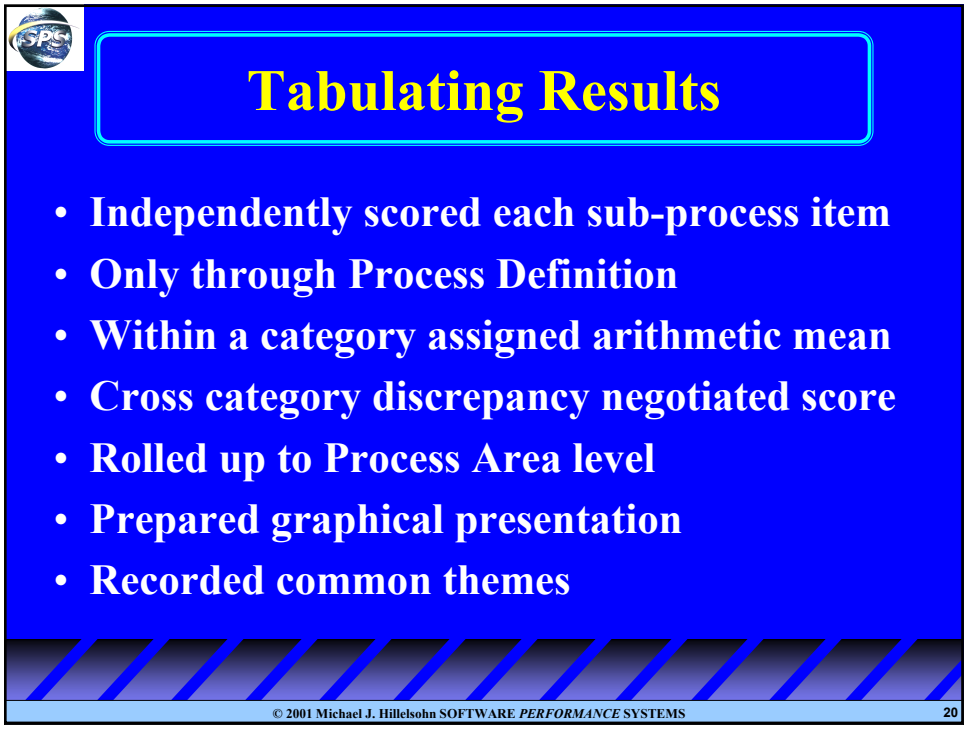
Data Collection Form - Scoring

Well managed, improving Process	86 to 100% Fully Achieved
Metrics & Controls	51 to 85% Largely Achieved
Documented Process	16 to 50% Partially Achieved
No Process	0 to 15% Not Achieved

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

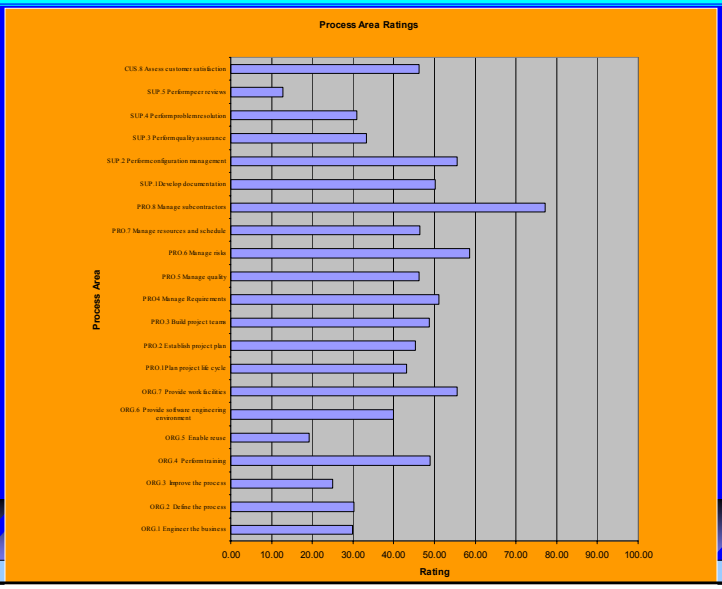
© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com





© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com

Sample Data



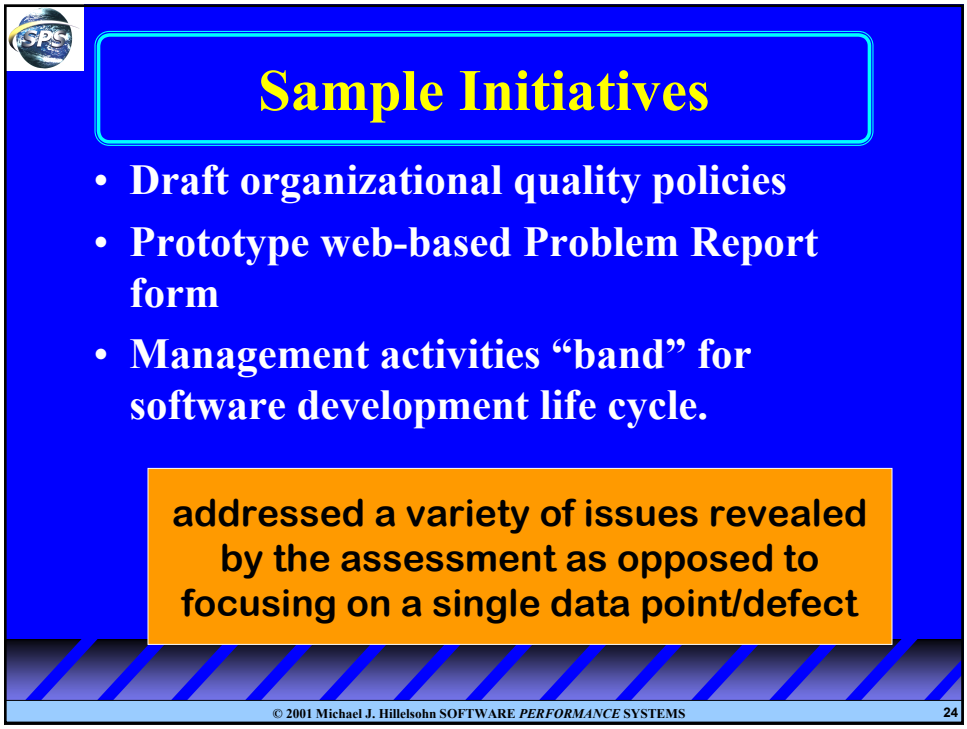
21

Root Cause Analysis

- Subsequent discussions with senior staff
- Quantitative data didn't give complete picture
- Organizational issues
- SPICE framework both provided insights and glossed over some systemic defects

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS
22

© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com



© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com

Value of the Assessment

- Sensitized staff to senior management priorities
- Gathered insights, using a common software engineering framework, across organization
- Created a baseline "state of the practice"
- Revealed systemic issues to be addressed by innovative solutions
- Prompted some organizational changes
- Provided set of software engineering practices to be used by the organization as a frame of reference

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Summary

- SPICE is comprehensive enough to reveal insights into an organization's software engineering practices.
- Senior management sponsorship of adoption of software engineering frameworks moves organizations on the path to success.
- Software engineering frameworks are useful quality tools without targeting certification.

© 2001 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com

© 2000 Michael J. Hillelsohn
SOFTWARE *PERFORMANCE* SYSTEMS
(703) 797-7707 mhillelsohn@goSPS.com

QWE2002 Paper 7M

Mr. Peter Gooch
(Computer Associates)

Process Management Maturity

Key Points

- Process Management Maturity.
- Enterprise Project Management.
- Quality Assurance.

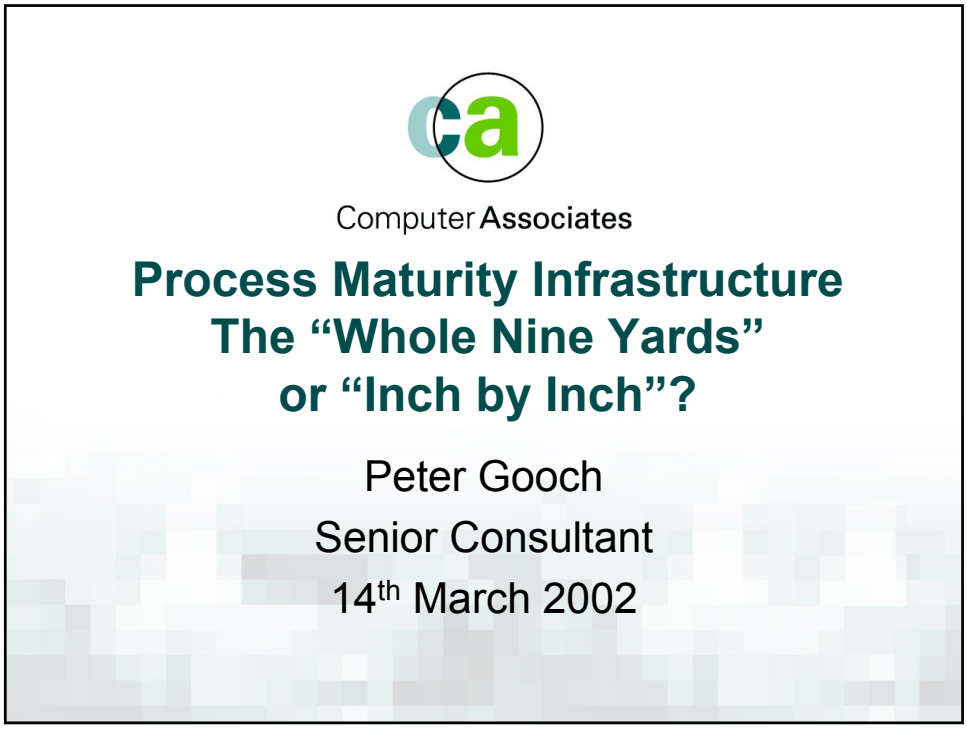
Presentation Abstract

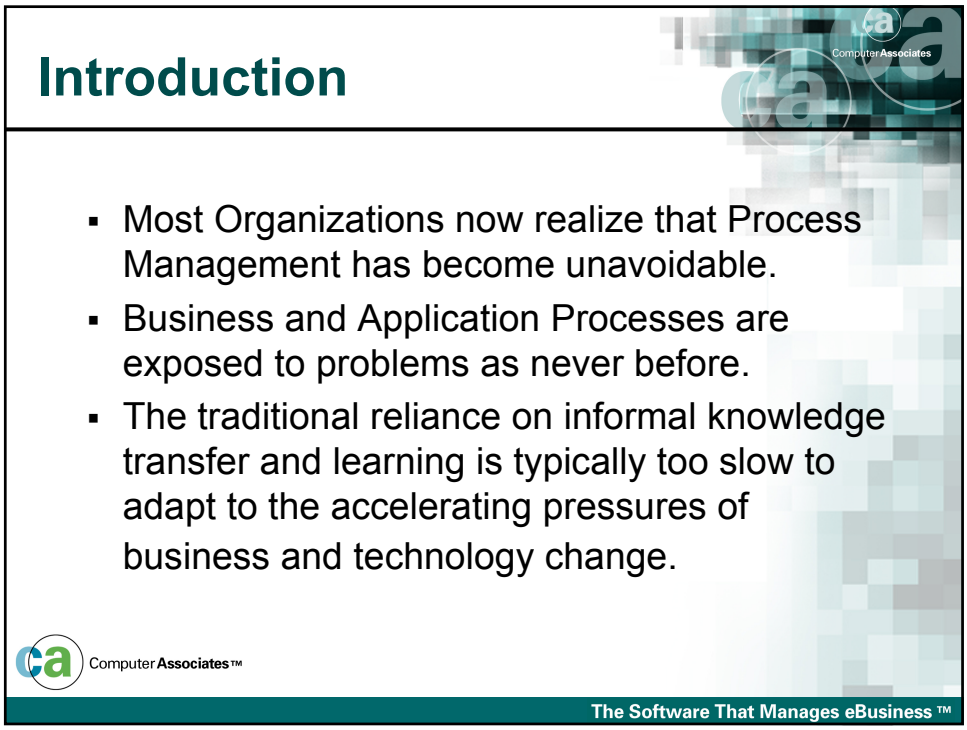
The Whole Nine Yards or Inch by Inch? - Which strategy is the right one? As IT organizations become more involved in the brave new world of e-business development, most have discovered that Process Management has become unavoidable. Both the business and application processes are exposed to problems as never before. The traditional reliance on informal knowledge transfer and learning is typically too slow to adapt to the accelerating pressures of business and technology change.

By factoring in an organizations business goals, culture, industry, and demographics into each of the available options, it is a fairly simple process to determine which options should be quickly eliminated, and which process management solutions merit further consideration. From that point, we will be able to consider the pros and cons of the short list of process management options. This session is ideal for managers who are in the process of evaluating (or re-evaluating) enterprise process management and what impact it would have upon their enterprise.

About the Author

Peter Gooch is a Senior Consultant for Computer Associates in the United Kingdom. He has over 18 years experience as an IT professional, the last 8 of which have been spent specialising in the Process and Project Management market space. Peter provides assistance to companies in improving their overall productivity via the implementation of 'Best Practice' and formal 'Process Management'.





Copyright © 2001 Computer Associates International, Inc. All rights reserved.

Analysts predict that by 2004:

- Electronic Commerce will hit $6.8 trillion – Forrester Research
- Transactions on e-marketplaces will reach $2.8 trillion – AMR Research
- Businesses will buy $2.8 trillion in supplies over the Internet – AMR Research
- Companies will spend $10.2 billion to store and share their employees' knowledge over the Net – IDC
- Corporations will invest $12.2 billion by 2004 on linking customers, sales and marketing over the Web – Meta Group

Source: Business Week, *"E-Biz: Down But Hardly Out,"* March 26, 2001

Computer Associates™

The Software That Manages eBusiness ™

So – what should we do?

- IT organizations are searching for the best way to deliver high quality software applications quickly and cheaply.
- What is needed is a process oriented approach to software development.
- But, where do we begin?
- And, what of the many options is right for your organization? The answer is…..

Computer Associates™

The Software That Manages eBusiness ™

Computer Associates

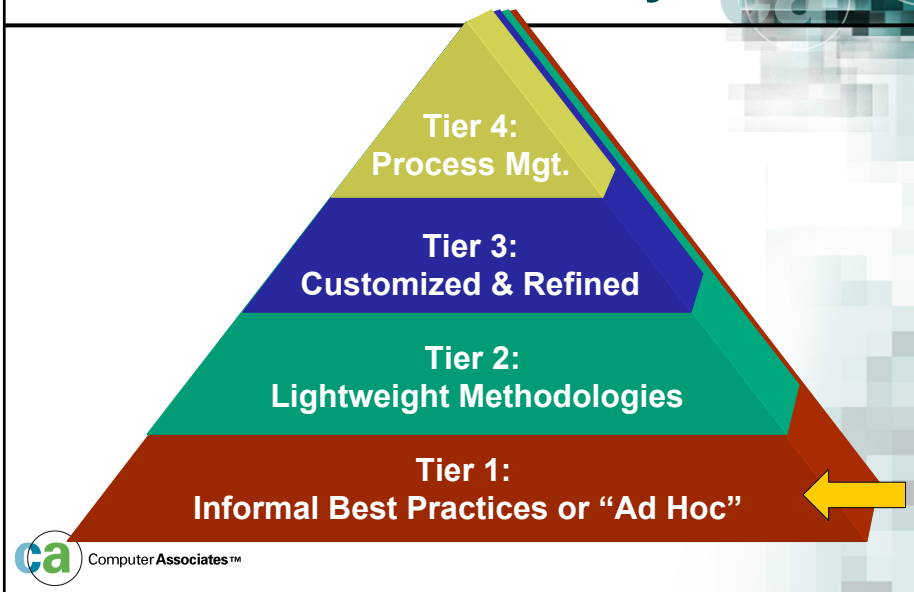
Copyright © 2001 Computer Associates International, Inc. All rights reserved.

Computer Associates

IT DEPENDS!

Lets examine some of the different ways available to improve Process Maturity

Tiers of Process Maturity



Computer Associates™

The Software That Manages eBusiness ™

Computer Associates

Copyright © 2001 Computer Associates International, Inc. All rights reserved.

Tier 1: Informal Best Practices

- "Ad Hoc"
- Every organization uses some Tier One characteristics
- "It's the way the we have always done things around here"
- "Walk over and ask the expert"

Computer Associates™

The Software That Manages eBusiness ™

Tier 1: Informal Best Practices

Advantages

- Already in Place
- No (additional upfront) cost
- No additional Management Effort is needed

Computer Associates™

The Software That Manages eBusiness ™

Copyright © 2001 Computer Associates International, Inc. All rights reserved.

Tier 1: Informal Best Practices

Disadvantages

- Difficult to Modify and Improve
- Difficult to clearly communicate to everyone
- Slow to adapt to changing conditions
- Process may be incorrect or flawed
- Hard to incorporate feedback
- Organization does not capture the information as resources relocate

Computer Associates™

The Software That Manages eBusiness ™

Tier 1: Informal Best Practices

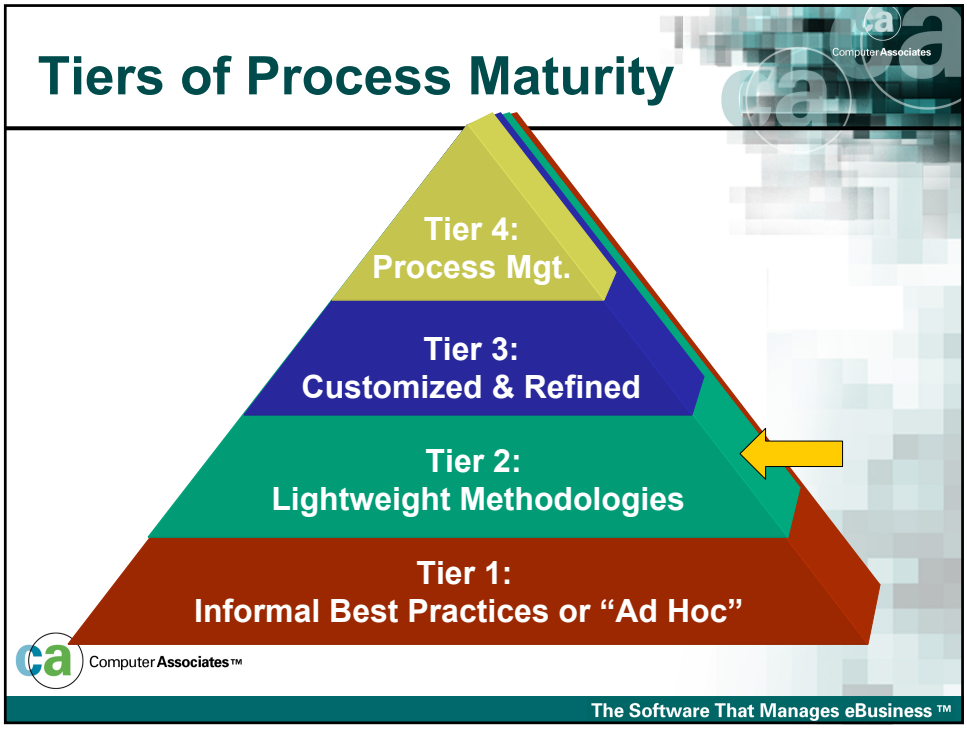
When to use…

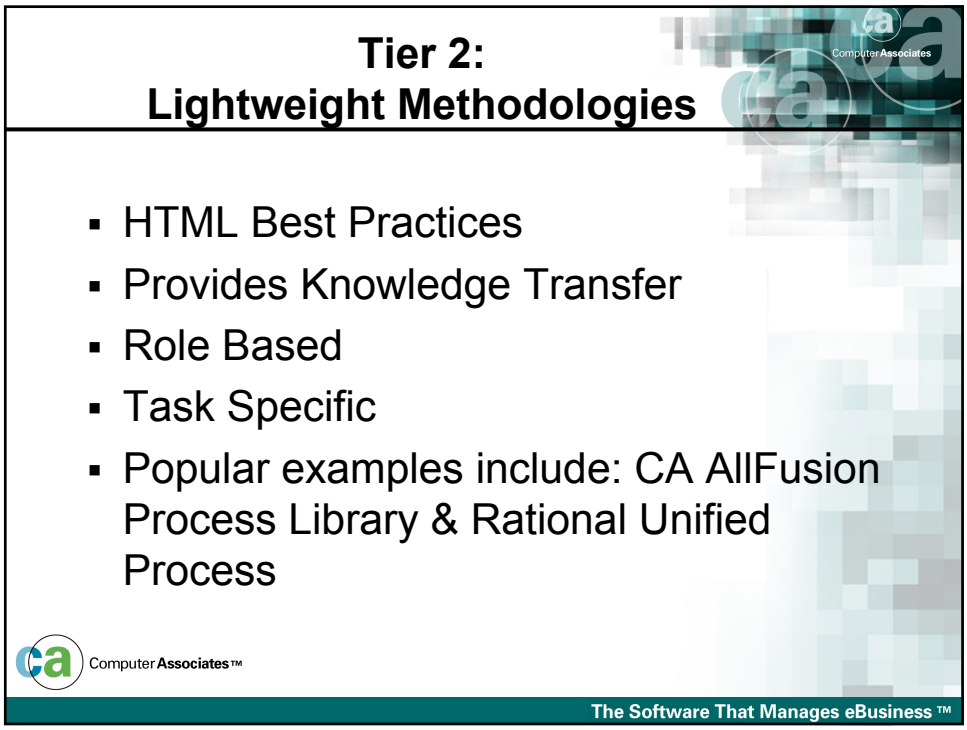
- Small IT Shops
- Centralized Development – One location
- No Organization or Support for Process Management

Computer Associates™

The Software That Manages eBusiness ™

Copyright © 2001 Computer Associates International, Inc. All rights reserved.





Computer Associates

Copyright © 2001 Computer Associates International, Inc. All rights reserved.

Tier 2: Lightweight Methodologies

Advantages

- Simple to Implement
- Little to No Training Needed
- Retail Best Practices can offer very effective Knowledge Transfer for simple or well defined tasks
- Requires Absolutely No Commitment
  - from Executives
  - from Management
  - from Team Members
- Lets us to show that we are doing something – even if nothing changes.

Computer Associates™

The Software That Manages eBusiness ™

Tier 2: Lightweight Methodologies

Disadvantages

- Retail Processes may not be available or applicable to complex or unique tasks
- This purchased knowledge may never make it into the project plan
- Impossible to Modify and/or Improve
- Organization does not capture the lessons learned
- No way to Manage the Process
- Difficult to determine the ROI to the Organization

Computer Associates™

The Software That Manages eBusiness ™

Computer Associates

Copyright © 2001 Computer Associates International, Inc. All rights reserved.

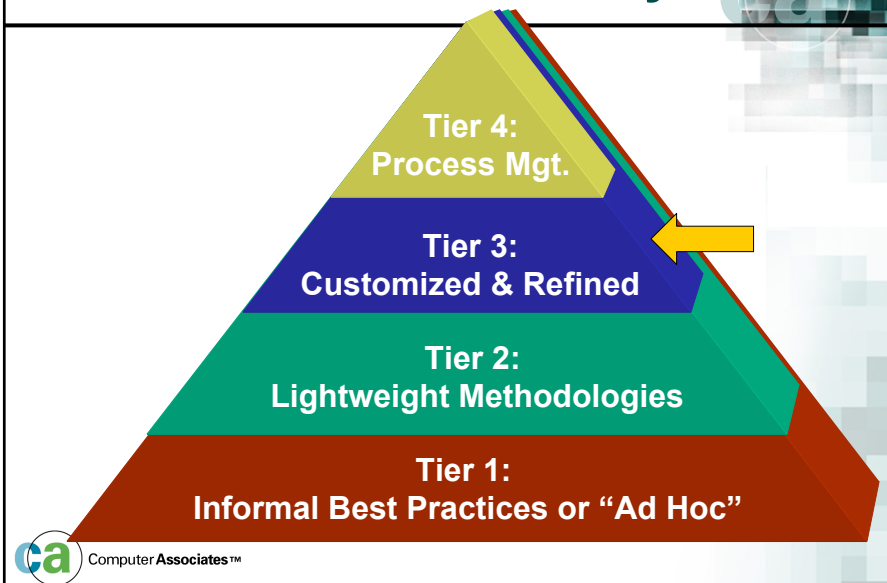
Tier 2: Lightweight Methodologies
When to use…

- Highly Skilled Team Members
- No or Limited Management and/or Executive Support or Interest implement Process Mgt
- Quick Fix Needed - “We got to do something to get Management off of our back”
- No External Motivation to Improve
  - Applications are not mission critical
  - No Competitive Pressures
  - Clients are powerless to force issue
  - Application Development is not Revenue Dependent or Revenue Related

Computer Associates™

The Software That Manages eBusiness ™

Tiers of Process Maturity



Computer Associates™

The Software That Manages eBusiness ™

Copyright © 2001 Computer Associates International, Inc. All rights reserved.

Tier 3: Customized and Refined Processes

- The modification and/or improvement of retail or homegrown Methodologies
- Best Practices are tailored to the organization
- Processes are integrated into the Project Plans
- Projects are Estimated and Defined
- Resources are given skill level classifications
- Examples include CA AllFusion Process Management Suite

Computer Associates™

The Software That Manages eBusiness ™

Tier 3: Customized & Refined Processes

Advantages

- Increased Productivity of Team Members through more specific and role defined processes
- Continuous Process Improvement
- Captured Knowledge and Experiences
- Ability to better manage the Application Development through real time Exception Reports
- Reduced Development Time and Cost
- Documented Financial Justification

Computer Associates™

The Software That Manages eBusiness ™

Computer Associates

Copyright © 2001 Computer Associates International, Inc. All rights reserved.

Tier 3: Customized & Refined Processes

Disadvantages

- Requires executive support to succeed
- System must be managed
- Takes time to implement
- Results must be tracked and justified
- Requires an upfront cost
- May result in initial Team Member pushback

Computer Associates™

The Software That Manages eBusiness ™

Tier 3: Customized & Refined Processes

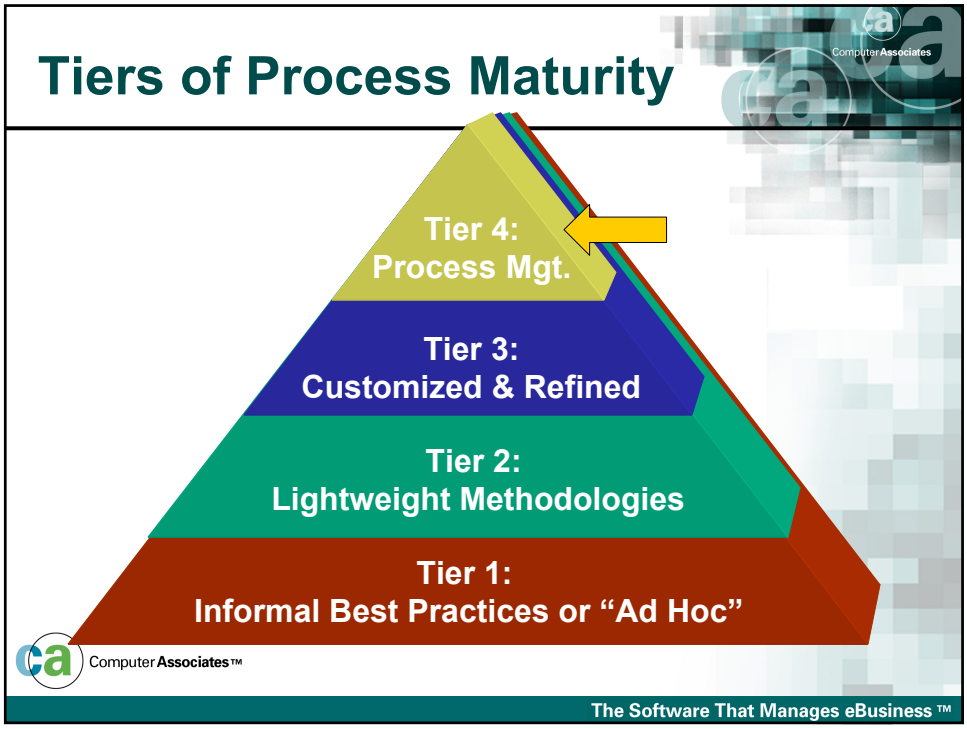
When to use…

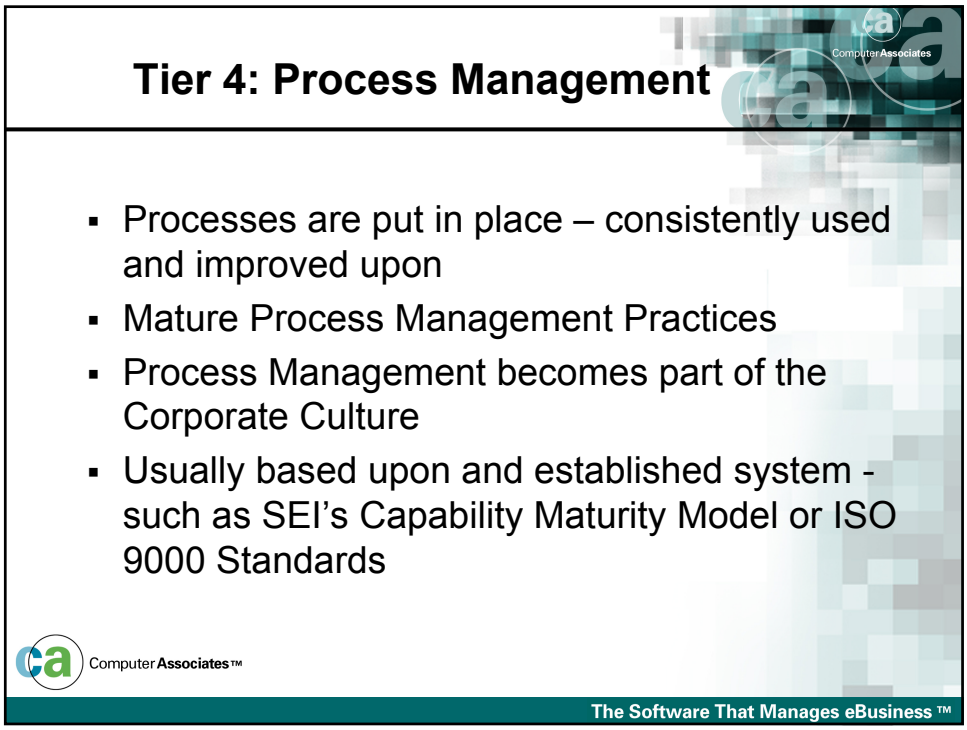
- Medium to Large Development Shops
- Decentralized IT Development
- Varying Team Member Skill Levels
- IT benefits by Project Reuse
- Real Management and/or Executive Support
- External Motivation (Pressure) to Improve
  - Applications are mission critical
  - Competitive Motivation
  - Clients have the power to demand good applications through their purchase power
  - The Application Developed effects the Revenue

Computer Associates™

The Software That Manages eBusiness ™

Copyright © 2001 Computer Associates International, Inc. All rights reserved.





Computer Associates

Copyright © 2001 Computer Associates International, Inc. All rights reserved.

Process Management consists of….

- Knowledge Transfer
- Intellectual Capital Reuse
- Business Intelligence
- Project and Resource Management
- Requirements Management
- Team Collaboration
- Knowledge Portals

Computer Associates™

The Software That Manages eBusiness ™

Tier 4: Process Management

Advantages

- IT Development now becomes a Strategic Asset to the organization
- Provides for Revenue Generation
- Shows Cost Savings through Increased Development Productivity
- Results in a Better Technical Product and/or Service

Computer Associates™

The Software That Manages eBusiness ™

Computer Associates

Copyright © 2001 Computer Associates International, Inc. All rights reserved.

Tier 4: Process Management

Disadvantages

- Takes Effort
- Takes Time to Implement
- Requires a Financial Investment
- Requires Organizational Commitment
- Positive Results are not immediate
- Usually requires someone to take a degree of professional risk - as the program could possibly fail to show results
- Often involves a change in corporate culture

Computer Associates™

The Software That Manages eBusiness ™

Tier 4: Process Management

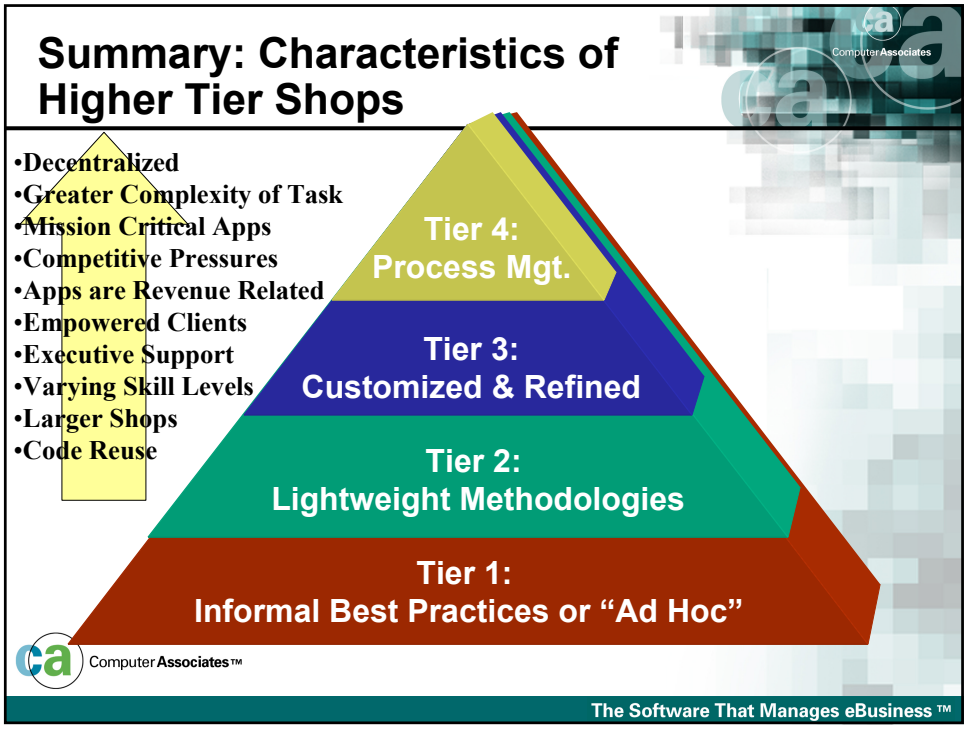
When to use…

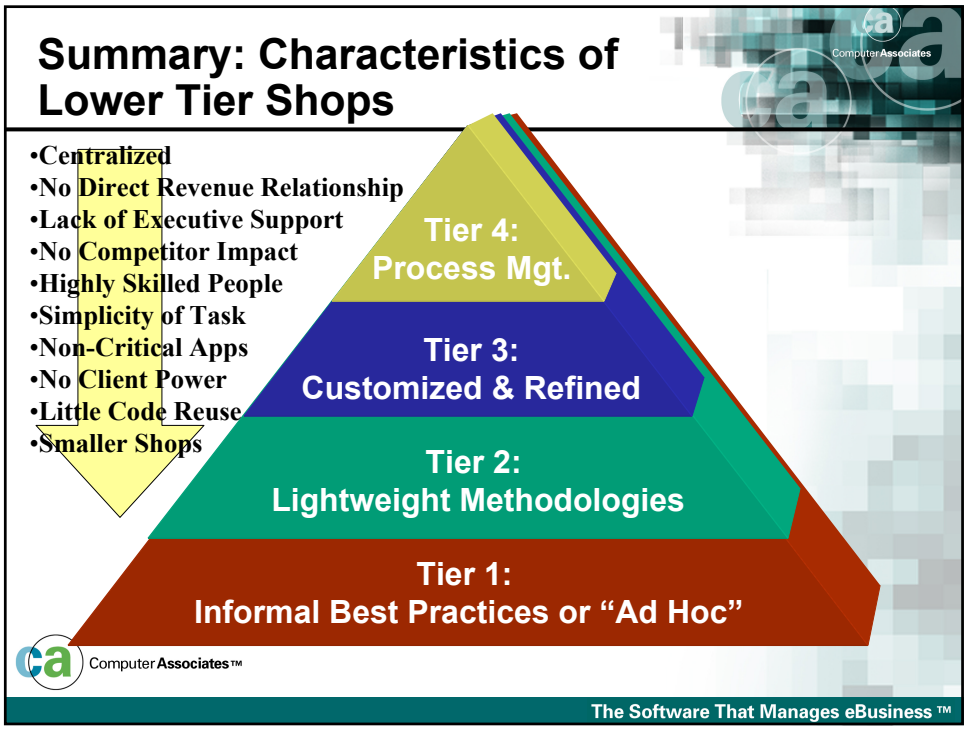
- Natural Progression of Level 3
  - Medium to Large Development Shops
  - Decentralized IT Development
  - Varying Team Member Skill Levels
  - IT benefits by Project Reuse
  - Real Management and/or Executive Support
  - External Motivation (Pressure) to Improve
- Should be justified by ROI
- Once we have quantified the benefit that defined processes have provided – we decide if it is a benefit to go the whole distance.

Computer Associates™

The Software That Manages eBusiness ™

Copyright © 2001 Computer Associates International, Inc. All rights reserved.





Copyright © 2001 Computer Associates International, Inc. All rights reserved.

Conclusions

- There is no single right answer for every organization
- Determine what solution is realistic for the organization
- Determine and document the level of executive commitment before beginning
- Develop a roll-out plan with benchmarks
- Address the areas that most benefit your particular organization
- Quantify and Qualify Results to justify the benefits of the project

Computer Associates™

The Software That Manages eBusiness ™

Copyright © 2001 Computer Associates International, Inc. All rights reserved.

QWE2002 Paper 8M

Mr. Anders Claesson
(Enea Realtime AB)

A Risk Based Testing Process

Key Points

- Learn the steps and procedures in risk based testing.
- Learn how to identify, prioritize and evaluate system risks.
- Be able to assess the current quality of the system.

Presentation Abstract

The approach of using risk based testing has become a widespread technique in software testing. By using risk based testing you can focus the testing effort on the critical areas of the system. This presentation describes a process based on experience of how to apply a risk based testing strategy.

The advantage of a risk based approach is to show the people involved in the project where problems and risks in the system might occur in advance. The purpose of testing is to evaluate those identified risks. If no faults are discovered the risks can be lowered. If faults are found the risks can be lowered by conducting more testing in the risk area and to make the designers correct the faults found. The quality of the system can be determined anytime during the project by looking at the current status and priority of the risks in your risk list.

About the Author

Anders have been working with testing of software and systems, mainly real-time and telecommunication systems since 1982, but also large client server systems. Anders is currently working as a consultant at Enea Realtime AB. Anders is writing test processes and methods for system testing. He also works with test process improvements for different customers.

ENEA Realtime

A Risk Based Testing Process

ENEA Realtime

Contents

- Maturity levels in testing.
- Why risk based testing?
- General steps in the testing process.
- Workflow.
- Risk focus not enough.
- How to start using risk based testing.
- References.
- Summary.

© Anders Claesson

Maturity Levels in Testing

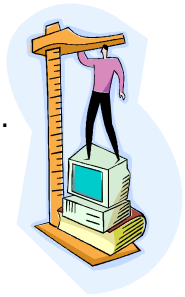
5 Preventive and optimizing activities.

4 Identify/evaluate the risks of using the system.

3 Show that the system doesn't work.

2 Show that the system works.

1 No difference between testing and debugging.



Why Risk Based Testing?

Shift focus

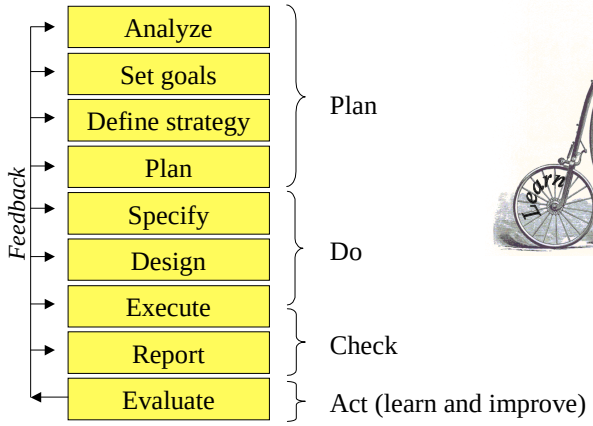
From

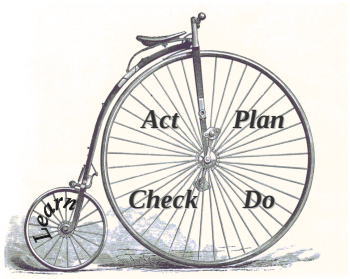
- Test cases based on specifications.
- Report executed no. of test cases and faults found.

To

- Identify system risks.
- Prioritize tests according to risks.
- Report current risk level.

General Steps in the Testing Process





A Risk Based Testing Process Anders Claesson 2001-09-15 Page 5 (20) www.enea.se

Workflow

1. Plan the analysis
2. Identify risks
3. Analyze risks
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. Monitor risks
9. Report test results

1. Plan for the risk based testing.
2. Identify and list potential risks.
3. Analyze the risks.
4. Prioritize the risks.
5. Plan for how to reduce/eliminate and test each risk.
6. Resolve and test the risks according to the plan.
7. Analyze the test results.
8. Monitor the risks.
9. Report test results.

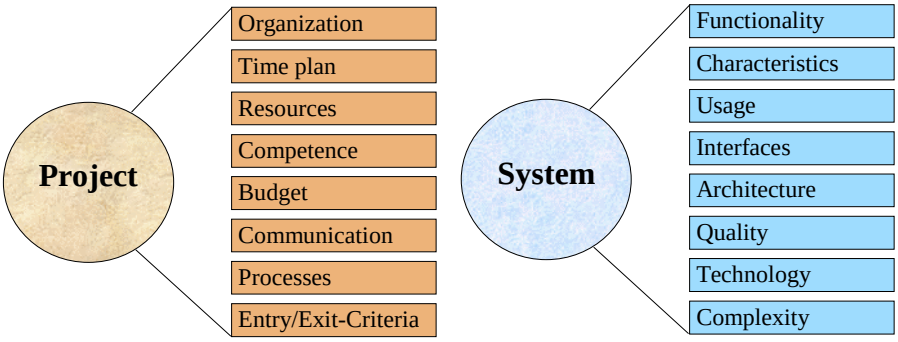
A Risk Based Testing Process Anders Claesson 2001-09-15 Page 6 (20) www.enea.se

ENEA Realtime

1. Plan the Risk Analysis

1. What should be analyzed and when?
2. Who participates?
3. How should the analysis be performed?

1. Plan the analysis
2. Identify risks
3. Analyze risks
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. Monitor risks
9. Report test results

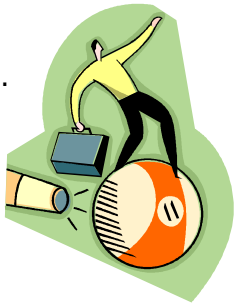


ENEA Realtime

2. Identify Risks

1. Gather input information.
2. Identify risk factors:
   - Weak/unclear specifications.
   - New technology.
   - Usage of third-party products.
3. View the system as a whole.
4. Identify events that might lead to failure.
5. Describe possible consequences.
6. Analyze previous trouble reports.
7. Analyze problems in current version.

1. Plan the analysis
2. Identify risks
3. Analyze risks
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. Monitor risks
9. Report test results



© Anders Claesson

ENEA Realtime

2. Identify Risks

1. Plan the analysis
2. **Identify risks**
3. Analyze risks
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. Monitor risks
9. Report test results

Risk identification methods:

1. Risk checklists.
2. Past experience and historical data.
3. Risk brainstorming meeting.
4. “What-iffing”.
5. What, where, when, how, who – analysis.
6. Six thinking hats questioning.
7. Risk factor analysis and questioning (will the system be reliable?).
8. Cause and effect graphing.

ENEA Realtime

2. Identify Risks

1. Plan the analysis
2. **Identify risks**
3. Analyze risks
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. Monitor risks
9. Report test results

Risk testability:

1. Clarify the risk (e.g.“unstable system”).
2. Actors involved.
3. System parts involved.
4. Scenario/use cases.
5. Configuration.
6. Actions triggering the risk.
7. Input data.
8. System state.

ENEA Realtime

3. Analyze Risks

1. Break down the risk into subcategories.
2. Analyze probability and consequences.

1. Plan the analysis
2. Identify risks
3. **Analyze risks**
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. Monitor risks
9. Report test results

Probability	Consequence
(4) Unavoidable	(4) Catastrophic
(3) Frequent	(3) Damaging
(2) Occasional	(2) Hindering
(1) Rare	(1) Annoying

A Risk Based Testing Process Anders Claesson 2001-09-15 Page 11 (20) www.enea.se

ENEA Realtime

4. Prioritize Risks

High Must be resolved/evaluated.
Medium Needs attention.
Low Looked into if time allows.

1. Plan the analysis
2. Identify risks
3. Analyze risks
4. **Prioritize risks**
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. Monitor risks
9. Report test results

Probability

Consequence	Rare (1)	Occasional (2)	Frequent (3)	Unavoidable (4)
Catastrophic (4)	High/medium	High	High	High
Damaging (3)	Low	Medium	High	High
Hindering (2)	Low	Low	Medium	Medium
Annoying (1)	Low	Low	Low	Medium/low

A Risk Based Testing Process Anders Claesson 2001-09-15 Page 12 (20) www.enea.se

ENEA Realtime

5. Plan Reduction/Test

1. Preventive activities.
2. Specify test strategies.
3. Specify type of test cases.

1. Plan the analysis
2. Identify risks
3. Analyze risks
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. Monitor risks
9. Report test results

Prio	Risk description	Probability * consequence	Actions
1	**Risk:** Unstable system platform. **Cause:** Platform components developed by independent vendors. Components not tested together.	4 * 4 = 16	**Risk reduction and evaluation strategy**: Perform platform tests to evaluate the interactions between components. **Test objects**: Platform test of system xxx. **Test cases**: - Test of individual important functions. - Tests of related function groups, bottom up to the application interface level.

ENEA Realtime

6. Test/Evaluate the Risks

1. Group test cases.
2. Start with a broad coverage.
3. Test the highest risks first.
4. Run additional test cases in unstable areas.
5. Report found faults with reference to the associated risk.
6. Use the risk list to prioritize corrections.

1. Plan the analysis
2. Identify risks
3. Analyze risks
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. Monitor risks
9. Report test results

ENEA Realtime

7. Analyze Test Results

1. Identify test coverage.
2. Identify where failures occurred.
3. Analyze occurrence and severity of the found faults.

1. Plan the analysis
2. Identify risks
3. Analyze risks
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. **Analyze test results**
8. Monitor risks
9. Report test results

Occurrence

Severity	Never	Rarely	Sometimes	Usually	Always
Catastrophic	Medium	High	High	High	High
Hazardous	Low	Medium	Medium	High	High
Major	Low	Low	Medium	Medium	High
Minor	Low	Low	Low	Low	Medium
No effect	Low	Low	Low	Low	Low

ENEA Realtime

8. Monitor the Risks

1. Identify areas with highest risks.
2. Identify possible causes to severe faults.
3. Change test strategies and plans.
4. Repeat any of the previous steps in the risk process.

1. Plan the analysis
2. Identify risks
3. Analyze risks
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. **Monitor risks**
9. Report test results

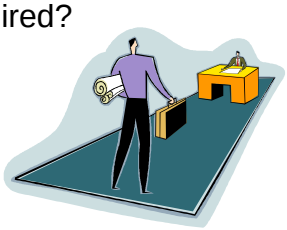
Prio	Risk	P * C	Actions	Results
2(1)	Unstable system platform.	3*3=9 (4*4=16)	**Risk reduction and evaluation strategy**: Perform platform tests to evaluate the interactions between components. **Test objects**: Platform test of system xxx. **Test cases**: - Test of individual important functions. - Tests of related function groups, bottom up to the application interface level.	- Test results reviewed from vendor tests of third-party products. - Integration tests performed on all platform components. - 48 test cases executed for individual functions. 3 major faults found, 1 minor (TR-ref). - Major problems in execution were the definition of the configuration parameters for the platform components.

ENEA Realtime

9. Report the Test Results

1. Plan the analysis
2. Identify risks
3. Analyze risks
4. Prioritize risks
5. Plan reduction/test
6. Test/evaluate
7. Analyze test results
8. Monitor risks
9. Report test results

1. Where are the major problem areas?
2. How much has been tested?
3. How many and which kind of faults were found?
4. Which risks have been evaluated and how?
5. What is the current risk level?
6. How much additional testing is required?
7. Is the system ready for release?



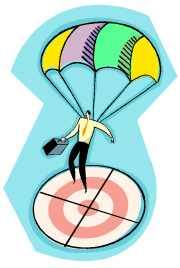
ENEA Realtime

Risk Focus not enough

User Requirements, needs, expectations, values.

System Functions, characteristics.

Risks Failure to meet or fulfill the above.



How to start using Risk Based Testing

1. The risk analysis is the most important part.
2. Select and execute test cases based on the risks.
3. Adjust the testing effort based on the current risk areas.
4. Report the test results against the risks that have been evaluated.

Summary

- A risk based approach focus on the stability and the reliability of the system.
  1 - Risk analysis planning
  2 - Identification.
  3 - Analysis.
  4 - Prioritization.
  5 - Planning.
  6 - Test/Evaluation.
  7 - Analysis.
  8 - Monitoring.
  9 - Reporting.

- You also need to cover requirements, functions and characteristics.

A Risk Based Testing Process

Anders Claesson, Enea Realtime - Test
Enea Realtime AB
www.enea.se
Box 232, S-183 23 Täby, Sweden
Tel: +46-8-50 714 000
Fax: +46-8-50 714 040
E-mail: ancl@enea.se

© 2001 Enea Realtime AB
All rights reserved

Abstract

The approach of using risk based testing has become a widespread technique in software testing. Risk based testing is a way of preventive care for the system under development. You identify any potential problems (risks), analyze and evaluate the risks involved.

This paper describes how to apply a risk based testing strategy in your testing process. By using risk based testing you can focus the testing effort on the most critical areas of the system.

The advantages of a risk-based approach are both to show, and involve, the project management, requirement authors, system designers, testers and maintenance people to discuss where potential problems and risks in the system might be. The purpose of testing is to evaluate those identified risks. If no faults are discovered the risks can be lowered. If faults are found the risks can be lowered by conducting more testing in the risk area and to make the designers correcting the faults found.

The quality of the system can be determined anytime during the project by looking at the current status and priority of the risks in your risk list.

The **basic steps** in the risk based testing process are:

1. Plan the risk analysis and the risk based testing.
2. Identify potential risks.
3. Analyze the risks.
4. Prioritize the risks.
5. Plan for how to test, reduce and/or resolve each major risk.
6. Test (evaluate, reduce and resolve) the risks.
7. Analyze the test results.
8. Monitor the risks.
9. Report the test results.

The risk based testing process can be used to:
- Give directions for which test strategies to use.
- Select test objects and the most important test cases.
- Explore the risks when performing the test execution.
- Evaluate the current level of system quality and risk.
- Report test results related to the risks that were evaluated.
- Provide input for the decision to continue or stop testing.

Contents

1 Introduction 3
2 The steps in the risk based testing process 4
2.1 Plan for the risk based testing 6
2.2 Identify and list potential risks 7
2.3 Analyze the risks 10
2.4 Prioritize the risks 11
2.5 Plan for how to test, reduce and/or resolve each major risk 12
2.6 Test (evaluate, reduce and resolve) the risks 13
2.7 Analyze the test results 14
2.8 Monitor the risks 15
2.9 Report the test results 15
3 How to start using a risk based testing process 16
4 Conclusions 16
5 References 17
Appendix A: The general steps in a testing process 18
Appendix B: Maturity levels in testing 19
Appendix C: The six thinking hats 20

1 Introduction

As a tester you might be challenged by having to test a product with very limited amount of specifications of how it is supposed to work, with limited resources and within a short time frame. In most cases it might be more of a rule than the exception.

Your job is to find as many faults in the product as possible within the short time period available (and making the designers correcting the faults). Your job is also to give the project manager information regarding the products status and if it might be ready for release. The readiness for release can be expressed as the risk level for the system to fail under any conditions of the expected usage or operation. Potential problems, or risks, are identified in a risk analysis.

Problems can be dealt with in two ways:

Reactive You react only to what is happening. I.e. the system crash and you try to fix it.

Proactive You try to predict problems that might happen and take actions to avoid/reduce and evaluate the problem areas.

To act proactively is the most cost and time effective solution, but since you cannot predict everything you also need to have a reactive strategy. The proactive approach involves risk analysis, cause and effect analysis and contingency planning. The reactive approach involves a good ad-hoc and exploration strategy. It can also be combined with a more scientific approach of investigating unexpected or unknown events (see [Claesson1 -01]).

The most important part of the risk based testing approach is the risk analysis. In the risk analysis you can find out which parts of the system that needs the most attention in testing [Bach -99].

The main purpose of risk based testing is to:

- Identify issues (early) that might become problems.
- Specify the potential problems/risks in a prioritized list (to make them visible).
- Reduce the likelihood of the problems to occur.
- Evaluate/explore the (identified) potential problems by performing tests.
- Reduce the impact if the problems do occur.
- Increase the chances of detecting and removing any hazards, faults, inconsistencies, ambiguities and problem issues in the system.
- Continuously adjust the testing effort to stay focused on the currently most important risks.

Fig. 1 A potential risk that has not been taken care of might show itself in a less desirable way.

2 The steps in the risk based testing process

The risk based testing process only deals with technical issues in the product under development. That is technical issues that can be evaluated in testing. To handle the risks you need a good process and strategy where you identify, plan and control any potential problems. The process below outlines the steps involved in the risk based testing process

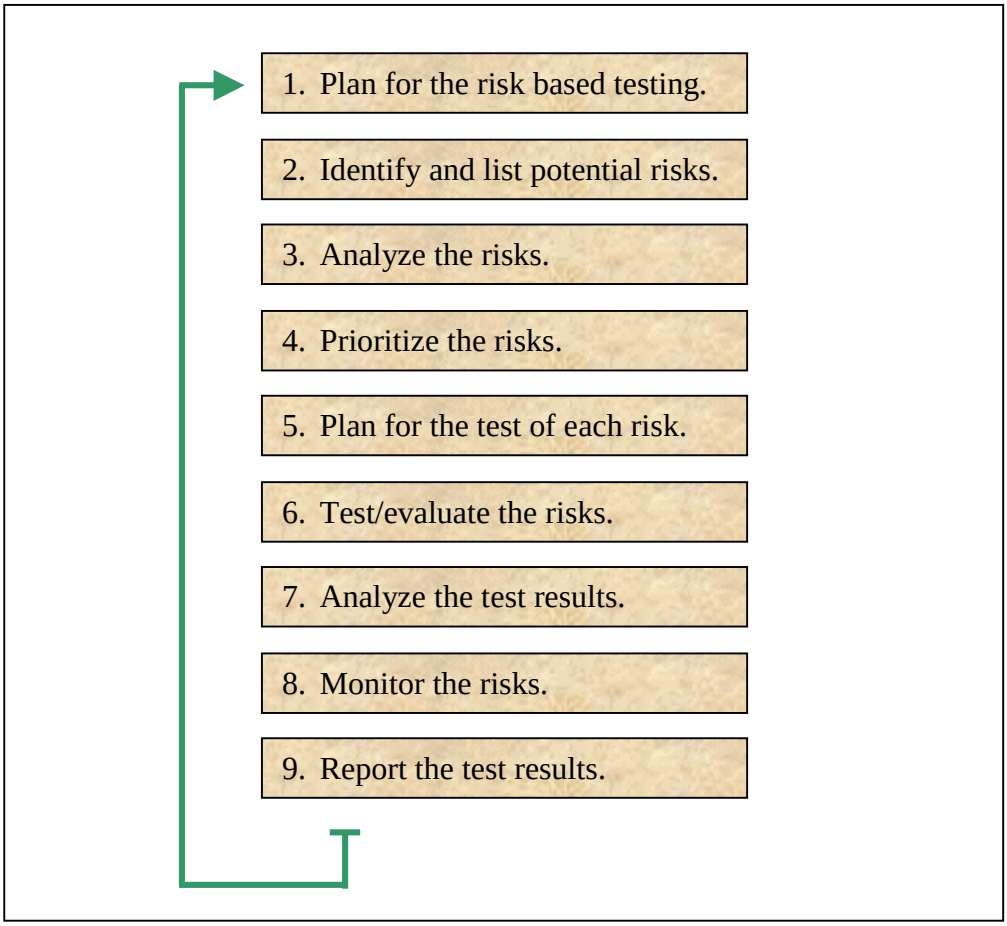


Fig. 2 The figure shows all the steps in the risk based testing process. The process is repetitive where several iterations can be made both between each of the steps and for the whole process.

Overview of the steps in the process:

1 **Plan for the risk based testing.**
Specify your goals for the risk based testing. What do you want to achieve (see *appendix A* and *B*)? What is your strategy for how to perform the risk identification, prioritization, mitigation, evaluation and monitoring of the risks? Also identify when in the development phase the risk analysis should take place, who should be involved and how the risks will be followed up and tested/evaluated.

2 **Identify and list potential risks.**
Identify important risk factors applicable for your system. I.e. what events, issues and conditions may cause problems in the system (e.g. weak specifications, complex areas, critical areas)? Identify the risks associated with the functionality, characteristics and usage of the system. Identify the actions/events in the system that might lead to failure. Describe the risks with possible causes.

3 **Analyze the risks.**
Determine the probability of an actor/user of the system will experience a failure in the identified potential problem area. Determine the impact and consequences a failure would have for the actor/user (may be different for different actors). Also consider any possible negative effects for the testing (show stoppers that might halt all test activities).

4 **Prioritize the risks.**
Prioritize each risk based on its probability and consequences. Divide the risks in three categories: high, medium and low risks. For high-level risks you need to lower either the probability, the consequences or both for the risk. For medium-level risks you need to pay attention to it but only resolve it when your tracking mechanisms indicates. For low-level risks you only have to monitor them to make sure they do not increase.

5 **Plan for how to test, reduce and/or resolve each major risk.**
Make a plan for how to deal with the risks having the highest priority. Also include a proposal for how to reduce/eliminate each risk (before the actual test execution starts). Decide which test strategies to be used for the evaluation (test specification, design and execution). Identify which test objects and test cases that are needed.

6 **Test (evaluate, reduce and resolve) the risks.**
Perform the tasks specified in the plan to reduce/eliminate the risks. Evaluate the risks by executing the test cases associated with each risk. How many test cases you need to execute per risk is determined by the priority of the risk (considering the possible system impact). Record all your test results and observations.

7 **Analyze the test results.**
Analyze and draw conclusions from the test results. The amount of test cases executed gives a certain degree of confidence and knowledge about the risks (probability and consequences). Identify if any unexpected events occurred (faults or issues) that might need further investigation. Group the identified faults in categories according to severity and occurrence. Map the found faults to corresponding risks. Consider increasing the risk level and priority if the found faults are severe, decrease if they are less severe or if no faults are found in the risk area.

8 **Monitor the risks.**
Continuously monitor the progress of the risk resolution and testing. As risks disappear or decrease in importance and new ones emerge, adjust your risk list and strategy to stay focused on the present risks and their priorities. By continuously adding information about the evaluated risks you can provide detailed information about the quality of the system anytime (to the project manager or any other part interested in the outcome of the project).

9 **Report the test results.**
Report the results of the risk based testing. Identify the remaining risk level in the system based on all planned and added test cases that have been executed, and the amount of faults found. Give recommendations regarding the system if it is ready for release or not. Identify remaining issues to be resolved.

Risk based testing is an iterative process where new discoveries, evaluation results, knowledge and new ideas continuously are added to any of the steps in the process. Since the prerequisites might change for both the project and the system from day to day it is really important to keep information up to date. Preferably by using an internal web site where the current risk list can be found (including test results and a risk evaluation). A top-ten list of found open trouble reports should also be kept with references to the associated risks.

2.1 Plan for the risk based testing

As for any test activity the risk based testing needs careful planning. Plan for what, how, when, where and by whom the risk analysis should be performed [Bach1 -99].

1. **Decide when the risk analysis is going to be performed.**
   The risk analysis can be performed during the requirement analysis, system design, unit design or in the test phase? Try to perform the risk analysis as early as possible (preferably when the requirements have been specified).

2. **Identify the prerequisites for the risk analysis.**
   - Are all people/roles available to perform the analysis?

     Example:
     - The project manager.
     - The test leader (and his/her mentor).
     - Requirement author (orderer of the system/product).
     - The person responsible for the design of the system.
     - The person responsible for quality assurance (QA).
     - The person responsible for the integration and delivery of the system (to the customer and internally to testing).
     - A designer.
     - A tester.
     - The person responsible for the operation and maintenance of the system.

   - What are the project goals and constraints?
   - What time constraints exist for the test phase?
   - To what extent will internal milestones be followed in the project (are delays accepted in design but not in testing)?
   - Do requirements (and/or other design specifications) exist and are they known/understood by all participants?
   - Does everyone know what functionality is included in the project?
   - How much functionality should be delivered, and when?
   - Which characteristics must be fulfilled?
   - What are the quality requirements?
   - What are the cost limits?
   - Are all logical and physical units and interfaces in the system (both internal and external), identified and known/understood by all participants?

2. **Decide what should be analyzed.**
   - Identify which parts of the functionality, characteristics and/or usage should be analyzed. Also analyze the system as a whole?
   - Decide how the needed input information should be obtained.
   - Which specifications are needed?
   - Which peoples input are needed?

3. **Decide how the analysis should be performed.**
   - Decide who should be responsible for the risk analysis and the follow up.
   - Which meetings should be held?
   - Which process/work procedure should be used?
   - How should the results be presented and followed up?

The risk based testing process focus only on technical issues. Traditional project risk analysis often miss this part and only concentrate on issues like: not enough people available, too short lead time, equipment may arrive late, lack of knowledge in key areas, or other organizational issues.

Therefore it is recommended to perform two separate risk analyses, one for the project issues and one for the technical issues. The project issues will probably have a high influence on the technical issues and vice versa. After the two risk analysis sessions have been held try to connect the related risks.

Output:	A plan and strategy for the risk based testing.

2.2 Identify and list potential risks

1. **Gather needed input information for the risk analysis.**
   Gather all the facts you can get in the selected areas that should be analyzed (e.g. requirements, known faults in the current version of the system).

   You may also want to interview people (e.g. customer support, designers, system architects, production responsible). Ask them to use the "six thinking hats" when giving you the information (see *appendix C* and [Focus -01], [Claesson1 -01]).

2. **Identify important factors to consider in the risk analysis.**
   The risk factors may help you to find what aspects or conditions that may cause problems and risks for the system [Schaefer -99], [Bach -99] (e.g. weak specifications, complex areas of the system, critical areas, etc.). Look for areas or issues of uncertainties [Claesson1 -01].

   You may also want to create a risk checklist specific for your system based on the general risk factors above.

3. **Identify areas to analyze.**
   Identify critical and important areas of the specifications/information for the system to be analyzed.

   Example:
   - Requirements that are missing, unclear or difficult to implement and test.
   - Design specifications which are missing or unclear.
   - Any concerns expressed by people that were interviewed.

   Identify important parts/aspects of the system to be analyzed.

   Example:
   - The system as a whole. Is the system solving the users problems? Will the user be efficient in achieving the tasks the system is intended for?
   - Functions, function groups and functional behavior.
   - System characteristics, i.e. performance, capacity, security.

4. **Identify the risks.**
   Identifying the risks is the most difficult part of the risk based testing. There are a number of different approaches that can be used.

   The risk analysis-brainstorming meeting.
   The simplest one (and yet effective) is to have a brainstorming meeting including key participants according to the risk planning (see chapter 2.1). Gather experienced testers, trouble-shooters and designers to come up with ideas about problem areas.

   View the system as a whole. Start identifying the risks from a usage perspective, i.e. does the system solve the users problems? You can also use the "six thinking hats" to explore the possibilities of identifying risks with different mind sets (see *appendix C*).

   Look at the previously analyzed uncertainties, which may impact the functionality, characteristics and/or usage of the system. Each uncertainty of importance may be translated into a risk.

   Identify known problems in the current version of the system in operation. Analyze trouble reports both from the customer and internal. Identify the cause and effect of the problems. Draw conclusions about where similar problems may occur in the new version of the system.

   Identify which actions/events in the system that leads to failure.

   Describe the risks with possible causes. The (root) cause description is the main input to determine what preventive actions may be efficient to perform.

Risk description
<Description of the risk, and the possible underlying problems (root cause) to the risk>

Table 1. Example of the format of a risk description.

What-iffing [Thinking -98], [Claesson1 -01].
The "what-iffing" approach means to ask **what if** many times related to the area to be investigated for potential risks. What-iffing involves describing an imagined situation, condition, action, state or event, and then examining associated facts, consequences and effects.

- What if we do this, what would happen then?
- What if the requirements are wrong, or the estimation of the total number of users? What are the risks involved?
- How does this function actually work? What are the risks that it will not work in this way?

Risk specification (what, where, when, how, who) [Hale -97].
This is a technique to clarify the risk by asking the same set of questions to them all. With this approach you recognize both what the risk is as well what it is not. By asking what, where, when, how, and who, you can build a clearer picture which helps in defining the potential problem or risk. By also looking at what the risk is **not** related to, it is easier to define what it actually is.

Keyword	The risk is …..	The risk is not ….
What?	The system platform is unstable.	Related to the underlying database.
Where?	On the application server where the system platform is installed.	In the load sharing function where different parts might be located on different servers.
When?	The application sends transactions to the database.	The user just reads data from the database.
How?	The system crashes and generates a core dump.	That the database gets blocked.
Who?	The user will not be able to store any information.	The super user can still access the system.

Table 2. Example of a risk specification.

Risk factor analysis [Down -94].
State questions about what is most important for the system. Specify possible factors that affect the primary question the most [Down -94]. Which questions are the most important to answer for your system [Claesson1 -01]?

Risk factors for system characteristics			
Primary question	**Secondary factor**	**Tertiary factor**	**Risk**
Will the system be reliable?	1. Error handling.	Requirements and design specifications.	Vague or missing.
		Design methods.	Not followed.
	2. Redundancy.	Load sharing.	Not fully implemented.
	3. Input validation.	Boundary value check.	Not performed consistently.
		Design rules.	Not enforced or checked.
	4. Design and code quality.	Defect prevention.	No time or motivation.
		Defect removal.	Serious faults left.
		Test coverage.	Inadequate because of lack of time.
		Test case selection.	Insufficient test analysis and test strategy.

Table 3. Example of a risk factor analysis.

What is most important for the customer and the user of the system? For a safety critical system reliability would be one of the most important issues.

The advantage of the table of questions and factors is that it provides a checklist and a useful way of integrating all the factors to come up with a risk assessment of the original question.

Preventive activities, as well as suggestions for evaluation/test of the overall or detailed risks, are also easier to find.

Cause and effect graphing.
A similar way to the risk factor analysis is to look at the cause and effect relationships of potential problems. The relationships can be specified in a fish bone diagram.

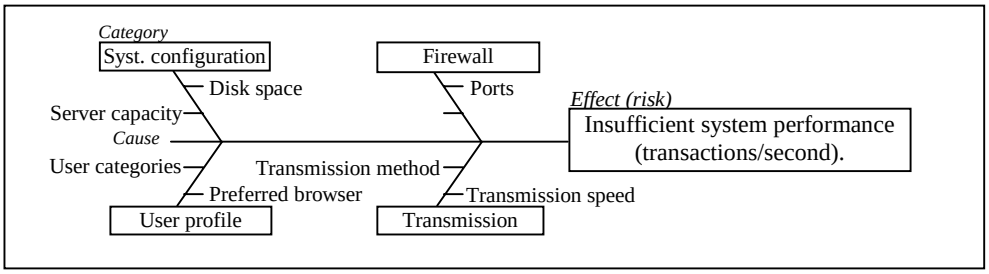


Fig. 3 Example of a cause and effect diagram where you can see what factors that might have an impact for a risk to occur [Quality -01], [Mears -95].

The fish bone diagram is useful as a basis when designing test cases. By looking at what causes a risk you get an idea of what preconditions should be set up for the test cases and what variables might be useful to change when evaluating the risk (e.g. testing with different user profiles and system configurations).

Risk testability
Just like any specification you use as input for the design of test cases, the risk must be testable. To make a risk testable it must be well defined and put into context. Risk descriptions from a brainstorming meeting are sometimes vague and would need some more analysis and definition (if you just have a yellow post-it-note saying “unstable system” it will not help that much for the specification of a suitable test strategy and associated test cases).

The following may be considered when defining a risk:

1. A suitable risk heading/summary.
2. Why it is a risk and why it may occur?
3. What assumptions have been made regarding the system, its configuration and usage?
4. When and how the risk may show itself.
5. Which actors might be involved?
6. Which system parts/components might be involved?
7. Which user observable objects (e.g. screens, menus) are involved?
8. Are there any requirements or design specifications related to the risk?
9. Is there any specific scenario (use case) involved for the risk?
10. Which actions or events might trigger the risk to occur?
11. Is there any specific input data assumed for the actions that might trigger the risk?
12. What are the expected results (outcome/system state) if the risk occurs?
13. Are there any logical conditions involved for the risk to occur (i.e. if this and that happens, then…)?
14. What is the root cause of the risk and the events leading to the occurrence of the risk?

Output: A list with specifications of potential risks.

2.3 Analyze the risks

Probability
Determine the probability that an actor/user will experience a failure in the identified problem area (function, characteristic or part) of the system [Schaefer -99].

(4) Unavoidable
An area of the system that most users will come in contact with during an average user session (e.g. Login, print or save).

(3) Frequent
An area of the system that most users will come in contact with frequently, but not every user session.

(2) Occasional
An area of the system that an average user may never find or come in contact with very often, but that deals with functions that might be important to more experienced or advanced users.

(1) Rare
An area of the system that will be found only when a certain set of steps is taken. Most users will never find this, but critical failures are still important to find.

Consequence
Determine the impact/consequence a failure would have for an actor/user (may be different for different actors).

(4) Catastrophic
The system stops. The failures lead to major financial (or other) losses.

(3) Damaging
The system may not stop, but data may be lost or corrupted, or functions may cease to work.

(2) Hindering
The user is forced to work-arounds to more difficult actions in order to reach the same results.

(1) Annoying
The problem does not affect the functionality but rather makes the system less appealing to the user or customer.

Also consider the impact for testing. If there is a high risk of vital functions not working it might stop the testing of other parts of the system (preventive actions might be necessary).

It is important that the project manager, system designers, integrators and testers agree on where the problems are (and the possible impact). Otherwise it will be difficult to initiate preventive actions to reduce or eliminate the risks.

Risk description	Probability/ consequence
< Description of the risk, and the possible underlying problems (root cause) to the risk>	< Probability of a risk to become a problem * Consequence if the risk occurs >

Table 4. Example of the risk description with probability and consequence added.

Output:	An estimated probability and consequence of each risk.

2.4 Prioritize the risks

Assign a priority to each risk. The priority can be calculated by multiplying the probability with the consequences. The risks can then be divided into priority levels according to below:

High The risk must be resolved and evaluated by one or more test cases. The underlying problems causing the risk must be addressed first and if possible prevented early.

Medium The risk needs attention but may be given less development and testing time than the high priority risks.

Low The problems are rare and usually only annoying, i.e. minor design issues that have little effect on the functionality of the system. Problems should be fixed only if time allows.

How the prioritization should be performed must be decided by the project.

		Probability			
		Rare	**Occasional**	**Frequent**	**Unavoidable**
Consequence	**Catastrophic**	High/Medium	High	High	High
	Damaging	Low	Medium	High	High
	Hindering	Low	Low	Medium	Medium
	Annoying	Low	Low	Low	Medium/Low

Table 5: Example of how to specify priority levels with respect to probabilities and consequences.

Priority	**Risk description**	**Probability/ consequence**
<1>	< Description of the risk, and the possible underlying problems (root cause) to the risk>	<Probability of a risk to become a problem * Consequence if the risk occurs>

Table 6: Example of how to add priority to the risks.

Output:	A prioritized list of risks.

2.5 Plan for how to test, reduce and/or resolve each major risk

Make a strategy and a plan for how to deal with the identified risks having the highest priority. Include a proposal for reducing/eliminating the risk with a test strategy for evaluating the risk in the risk list.

Keep the risks with lower priority in the list but wait with the planning for the resolution until they have been given higher priority.

Use the risk list to negotiate time and testing resources in the project.

Identify test objects. Specify which types of test cases are needed. Plan for the specification and execution of the test objects and test cases in conjunction with the overall test planning.

Priority	Risk description	Probability/ consequence	Actions
1	**Risk: Unstable system platform.** - Instability may occur when invalid or wrong function calls are performed in the API of the platform components. - Instability may occur when platform components communicate by sending faulty data or by not expecting the given input/response from another component. Possible cause: - Platform components developed by independent vendors (third party products). - Platform components have not been tested together with other platform components. - It is unknown which test environment the different vendors have used for their testing. - The configuration of the components has not been defined (with consideration of other components). - There is no specification of which versions of platform components that works together (compatibility not guaranteed).	Probability = 4 Consequence = 4 Risk = 4 * 4 = 16	**Risk reduction and evaluation strategy:** 1. Evaluate the criteria used for the acceptance of each product/ component selected for the system platform. 2. Analyze the test results from the vendor testing of components. Identify which test environment that has been used and the configuration. 3. Define a separate test activity (platform test) with the scope of testing interactions between the components in the platform and with components using the platform. **Test objects and test cases:** Platform test of system xxx. Test cases: - Test individual functions in the platform. - Identify and test function chains - bottom up. (Testing of functions related to each other in the platform up to the application level). - Test all application features – Top down. (A selection of normal cases of user features/ functions using different parts of the platform functions).

Table 7: Example of a prioritized list of identified risks with strategies for prevention and evaluation (test).

Output:	A prioritized list of risks with suggested strategies for how to reduce/eliminate and evaluate the risks.

2.6 Test (evaluate, reduce and resolve) the risks

1. **Identify risks to be evaluated.**
   Identify which risks that can be evaluated in your current increment delivery (i.e. system release to be tested).
   - Which functionality has been delivered?
   - Which characteristics are possible to evaluate? I.e. load performance might be possible do test in a small scale or part of the existing system release.
   - Which user interfaces are possible to use?
   - Which test interfaces (or stub code) can be used to test (parts of) the functionality?
   - Which are the most difficult test cases to execute?
2. **Execute a broad coverage of test cases first.**
   Before starting to execute specific areas that have been identified having the highest risks, run a selection of test cases including a broad coverage for all delivered functions and characteristics. The purpose is to quickly find any problem area that might have been missed in the risk analysis.
3. **Reduce or eliminate risks.**
   Perform the tasks specified in the plan to reduce and/or eliminate/prevent the identified risks to occur.
4. **Execute the test cases specified for each risk.**
   Execute the test cases associated with each major risk in order to evaluate the risk level. How many test cases you need to execute per risk is determined by the priority of the risk (considering the possible system impact).

   Identify what kind of actions should be performed and how to perform the actions. In GUI tests, for example, you can give input or manipulate the system in many different ways.
   - Perform the actions including input data (if required).
   - Observe the results.
   - Make sure all results are observed and recorded.

   Example of system response/results:
   - Result printouts.
   - A changed system state.
   - Added or removed information in the database.
   - Messages on the screen (e.g. error messages).
   - New or changed screen contents.
   - Indicators activated (flashing lights, sound alerts).
   - Events/control functions activated (parts moving, signals are sent to other systems).
   - New entries in system log-files, error log-files or security log-files.
   - Any events/results in the system not directly related to your test object.

   Verify/compare the observed result with the expected result.
5. **Improvise more test cases in faulty or risky areas.**
   Design additional test cases ad-hoc to perform additional evaluation or investigation when:
   - An unexpected event occurs.
   - A fault has been discovered and you want to cover surrounding areas to show that there exist no related faults close to the found fault.
   - There are uncertainties about how a function is supposed to work.
   - The specifications are vague or missing.
   - You have a general suspicion regarding the reliability or stability of the system (or parts of the system).

   Remember to document all steps taken in the additional evaluation/test. This is especially important if you need to rerun a test case due to found faults.

   Also record your findings. What conclusions do you make from the test results?

Output:	Log-files, test records with test results, trouble reports with references to the evaluated risks.

2.7 Analyze the test results

1. **Identify areas of the system covered in the risk list.**
   Identify which:
   - Risks were identified in the previous risk analysis.
   - Risks that have been evaluated so far by performed tests.
   - Usage scenarios, functions, characteristics, interfaces or components/ system parts that were associated with the highest predicted risks.
   - Risk definitions/descriptions that need further investigation and specification. Which risks were found not clear enough when the evaluation was performed? What additional knowledge was gained about the risks, which can be added to the definition and the description of possible causes of the risks?
2. **Identify which and where failures occurred during testing.**
   Identify all unexpected events and failures that occurred during testing. It is recommended to also list all the unexpected events to get a more complete picture of the test results. Indicate if the unexpected events were caused by testing errors or faults in the system.

   Identify which:
   - Usage scenarios failed?
   - Functions or characteristics failed?
   - Parts of the system were involved in the failures?
   - Interfaces were involved in the failures?

   Identify which and how many faults:
   - Have been corrected.
   - Are in the process of being corrected (e.g. to the next software delivery).
   - Are not corrected and not yet planned to be corrected.
3. **Analyze occurrence and severity of the found faults**
   Group the found faults in categories according to severity and occurrence. A suggested **priority** is included relating to the severity and occurrence of the found faults.

Severity \ Occurrence	Never	Rarely	Sometimes	Usually	Always
Catastrophic	Medium	High	High	High	High
Hazardous/severe	Low	Medium	Medium	High	High
Major	Low	Low	Medium	Medium	High
Minor	Low	Low	Low	Low	Medium
No effect	Low	Low	Low	Low	Low

Table 8. The table shows the relation between severity and occurrence of the faults found. A suggested fault priority is also included.

   List the faults that are in the high, medium and low priority categories in a table according to above.
4. **Analyze the root cause of the found faults.**
   Find the root cause to the most severe faults [Ammerman -98], [Gano -99]. Use cause and effect graphing techniques (see chapter 2.2) to find what caused the fault to happen and its effect (failure) on the system. If needed initiate additional risk prevention/evaluation activities to avoid more problems of the same kind. Change the design and test processes to cover and prevent similar events from happening again.

Output:	Risk analysis results.

2.8 Monitor the risks

Continuously monitor the progress of the risk resolution and testing. As risks disappear or decrease in importance and new ones emerge, adjust your risk list and strategy to stay focused on the present risks and their priorities.

By continuously adding information about the evaluated risks you can provide detailed information about the quality of the system anytime (to the project manager or any other part interested in the outcome of the project).

Example:

Pri	Risk descr.	Prob/ Cons	Actions	Results	Analysis/comments
2 (1)	**Risk:** Unstable system platform.	Prob =3 (4) Cons=3 (4) Risk = 3*3=9 (16)	**Test objects and test cases:** Platform test of system xxx.	**Test execution results:** A total number of 48 test cases were executed. A total number of 5 faults were found (0 high priority, 2 medium and 3 low priority).	By executing 48 test cases and finding 2 medium and 3 low priority faults resulted in more confidence of the platform not having the highest risk level as first anticipated. The new risk level is still high but less than catastrophic. More testing is required.
			Test cases: - Test individual functions in the platform.	15 functions were used in the platform and tested from the application interface. One medium level fault was found.	Each individual function was considered stable enough for the application.
			- Identify and test function chains - bottom up. (Testing of functions related to each other in the platform up to the application level).	Each layer of functions in the platform from the database in the bottom up to the middleware interface to the application was tested. Two minor faults were found.	The interconnections/ interactions between the functions had some problems. The application parameters were not set correctly causing failures in the communication and functional behavior.

Table 9: Example of a prioritized list of identified risks with results and analysis/comments included for each risk.

Prioritize the remaining risks to be tested based on the current results and analysis.

Change testing strategies if needed.

Plan for how to test the remaining risks.

Output:	A prioritized risk list reflecting the current level of risk in the system.

2.9 Report the test results

Report the results of the risk based testing. Describe the remaining risk level in the system based on all planned and added test cases that have been executed, and the amount of faults found. Give recommendations if the system is ready for release or not. Identify remaining issues to be resolved.

Use the following steps when reporting the test results:

1. Gather test results, trouble reports, test records and the risk evaluation information from the risk list.
2. List all functions and characteristics to be delivered to the customer/orderer of the system.
3. Identify which test objects, test cases, test data that have been selected and executed.
4. Specify which test cases passed, failed, were not possible to run and which were not applicable.
5. Describe how and when you performed the tests.
6. Describe which test tools, methods and techniques that were used.

7. Specify which system configuration that was used. Also specify if it was a “typical” configuration or a specific configuration aimed at revealing certain risks.
8. Identify the number of faults found per risk, function and characteristic that were tested. Analyze the faults and try to find their cause. Suggest how to avoid similar problems in the future.
9. Identify which faults remain to be corrected by the designers.
10. Specify which level of test coverage you manage to achieve and the confidence level when analyzing the risks.
11. Estimate the quality level per function, characteristic and the system as a whole based on the risks that have been evaluated.
12. Summarize the results from the risk list. Refer each risk to the related quality attributes above.
13. Identify remaining risk areas that might need further evaluation and investigation. Specify a suggested completion plan.
14. Describe your experiences of the risk based testing, any problems encountered, what to think about the next time you or someone else perform risk based testing (in the next project).
15. Give your recommendation for how to proceed. Should testing continue or stop. Is the probability of remaining faults considered to be reduced to a level that can be accepted by the user and the customer?

Output:	A report with the results from the risk based testing.

3 How to start using a risk based testing process

A risk based testing approach can be introduced in an organization quite easily. The risk analysis is the most important part. It can be performed as a one-day activity/meeting where all the risks are identified. Make sure all roles of people in the project participate.

The used process for the risk based testing can be gradually improved to include all the steps described in this paper to get a more efficient risk handling and evaluation.

A common problem when you analyze the risks in the project is that they are not followed up. Preferably let the quality manager or the test leader in your project has this role.

It is also recommended that the risk analysis is performed in conjunction with the risk analysis included in the application of RUP (Rational Unified Process) [Rational -00]. Risk based testing fits well into the risk based thinking when designing the system.

A risk-based approach is however not enough if you want to perform a full test of your system. You still need to fulfill the lower levels of the testing maturity steps (see *appendix B*). You also need to perform requirement and design based testing to make sure all functionality works.

4 Conclusions

Risk based testing is a method to:

- Make potential technical problems in the system visible at an early stage of development.
- Select a test strategy, test objects, test cases, test data, and system configuration related to the risks to be evaluated.
- Prioritize the most important risks to be tested/evaluated and/or eliminated/reduced.
- Explore and evaluate the risks by executing test cases.
- Report and follow up test results that give a clear indication of the current quality level of the system.
- Provide information to the project manager whether testing should continue or stop by considering the current level of risk in the system.

Testing has always been risk based in one way or another. There is always risks involved in what is presumed to work or not when releasing to the customer. The risk based testing approach provides a framework for how to find and evaluate those risks in a structured and efficient way.

5 References

[Ammerman -98] The root cause analysis handbook ISBN 0-527-76326-8

[Bach -99] James Bach on risk-based testing STQE Magazine nov/dec99 http://www.stqemagazine.com http://www.satisfice.com

[Bach1 -99] Satisfice test strategy model. General functionality and stability test procedure. http://www.satisfice.com

[Beizer-90] Software Testing Techniques, second edition ISBN 0-442-20672-0

[Claesson1 -01] How to use scientific methods in software testing (QWE -01) www.soft.com

[Claesson2 -01] ETP - Enea Test Process www.enea.se

[Connell -96] Rapid development. ISBN 1-55615-900-5

[De Bono-85] Six thinking hats ISBN 0-316-17791-1

[Down -94] Risk management for software projects ISBN 0-07-707816-0

[Focus -01] Focus groups http://universe.indiana.edu/clp/rf/focusg.htm

[Gano -99] Apollo Root Cause Analysis ISBN 1-883677-01-7

[Hale -97] Practical problem solving and decision making ISBN 0-7494-2219-X

[Jones -95] The thinkers toolkit ISBN 0-8129-2601-3

[Lutz -96] Experience report: Contribution of SFMEA to requirements analysis http://citeseer.nj.nec.com/69117.html

[Mears -95] Quality improvement tools and techniques ISBN 0-07-041219-7

[Quality -01] The Guide to Managing for Quality http://erc.msh.org/quality/map.cfm

[Rational -00] Rational Unified Process 5.1 http://www.rational.com

[Schaefer -99] Strategies for prioritizing tests against deadlines – risk based testing. http://home.c2i.net/schaefer/testing.html

[Thinking -98] Tools for the Age of Knowledge http://www.virtualsalt.com

[Wallace -95] A Study on Hazard Analysis in High Integrity Software Standards and Guidelines http://hissa.ncsl.nist.gov/

Appendix A: The general steps in a testing process

This appendix consists of a general description of a testing process [Claesson2 -01]. The purpose of a testing process is to provide a structure for the general steps performed in testing. These steps are used as a reference to the application of the risk based testing process.

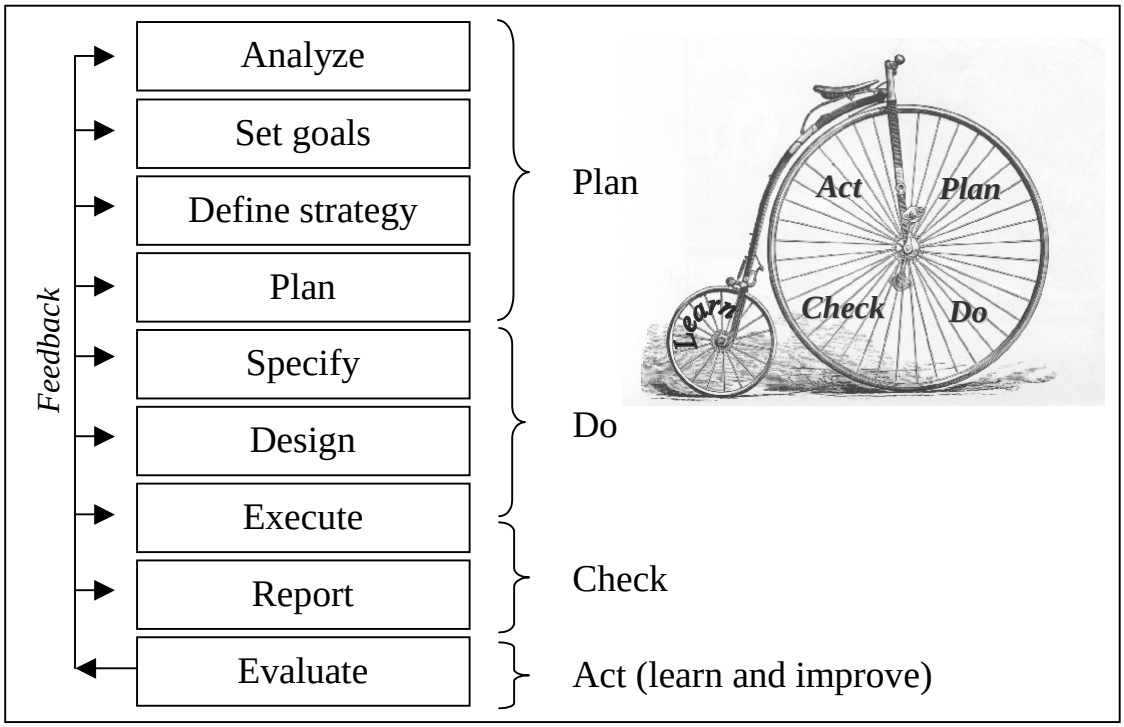


Fig. 4 The 9-step testing process representing the order in which different tasks are performed in testing.

The general steps performed in testing can be described in the following way:

1. Analyze the system to be developed, the requirements, the customer/users and the project.
2. Set goals for what and how much should be tested, and in which time frame.
3. Define a strategy how to reach the testing goals, i.e. how to organize the tests, which processes and methods to be used.
4. Plan the tests according to defined goals and strategies. What should be tested when and by whom?
5. Specify the test cases that should be run.
6. Design the test cases; i.e. add test data and configuration information.
7. Execute the test cases and check the results.
8. Report the result of the testing.
9. Evaluate the result of the whole testing process. Feed back experiences for improvements.

The scientific method can both be used as part of the whole testing process and for specific subjects or areas to be investigated.

The Deming PDCA cycle can also be used in the testing process as well as for the scientific method to obtain an overall structure of the steps to be performed (both as a whole and in parts).

Plan	Collect information, analyze, define goals and strategies, decide and plan actions.
Do	Prepare and execute the actions in your plan.
Check	Check and evaluate the results.
Act	If needed, change your goals, strategies and plans based on the results from the previous actions.

In each iteration of the PDCA cycle you gain valuable knowledge, which you can add to the answers of your original questions (e.g. have all major faults been found, have we performed enough testing?).

Appendix B: Maturity levels in testing

This appendix consists of a description of how different maturity levels may affect your testing goals.

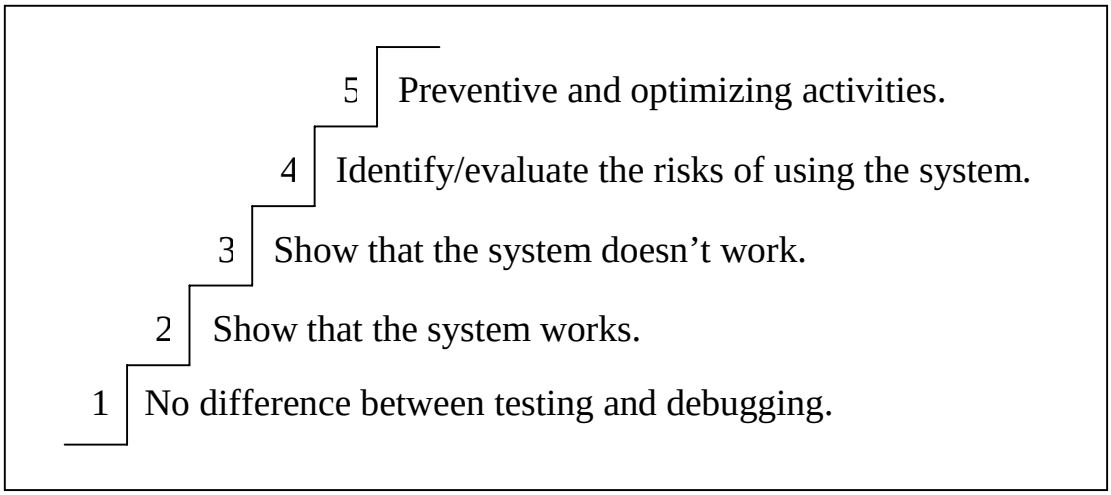


Fig. 5 The figure shows the five levels of testing awareness and maturity based on [Beizer-90].

Level 1 (ad hoc fault finding):

> Testing is unstructured and a "code and fix" strategy is used. No efficient testing and no quality can be obtained at this level.

Level 2 (normal case testing):

> Testing is structured and a requirements based test strategy is used. The main problem is the large (almost infinite) amount of possible combinations to test. How many combinations are needed to show that the system works? The more you test the likelier you are to find a fault.

Level 3 (negative testing):

> Testing is focused on trying to break the system. Independent testing groups runs a lot of tests that provoke the error handling of the system. How many negative tests does it take to show that the system is robust? The more you test the likelier you are to break the system.

Level 4 (risk based testing):

> Testing is focused on identifying and evaluating risky areas of the system. By prioritizing test cases according to risk the amount of test cases can be limited and the probability of remaining faults reduced to a level that can be accepted by the customer.

Level 5 (fault prevention):

> Testing is focused on preventive and optimizing activities. Fault prevention is initiated by using the results from the risk analyses of the system (used in the risk based approach at level 4).

The objective of reaching higher levels of maturity is to enable a more efficient testing (i.e. effectiveness in the planning, preparation, execution, reporting and evaluation of the results).

But even if you are at level 5 you still need to:
- Perform debugging (level 1).
- Show that the system works (level 2).
- Show that the system is robust (level 3).
- Evaluate the risks in using the system (level 4).

Your higher-level strategies will make the lower levels more focused towards where it is most important to test.

Appendix C: The six thinking hats

In the book “Six thinking hats” Edward de Bono describes how to apply different techniques in exploration, problem solving, planning and decision making [De Bono-85] which may be useful when investigating or asking about risks. There are two main thinking styles:

- Reactive thinking.
  You react moment to moment to what’s happening without any mental maps or much analysis of the situation. Imagine playing chess by just reacting to the moves of the other player (using an ad-hoc strategy of how you play). Without a map you don’t know where you are going.
- Proactive thinking.
  In proactive thinking you draw maps where you use different thinking skills. Some useful skills are evaluation, visualization, imagination, classifying, abstracting, questioning, judging, interpreting, critical thinking, logical thinking, emotions and creative thinking.

In the proactive thinking the so called PMI method can be used to find the direction in which you want to conduct your research or making decisions during the research.

P The “Plus” stands for what positive aspects exist based on your observation to conduct a research (i.e. why you want to investigate something).

M The “Minus” stands for negative aspects (i.e. why not doing the research).

I The “Interesting” stands for which aspects are worth noting but do not fit under either “plus” or “minus”.

With a list of “+”, “-“ and “interesting” decisions will be easier and faster than arguments, discussion, new argument, more discussion.

The six thinking hats defined by De Bono consists of:

White hat Neutral and objective, facts and figures.
To provide just the facts with no interpretations or judgments. Consider if it is a fact, belief or likelihood. How true are the facts? Are the facts checked and proved? Use the facts to draw maps of what you want to investigate.

Red hat Emotions.
The opposite of the neutral and objective information. It is all about emotions without the need to justify, explain, give reasons for or the basis of the expressions. The emotions can be expressed as hunches, intuitions or impressions.

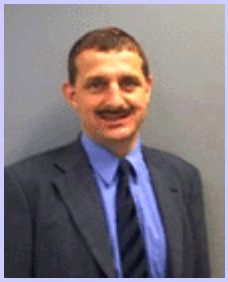
Black hat Critical and negative.
The logical-negative aspects. Why it will not work. Reasons for something that doesn’t fit our knowledge and experience. Critical judgment and pessimistic view. Errors of thinking, why things does not follow another. Possible conclusions. How does this fit my past experience patterns? What are the risks? Are the facts right? Are the facts relevant? Is this worth doing? The black hat thinker points out what is wrong, incorrect and in error.

Yellow hat Sunny and positive.
The optimistic view focusing on benefits. Constructive thinking that makes things happen. What is the positive view based upon? Why do you think it will happen in this way? What are the background reasons for the optimism? What are your proposals and suggestions? How do the best possible scenarios look like?

Green hat Creativity and new ideas.
New ideas, concepts and perceptions. Alternatives and new approaches to problems. Follow the forward effect if an idea. Provocation of given facts or ideas. Outlines routes, options, choices and alternatives.

Blue hat Control and organization of the thinking process.
The organization of thinking as a step-by-step approach. It is the control over the other hats. To summarize, conclude and report. The difference between a good thinker and a poor thinker often lies in the ability to focus.

The six thinking hats method is a two-stage process. The first stage is to make a map. The second stage is to choose a route on the map. If the map is good enough, the best route will often become obvious.



QWE2002 Paper 9M

Mr. Rob Baarda
(IQUIP Informatica BV)

Risk Based Test Strategy

Key Points

- Well defined steps from business risks to test coverage.
- Early and stronger test involvement of all parties concerned.
- Useful for all tests.

Presentation Abstract

Testing of an information system should be based on the business risks for the organisation in using that information system. In practice, the test manager often takes the steps to go from risks to test coverage in an intuitive way. In this presentation, the steps to define a testing strategy are made explicit.

The risk assessment requires the involvement of users and managers of the client organisation and of project members like developers, testers, QA personnel and project manager. Discussing risks and testing in the above way proves in practice to be real eye-openers for all parties concerned. This also enables negotiating about testing depth by letting the customer choose what should be tested how thoroughly. The stepwise defining of the test strategy can be used for any test and also for an overall strategy, including even inspections.

About the Author

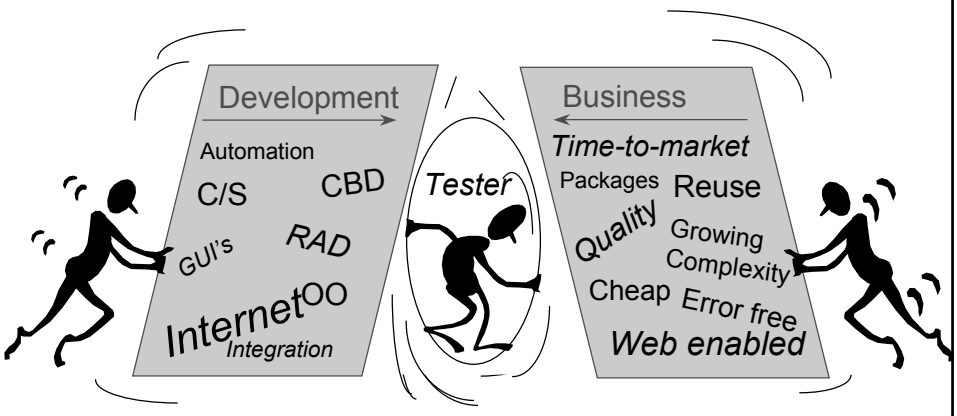
After a technical study Rob is working more than 20 years in the IT-industry. He works in the field of testing since 1986 where he started developing test automation. Later Rob changed to the methodology of testing. He is now working on test subjects in the R&D department of IQUIP, working as an (inter)national test consultant and teaching TMap® in the Netherlands and in Belgium. He presents about Risk Based Test Strategy at (inter)national conferences.

Risk Based Test Strategy

Rob Baarda
baardaro@iquip.nl

IQUIP
00 nr Be 1

Tester under Pressure



Needed: communication between
Development, Business and Tester

IQUIP
00 nr Be 2

Copyright © IQUIP Informatica B.V.

Risk Based Test Strategy helps!

Agenda
- Test Strategy
- Risk Analysis
- Procedural steps
- Conclusion

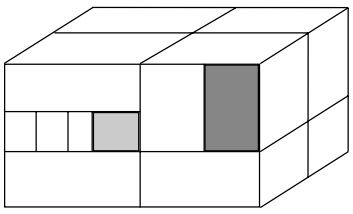
IQUIP
00 nr Be 3

What is Test Strategy?

Test Strategy =
the distribution of the effort for testing
-based on the test coverage-
over parts and quality characteristics
of the system under test

Quality characteristics:
- Functionality
- Performance
- Usability
- Security

Etc.



IQUIP
00 nr Be 4

Copyright © IQUIP Informatica B.V.

Aim of Test Strategy

Aim: To detect the most important defects as early as possible at the lowest costs!

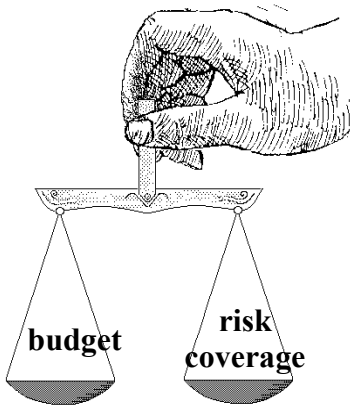
What are the most important defects?

Risk thinking gives the answer!

IQUIP
00 nr Be 5

Risk thinking: an introduction

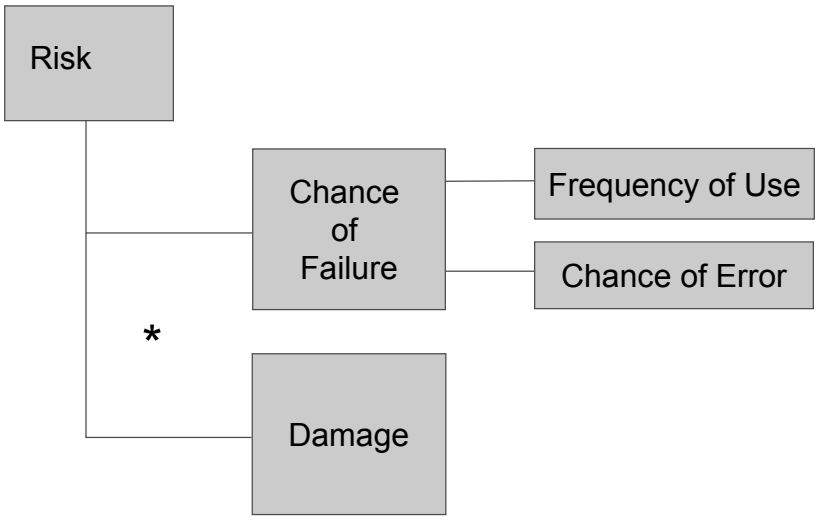
- Business reasons
- No risk, no test
- Risk analysis approach



IQUIP
00 nr Be 6

Copyright © IQUIP Informatica B.V.

Risk: definition



IQUIP
00 nr Be 7

Risk definition Chance of Error

- Chance of Error
  - globally
    = size * complexity
  - in detail
    = knowledge of development quality
- Possible development problems
  - did they think about different browser-versions?
  - did they ask potential users?
  - was performance a design issue?
  - insufficient quality of low-level tests
  - extreme time pressure

IQUIP
00 nr Be 8

Copyright © IQUIP Informatica B.V.

Risk definition Damage

- Damage
  - direct financial including repair cost
  - loss of faith of customers
  - damage to corporate identity
  - impact on other functions and/or systems
  - detection and repair time

IQUIP
00 nr Be 9

Risk analysis applicable-1

- Applicable on the level of:
  - system
  - subsystem
  - individual function (e.g. interest calculation)
- Applicable:
  - New developments
  - Maintenance

IQUIP
00 nr Be 10

Copyright © IQUIP Informatica B.V.

Applied in maintenance

- Risk =
  Chance of Failure * Damage
- Chance of Failure =
  Frequency of Use * Chance of Error

IQUIP
00 nr Be 11

Risk analysis applicable-2

Applicable on quality characteristics

- What is the chance that a failure will happen and what will be the damage for:
  - functional defects
  - low performance
  - poor security
  - ...

IQUIP
00 nr Be 12

Copyright © IQUIP Informatica B.V.

Example

Euro & Internet update	Failure 1-5	Damage 1-5	Risk 1-25
System:			
Debt management	5	5	25
Cash management	3	2	6
Stock transactions	3	5	15

Risk classes:

- Great risk (25)
- Middle risk (15)
- Small (6)

Organization can decide on test depth

IQUIP
00 nr Be 13

Important hints

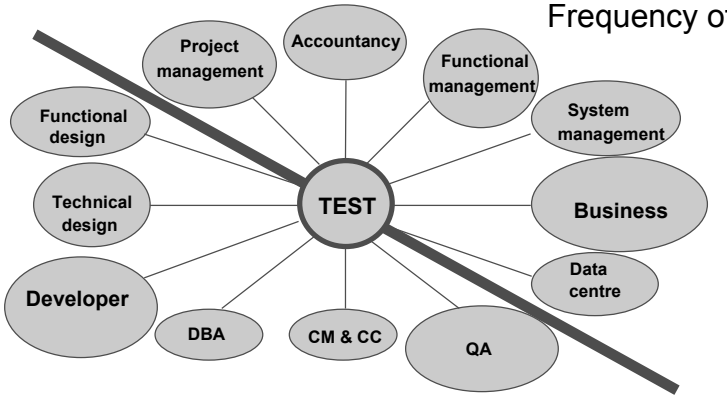
- Risk Analysis should lead to a limited number of classes of (more or less) equal risks
- Pay attention to the scales of measuring
- Get the right parties involved
- RBTS is a learning process

IQUIP
00 nr Be 14

Copyright © IQUIP Informatica B.V.

Parties involved

Damage
Frequency of Use



Chance of Error

IQUIP
00 nr Be 15

Risk Based Test Strategy helps!

Agenda

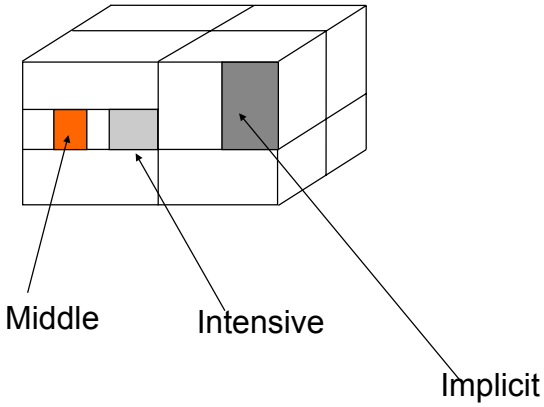
- Test Strategy
- Risk Analysis
- Procedural steps
- Conclusion

Test Strategy =
the distribution of the effort for testing
-based on the test coverage-
over parts and quality characteristics
of the system under test

IQUIP
00 nr Be 16

Copyright © IQUIP Informatica B.V.

Coverage



IQUIP
00 nr Be 17

Test techniques

Intensive
UL Usability Laboratory

Intensive
HT Hackers Team

Intensive
ECT Elementary Comparison Test

Middle
PCT Process Cycle Test

IQUIP
00 nr Be 18

Copyright © IQUIP Informatica B.V.

Procedure visualized

Quality characteristics	Relative importance %	System Debt	Cash	Stock
Security	25			++ HT
Functionality	55	+++ ECT	++ PCT	+ PCT
Usability	20	+ PCT		++ UL
…	0			
	100	60	15	25

IQUIP
00 nr Be 19

Steps

- **Master Test Plan**
  - selection of quality characteristics
  - relative importance of quality characteristics based on risks
  - quality characteristics allocated to the test levels
- **For one Test Level Plan**
  - determine (relative importance) of system parts based on risks
  - specify test importance per system part and quality characteristic
  - choose test design techniques (TT) and testing depth

IQUIP
00 nr Be 20

Copyright © IQUIP Informatica B.V.

Test Strategy supports communication

A documented Risk Based Test Strategy helps:

- To be complete in assessing the risks
- Discussions with the parties involved
- To get commitment from the customer
- In the future to:
  - have documented proof if needed
  - make a switch in test manager position easier
  - do a re-planning
  - use in the maintenance situation

IQUIP
00 nr Be 21

Conclusion

Risk Based Test Strategy

- one line from risks to test design techniques
- less intuition, more business reasons
- applicable for different test levels
- supports customers to make choices
- is a medium to have communications between users, developers and testers
- makes the test process more manageable

IQUIP
00 nr Be 22

Copyright © IQUIP Informatica B.V.

Risk Based Test Strategy

Rob Baarda

IQUIP Informatica B.V.

www.iquip.nl

P.O. Box 263, 1110 AG Diemen, The Netherlands

Tel: +31 20 660 6600

Fax: +31 20 695 3298

E-mail: baardaro@iquip.nl

1. Introduction

Nowadays the pressure on the tester increasing due to demands from the business on one side and faster development of software on the other hand. A good approach for the tester to establish a good communication between all the parties is the use of a test strategy.. A test strategy is, in short, the choice which aspects of a (sub)system will be tested with what testing depth. In recent years we have had splendid opportunities for the further development and implementation of test strategies

We were able to combine theory on **business risks** with the existing theory of test strategy as described in TMap® (see references). This leads to a straight line of thinking from business risks into detailed testing depth per function. Some organizations started in parallel to change from the intuitive way of making a test strategy into a more rational one based on business risks.

In this paper we first describe the theory from business risk unto test techniques, followed by two concise case-stories of the previous described situations.

We will make clear that the described manner to define a test strategy has the advantage of:

- improved communication between the tester and other parties concerned;
- better test coverage on the right spot.

Copyright © 2001 IQUIP Informatica B.V. The Netherlands

2. Test strategy and risks

The development of a test strategy is a means of communication with the customer commissioning the test on such matters as the organization of testing and the strategic choices that go with it. The test strategy indicates how testing is to be carried out. In order to make the best possible use of resources and time, it is decided on which parts and aspects of the system the emphasis should fall. The test strategy forms an important basis for a structured approach to testing and makes a major contribution to a manageable test process.

The customer who commissions the test will expect specific qualities of the system when in production, and wants to know whether the released system will meet these requirements. If the system qualitatively does not meet the requirements or only to a limited extent, this implies high damage for the organisation, for instance since high rework costs will be needed or clients/users will be unsatisfied. Therefore, this situation forms a risk for the organization. 'Risk' in this paper is defined as:

A risk is the chance of an error[1] occurring (chance of failure) related to the damage expected when this error does occur

---

Testing covers such risks by giving insight into the extent to which the system meets the quality demands. When quality turns out to be insufficient timely measures can be taken, e.g. rework by developers. If the shipping of the system implies many risks for the organization, better testing is obvious as a solution. And the reverse also holds:

No risk, no test

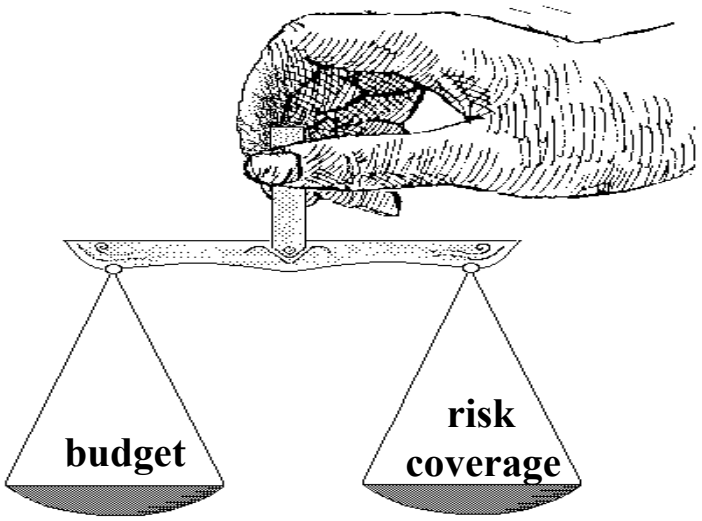
Although in the above we refer to quality and risks in a general sense, there may be large differences depending on the situation. It is of great importance to discuss this with the customer, and to translate the customer's wishes in this respect into the way testing will be performed. Thus, the test strategy is directed towards finding the optimal balance between the test effort to be exerted and the coverage required for the risks. To this purpose the risks are specified up to the level of quality characteristics and separate subsystems. In doing so it becomes possible to find a suitable test coverage for the assessed risks. Here a higher test coverage usually results in more test effort. In order to reach at the variation in test coverage needed, the use of more than one test specification technique (test design technique), each offering a specified test coverage, is crucial.

An analogy with insurance industry may clarify this matter a bit more. A person wants to cover a relevant risk and takes an insurance with a coverage fitting this risk as best as possible. This insurance takes a certain premium. If the person wants to pay less, an insurance with a lower coverage is bought. The consequence is that there will be no payment if the uncovered risk occurs. On the other hand, if coverage was too large, then too much premium is paid, since a situation has been insured which is unlikely to occur for this person.

---

[1] The terms error, defect and failure are not used as exactly as IEEE advocates. In this paper error = fault or mistake; failure = the result or manifestation of one or more errors.

Copyright © 2001 IQUIP Informatica B.V. The Netherlands



The balance between budget and risk coverage

3. Risk Assessment

Test strategy is based on risk assessment. This means assessing the damage of the consequences of failures, both undetected prior to operation and occurring during operation.

Risk assessment takes place on the basis of quality characteristics and subsystems. For instance, if the system is insufficiently user-friendly, what will be the negative consequences. And what will be the damage when the salary calculation module in a payroll system does not work correctly.

In order to be able to perform this assessment well, the separate aspects of a risk are considered:

*Risk = chance of failure * damage,*

where chance of failure is related to aspects including frequency of use and the chance of an error occurring.

These aspects are listed below:

- Frequency of use

In a function which is used dozens of times each day the chance of an error demonstrating itself is much bigger than with a function used once a year.

- Chance of error

For the assessment of the chance of errors the following list can be helpful. It presents the locations where errors tend to cluster. It is partly based on H. Schaefer, 1996 (*Surviving under time and budget pressure*, in: Conference Proceeding EuroSTAR1996, Amsterdam, the Netherlands):

  - ◊ Complex functions;
  - ◊ Completely new functions;
  - ◊ (Especially frequently) adjusted functions;
  - ◊ Functions for which certain tools or techniques were employed for the first time;

Copyright © 2001 IQUIP Informatica B.V. The Netherlands

- ◊ Functions which were transferred from one developer to another during development;
- ◊ Functions that were realized under extreme time pressure;
- ◊ Functions which had to be optimized more frequently than on average;
- ◊ Functions in which many errors were found earlier (e.g. in previous releases or during earlier reviews);
- ◊ Functions with many interfaces;
- ◊ Inexperienced developers; Did they think about different browser types? Was performance a design issue?
- ◊ Insufficient involvement of users;
- ◊ Insufficient quality assurance during development;
- ◊ Insufficient quality of low-level tests;
- ◊ New development tools and development environment;
- ◊ Large development teams;
- ◊ Development teams with sub-optimal communication (e.g. owing to geographical spread or personal causes);

- Damage

If and when the error manifests itself, what will be the damage for the organization. Aspects are costs of repair (both of the system and of the consequences), forgone income and loss of clients or of confidence. Usually the damage increases if the error has its impact on other functions or systems. In the case of errors occurring in batch processes there may be a possibility to prevent them from hampering users, so that the eventual damage will be smaller than with similar on-line processes. Of course, this only holds if errors are detected on time.

Because of the complexity of the matter, it is impossible to assess risks with complete objectivity and in detail: it is a global assessment. It is therefore important for the risk assessment not to be carried out by the test manager alone. A large number of people involved in the scheme should contribute customer, users, development team, accountants, IT auditors and so on. This not only increases the quality of the strategy, but it also has the advantage that the different parties are more aware of the risks and the extent to which testing contributes to making these risks manageable in a better way.

The developer of the test strategy should realize that 'users' are the best people to assess the damage and the frequency of use when valuing the risks (end-users, system managers and application managers, line management), whereas project team members are best to assess the chance of error (project managers, designers, programmers, project quality staff, test manager).

The focus in risk assessment is on product risks, or, in other words, what is the risk for the organization if the product does not demonstrate the expected quality. In addition to this, there are also (test) project risks. If the system must be in production on January 1st, if functional specifications are produced too late, if no experienced testers are available, or if the test infrastructure is not ready on time, then we speak of (test) project risks. These are not taken into account in determining the test strategy; they do play a role in the test plan.

In developing a test strategy the aim is to see to it that the test will be organized in such a way that with a certain extent of reliability

- the most important problems will be found;
- the problems will be found in an early stage;
- the problems that require the most rework time will be found first:
- efficient use is made of resources;
- and eventually an accurate quality advice can be given.

This can be summarized as:

Test strategy aims at finding the **most important** errors as **early** as possible against the **lowest costs**

In practice, the development of a test strategy is often planned to coincide with preparing the budget, for example with the help of test point analysis. The advantage is that the consequences of the adopted strategy are immediately translated into time required for testing, and consequently the cost of testing, which makes the strategic choices manageable. If the time available for testing is more or less fixed, it is also possible to use test strategy combined with test point analysis to determine what is achievable within the time limits. It is probably even more important to make it clear at this time which parts cannot be tested, or cannot be fully tested, and what risks will therefore be incurred.

4. Quality Characteristics

The quality characteristics we distinguish can be divided into dynamic and static quality characteristics. The dynamic quality characteristics deal with features of the information system in use; examples are security, usability, continuity, traceability, functionality, userfriendliness, suitability, efficiency, and performance. The static are concerned with intrinsic characteristics of the information system and the documentation, as considered from the standpoint of developers and future system managers. Examples are manageability, maintainability, connectivity, reusability, portability, testability.

5. Steps in different situations

The development of a test strategy is not something that can be done purely methodically or formally. The below steps are aids and indicators. Experience and skills of the performer of this activity in the area of testing is a major success factor for a sound test strategy.

One should also realize that test strategies arise as a result of iterative processes and in connection with other activities for a test plan. If the first test strategy produces an amount of test effort needed or a certain time schedule, which is unacceptable for the customer, the strategy should be adjusted. The lack of test skills or suitable infrastructure can also result in adjustments of the test strategy.

The stepwise defining of the test strategy can be used for any test level and also for an overall strategy (master test plan), including and coordinating all test levels and even inspections. The steps differ for both situations.

Also we describe shortly considerations managing the test process and for the maintenance situation.

5.1 Strategy in Master Test Planning

The steps to be taken for a test strategy are:

- Decide on the quality characteristics;
- Determine relative importance of quality characteristics;

- Attribute quality characteristics to test levels.

5.1.1 Step 1: Selection of Quality Characteristics

In close liaison with the customer and other parties involved a selection of quality characteristics is made on which the tests must focus. In doing so one should take risks for the business into account as well as aspects including system requirements, business objectives concerning the information system, directions and standards set by the computer center. These quality characteristics are also used for reporting to the customer during test execution and completion.

Some characteristics are difficult to test. There may be a wish for a system to be user-friendly and flexible, for instance, but these wishes turn out not to have been translated into measurable requirements. That is why a substantial part of the effort here is devoted to formulating the relevant quality demands as measurably and unambiguously as possible. It is also the case that some quality characteristics demand relatively much effort in testing. Since it is not useful to offer possibilities which cannot be fulfilled, it should be determined beforehand what will be the estimated effort needed for a decision made.

For non-IT people our quality characteristics may be hard to handle. It helps when we translate them to the conceptual environment of our conversational partners. This can be done by finding illustrative examples of problems or errors that may occur in production and the damage that would be caused by this. This is one of the most difficult aspects of the formulation of a test strategy.

5.1.2 Step 2: Relative Importance of Quality Characteristics

On the basis of the results from Step 1 the importance of the selected quality characteristics is determined in relation to one another. This is done in the Matrix of Weights (see below), by weighing the relative risks per quality characteristic. Here the relative importance is indicated (in percentages). Note that it is not of importance to have exact percentages: the objective is to arrive at a general picture of the relative importance of the various quality characteristics. The filling in of the matrix helps evaluating the risks.

The customer should be forced to make choices. Therefore, as a directive we ask for a percentage of 5 as the minimum. The sum of all percentages should not exceed 100. An example of a Matrix of Weights is given below:

Quality characteristic	Relative importance
Manageability	5
Security	5
Usability	-
Connectivity	-
Continuity	10
Traceability	-
Flexibility	-
Functionality	50

Copyright © 2001 IQUIP Informatica B.V. The Netherlands

Userfriendliness	10
Reusability	-
Infrastructure	-
Suitability	10
Maintainability	5
Performance	5
Portability	-
Testability	-
Efficiency	-
Total	100%

The Matrix of Weights

The high percentage for functionality in this matrix may strike the reader. This is in conformance with practical experience: generally 50% of the importance or more is attributed to this characteristic. The reason for this is that risks usually are larger for incorrect performing systems (Functionality) than for slow systems (Performance) or awkward systems (Userfriendliness).

5.1.3 Step 3: Quality Characteristics Attributed to Test Levels

With the aim of spending the total test effort as efficiently as possible, during test strategy development it is decided with which test level or combination of test levels the various selected quality characteristics will be tested. Also inspections may fall under the scope of the master test plan and under the test strategy. In the remaining sections when 'test' is used, inspections are also included.

In this way the various test levels within a project are brought into balance. It is obvious that the different responsibilities and authorities remain intact.

A +-sign in a matrix (for an example, see matrix below) indicates whether the test strategy takes a quality characteristic into account. '++' or '+++' indicate that relatively much attention is to be paid to the quality characteristic for the specified test level. It is obvious that one quality characteristic can be in effect for more than one test level, but depth will often vary. If structured test specification techniques are used, the acceptance test, for example, may use results of previous tests levels, on the basis of which it may be decided to test with less depth.

	Insp RQMS	Insp Specs	Insp Design	PT	IT	ST	FAT	PAT	Relative importance
Manageability	+	+	++					+	5
Security	+	+	+				+	+	5
Usability									-
Connectivity									-
Continuity	+		+					++	10
Traceability									-
Flexibility									-
Functionality	++	++		+	+	+++	++		50
Userfriendliness		++					++		10
Reusability									-
Infrastructure									-
Suitability	+	++					++		10
Maintainability		+				+			5
Performance			+				+	+	5
Portability									-
Testability									-
Efficiency									-
									100%

Example of a Test Strategy for Test Levels

Legenda:

Insp RQMS	Inspection of Requirements
Insp Specs	Inspection of Functional Specification
Insp Design	Inspection of Technical Design
PT	Program Test
IT	Integration Test
ST	System Test
FAT	Functional Acceptance Test
PAT	Production Acceptance Test

Copyright © 2001 IQUIP Informatica B.V. The Netherlands

5.2 Strategy for a Test Level

The steps to be taken for a test strategy for a specific test level are:
1. Decide on the quality characteristics;
2. Determine relative importance of quality characteristics;
3. Divide the system into subsystems;
4. Determine relative importance of subsystems;
5. Specify test importance per subsystem and quality characteristic;
6. Establish test techniques to be used.

The strategy determination for a specific test level naturally has the master test plan strategy as a precondition and a starting point. If a master test plan, including a test strategy, is there, step 1 can be omitted and step 2 will be an easy and fast performed activity. Nevertheless, all steps are worked out below.

5.2.1 Step 1: Decide on Quality Characteristics

In collaboration with the customer and perhaps other parties concerned the quality characteristics are determined on which the test will focus, in relation to business risks. During the test and in the completion phase, results are reported on the basis of these quality characteristics.

5.2.2 Step 2: Determine Relative Importance of Quality Characteristics

Based on the results of step 1 the relative importance of the selected quality characteristics is determined. Determination of the importance takes place by weighing the risks per quality characteristic. This is shown in a Matrix of Weights by a percentage in the column Relative importance. In order to force the customer to make choices, a percentage of 5 is the minimum.

An example of a matrix for a functional acceptance test is given below:

Quality characteristic	Relative importance
Security	5
Functionality	60
Userfriendliness	10
Performance	5
Suitability	20
Total	100%

The Matrix of Weights for a Functional Acceptance Test (Example)

Copyright © 2001 IQUIP Informatica B.V. The Netherlands

5.2.3 Step 3: Divide System into Subsystems

During this step and the following steps the test strategy is refined more and more. This implies that the quality characteristics and their relative importance as indicated in the Matrix of Weights are to be broken down for the combination of test specification technique and subsystem, later even for test specification technique and test unit.

The information system is divided into subsystems. The reason for this is that the same quality demands do not have to be valid for each subsystem. Moreover, the various subsystems may have different risks for the organization. In principle the division is the same as given in the design documentation. If we deviate from this one, we must clearly indicate the motivation for this. Examples of alternative divisions are on the basis of extent of risk or on the basis of order of release by the developer. If a conversion module is there, this is to be treated as a separate subsystem. Often the subsystem 'Total system' is distinguished. This serves the purpose of indicating that some quality characteristics can be evaluated effectively only with the help of an integral test, testing the coherence of the various subsystems.

In a later stage the various subsystems are further divided into independent test units. E.g. in a logistics system the subsystem Sales may be divided into the test units Quotations (all functions regarding quotations) and Orders.

5.2.4 Step 4: Determine Relative Importance of Subsystems

On the basis of the result of the previous step the relative importance (in percentages) of the subsystems should be indicated in the Matrix of Weight. This is done by weighing the risks per subsystem. It is not a matter of exact percentages; rather it is a matter of getting a general image of the importance of the subsystems as seen through the eyes of the customer and other parties concerned. This step helps in asking people to form an opinion of this.

The relative importance is determined of each subsystem within the information system. In the Matrix of Weights this is indicated with a percentage in the column Relative importance.

An example of a Matrix of Weight for a functional acceptance test (based on the Strategy Matrix for test levels in the master test plan, shown above) is given here:

	Relative importance
Subsystem 1	30
Subsystem 2	15
Subsystem 3	20
Conversion	15
System	20
Total	100%

Relative Importance of Subsystems for a Functional Acceptance Test (Example)

5.2.5 Step 5: Specify Test Importance per Subsystem and Quality Characteristic

Finally a refinement is made by assessing the importance of the combination quality characteristic - subsystem. E.g., a refinement may be that userfriendliness is important (relative importance of 10), but this holds predominantly for subsystem 1 and not at all for subsystem 3. Again it is emphasized that test strategy determination is not a mathematical affair: it is meant to get an image of the relative test importance of the various subsystems and quality characteristics. This is also the reason why we choose +, ++ and +++ as notation symbols, rather than opting for the pseudo-certainty of a mathematical formula. An example of this is the following: suppose both userfriendliness and a specific batch subsystem are very important, a mathematical formula would probably result in large test effort to be spent on the userfriendliness of the batch procedure. The Matrix of Weight may look like this:

	Subsystem 1	Subsystem 2	Subsystem 3	Con-version	Total system	Relative importance
Security	+	+				5
Usability						-
Continuity						-
Traceability						-
Functionality	++	+	+	++	+	60
Userfriendliness	++	+				10
Performance	+		+			5
Suitability	+	+	+		++	20
Efficiency						-
Relative importance	30	15	20	15	20	100%

Relative Importance of Subsystem x Quality Characteristic (Example)

5.2.6 Step 6: Establish Test Specification Techniques to be Used

The final step in test strategy involves the selection of the techniques that will be used to test the combination of the selected quality characteristics and subsystems. A high importance implies the use of techniques with a high coverage or the use of more techniques, a low importance implies the use of techniques with a lower coverage or the use of fewer techniques.

In choosing the techniques one should also take into account various other factors, a number of which are listed below.

- Quality characteristic to be tested

A technique is fit for testing one or more quality characteristics. Some quality characteristics can best be tested with one (set of) techniques, others with another one.

- Area of application

Some techniques are specifically suitable for testing the interaction (screens, reports, on-line) between the system and the users, others are better in testing the

processing of systems (batch processes). There is a relation with the type of error to be found with a technique, e.g. false input checks, incorrect processing or errors of integration.

- Availability of test basis

Each techniques starts from a certain test basis. This may be the functional specification, the technical design, program code or descriptions of the end-user organization. The exact form of the test basis is also relevant to the choice of a technique, e.g. decision tables, pseudo-code, structured language or unstructured prose.

- Extent of formality

Informal test specification techniques offer more freedom for the tester in making the test cases than do formal techniques.

- Use of resources

The application of a techniques requires a specific amount of resources, in terms of man capacity as well as machine capacity. The use of resources has a direct relation with costs.

- Required knowledge and skills

Not each tester is equipped for each technique. For the useful application of some techniques much business knowledge is needed. For other techniques more analytic talent is required. Therefore, the knowledge and skills of the test staff also influences the choice for techniques.

For practical reasons one should attempt to cover all selected quality characteristics with a minimal set of test specification techniques.

The selection of the test specification techniques should be done in an early stage of the test process, for then the test team can take the appropriate actions in training for techniques and the necessary checklists can be made or adjusted for the specific situation.

As a result of this step the techniques that will be used per subsystem are defined. Optionally, especially with large test projects, this last step in the test strategy is performed slightly later in the process, namely during the preparation phase. As a part of this the priority order of the tests to be performed is determined. The aim of this is to have the most important tests take place as early as possible.

5.3 Strategy during the Test Process

The test strategy determined in advance often will be put under pressure in a later stage of the test project. In such a situation the test manager is asked to perform less tests or shorter tests in order to conform to the adjusted schedule. The consequences are to be seen mainly in the last step of strategy development: suddenly some tests must be cancelled or must be carried out with less depth. Using the test strategy as a basis the test manager may discuss with the customer which tests can be dropped or where less thorough testing can be done. By indicating which parts are to be tested less in relation to the risks assessed (translated into importance levels in the strategy), the test manager can report in a solid fashion on the increased risks after the testing phase. Therefore it is essential not to change the steps 1 to 5: the risks and the importance levels do not change. The result is that, when testing is reduced, there will be more risks after the system has been implemented.

Apart from this there is also the situation that during testing it turns out that part of the system contains an excessive number of errors or excessively few errors. Both cases justify adjustments in the test strategy, namely the increase and decrease of test effort, respectively. Contrary to the situation in the previous

paragraph here risks will remain the same after implementation of the system. The correction can be summarized as follows:

Testing should continue as long as the costs for finding and correcting errors during testing are lower than the costs connected to the error occurring in production

---

In 'finding and correcting errors' more costs than just test costs play a role; other, extensive costs may be concerned with the delay in shipping the product. For 'the error occurring in production' one should also take into account the chance that the error will actually occur: an error that will never occur is no error (defect)!

5.4 Strategy during Maintenance

The main difference between the development of new systems and maintenance for the test strategy is the chance of error. In the case of maintenance changes are made to an existing information system. These changes should be tested. During maintenance there is a risk that new errors are introduced, with a decrease in quality for the system as a result (regression).

The implication of this different chance of error in the case of maintenance implies for the strategy that the relative importance of the subsystems may change: a subsystem which had a high importance when it was developed, may be unchanged in maintenance. Since the chance of regression is the only risk in this case, the test importance is much lower. Therefore, test strategy development for a test level can be modified by changing the concept of 'subsystem' in the steps to 'change'. For each change it is analyzed which system parts were mutated, which parts may have been influenced by the change, and which quality characteristics are relevant. There are various possibilities for testing each change, dependent on the risks:

- A limited test, only focused on the change;
- A complete (re)test of the function which had been changed;
- A test of the coherence of the changed function and the adjacent functions.

There should also be a regression test for the system as a whole. This test focuses on the coherence of the changed and unchanged parts of the system, since the chance of regression is largest here. If the test strategy for the new developed system is available, the importance levels attributed to the subsystems here can be of use.

Apart from the changed chance of error there are more differences between a system developed and a system under maintenance. However, they have no influence on the *technique* of test strategy development. Examples of these differences include:

- Test basis is missing, incomplete or not up-to-date

This situation, frequent in maintenance, has consequences for the test specification techniques to be chosen.

- Predictable versus ad-hoc maintenance

The majority of maintenance situations are predictable and can therefore be planned. The strategy determination is easily to be applied to this type of maintenance. The situation is more difficult in the case of ad-hoc maintenance, where the focus is on putting right a production break-down and getting the system in the air as soon as possible. A formal strategy determination costs too much time here. It is feasible, however, to have some test strategy scenarios available: if program x goes wrong and is repaired, what should be tested? These scenarios support an optimal test for ad-hoc maintenance.

Copyright © 2001 IQUIP Informatica B.V. The Netherlands

6. Conclusion

The testing of information systems can be based on the business risks, which the organization will experience in using these information systems if the system is not tested. In practice, test managers often take the steps to come from risks to test coverage in an intuitive manner. In this paper, the steps needed for the definition of a test strategy are made explicit. The result of such a test strategy is better insight for all parties involved and a sound basis for negotiating testing depth.

Good risk assessment is a part of these steps. It is essential to realize that this explicit way of looking at risks cannot be done by a test manager or tester alone. It is necessary to ascertain for the involvement of users and managers of the client organization, of auditors, and of project people such as developers, testers, QA staff and project managers. In practice, the discussion of risks and related testing strategies in this way proves to be a real eye-opener for all parties concerned. It also enables negotiation of testing depth by having the customer decide which elements should be tested how thoroughly.

The stepwise definition of the test strategy can be used for any test level (e.g., system test, acceptance test) and also for an overall strategy (master) test plan, including and coordinating all test levels and inspections/reviews.

7. References

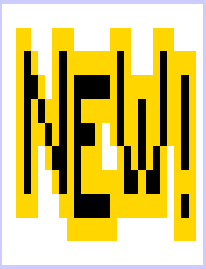
In English with a short description of test strategy
Pol, Martin and Veenendaal, Erik van, Structured Testing of Informations Systems (1998), tenHagenStam, The Netherlands, ISBN 90-267-2910-3,

In Dutch with an extensive description of test strategy
Pol, Martin, Teunissen, Ruud and Veenendaal, Erik van, Testen volgens TMap (1999), Tutein Nolthenius, s' Hertogenbosch The Netherlands, ISBN 90-72194-58-6.

Deutsch
Pol, Martin, Koomen, Tim und Spilner, Andreas, Management und Optimierung des Testprozesses, Ein praktischer Leitfaden für erfolgreiches Testen von Software mit TPI und TMap, dpunkt verlag Heidelberg, ISBN 3-932588-65-7

The translation into English is planned for November 2001.
Pol, Martin, Teunissen, Ruud and Veenendaal, Erik van, Software Testing, Introducing the TMap approach, Addison Wesley London, ISBN 0201745712

Copyright © 2001 IQUIP Informatica B.V. The Netherlands



QWE2002 Paper 10M

Dr. Robert Darimont, Ms. Emmanuelle Delor & Mr. Andre Rifaut
(CEDITI)

Quality Starts by Defining Goals

Key Points

- Inadaquate requirements analysis leads to major quality issues.
- The Kaos methodology allows analysts to produce very high-quality requirements documents.
- The Kaos methodology is goal-oriented and is supported by a tool: Grail.

Presentation Abstract

The paper presents the principles of the Kaos method, the functions of the Grail tool, the process followed during analyses, and feedback from experience to carry out requirements analyses, to draw up IT master plans, strategic analyses and IT specifications.

About the Author

Robert Darimont received the MS and PhD degrees in applied sciences (softare engineering orientation) from UCL (Université catholique de Louvain), Belgium. He is the manager of the IT consulting and R&D department at CEDITI, the IT transfer center of the University of Louvain. He leads industrial projects using Kaos; and helps companies to adopt up to date and precompetitive software engineering technologies.

Emmanuelle Delor received the MS degree in computer science from the University of Namur, Belgium. She is senior consultant at CEDITI, the IT transfer center of the University of Louvain. She participates to industrial projects using Kaos, and is responsible for the QA program at CEDITI.

Andre Rifaut received the MS degree in applied sciences (applied mathematics orientation) from UCL (Université catholique de Louvain), Belgium. He is senior consultant at CEDITI, the IT transfer center of the University of Louvain. He participates to industrial projects using Kaos, and is involved in the design of advanced features of Grail, the tool supporting the Kaos methodology.

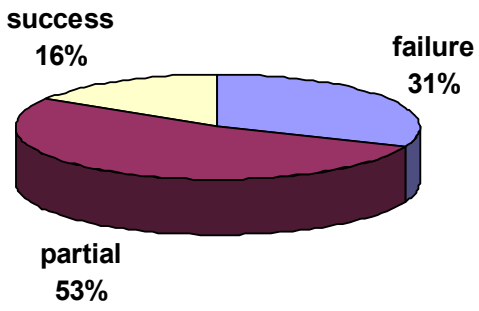
Quality starts with the definition of goals

Robert Darimont, Emmanuelle Delor, André Rifaut

e-mail: {Robert.Darimont, Emmanuelle.Delor, André.Rifaut}@cediti.be

© CEDITI

Why Focus on Requirements Engineering?



Standish group USA 1995 ; sample over 365 companies and 8380 IT projects

© CEDITI

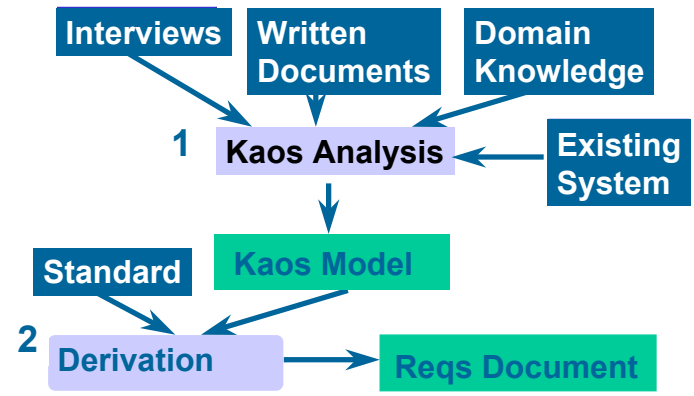
10 first factors of failure

• **Lack of user input**	**13 %**
• **Incomplete reqs & specs**	**12 %**
• **Changing reqs & specs**	**11 %**
• lack of executive support	8 %
• technology incompetence	7 %
• lack of resources	6 %
• **Unrealistic expectations**	**6 %**
• **Unclear objectives**	**5 %**
• unrealistic time frames	4 %
• new technology	3 %

© CEDITI

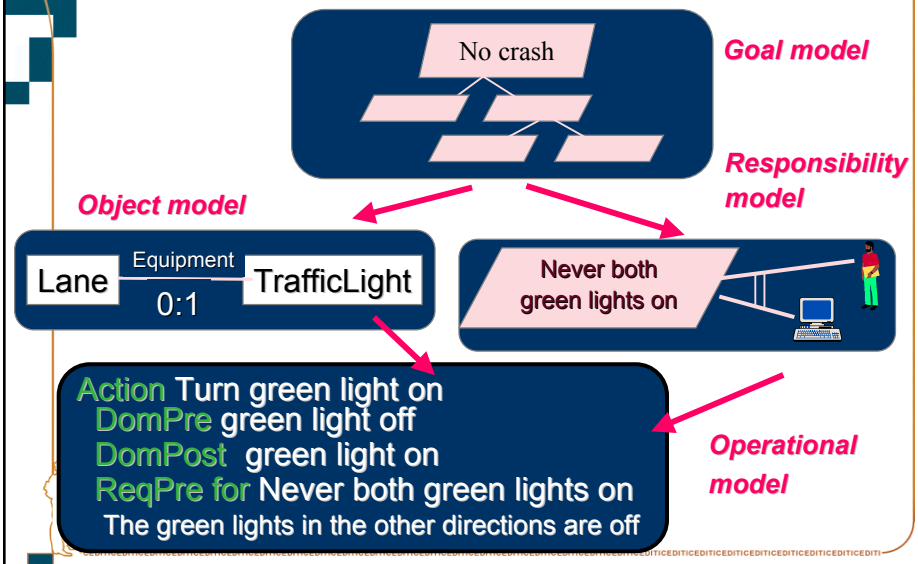
Kaos approach

Study the **problem** first, then specify a solution
Study the software system and its **environment**



© CEDITI

Kaos method : overview



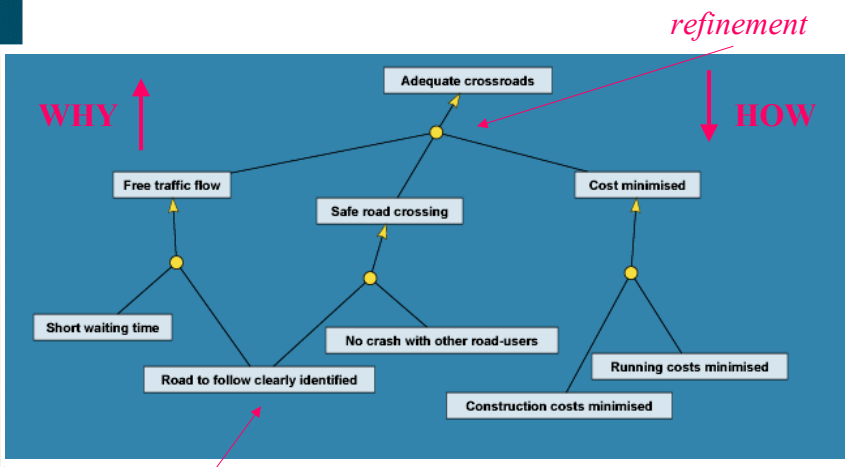
© CEDITI

Goal model

- Who are the stakeholders and what do they want?
- Goals defined in terms of system states to reach/maintain (leave/avoid) + optimisation
- Goal graph: goals + relationships
  - refinement: goal/subgoal
    - why question : subgoal → goal
    - how question: goal → subgoals
  - alternatives
  - conflicts

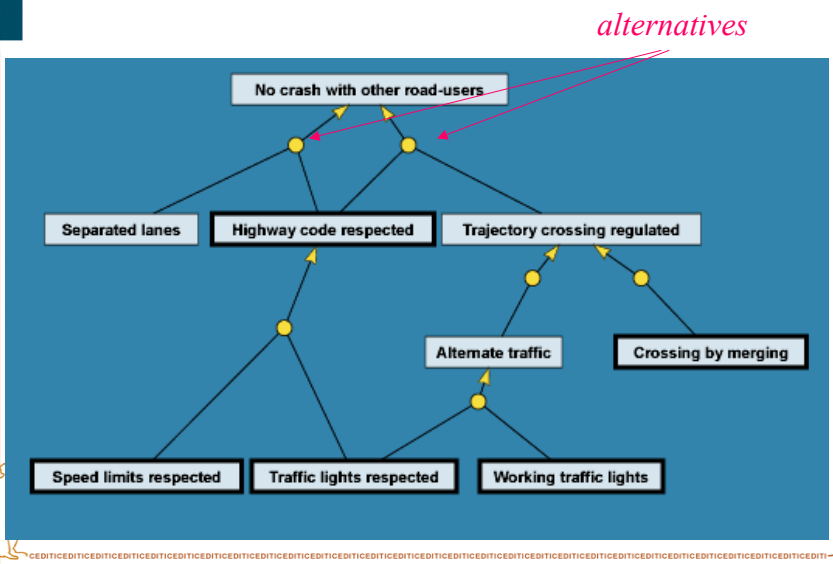
© CEDITI

Goal model (example)



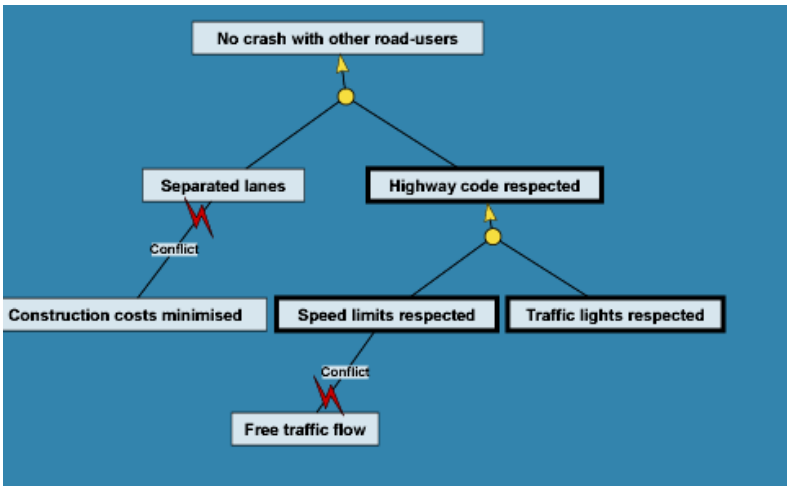
© CEDITI

Goal model (example)



© CEDITI

Goal model (example)



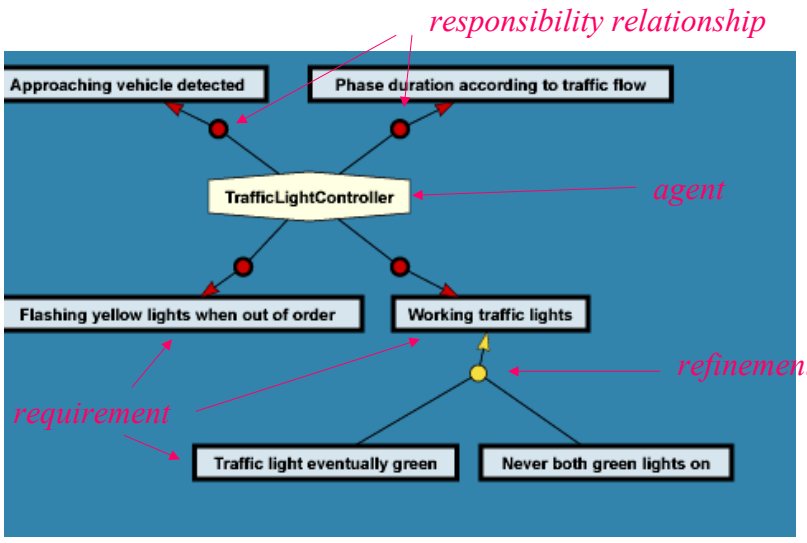
© CEDITI

Responsibility model

- Agent = human or software processor
- Requirement = goal under the responsibility of an agent
- Responsibility diagram: shows the responsibilities assigned to an agent

© CEDITI

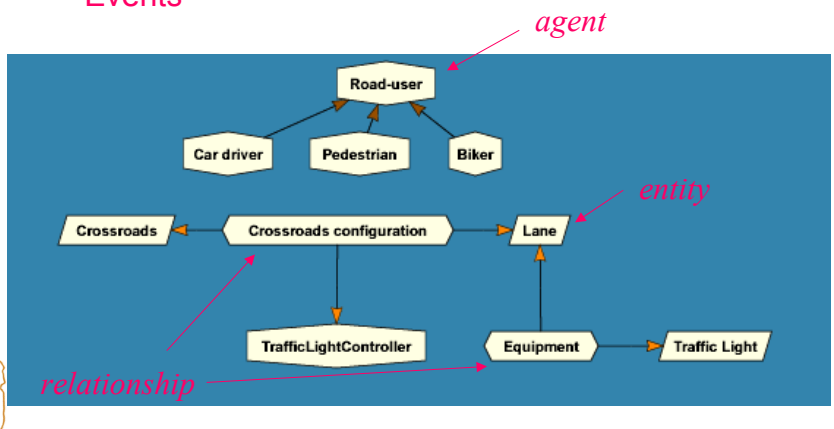
Responsibility model (example)



© CEDITI

Object model

- Agents
- Entities & Relationships
- Events



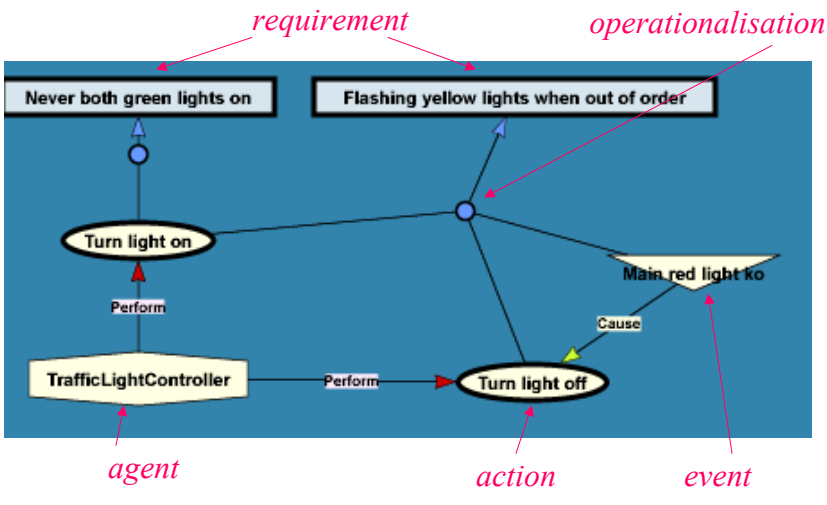
© CEDITI

Operational Model

- Behaviours that agents have to adopt to satisfy assigned requirements
- Action = state transition
  - domain pre-, post-, trigger-condition : standard, component-based behaviour
  - strengthened pre-, post-, trigger-condition : customised behaviour to satisfy the assigned requirements
- Capability & Perform link (agent -- action)
- Cause & Stop link (event -- action)
- Operationalisation link (action/object to requirement)

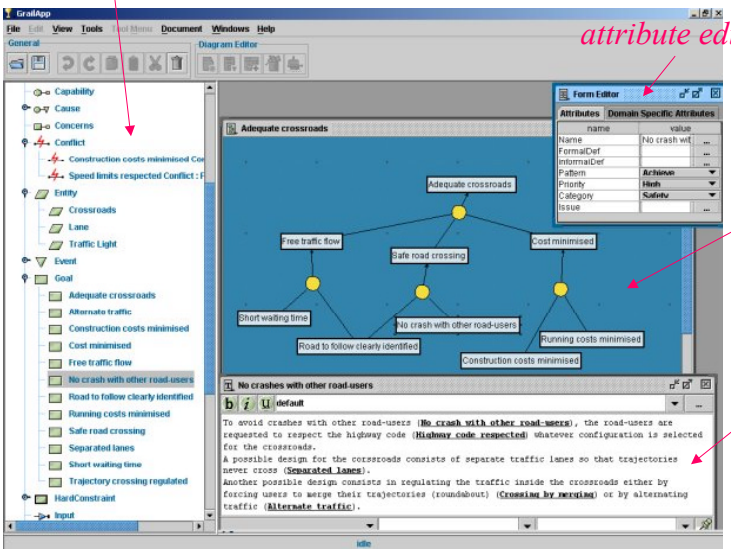
© CEDITI

Operational model (example)



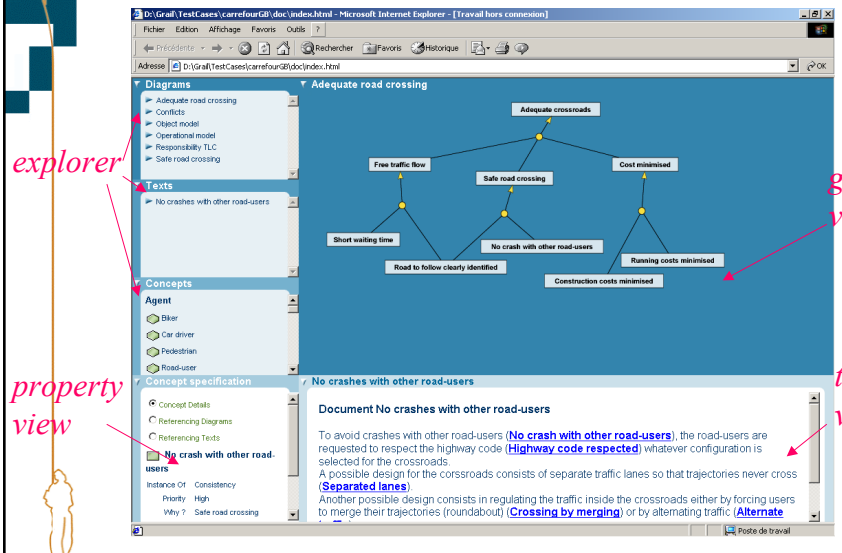
© CEDITI

Grail tool



© CEDITI

Grail generated documentation



© CEDITI

Experience Report

- Publishing (copyright; media sales, distribution, advertising)
- Aeronautics (ATC procedures)
- Pharmaceutical industry & distribution
- Language industry
- Telecoms (cable phone; SPI tool; R&D)
- Hospitals (clinical reporting; IT plan)
- Mechanical industry (Finite scheduling optimisation)

Requirements documents produced:
- 100-150 pages long
- IEEE 830 standard compliant
- exhaustive glossary

© CEDITI

Experience report

- Well-structured, self-contained, motivated, easily understandable documents
- Calls for tender managed more easily
- Highly effective way to communicate about the system
- Vertical traceability

© CEDITI

Conclusion

- Kaos : one of the most prominent RE methods
- Grail tool : effective and available soon

Quality definitely starts
with the accurate definition of goals.

© CEDITI

Quality Starts with the Definition of Goals

Robert Darimont, Emmanuelle Delor, André Rifaut
CEDITI, Avenue Georges Lemaître, 21, B-6041 Charleroi -- Belgium
{Robert.Darimont, Emmanuelle.Delor, Andre.Rifaut}@cediti.be

1. Introduction

A weekly journal for private sector executives and directors recently featured an article entitled: "No baker would survive with only 16% satisfied customers. Half of the failures in IT projects are due to inadequate communication". This article referred to a statistic published in 1995 by the Standish Group [STG 95] which reveals a very harsh reality from a sample of more than 8000 IT projects carried out by some 350 American firms: only 16 % of projects can be described as complete successes. They eventually succeed while keeping to the anticipated costs and functions. Almost one in three projects never succeed and the remainder (over half) show a half-measure result with partially installed functions, a cost approaching 200% compared to the initial budget, and with a significant delay.

When the causes of these failures are analysed, it emerges that almost half of them are due to **inadequate analysis of requirements.** The survey goes further and breaks down that inadequacy as follows: lack of information for the user represents 13% of the causes of failure, inadequacy of the specifications 12%, their changing nature 11%, unrealistic expectations 6%, and confused objectives 5%. Discovering a requirement during the course of development leads to cost overruns and delays, which are even larger when this discovery is made late in the day.

This paper presents the principles of the Kaos method, the functions of the Grail tool, the process followed during analyses, and feedback from experience from carrying out requirements analyses, from drawing up IT master plans, strategic analyses and IT specifications.

2. The Kaos method

Most of the software engineering methodologies currently in use are dedicated to the specification of a solution and do not really address the description of the problem, that is, specifying the goals to be met, describing the application domain, deciding who will perform which actions to satisfy the goals, etc.; in one word, to perform a requirement analysis.

A requirements analysis in Kaos [DAR 93, DAR 96, VLA 98{a,b,c}] is broken down into three phases:

- gathering of the information to be used as a guide for the goals to be achieved
- modelling
- drawing up of a report (specifications, synthesis, IT master plan, etc.)

The aim of the modelling phase is to identify all the relevant concepts by completing an information framework having a pre-established structure (metamodel). This framework enables the analyst to classify the concepts that he/she identifies and to relate them to each other.

The framework pre-defines different types of concepts and different types of relationships between concepts. Those concepts types and relationships are explained in the following sections. A single example will be used throughout the paper to illustrate the approach. This example is described first.

2.1 Case study

> *We have been called by the authorities of a large city to help select the best configuration for a new crossroads. The authorities expect a report that motivates the choices we advocate and that anticipates the requirements document for the actual construction of the crossroads.*

Let us observe that we are asked to analyse a problem, not *necessarily* a software problem. If a software component is needed to implement the solution, it will be introduced later on. This yields two key ideas of the Kaos method :

Key idea. A requirement analysis should not only study the software system we have in mind but also the part of its environment with which the system will interact.

Key idea. Before attempting to describe a **solution** in a requirements document, it is necessary to describe and analyse the **problem** which we are faced with. In fact, we strongly believe that a good requirements document specifies the problem to be solved and minimal technical constraints on the solution to be built, no less and no more.

2.2 Goal model

The first task of the analysis will be to address the following questions: 'who are the stakeholders implied in this problem?' and 'what do they want?'. Answers to these questions will be gathered from interviews, written documents, observation of an existing system, and knowledge about the problem domain. The more an analyst is acquainted with a specific domain, the easier it will be for her to find good abstractions and to elicit the needed properties.

Case study. The following table shows some of the stakeholders and some of their wishes for the road crossing example.

Table 1: Stakeholders and their wishes

Stakeholder	Stakehoder's wishes
Car driver	no crashes with other road-users free traffic flow road to follow is clearly known
Pedestrian	safe road crossing short waiting time
Public authorities	construction costs minimised running costs minimised no casualties
Biker	not to be run over by cars free traffic flow

In Kaos, stakeholder's wishes are named **goals**. Goals can be shared by different stakeholders and goals typically need the cooperation of different stakeholders to be satisfied.

For instance the goal *Free traffic flow* is shared by car drivers and bikers. It needs the cooperation of all road users; the way the crossroads is configured and managed will probably contribute to satisfy this goal.

Goals can often be defined in terms of system states to reach (or leave) or system states to maintain (or avoid). A goal can also require the optimisation of some parameter(s).

The Kaos method invites the analyst to collect and relate goals with each other: some goals contribute to other ones; others can be conflicting with each other. Abstraction can be introduced to highlight common goals. To help the analyst relate goals with each other, he/she is prompted to ask the following questions for each goal that has been identified:

- the '**why**' question: why is this goal needed ? What is the upper-level goal that motivates this goal?
- the '**how**' question: how can this goal be satisfied? What are the lower-level goals that will contribute to the satisfaction of this goal?

In Kaos, a goal **refinement** is a relationship between goals that allows the analyst to connect a goal with its contributing subgoals.

Case study. Figure 1 shows a Kaos goal diagram related to the case study. Rectangles stand for goals. Circles stand for refinements of upper goals into a set of lower subgoals. Certain goals have been introduced to put some structure in the graph as for instance *Costs minimised* or *Adequate crossroads*. A goal can contribute to more than one parent goal; for instance, the goal *Road to follow clearly identified* contributes to the *Free traffic flow* as the driver will not hesitate about the direction to follow if road signs are clear; it also contributes to the goal *Safe road crossing* as a driver hesitating about his/her way or looking for road signs increases the risk of a crash.

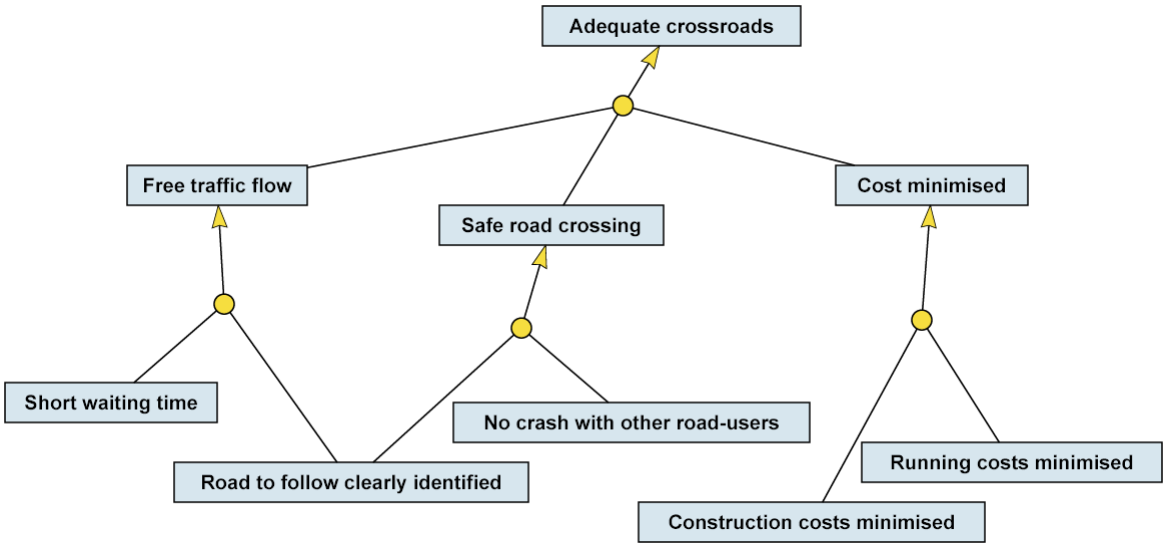


FIGURE 1 : High-level goals

Alternatives describe different ways to achieve goals. They can be motivated by different policies or design decisions as for instance the technology used or the quality of the solution they propose (reliability, performance, ...). Alternatives are not necessarily exclusive, that is, one can decide to implement a system with several of them included, for instance, to improve system reliability. Alternatives do not define partitions as different alternatives may share common subgoals.

Case study. Figure 2 shows some alternatives refining the goal *No crash with other road-users*. To avoid a crash, that is, two crossing road-users at the same place at the same time, one can choose to never have two crossing road-users at the same place by separating the lanes they follow (as in interchanges). One can also choose to allow crossing road-users to be at the same place but never at the same time by regulating the traffic flow. Both alternatives share a common subgoal: highway code must be respected by the road-users. Traffic regulation, in turn, can be achieved either by forcing road-users to merge their trajectories inside the road crossing (roundabout) or by alternating traffic in each crossing direction. A way to implement the latter is to require traffic lights.

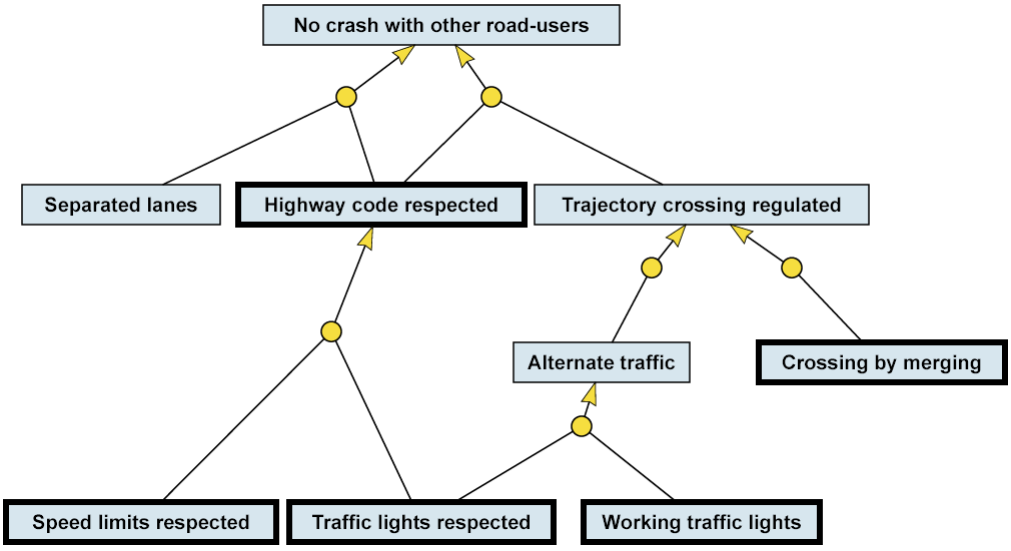


FIGURE 2 Alternatives

Conflicts. Goals expressed by different stakeholders may be conflicting. In Kaos, a conflict stands between two goals if there can be system states in which both goals are logically contradictory. It is important to identify conflicting goals as early as possible in the life cycle to minimise the risk of system rejection by some category of stakeholders.

Case study. The goal *Separated lanes* in Figure 2 may be conflicting with the goal *Construction costs minimised* in Figure 1as the construction of an interchange is one of the most expansive designs to organise a road crossing. The goal *Speed limits respected* in Figure 2 may be conflicting with the goal *Free traffic flow* as they could down the traffic flow. Conflicts are graphically represented in a goal graph by connecting the conflicting goals with a labelled link (see Figure 3).

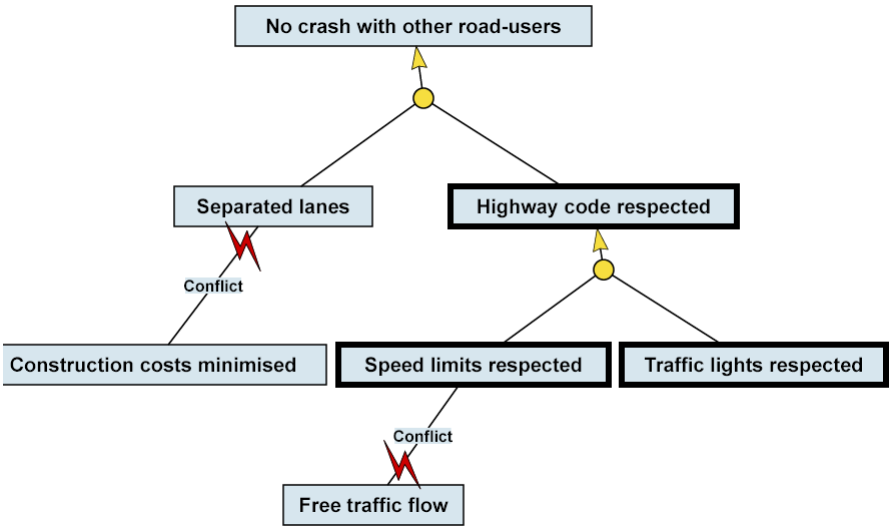


FIGURE 3. Conflits

2.3 Responsibility model

Key idea. In Kaos, a **requirement** is a goal, the satisfaction of which can be put under the responsibility of an agent.

An **agent** in Kaos is a processor (human or software) capable of behave to achieve assigned goals.

Case study. Requirements have already been shown in Figure 2. There are represented as goal rectangles appearing with a thick border. For instance, *Highway code respected* and *Working traffic lights* are examples of requirements. The figure also shows that requirements, like goals, can be refined. For instance the requirement *Highway code respected* is refined into two requirements: *Speed limits respected* and *Traffic lights respected*.

A Kaos responsibility model typically contains several diagrams. Each diagram focuses on the responsibilities of an agent in the system. The agent is drawn at the centre of the diagram and all the requirements for this agent are displayed around it.

Case study. Figure 4 shows requirements for the traffic light controller agent. Agents are represented by hexagons. Responsibility relationships between an agent and a requirement are also represented graphically.

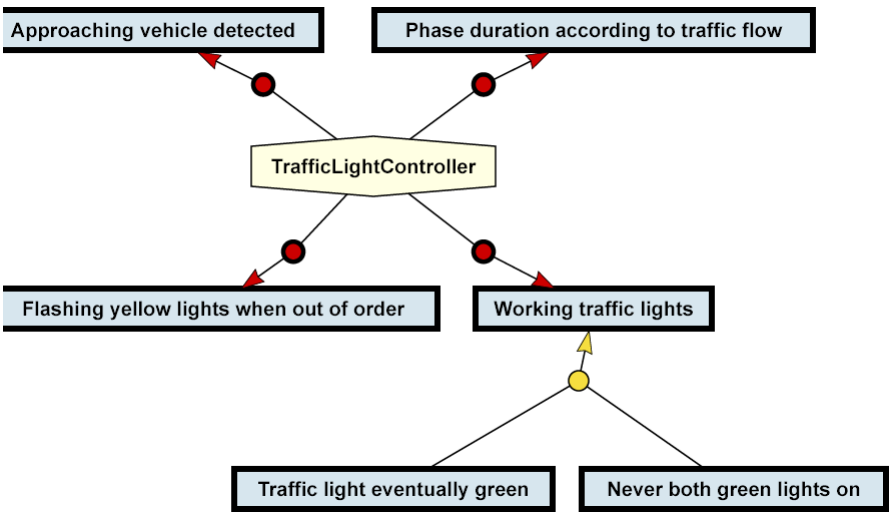


FIGURE 4: Responsibility relationships

During analysis, different candidate agents can be assigned to fulfill a requirement. Selection among agents can result from a cost, reliability or performance comparison. Even if, in the final requirements document, only one agent is responsible for a requirement, it is important to keep track of the rationale behind the choice of a particular agent for satisfying a requirement as this choice can be revisited later on for next releases of the system.

Key idea. A Kaos model is elaborated non only to produce a particular result at a particular instant in time (as for instance a requirements document), but also to accompany the system documentation during its lifetime.

2.4 Object model

Goals are defined by means of properties relating system states with each other. In Kaos, these system states consist of object states. Object states aggregate attribute name-value pairs defining the underlying objects. The Kaos object model allows the analyst to define Kaos objects almost as a UML analyst defines classes in a class diagram with the following differences:

- A distinction between four kinds of objects is made: entities, relationships, agents, and events. Agents are active objects capable of actions to achieve goals. **Entities** are passive objects whose states are modified by actions performed by agents. **Events** are instantaneous objects that cause and stop actions. Agents, entities and events are autonomous, that is, their definitions do not rely on the definition of other objects. **Relationships**, on the other hand, are objects dependent on other linked objects.
- Relationships are first-class objects. They can be binary, ternary or n-ary. Attributes can be attached to them; they can be specialised. Minimum and maximum cardinalities can be specified on links. A cardinality on a link indicates a number of times a same instance of an object can be linked through the relationship.

Case study. Figure 5 shows an object model in which the agents for the crossroads have been put together. The agent *Roaduser* has been introduced as a common abstraction for *Car drivers, Bikers,* and *Pedestrians.* This agent will be used in the model when the indication of a specific agent category is not relevant. Entities are represented by parallelograms and relationships by flattened hexagons (compare with agents). Links between a relationship and its linked objects are shown by arrows linking the relationship to the linked objects. For instance, *Crossroads configuration* is a ternary relationship between two entities *Crossroads* and *Lane*, and an agent: *Traffic Light Controller*.

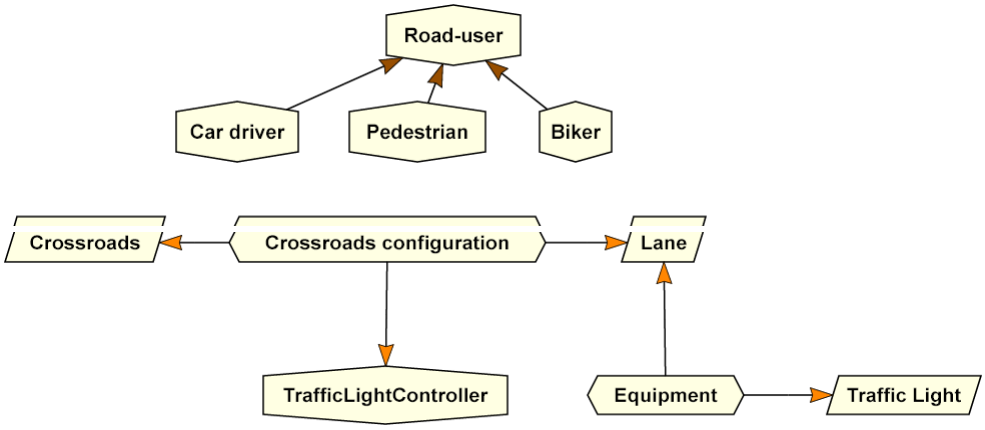


FIGURE 5. Object model

2.5 Operational model

Besides defining goals, objects and requirements under the responsibility of agents, Kaos allows the analyst to describe the behaviours that agents have to adopt in order to satisfy the requirements assigned to them. In Kaos, behaviours are named **actions** and represent state transitions. They are described in terms of preconditions (minimal conditions on the system state which must be met before executing the action), trigger conditions (conditions on the system state making execution of the action mandatory), and postconditions (conditions on the states resulting from performing the action if the precondition holds). Each kind of condition is split into two categories:

- domain conditions: conditions that hold independently of specific requirements. They describe a "standard" behaviour according to predefined agent capabilities.
- strengthened conditions: additional conditions that are introduced to strengthen agents' standard behaviours so that they will be able to achieve preset requirements.

When an action allows an agent to fullfill a requirement, an **operationalisation** relationship is created between the requirement and the action.

Key idea. A clear distinction between domain and strengthened conditions favors traceability from goals to operational required behaviours. Impact and risk analyses will also be easier to produce when future releases of the system put certain requirements and behaviours in question.

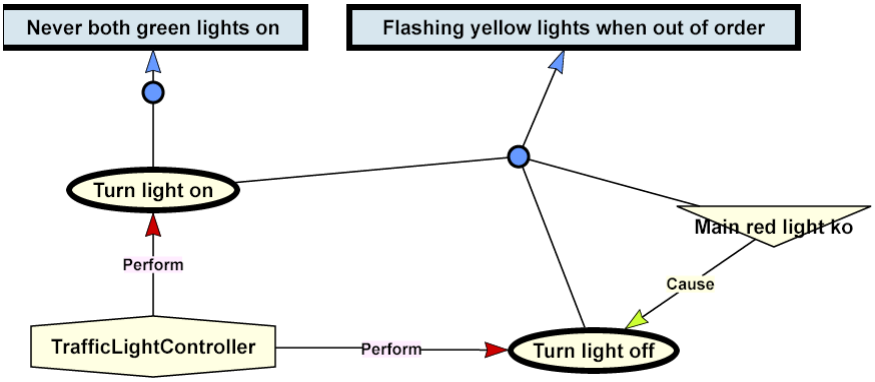


FIGURE 6. Operational model

Case study. Figure 6 shows the action *Turn light on* performed by the agent *Traffic Light Controller*. The action operationalises the constraint *Never Both Green Lights on.* Details on the specification of the action show that the domain precondition for the action is 'the light must be turned off' and that the domain post-condition for the action is 'the light is switched on'. A strengthened precondition will prescribe that if the action is applied on a green light, the green lights for the other directions must be off. The figure also shows how the constraint *Flashing yellow lights when out of order* is operationalised by two actions (*Turn light on* and *Turn light off*) and one event that triggers the *Turn light off* action (*Main red light ko)*.

3. The Grail tool

A prototype tool has been developed to support Kaos analyses [DAR 97]. It consists in a graphical editor to represent the concepts graphically, a form editor to describe each concept textually, a report generator extracting information from the model and presenting it according to a predefined template, a database for querying about concepts and their "neighbourhoods", and a hypertext generator for browsing through the model.

To address the increasing interest in the methodology, a commercial version of the tool, Grail, is currently being developed. Version 1 of the tool will contain the following components:

- a graphical editor to represent the concepts and their relationships,
- a text editor allowing the analyst to record interview summaries or to associate descriptive texts to diagrams,
- an attribute editor to specify predefined attribute values or user-defined attribute-value pairs
- an explorer to retrieve diagrams, text documents, and concepts by names, types or occurrences
- a hypertext documentation generator that allows one to inspect the entire model with a Internet browser tool in a very user-friendly way.

The Grail tool is a ***meta*** case tool: the Kaos methodology is not hard-coded in the tool. Evolution or specific customisations of the tool will be easier to implement. Specifications are saved in XML format, including diagrams (in the SVG format).

The tool will be available on all major platforms since it is implemented in Java. It has been designed to be easily integrated with other CASE tools. Integration can be achieved in three ways:

- by exchanging data in XML format;
- by interfacing with an event-based API;
- by querying the tool with OQL (an object-oriented query language standard similar to SQL).

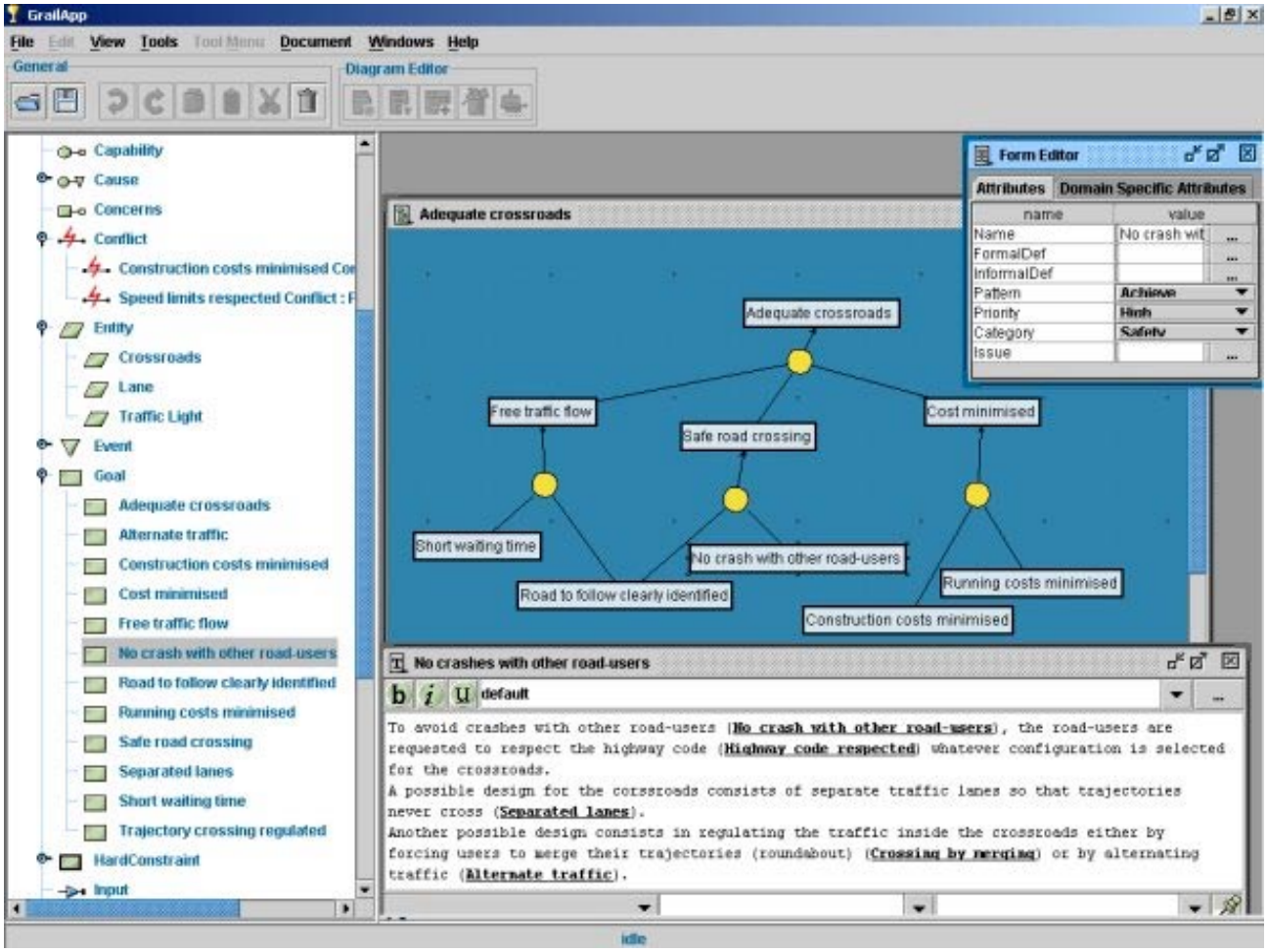


FIGURE 7. Grail

Figure 7 shows a screen shot of the tool. The explorer appears on the left of the screen. The right part shows a diagram edited by the graphical editor, the form editor and a descriptive text edited by the text editor.

Figure 8 shows the same information but in the Web-based generated documentation.

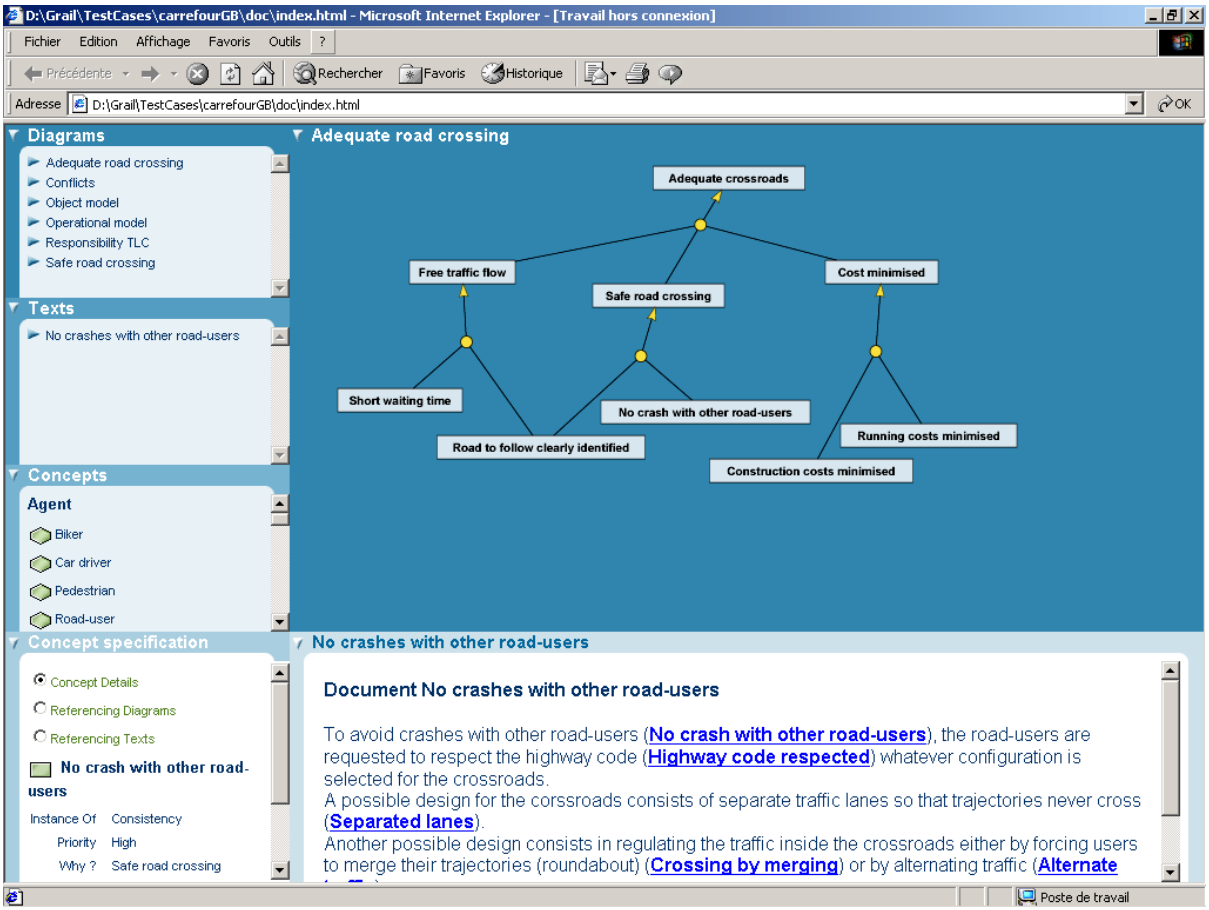


FIGURE 8. Grail Web-based documentation

4. Experience report

Experience acquired from numerous industrial studies performed by CEDITI has shown that the Kaos framework is highly efficient when carrying out requirements analyses, devising IT master plans or producing strategical analyses.

The following table summarises the kinds of projects already realised.

Publishing	Reqs engineering for a complex copyright management system; Reqs document for Media Sales, Distribution & Advertising Management
Aeronautics	Reqs traceability for Air Traffic Control Procedures
Drugs Industry & Distribution	Strategic analysis; Reqs for an e-learning system
Telecommunications	Requirements re-engineering of a cable telephone system
Language Industry	Reqs for Web-based professionnal and on-the-fly translation tools
Hospitals	IT plan, Reqs for standard clinical reporting
Earthmover Factory	Finite scheduling optimisation

Requirements documents (typically 100 to 150 pages long) produced by the method and its associated tools are IEEE 830 standard compliant. About 20% of such documents consist in an exhaustive glossary of terms used in the rest of the document. The glossary is directly derived from the Kaos object model. The requirements are progressively introduced in the text according to a top-down, linearised traversal of the goals graph.

Our experience has shown the following benefits and returns of the Kaos approach:

- requirements documents produced with the method have been unaminously recognised by our customers for presenting remarkable qualities: they are well structured, self-contained, motivated, easily understandable, and calls for tender is managed more easily;

- the Kaos model is a highy effective way to communicate about the system: it provides a common framework helping the different stakeholders to understand each other's viewpoints: managers acknowledge goals, requirements and assignments that underly operational models; technicians acknowledge operational models that are motivated by the goals and that can be easily mapped to UML;
- it turns out that many companies lack vertical traceability between the company strategy and the requirements documents produced on the one hand, and between the requirements document and the solution specification on the other hand. A Kaos model allows one to trace requirements to goals and to trace high-level, coarse-grained behavioral specifications to requirements.

5. Conclusions

In the requirements engineering research community, the Kaos approach is considered as one of the most prominent methods for eliciting, analysing, and formalising requirements (see the References section below). Experiences in various industries have proved the benefits of the approach and have shown how well it improves the quality of requirements documents and, therefore, the quality of solutions based on such documents.

We now feel that the time has come to allow a larger number of companies to benefit from the approach: this explains why CEDITI will henceforth commercialise Grail and provide support for the Kaos approach.

New features that will soon extend the tool's capabilities include integration with existing market leader tools in requirements management and integration with existing UML modeling tools. More advanced features will be progressively incorporated. Among them, the edition of formal assertions and embedded tools to enable features like model checking, simulation, and test generation.

We strongly believe that all these features put together will easily convince professionals that quality definitely starts with the accurate definition of goals.

Acknowledgment. The Kaos approach results from several research projects led by Prof. A. van Lamsweerde (University of Louvain) and funded by the European Union, the Belgian and the Walloon governments. The Grail tool which will soon be commercialised is being built by a strongly motivated team of Java developers: Jean-Luc Roussel, Cédric Nève, Philippe Legrain, Denis Genard, Denis Ballant and Philippe Massonet, with the significant contribution of André Rifaut for providing requirements, feedback, and prototypes for the tool. Kaos analyses that have proved the benefits of the approach have been led by the authors of this paper. Several PhD theses have contributed to explore advanced features eventually deployed in the tool; more are on their way.

6. References

[DAR 93] Dardenne, A., van Lamsweerde, A., Fickas, S., Goal-directed Requirements Acquisition, in Science of Computer Programming, Vol 20, 1993, pp 3-5

[DAR 96] Darimont, R., van Lamsweerde A., Formal Refinement Patterns for Goal-Driven Requirements Elaboration, Proc. FSE-4, ACM 10/96, pp 179-190

[DAR 97] Darimont, R., Delor, E., Massonet, P., van Lamsweerde, A., Grail/Kaos : An Environment for Goal-driven Requirements Engineering, in Proc. ICSE 19, 05/97, pp 612-613

[STG 95] Standish Group, http://www.standishgroup.com/chaos.html

[VLA 98a] van Lamsweerde, A. , Darimont, R., Letier, E., Managing Conflicts in Goal-driven Requirements Engineering, IEEE Transactions on Software Engineering, 11/98

[VLA 98b] van Lamsweerde, A., Willemet, L., Inferring Declarative Requirements Specifications from Operational Scenarios, IEEE Transactions on Software Engineering, 12/98

[VLA 98c] van Lamsweerde, A., Letier, E., Integrating Obstacles in Goal-driven Requirements Engineering, Proc. ICSE'98, IEEE-ACM, 04/98

A full bibliography of publications about the method can be found at URL http://www.info.ucl.ac.be/research/ projects/AVL/ReqEng.html.

7. Bibliographical sketch

Robert Darimont received the MS and PhD degrees in applied sciences (software engineering orientation) from UCL (Université catholique de Louvain), Belgium. He is the manager of the IT consulting and R&D department at CEDITI, the IT transfer center of the University of Louvain. He leads industrial projects using Kaos; and helps companies to adopt up to date and precompetitive software engineering technologies.

Emmanuelle Delor received the MS degree in computer science from the University of Namur, Belgium. She is senior consultant at CEDITI, the IT transfer center of the University of Louvain. She participates to industrial projects using Kaos, and is responsible for the QA program at CEDITI.

André Rifaut received the MS degree in applied sciences (applied mathematics orientation) from UCL (Université catholique de Louvain), Belgium. He is senior consultant at CEDITI, the IT transfer center of the University of Louvain. He participates to industrial projects using Kaos, and is involved in the design of advanced features of Grail, the tool supporting the Kaos methodology.

QWE2002 Paper 11M

Mr. Ruud Teunissen
(Gitek nv)

The Art of Managing Fixed Price Test Project

Key Points

- Fixed Price Test Project = Win/Win situation.
- How to manage a Fixed Price Test Project.
- Practical examples: Lessons Learned.

Presentation Abstract

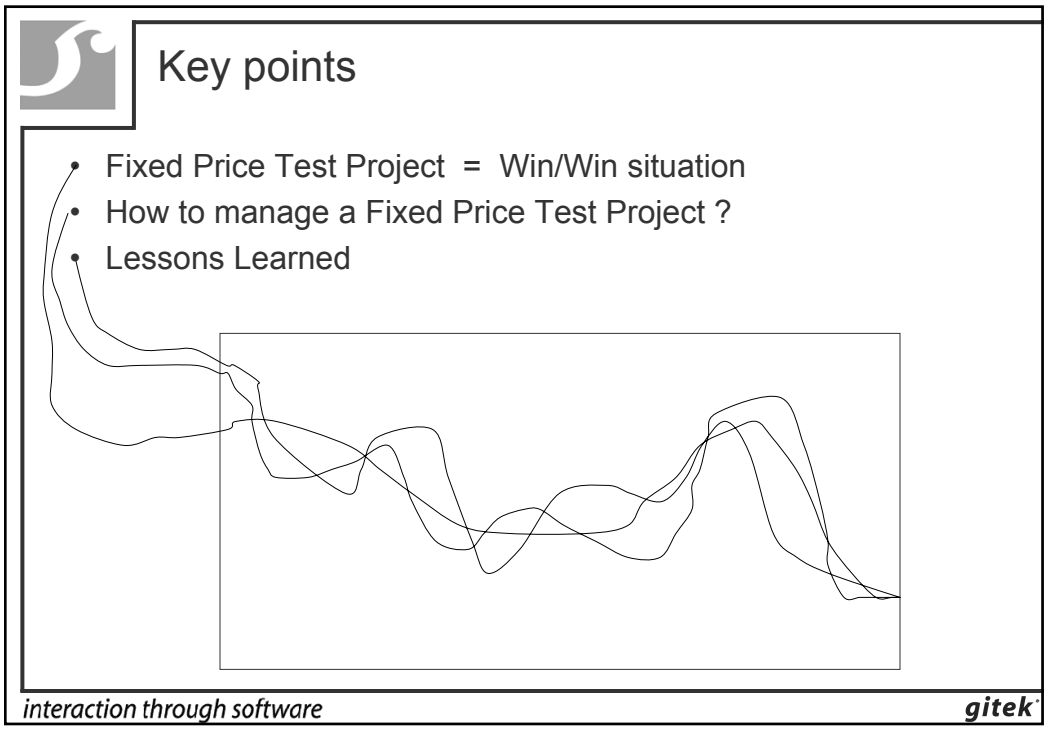
Fixed price projects have become more and more popular in software development, the demand for fixed price test projects is also increasing. But who is "mad" enough to do it? Controlling a “normal” test process is already a hell of a job, so who would be willing to sign a fixed price testing deal.
I will share our view on the possibilities on fixed price testing, discussing prerequisites, risks, test maturity, special skills required - including steel nerves.
With this paper I will give a satisfactory answer to the different questions that arise and scenario’s that happen during fixed price test projects. We have proven that fixed price test projects are possible and can be beneficial for all parties involved. Managing Fixed Price Test projects is the next step in controlling your test process - it is an art: The art of managing fixed price test projects

About the Author

Since 1989 Ruud Teunissen is employed in the testing world. He has been involved in a large number of ICT projects and has performed several functions within the testing organisation: tester, test specialist, test advisor, test manager, etc. Based on his experience, Ruud participated in the development of the structured testing methodology TMap® and is co-author of several books on structured testing. The last years Ruud is involved in implementing structured testing within organisations on the Belgian and Dutch market. At this moment Ruud has been working in Belgium for Gitek n.v. as Manager Testen and as such responsible for all testing activities at Gitek (over 50 professional testers).

Ruud is frequently speaking at conferences, universities, seminars, workshops etc. (for instance: QWE99, QWE 2000, ICS Test 2000, ICS Test 2001, EuroSTAR 2000, SAI, NOVI, Kluwer Education, IIR).

The Art of Managing Fixed Price Test Projects
Ruud Teunissen
Gitek nv - interaction through software
http://www.gitek.be
interaction through software
gitek

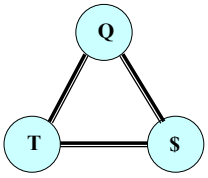

Key points
Fixed Price Test Project = Win/Win situation
How to manage a Fixed Price Test Project ?
Lessons Learned
interaction through software
gitek

Question!!!

- Is it possible to perform a test project as fixed price project ?
- Are you “mad enough” to do it… … ?
- Do you have steel nerves?

Definition of Fixed Project

- If you want more quality, you will need more time and since ‘time is money’, more money !
- If you want to reduce your project costs, you will have to live with lower quality …



- Definition
  A fixed project is a project where either quality, time, budget or a combination thereof is contractually fixed.

Four possible scenarios

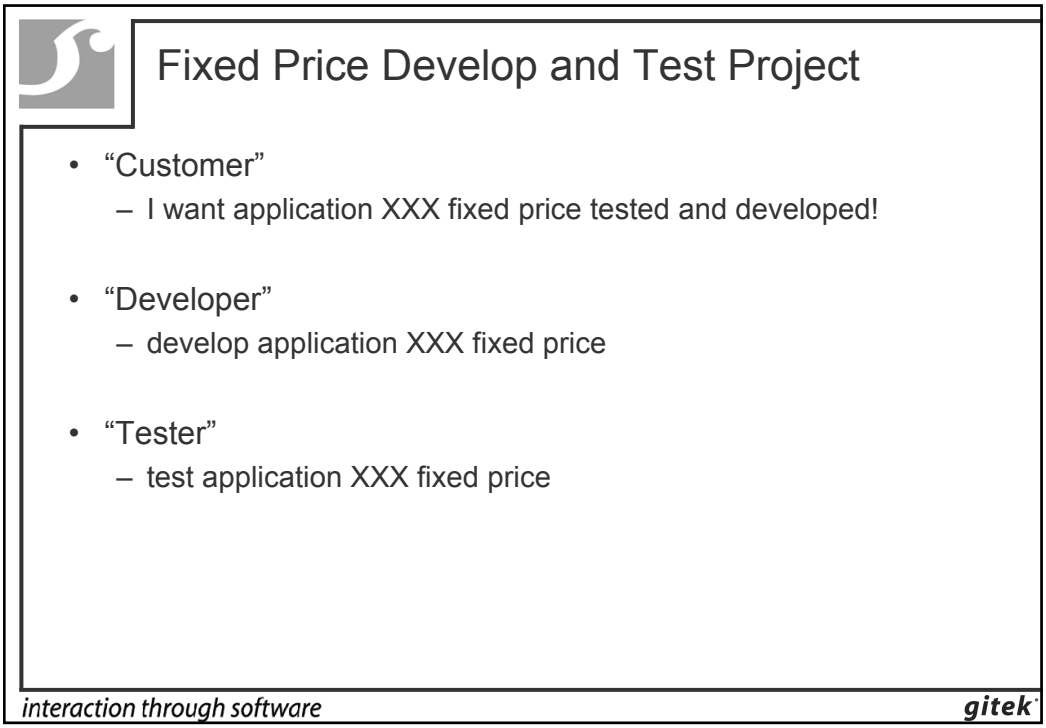
1. Fixed price
   - e.g. test an application for $1.000.000;
2. Fixed date
   - e.g. test a conversion before 31-dec-2001;
3. Fixed quality
   - e.g. build an application without critical bugs;
4. Fixed price fixed date
   - e.g. build an application before 1-jan-2002 for $1.000.000

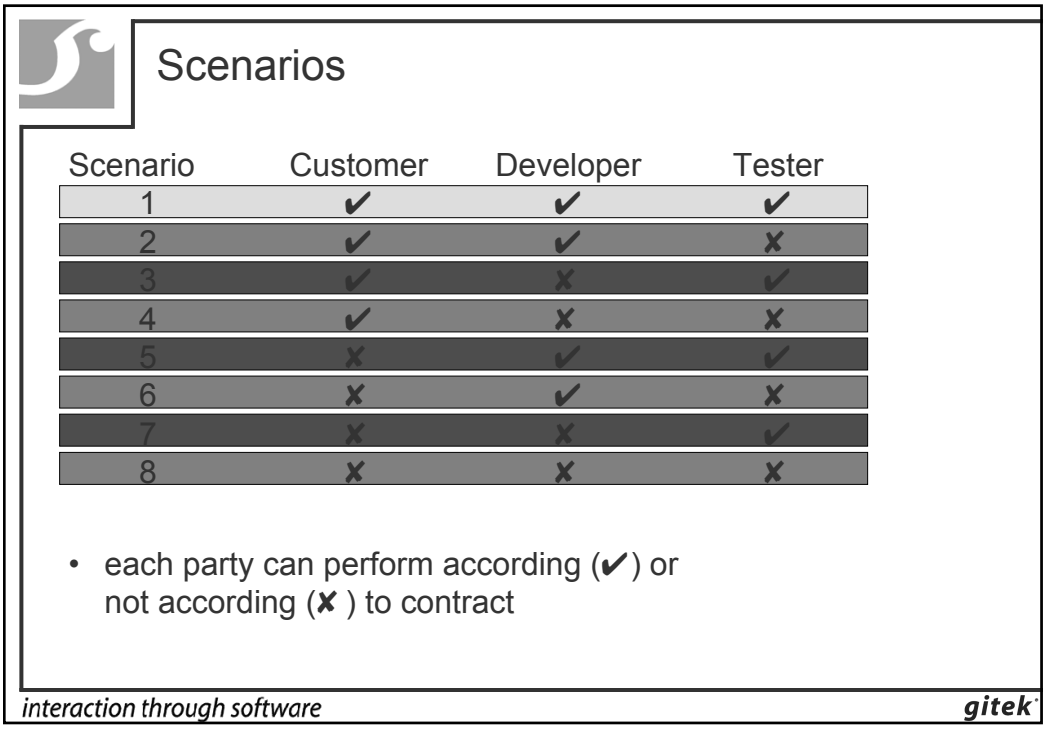
interaction through software gitek

Fixed Price = win/win !!

- Customer
  - Benefits
    - Easier to control your budgets;
    - The result of project is clearly specified;
    - Less risk (= shared risk !).
  - Concerns
    - Every change request has to be discussed with the supplier;
    - When the project remains on schedule, you were probably better of with a time and material based project.
- Supplier
  - Benefits
    - Confidence towards the customer: “We share your risks!”;
    - External marketing: “We control our test project!”;
    - Free in choosing human resources and project approach.
  - Concerns
    - A strict project follow up is inevitable;
    - Every change request has to be discussed.

interaction through software gitek


Fixed Price Develop and Test Project
• "Customer"
– I want application XXX fixed price tested and developed!
• "Developer"
– develop application XXX fixed price
• "Tester"
– test application XXX fixed price
interaction through software
gitek


Scenarios
Scenario Customer Developer Tester
1 ✔ ✔ ✔
2 ✔ ✔ ✘
3 ✔ ✘ ✔
4 ✔ ✘ ✘
5 ✘ ✔ ✔
6 ✘ ✔ ✘
7 ✘ ✘ ✔
8 ✘ ✘ ✘
• each party can perform according (✔) or not according (✘) to contract
interaction through software
gitek

Scenarios

Scenario	Customer	Developer	Tester
1	✔	✔	✔
2	✔	✔	✘
3	✔	✘	✔
4	✔	✘	✘
5	✘	✔	✔
6	✘	✔	✘
7	✘	✘	✔
8	✘	✘	✘

interaction through software **gitek**

Questions

- What to do when you are running out of budget and you did not cause the overrun ?
- Is there a way out for the tester ?
- What if only the developer is not performing conform contract and you are running out of budget ? Can you approach the developer directly ?

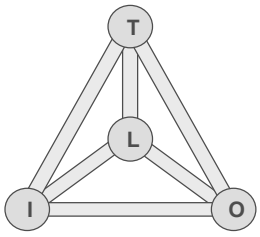
interaction through software **gitek**

Stick to your contract!

Scenario	Customer	Developer	Tester
1	✔	✔	✔
2	✔	✔	✘
3	✔	✘	✔
4	✔	✘	✘
5	✘	✔	✔
6	✘	✔	✘
7	✘	✘	✔
8	✘	✘	✘

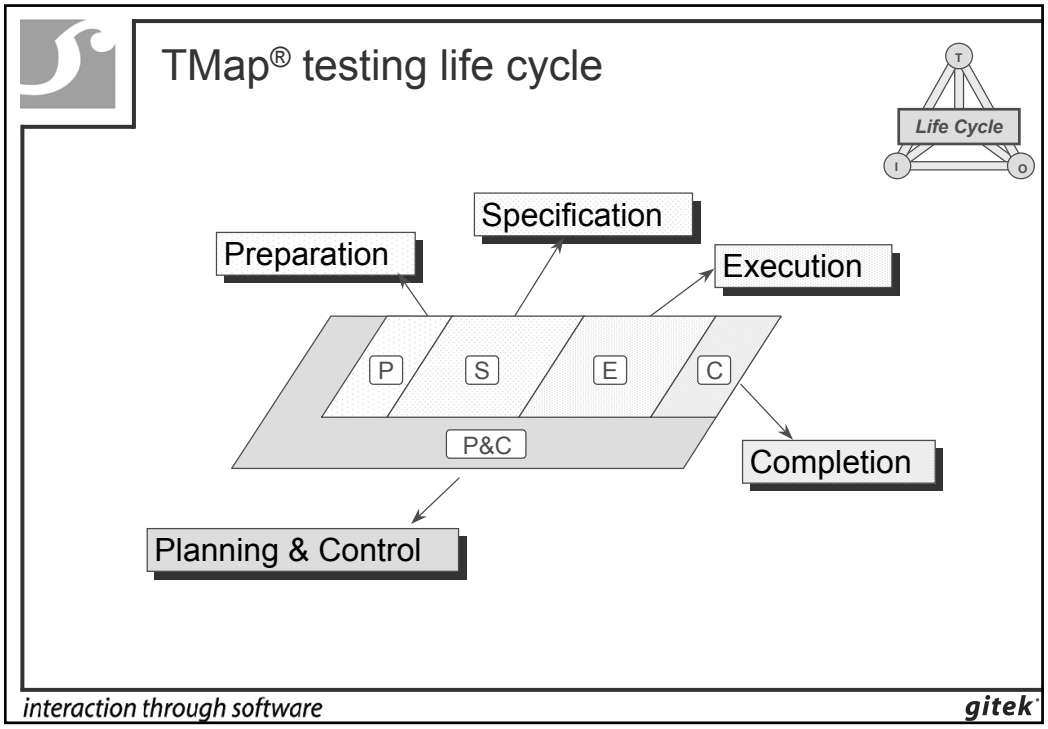
interaction through software **gitek**

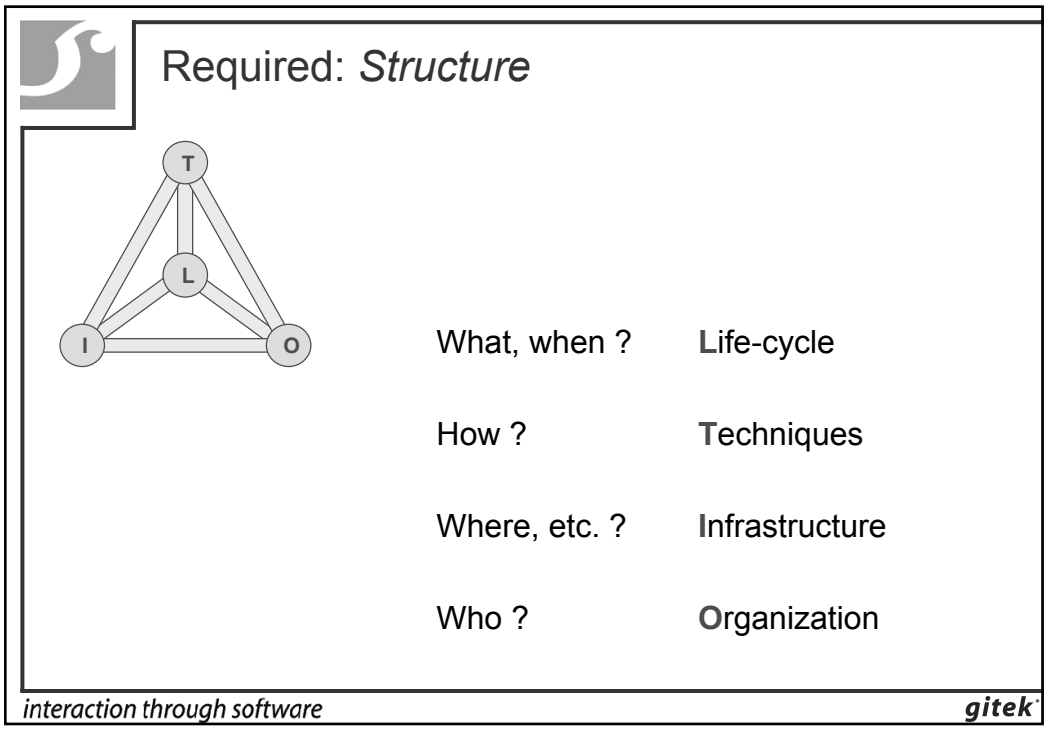
Required: *Structure*



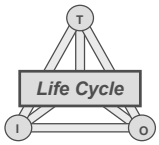
What, when ?	**L**ife-cycle
How ?	**T**echniques
Where, etc. ?	**I**nfrastructure
Who ?	**O**rganization

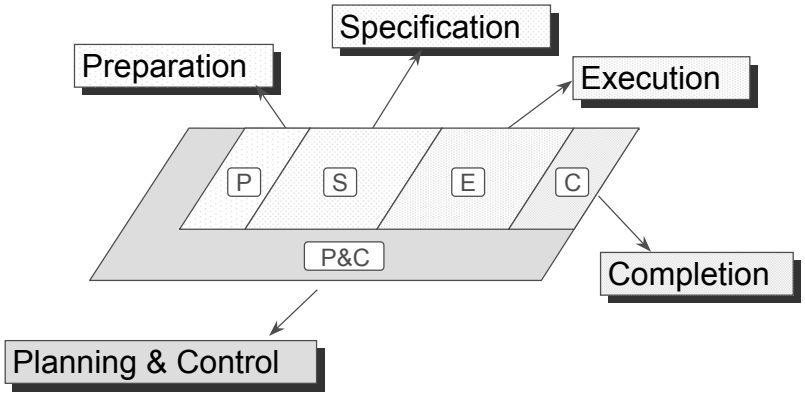
interaction through software **gitek**


TMap® testing life cycle
Life Cycle
T
I
O
Specification
Preparation
Execution
P
S
E
C
P&C
Completion
Planning & Control
interaction through software
gitek


Required: Structure
T
L
I
O
What, when ?
Life-cycle
How ?
Techniques
Where, etc. ?
Infrastructure
Who ?
Organization
interaction through software
gitek

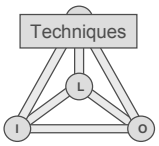
TMap® testing life cycle





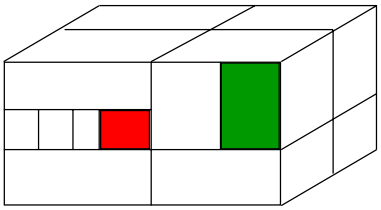
interaction through software gitek

Test Strategy



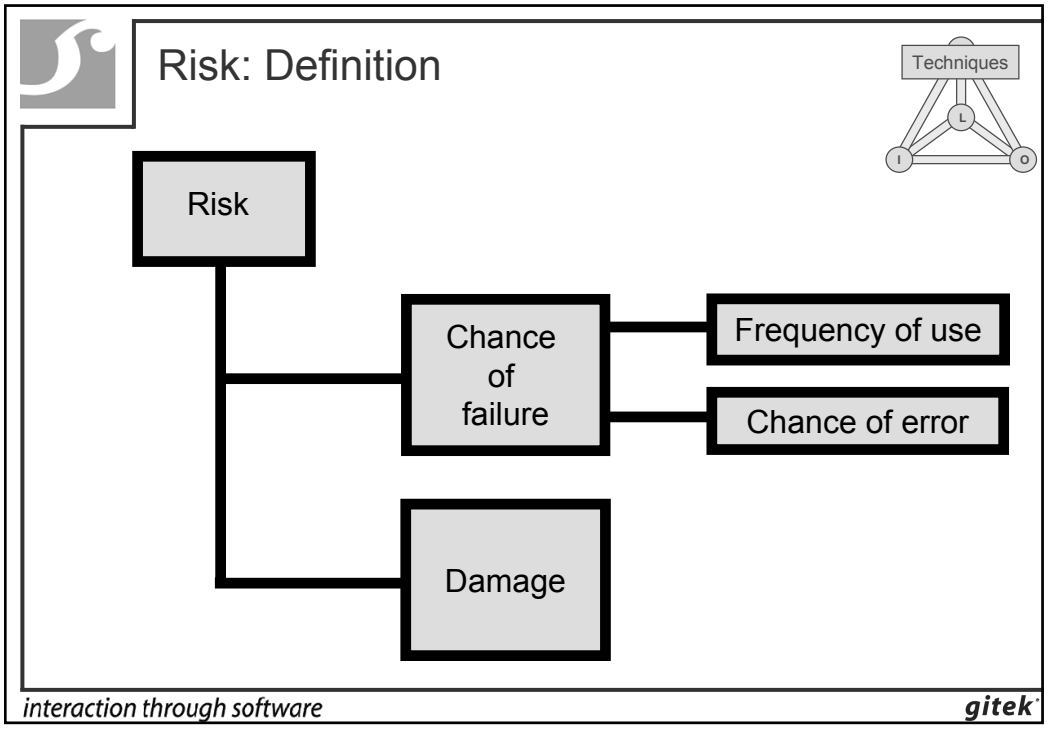
Aim: Find the **most important** defects as **early** as possible at the **lowest price**

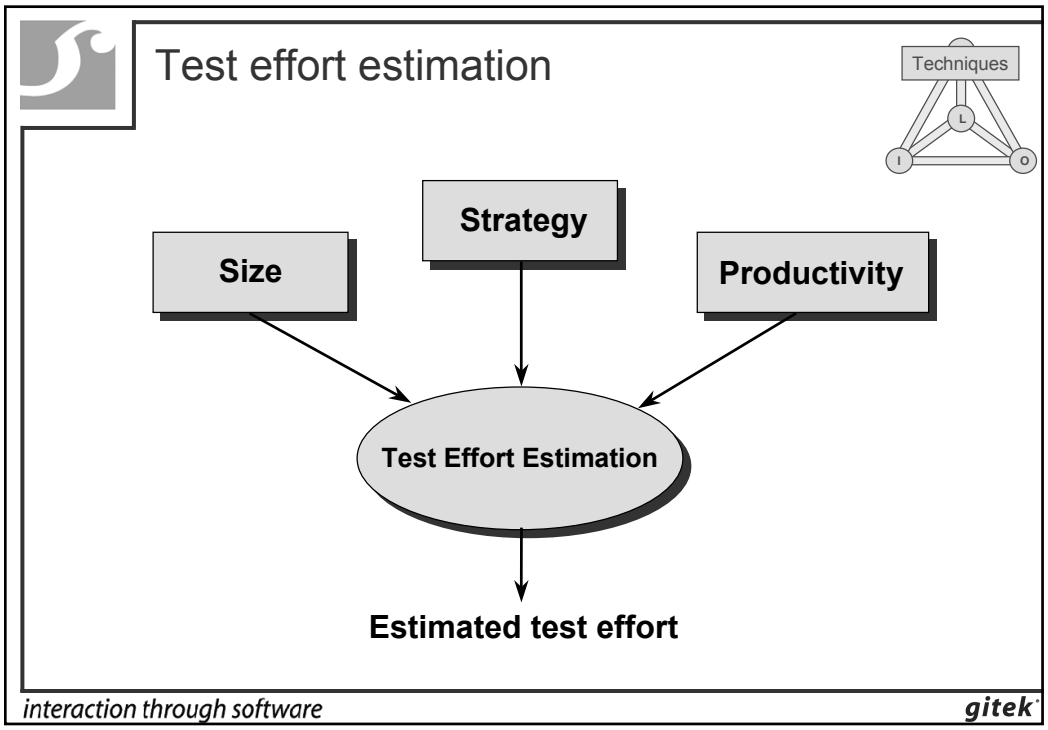
- Risks (business, project, test)
- Quality characteristics
- Available resources
  - people & expertise
  - infrastructure and tools
  - techniques
- Budget and time

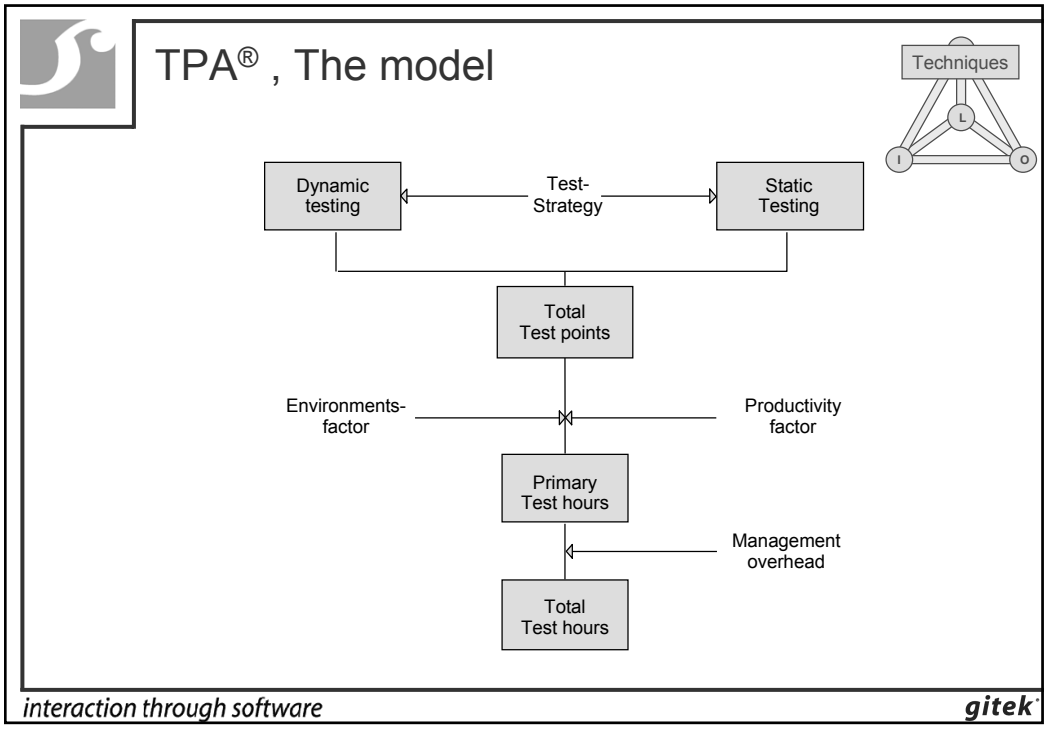


What are the most important defects?

interaction through software gitek


Risk: Definition
Techniques
L
I
O
Risk
Chance of failure
Frequency of use
Chance of error
Damage
interaction through software
gitek


Test effort estimation
Techniques
L
I
O
Strategy
Size
Productivity
Test Effort Estimation
Estimated test effort
interaction through software
gitek



Test plan(ning)

- Consolidation
- Communication
- Claim
- Commitment

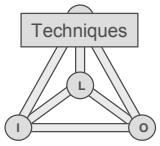
- Arrange:
  - Support
    - technical, functional, business
  - Change control
    - test basis, test object, test infrastructure
  - Defect tracking
    - analysis, decisions
    - response time
    - ……

Content Outline

- Assignment
- Test basis
- Test strategy
- Test effort estimation
- Test infrastructure
- Tools
- Organisation and staffing
- Management and control
- Reporting
- Schedule
- Resource claim

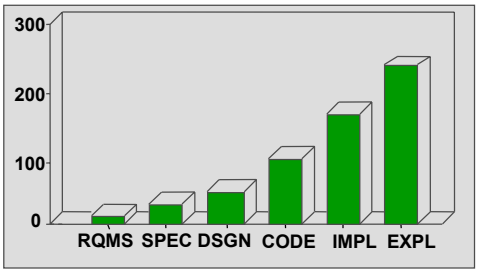
interaction through software gitek

Inspection of the test basis



Goal:

- Determine testability test basis
- Find ”early", potentially expensive defects
- Ensure “unrestricted” progress of test activities



Side effect: *Quality test basis increases*

Intake test infrastructure

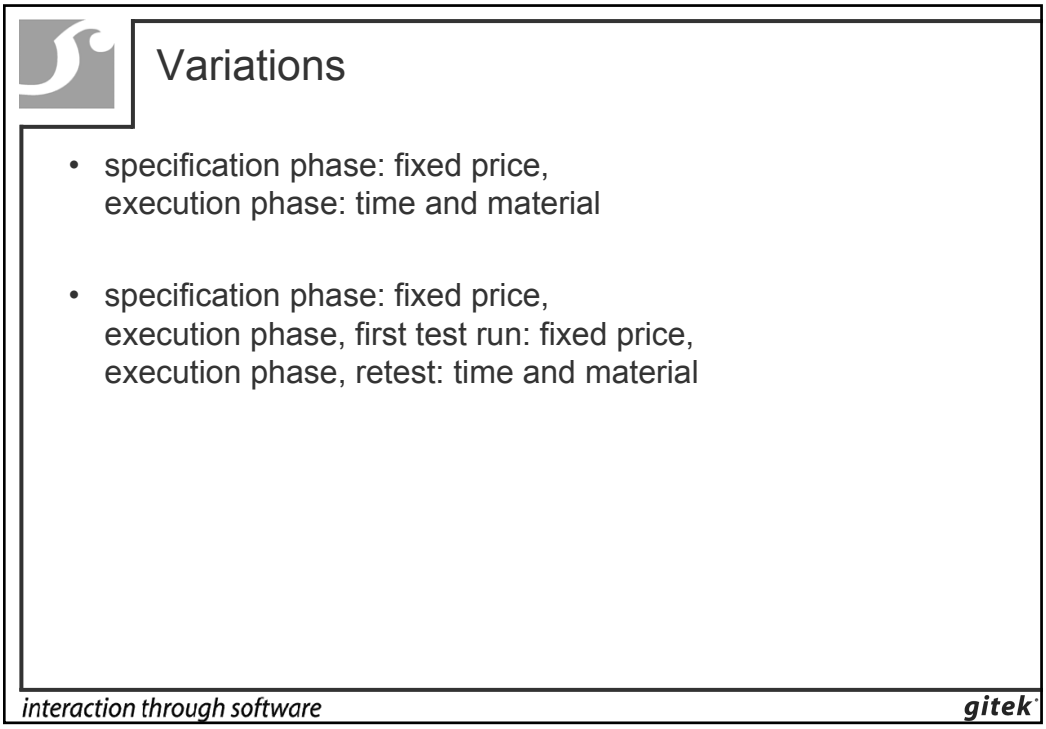
- Installation
  - test files
  - execution procedures
  - JCL, …..
  - organization
  - tools
- Means
  - hardware
  - system software
  - test tools procedures, facilities
  - workplace, logistics

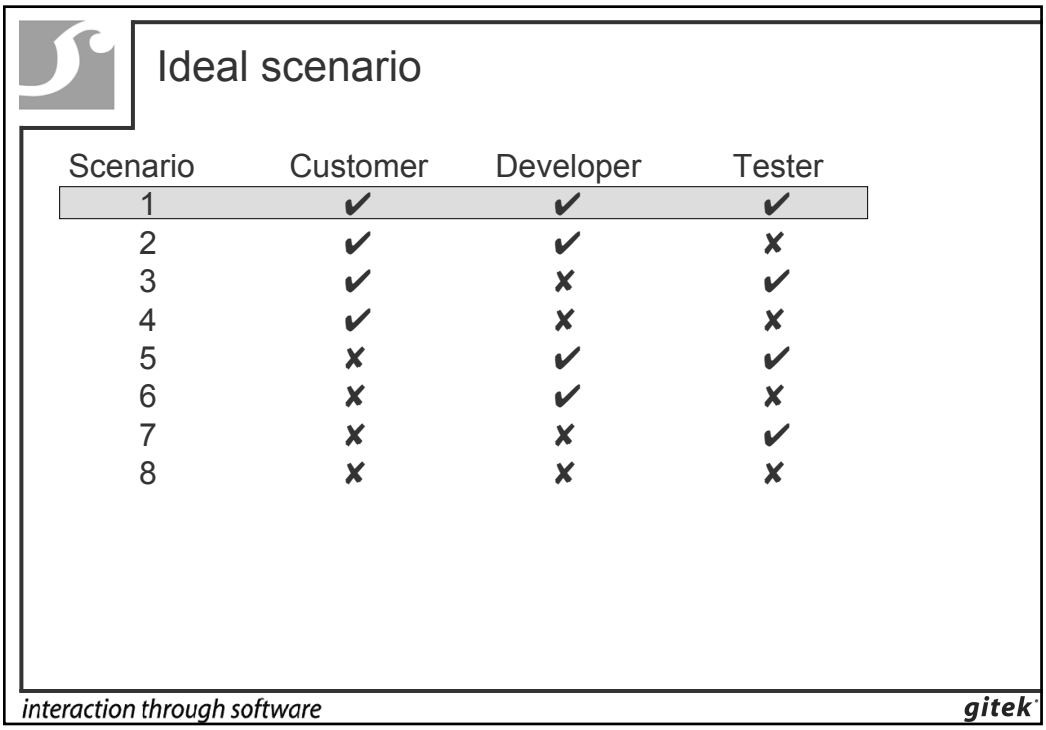
Intake test object

- Use of transfer procedures
- Intake
  - check on completeness
  - installation
- Pre-test / demo
  - testing integral installation (object and TIS)

Test progress reporting

- Weekly Reporting, containing:
  - overview effort spent/to be spend;
  - overview effort lost/extra effort required;
  - overview possible bottlenecks;
- Defect Reports:
  - asap, at least each day;
- Weekly project meeting;
- Ad hoc reporting *escalation*


Variations
• specification phase: fixed price,
execution phase: time and material
• specification phase: fixed price,
execution phase, first test run: fixed price,
execution phase, retest: time and material
interaction through software
gitek


Ideal scenario
Scenario Customer Developer Tester
1 ✔ ✔ ✔
2 ✔ ✔ ✘
3 ✔ ✘ ✔
4 ✔ ✘ ✘
5 ✘ ✔ ✔
6 ✘ ✔ ✘
7 ✘ ✘ ✔
8 ✘ ✘ ✘
interaction through software
gitek

Scenarios to settle for

Scenario	Customer	Developer	Tester
1	✔	✔	✔
2	✔	✔	✘
3	✔	✘	✔
4	✔	✘	✘
5	✘	✔	✔
6	✘	✔	✘
7	✘	✘	✔
8	✘	✘	✘

- each party can perform according (✔) or not according (✘) to contract

interaction through software gitek

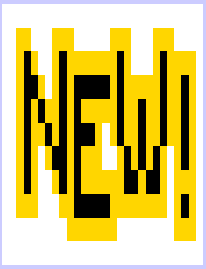
The Art of Managing Fixed Price Test Projects

Ruud Teunissen

Gitek nv - *interaction through software*

http://www.gitek.be

interaction through software gitek



QWE2002 Paper 12M

Mr. Bernd Eberhardt
(Rational Software)

Survival Guide for Applying a Software Development Process

Key Points

- Examine 10 essential elements of applying a new software development process.
- Discuss why the "one-size-fits-all" approach doesn't work.
- How to determine the needs and goals of your team.

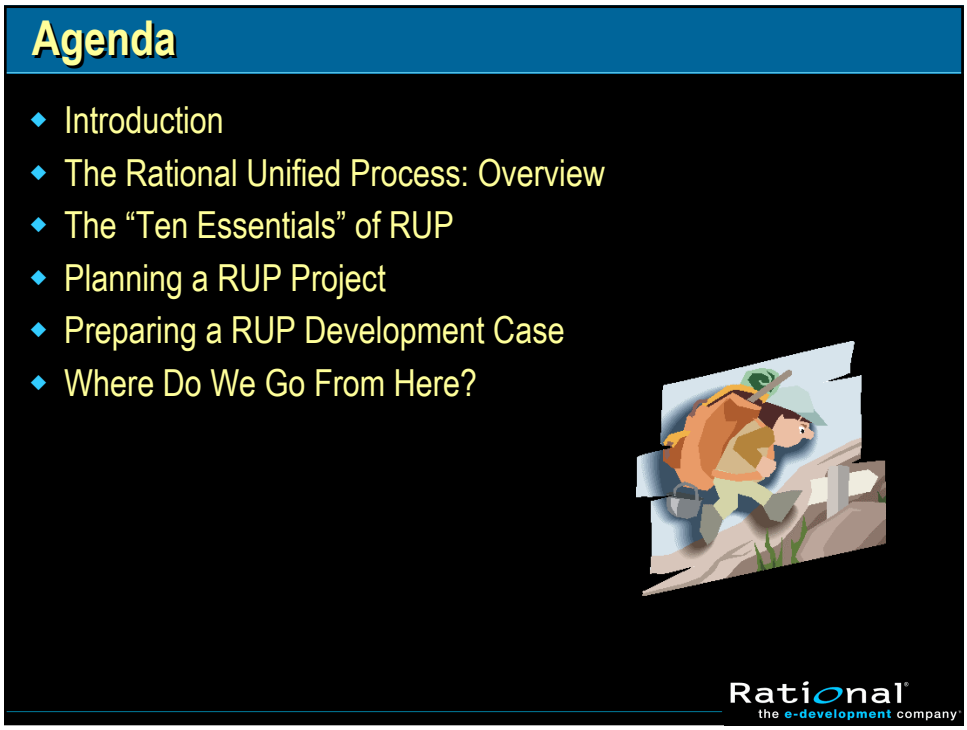
Presentation Abstract

This presentation will examine 10 essential elements of applying a new software development process to an organization and provide guidance on how to customize the process to fit team-specific needs. During this session, we draw upon Rationals 20 years of in-the-trenches experience in which we have harvested and synthesized many critical lessons. We make use of numerous analogies in order to re-enforce the key points of presentation, and to keep the presentation lively and interesting.

About the Author

Mr. Bernd Eberhardt is a Senior Consultant and Master Instructor for Rational Software Corporation. He has five years of professional experience in the area of software Quality Assurance. His expertise is in the area of software QA methodology as well as planning and implementing of QA systems. During his career, Mr. Eberhardt has been involved in numerous projects within the industrial, military and financial sectors. As a seasoned public speaker, he has performed more than 70 trainings (public and on-site) on the subject of testing methodology, test automation and performance test. He is a certified TestStudio Master Instructor/Consultant and a certified PerformanceStudio Master Instructor/Consultant. Mr. Eberhardt has an undergraduate degree in Engineering.

speed
quality
Rational®
the e-development company™
The Ten Essentials of RUP®
The Essence of an Effective Software Development Process
Bernd Eberhardt, Senior Consultant and Master Instructor
Rational Software Corporation


Agenda
Introduction
The Rational Unified Process: Overview
The "Ten Essentials" of RUP
Planning a RUP Project
Preparing a RUP Development Case
Where Do We Go From Here?
Rational®
the e-development company™

Introduction

- What problem are we trying to solve?
  - We need to deliver quality software in internet time (the "e-software paradox")
  - Experience shows us that effective application of a proven software development ***process*** improves product ***quality***
  - But… we can't afford the ***time*** and ***resource*** overhead required by full application of a "complete" software process
- How do we find the delicate balance?
  - "Just enough" process to deliver the ***quality*** we need within the limited ***time*** and ***resources*** available

Rational® the e-development company™

Audience Poll A

- How would you rate your level of knowledge of RUP?
  - I've used RUP on a project
  - I've not used RUP, but am very familiar with RUP
  - I've reviewed the RUP product
  - I've read about RUP (books, articles, etc.)
  - I barely know how to spell RUP (without cheating!)

Rational® the e-development company™

Audience Poll B

- Which best represents the role you play in your current/recent project?
  - Customer
  - Project Manager
  - System Analyst
  - System Architect
  - Programmer/Developer
  - Quality Engineer/Tester
  - All of the above!

Rational® the e-development company™

Audience Poll C

- What is the typical size of a project team in your experience?
  - Less than 5 members
  - 6 – 10 members
  - 11 – 20 members
  - 21 – 50 members
  - More than 50 members

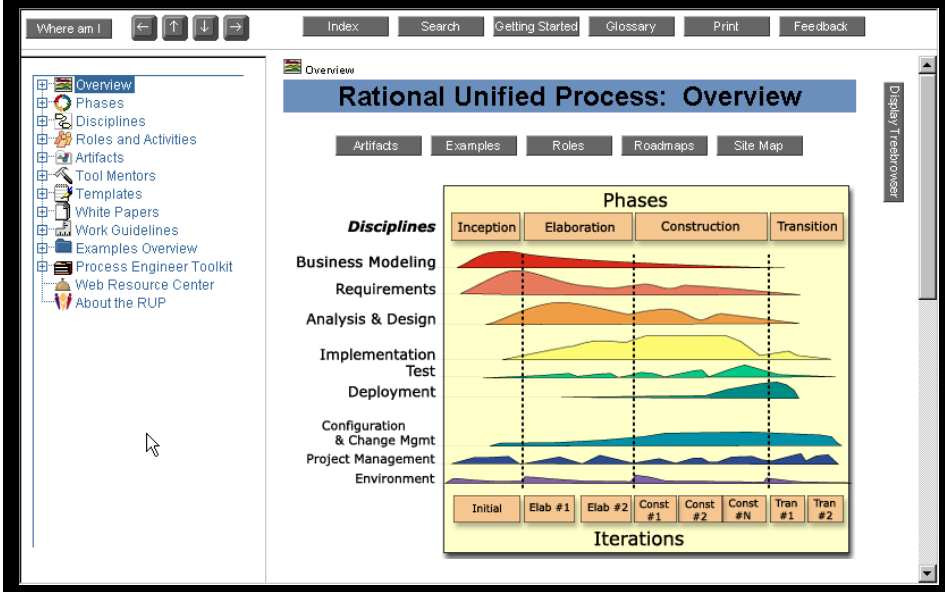
Rational® the e-development company™

Audience Poll D

- Which best describes the domain of your project experience?
  - Financial
  - Telecom
  - E-Business
  - Real-time/Embedded
  - Military/Aerospace
  - Software Development Tools
  - Information Processing

Rational
the e-development company

The Rational Unified Process: Overview



Rational
the e-development company

Key Elements of RUP

Disciplines

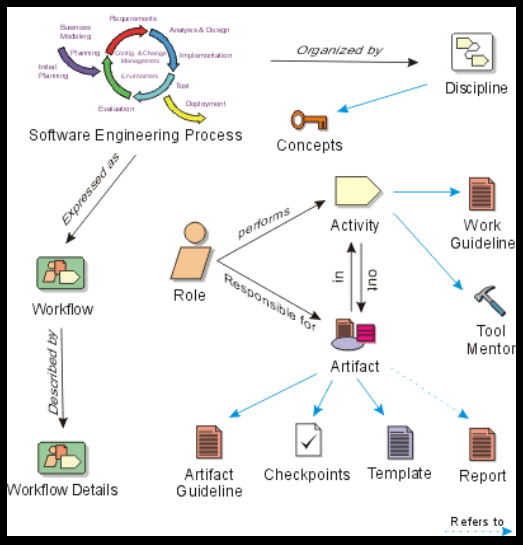
Workflows

Workflow Details

Roles

Activities

Artifacts



Concepts

Guidelines

Tool Mentors

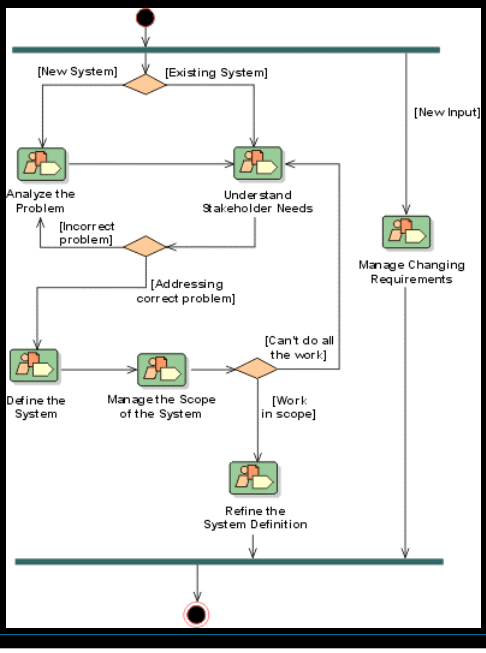
Reports

Checkpoint Lists

Document Templates

Rational the e-development company

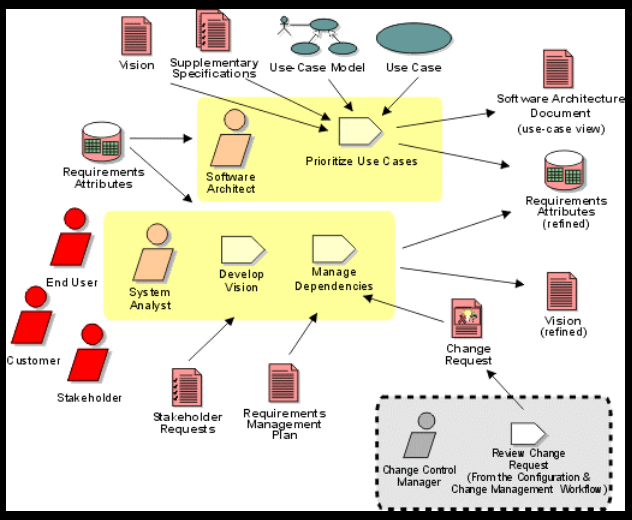
RUP Workflows



Sample RUP Workflow: ***Requirements***

Rational the e-development company

RUP Workflow Details

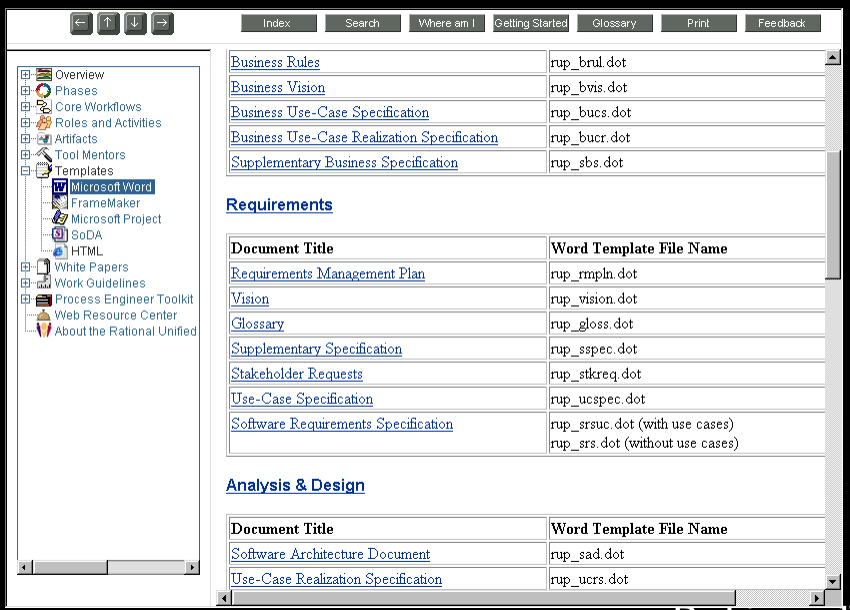


- Activities
- Roles
- Artifacts

Sample Workflow Detail: *Manage the Scope of the System*

Rational the e-development company

RUP Templates



Rational the e-development company

RUP Tool Mentors

- Include guidance on how to apply RUP activities using different software development tools

Rational
the e-development company

Which Do I Need for My Project?

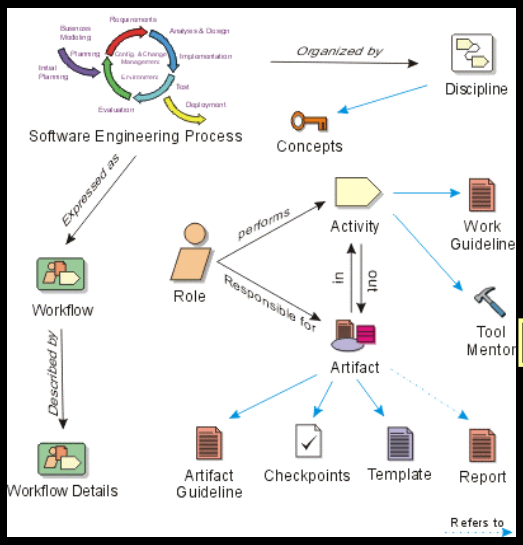
9 Disciplines & Workflows

59 Workflow Details

31 Roles

136 Activities

103 Artifacts



71 Concepts

139 Guidelines

122 Tool Mentors

16 Reports

44 Checkpoint Lists

43 Document Templates

Rational
the e-development company

No Simple Answer: Every Project Is Different!

Rational® the e-development company™

Different Domains

Rational® the e-development company™

Different Team Member Skills

Rational® the e-development company™

Different Safety and Security Considerations

Rational® the e-development company™

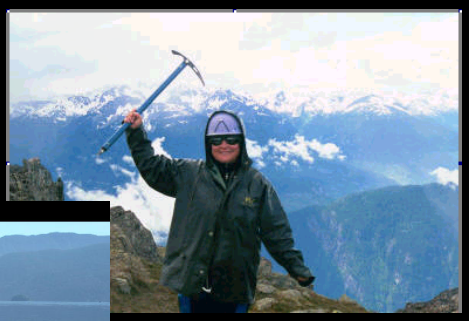
Each Has Different Risks

Rational
the e-development company

Each Needs Different Equipment

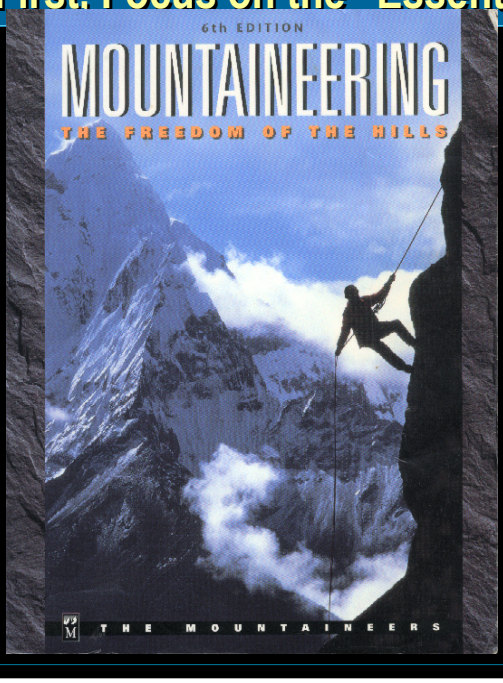
Rational
the e-development company

And Has Different Objectives!



Rational
the e-development company

First, Focus on the "Essentials"



ESSENTIAL EQUIPMENT

There is a small selection of critical items that deserve a place in almost every pack in addition to normal supplies of clothes, water, and food. You won't use every one of these items on every trip, but they can be lifesavers in an emergency, insurance against the unexpected.

Exactly how much "insurance" you should carry is a matter of debate. Some respected mindsetters argue that weighing down your pack with insurance items causes you to climb slower, making it more likely you'll get caught by a storm or nightfall and be forced to bivouac. "Don't carry bivy gear unless you plan to bivy," they argue.

The majority of climbers, however, take along carefully selected safety items to survive the unexpected. They sacrifice some speed but argue they will always be around tomorrow to attempt again what they failed to climb today.

The Ten Essentials

The special items most climbers believe should always be with you have become known as the Ten Essentials. They are listed here and then described in more detail. A memory aid is provided as well.

1. Map	Mom's
2. Compass	café
3. Sunglasses and sunscreen	serves
4. Extra food	everyone
5. Extra clothing	extra
6. Headlamp/flashlight	helpings of
7. First-aid supplies	French
8. Fire starter	fries,
9. Matches	mustard, and
10. Knife	ketchup

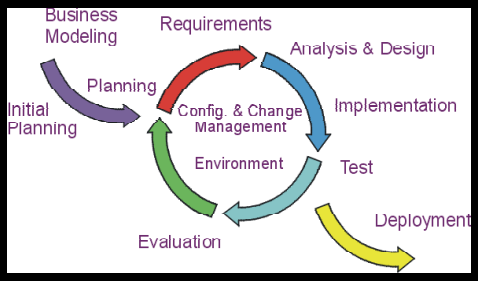
1. Map

Always carry a detailed topographic map of the area you are visiting, in a protective case or plastic

Rational
the e-development company

The "Ten Essentials" of RUP

1. Develop a ***Vision***
2. Manage to the ***Plan***
3. Identify and Mitigate ***Risks***
4. Assign and Track ***Issues***
5. Examine the ***Business Case***
6. Design a Component ***Architecture***
7. Incrementally Build and Test the ***Product***
8. Regularly ***Evaluate*** and Assess Results
9. Manage and Control ***Change Requests***
10. Provide ***User Support***



Rational the e-development company

1. Develop a *Vision*

- The "essence" of the RUP *Requirements* workflow
  - Define the Problem
  - Understand Stakeholder Needs
  - Define the System
  - Refine the System Definition
  - Manage Changing Requirements
- Key to developing a product that meets your stakeholders' "*real* needs"

Rational the e-development company

The *Vision* Helps to Keep the Focus on the Goal

- Answers the following questions (may be broken out to separate, more detailed artifacts, as needed)
  - Who are the *stakeholders*? Who are the *users*? What are their *needs*?
  - What *problem* are we trying to solve? (*Problem Statement*)
  - What are the definition of *key terms*? (*Glossary*)
  - What are the product *features*?
  - What are the *functional* requirements? (*Use Cases*)
  - What are the *non-functional* requirements?
  - What are the *design constraints*?

Rational the e-development company

2. Manage to the *Plan*

- The "essence" of the *Project Management* workflow
  - Conceiving a new project
  - Evaluating scope and risk
  - Monitoring and controlling the project
  - Planning for and evaluating each iteration and phase

- The *Software Development Plan*
  - Used to plan the project schedule and resource needs
  - Tracks progress against the schedule

"The product is only as good as the plan for the product"

Rational the e-development company

"The plan is nothing–the planning is everything"

- The *Software Development Plan* addresses:
  - Project Organization
  - Schedule (Project Plan, Iteration Plan, Resources, Tools)
  - Requirements Management Plan
  - Configuration Management Plan
  - Problem Resolution Plan
  - QA Plan
  - Test Plan
  - Test Cases
  - Evaluation Plan
  - Product Acceptance Plan

"For a small project, these do not have to be maintained as separate artifacts. They could be addressed as merely a section, or even a paragraph, in the SDP."

Rational the e-development company

3. Identify and Mitigate *Risks*

- The *Risk List* is intended to capture the perceived risks to the success of the project
  - Events which could lead to a significant negative outcome
  - A plan for mitigating each risk
  - Focal point for planning project activities
  - Basis around which iterations are organized

- It is essential to identify and attack the *highest risk* items early in the project

Rational the e-development company

4. Assign and Track *Issues*

- Regular *Status Assessments* provide a mechanism for addressing, communicating, and resolving management issues, technical issues, and project risks
  - Each is assigned a *due date*, with a *responsible person* who is accountable for the resolution
  - Regularly *tracked* and updated
  - Provide the heartbeat for management attention

Rational the e-development company

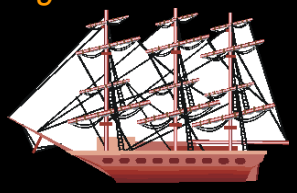
5. Examine the *Business Case*

- Provides the necessary information, from a business standpoint, to determine whether or not this project is worth the investment
  - An economic plan for realizing the project Vision
  - Used to make an accurate assessment of the return on investment (ROI)
  - Justification for the project
  - Economic constraints
  - Used to determine whether the project should move ahead

Rational the e-development company

6. Design a Component *Architecture*

- The "essence" of the *Analysis and Design* workflow:
  - Defining a candidate architecture
  - Refining the architecture
  - Analyzing behavior
  - Designing components of the system

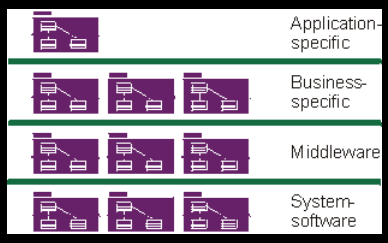


- *Software Architecture*: The organization or structure of the system's significant components interacting through interfaces, with components composed of successively smaller components and interfaces
  - What are the main pieces? How do they fit together?
  - Do we have a framework on which the rest of the software can be built?

Rational® the e-development company™

The *Architecture* Is Presented in Multiple Views

- Captured in the *Software Architecture Document*
  - Communication medium between the software architect and other project team members regarding architecturally significant decisions
- Presents multiple *architectural views*
  - Logical, Implementation, Process, Deployment, Use-Case
  - Each addresses some specific set of stakeholders' concerns: for end users, designers, managers, system engineers, maintainers, etc.



Rational® the e-development company™

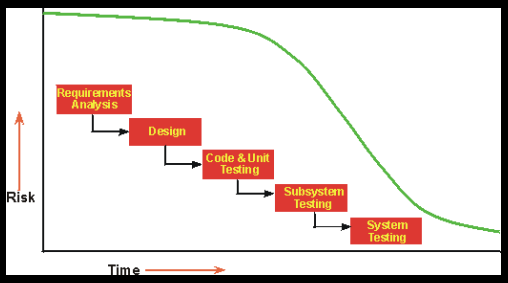
7. Incrementally Build and Test the *Product*

- The "essence" of the RUP *Implementation* and *Test* workflows
  - Incrementally build and test the *components* of the system
  - Starts with an architectural prototype at the end of the Elaboration Phase
  - Iterative approach of building, testing, and evaluating executable versions of the product
  - Identify problems and resolve risks and issues as early as possible

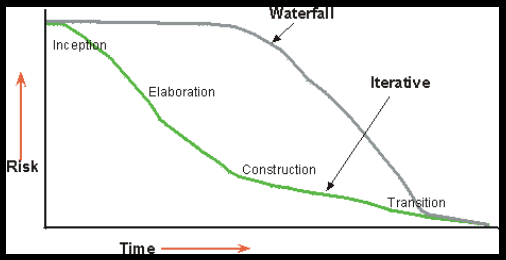
Rational the e-development company

Why Develop Iteratively?

- Initial design will likely be flawed with respect to its key *requirements*
- Late discovery of *design defects* results in costly over-runs and, in some cases, even project cancellation
- The earlier in the lifecycle you can verify that you have dealt with a risk, the more accurate you can make your plans



Risk Profile: Waterfall vs. Iterative Development



Rational the e-development company

8. Regularly *Evaluate* and Assess Results

- The *Iteration Assessment* captures the results of an iteration
  - Degree to which the *evaluation criteria* were met
  - *Lessons learned* and *process changes* to be implemented
  - May range from a simple record of demonstration and outcomes to a complete formal test review record
- Focus on *process* problems, as well as *product* problems

"The sooner you fall behind, the more time you will have to catch up"

Rational the e-development company

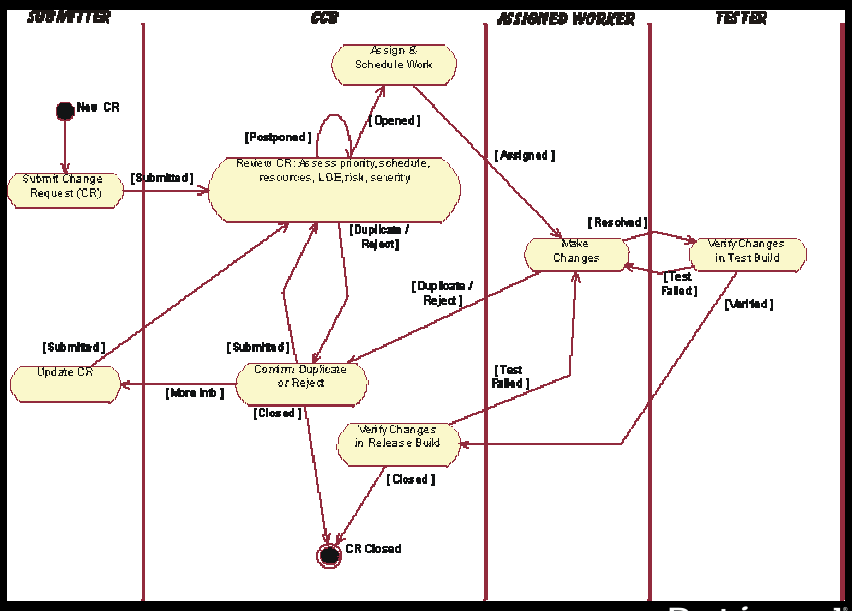
9. Manage and Control *Change Requests*

- The "essence" of the RUP *Configuration and Change Management* workflow
  - Manage and control the *scope* of the project, as *changes* occur throughout the project lifecycle
  - Use a *single point* of change control (CCB): could be a single person
  - Maintain the goal of considering all stakeholders' needs

"As soon as the first prototype is put before the users – and often even before that – changes will be requested."
(One of those certainties of life!)

Rational the e-development company

Sample *Change Request Management* Process



Rational the e-development company

10. Provide *User Support*

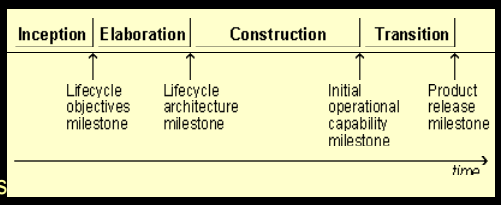
- The "essence" of the RUP *Deployment* workflow
- At a minimum:
  - Installation Guide and Release Notes
  - User's Guide (online help)
- Depending on the complexity of the product:
  - Product packaging
  - Bills of materials
  - Training materials

"The Goal is to produce a usable product!"

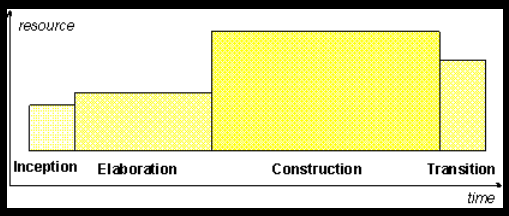
Rational the e-development company

Planning a RUP Project

- The Phases and Milestones of a RUP Project



- Planning Phases
  - Effort
  - Schedule



Inception	Elaboration	Construction	Transition
~5%	20%	65%	10%
10%	30%	50%	10%

Rational the e-development company

RUP Inception Phase

- Overriding goal
  - Achieve concurrence among all stakeholders on the *Lifecycle Objectives* for the project
- Essential Objectives
  - Formulate the *scope* of the project
  - Plan and prepare a *Business Case*
  - Synthesize a *candidate Architecture*
  - Prepare the *environment* for the project
  - Determine if the project should continue

Rational the e-development company

RUP Elaboration Phase

- Overriding goal
  - *Baseline the Architecture* of the system
- Essential Objectives
  - Define, validate and baseline the *Architecture*
  - Refine the *Vision*
  - Create and baseline detailed *Iteration Plans* for the construction phase
  - Refine the *Development Case* and put in place the *Development Environment*
  - Refine the *Architecture* and select components

Rational® the e-development company™

RUP Construction Phase

- Overriding goal
  - Clarify the remaining *Requirements* and complete the *development* of the system based upon the baselined *Architecture*
- Essential objectives
  - *Resource* management, control and *process* optimization
  - Complete component *development* and *testing* against the defined evaluation criteria
  - *Assessment* of product releases against acceptance criteria for the vision

Rational® the e-development company™

RUP Transition Phase

- Overriding goal
  - Ensure that the software is available for its *end users*
- Essential objectives
  - *Deploy* the product
  - Finalize *end-user support* material
  - *Test* the deliverable product at the development site
  - Create a product *release*
  - Get *user feedback*
  - *Fine-tune* the product based on feedback
  - Make the *product available* to end users

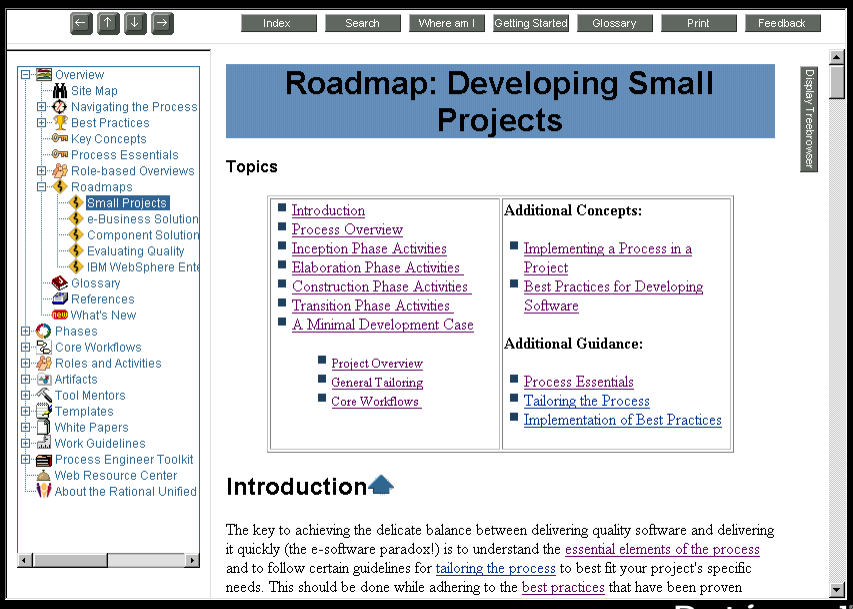
Rational the e-development company

Preparing a RUP Development Case

- Determine "essential" activities and artifacts to use for the project
  - Level of formality required
  - Tailoring required to fit project needs
- Map project team members to RUP roles
- RUP provides "roadmaps" to help
  - Small Projects
  - E-Business Solutions
  - Component Solutions
  - Evaluating Quality throughout the lifecycle

Rational the e-development company

Applying the "Essentials" to a Small Project



Where Do We Go From Here?

- For Very Small Projects
  - The "Ten Essentials" can be used in a light-weight fashion with minimal tool support

Rational
the e-development company

For Growing Projects

- Need more tool and process support
- Start with the "Ten Essentials"
- Add other RUP elements as needed

Rational the e-development company

For Mature Project Teams

- The "Ten Essentials" provide a quick method to assess the current process and identify areas where improvement is most beneficial
- Look for the areas of "pain" to determine where to implement more of the process and tools

Rational the e-development company

How Can You Use The 10 Essentials?

- Where is the “pain” in your projects?
- Evaluate the risk of ***not*** using each essential
  - Which can you afford to ignore?

Rational® the e-development company™

Feedback?

- Bernd.eberhardt@de.rational.com

Rational® the e-development company™

The Ten Essentials of RUP
The Essence of an Effective Development Process

Leslee Probasco

Rational Software White paper

Abstract: To effectively apply a software development process such as the Rational Unified Process[1] (affectionately known as "RUP"), it is important to first understand its key objectives, why each is important, and how they work together to help your development team produce a quality product that meets your stakeholders' "real" needs.

Rational®
the e-development company™

Table of Contents

A Plea for Help **1**

Overwhelmed with the Details **1**

Focus on the Essentials **1**

Start with a "Short List" **2**

What Should be on Your List? "Well, it Depends…" **2**

Build a Framework First **2**

The Ten Essentials of RUP **3**
- **Vision – Develop a Vision** **4**
- **Plan – Manage to the Plan** **4**
- **Risks – Identify and Mitigate Risks** **5**
- **Issues – Assign and Track Issues** **5**
- **Business Case – Examine the Business Case** **5**
- **Architecture – Design a Component Architecture** **5**
- **Product – Incrementally Build and Test the Product** **6**
- **Evaluation – Regularly Assess Results** **6**
- **Change Requests – Manage and Control Changes** **6**
- **User Support – Provide Assistance to the User** **6**

What about Requirements? **6**

What about Test? **7**

Summary: Applying the Ten Essentials **7**
- **For Very Small Projects** **7**
- **For Growing Projects** **7**
- **For Mature Project Teams** **8**

A Plea for Help

The other evening, my neighbor Randy came over to ask for some help: He was preparing for an weekend camping and hiking trip with his church group and was trying to determine what gear[1] to pack for the outing. He already had some clothing and equipment, and was undoubtedly going to have to buy some more. Could he please borrow my gear list?

My gear list? Yes. It appears that he was over visiting one day when I was preparing for a backpacking and climbing trip. He knows that I am quite experienced in leading and participating in wilderness trips and was very impressed that I was able to quickly and efficiently determine the items to fit into my limited packing space by referring to a list I have of much of my inventoried equipment and clothing. Could he please borrow that list?

Well, sure, he was welcome to borrow my gear list – but I was afraid it wouldn't be much help. Why not? Didn't I think that everything he needed would be on that list?

Actually, I was quite sure that everything he needed would at least be *represented* by something on my list, but more than likely, the exact items he needed for this particular trip might not be on the list at all. For example, not only is his shoe size *much* bigger than mine, but he no doubt prefers different food (and quantities) than I might choose to take along.

Overwhelmed with the Details

The fact is that I have literally *hundreds* of items on my outdoor gear list, covering many different outing types from backpacking and climbing, to skiing, snow-shoeing, ice-climbing and kayaking – and covering trip lengths from simple day trips to multi-day expeditions. And I have different items I would take along if I were leading a trip for an organization (such as trip sign-up sheet and waiver forms), than if I were just heading out into the wilderness with some of my buddies. My main concern was that he would not be able to wade through the multitude of items on the list and figure out what *he* really needed for his relatively simple outing.

Randy threw up his hands in exasperation! He'd never be ready in time to be properly prepared for his hike. How would he know what to pack? He had started packing last week – he had done extensive research on the Internet, bought fancy new boots, jackets and clothing – his pack was already full!

And he hadn't even packed any food yet! Oh, yeah, and didn't he need to take water, too? And what if something went wrong? What if they got lost? Or someone got hurt?

Focus on the Essentials

As I sorted through the items Randy had already put into his pack, I could easily see that he did not have a balanced view of the essential items needed on a wilderness outing.

Do you have the "ten essentials"? I asked. Ten essentials? What are those?

Here, Randy, I have just the list you need. I pulled out a blank sheet of paper and wrote at the top: "The Ten Essentials". On it, I wrote 10 items:

1. Map
2. Compass
3. Sunglasses & sunscreen
4. Extra clothing
5. Extra food & water

[1] "Gear", a slang term used by mountaineers and other outdoor enthusiasts to denote any manner of equipment, tools, clothing, footwear or other artifacts used in their various and sundry sports. If you think RUP has a lot of artifacts, see: www.mgear.com.

6. Headlamp
7. First-aid kit
8. Fire-starter
9. Matches
10. Knife[2]

Start with a "Short List"

That's it? Yup! That's it! If you start with these 10 elements, making sure you've got each area covered, then the other things will fall into place as needed. Of course, each of these "essentials" will scale up or down, depending on the trip, but starting with a "short list" and expanding it, as needed, is much easier than starting from a long list and trying to decide what *not* to take.

As you gain experience, you'll evolve your own "long list" which will make sense to you as time goes on. At that point, it may be helpful for you to look at my list and others' to see how you might properly expand your own. But no two person's lists will ever look exactly the same (unless they've cheated!).

I memorized this same list many years ago when I first started mountaineering and I still refer to it no matter what type of trip I am preparing for or how long I intend to be gone. If you ask any experienced mountaineer, "Do you have the 'ten essentials'?", most of them will know exactly what it is you're asking about. Actually, different groups or individuals may have a different list of "ten essentials", but in essence, they are the same. And for each trip, the actual items I pack may be entirely different, but I can still say that I have the "ten essentials".

What Should be on Your List? "Well, it Depends…"

So, how does this relate to RUP? Often, as I help project teams sort through the many elements in RUP (at last count, I found 4 phases, 9 core workflows, 31 workers, 103 artifacts, 136 activities, plus more guidelines, checklists and tool mentors than I would care to count!), I hear questions such as: "How do I sort through all of these items and determine which I need for my project?" "Do I need this one?" "Isn't RUP is only for big projects?"

Most often the answer is (one of our favorites): "Well, it depends…"

What I really want, I've decided, is a list of "Ten Essentials of RUP", that I can give to folks searching for process guidance – like I did for my friend Randy. This list could serve as a reasonable starting point for determining the right elements to include for any project and could apply if they were going for a short day hike (a very small project); an overnight backpack, ski or kayak trip (a medium-sized project in different domains); or a major Everest-type expedition (a very large, critical project). The idea is to focus on what I call "the essence" of RUP – or really of any effective software process. This also ties back to our concepts of "best practices"[3] in software development.

Build a Framework First

A common problem I see in many projects, is that they will often focus heavily in one particular area to the extent that they get bogged down with the details of that particular area before making sure that they have a good idea of the "key" elements involved in the whole process lifecycle of producing a quality product.

Then, when the project falls behind they say, "See, I told you that requirements management would just slow us down!". You can replace "requirements management" in the previous statement with "use cases", "collecting

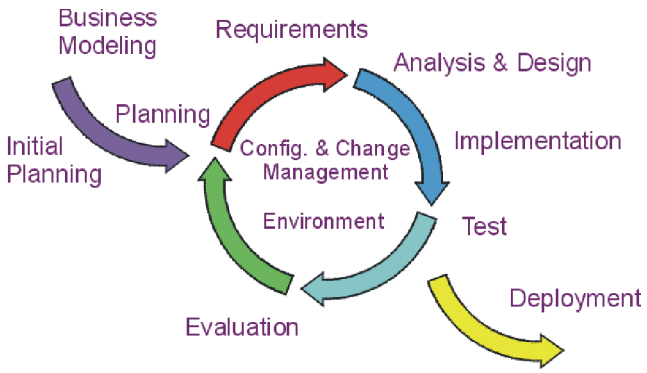
[2] For a complete analysis of this list of "Ten Essentials", I refer you to the excellent book, "Mountaineering – The Freedom of the Hills", 6th Edition, The Mountaineers, Seattle, WA, 1997. pp. 35-41.

[3] The "best practices" that Rational has been touting over the last few years include: (1) Develop Iteratively, (2) Manage Requirements, (3) Use Component Architectures, (4) Model Visually, (5) Verify Quality, and (6) Control Changes.

project metrics", "using configuration management", "using a defect tracking tool", "having status meetings" or any of a number of phrases that I'm sure you're familiar with.

I believe it is much more effective to take a more systematic and holistic approach, making sure that the key elements of a process are in place (an architecture, so to speak) before determining to focus on any one particular problem area.

Once this framework (or architecture) for a quality software process is in place, then I believe a project can effectively focus on a particular area which is identified as being a major contributor to their problems (and usually, I'll admit, requirements management is often right there at the top of the list[4]). Keep in mind, though, that the RUP approach is to base this selection on identifying and prioritizing risks for the project and determining early mitigation strategies for those identified risks.



The Ten Essentials of RUP

So, what are the "Ten Essentials of RUP"? The following list describes what I believe to be the minimal set of items a project will have in place if they are truly following the "essence" of the Rational Unified Process:

1. Vision
2. Plan
3. Risks
4. Issues
5. Business Case
6. Architecture
7. Product
8. Evaluation
9. Change Requests
10. User Support

[4]Since Requirements Management is one of the initial "Key Process Areas" of the Capability Maturity Model (CMM), this is not at all an unlikely scenario for those determining to improve their software development process.

Let's look at each of these items individually, see where they fit in RUP and try to understand why each made my "short list" of essential items.

Vision – Develop a Vision

Having a clear Vision is key to developing a product that meets your stakeholders' *real* needs".[5]

The Vision captures the "essence" of the *Requirements workflow* in RUP: analyzing the problem, understanding stakeholder needs, defining the system, and managing the requirements as they change.

The Vision provides a high-level, sometimes contractual, basis for more detailed technical requirements. The Vision captures very high-level requirements and design constraints, to give the reader an understanding of the system to be developed. It provides input to the project-approval process, and is therefore intimately related to the Business Case. It communicates the fundamental "why's and what's" related to the project and is a gauge against which all future decisions should be validated.

The contents of the Vision should answer the following questions, which might be broken out to separate, more detailed, artifacts, as needed:

What are the key terms? (Glossary)
What problem are we trying to solve? (Problem Statement)
Who are the stakeholders? Who are the users? What are their needs?
What are the product features?
What are the functional requirements? (Use Cases)
What are the non-functional requirements?
What are the design constraints?

Plan – Manage to the Plan

"The product is only as good as the plan for the product."[6]

In RUP, the Software Development Plan (SDP) gathers all information required to manage the project. It may enclose a number of separate artifacts developed during the Inception phase and is maintained throughout the project.

The SDP is used to plan the project schedule and resource needs, and to track progress against the schedule. It addresses such areas as: Project Organization, Schedule (Project Plan, Iteration Plan, Resources, Tools), Requirements Management Plan, Configuration Management Plan, Problem Resolution Plan, QA Plan, Test Plan, Test Cases, Evaluation Plan, and Product Acceptance Plan.

In a simple project, these may include only one or two sentences each. For example a CM Plan may simply state: "At the end of each day, the contents of the project directory structure will be zipped, copied onto a dated, labeled zip disk, marked with a version number and placed in the central filing cabinet."

The format of the Software Development Plan itself is not as important as the activity and thought that go into producing it. So, I don't even really care what it looks like – or what tools you use to build it. As Dwight D. Eisenhower said, "The plan is nothing; the planning is everything."

[5] "The goal is to develop a quality product, on time and on budget, that meets the stakeholders' real needs." – *Managing Software Requirements*, Dean Leffingwell & Don Widrig, Addison-Wesley Longman, January, 2000.
[6] Johnson Space Center Shuttle Software Group, "They Write the Right Stuff", Charles Fishman, *Fastcompany*, Issue 6, p. 95, December, 1996.

Essential items numbers 2, 3, 4, 5 and 8 capture the "essence" of the *Project Management workflow* in RUP: conceiving a new project; evaluating scope and risk; monitoring and controlling the project; planning for and evaluating each iteration and phase.

Risks – Identify and Mitigate Risks

An essential precept of RUP is to identify and attack the highest risk items early in the project. The risk list is intended to capture the perceived risks to the success of the project. It identifies, in decreasing order of priority, the events which could lead to a significant negative outcome.

Along with each risk, should be a plan for mitigating that risk. This serves as a focal point for planning project activities, and is the basis around which iterations are organized.

Issues – Assign and Track Issues

Continuous open communication with objective data derived directly from ongoing activities, and the evolving product configurations are important in any project. In RUP, this is done through regular status assessments, which provide the mechanism for addressing, communicating, and resolving management issues, technical issues, and project risks. In addition to identifying the issues, each should be assigned a due date, with a responsible person who is accountable for the resolution. This should be regularly tracked and updated as necessary.

These project snapshots provide the heartbeat for management attention. While the period may vary, the forcing function needs to capture the project history and resolve to remove any roadblocks or bottlenecks that restrict progress.

Business Case – Examine the Business Case

The Business Case provides the necessary information, from a business standpoint, to determine whether or not this project is worth investing in.

The main purpose of the Business Case is to develop an economic plan for realizing the project Vision. Once developed, the Business Case is used to make an accurate assessment of the return on investment (ROI) provided by the project. It provides the justification for the project and establishes its economic constraints. It provides information to the economic decision makers on the project's worth, and is used to determine whether the project should move ahead.

The description should not delve deeply into the specifics of the problem, but rather it should create a compelling argument why the product is needed. It must be brief, however, so that it is easy enough for all project team members to understand and remember. At critical milestones, the Business Case is re-examined to see if estimates of expected return and cost are still accurate, and whether the project should be continued

Architecture – Design a Component Architecture

In the Rational Unified Process, the architecture of a software system (at a given point) is the organization or structure of the system's significant components interacting through interfaces, with components composed of successively smaller components and interfaces. What are the main pieces? And how do they fit together?

RUP provides a methodical, systematic way to design, develop and validate a software architecture. This is the "essence" of the *Analysis and Design workflow* in RUP: defining a candidate architecture, refining the architecture, analyzing behavior, and designing components of the system.

To speak and reason about software architecture, you must first define an architectural representation, a way of describing important aspects of an architecture. In the RUP, this description is captured in the Software Architecture Document, which presents the architecture in multiple views.

Each architectural view addresses some specific set of concerns, specific to stakeholders in the development process: end users, designers, managers, system engineers, maintainers, and so on. This serves as a communication

medium between the architect and other project team members regarding architecturally significant decisions which have been made on the project.

Product – Incrementally Build and Test the Product

The "essence" of the *Implementation and Test workflows* in RUP is to incrementally code, build and test the components of the system, with executable releases at the end of each iteration after inception.

At the end of the elaboration phase, an architectural prototype is available for evaluation; this might also include a user-interface prototype, if necessary. Throughout each iteration of the construction phase, components are integrated into executable, tested builds that evolve into the final product.

Key to this essential element is an integrated set of test activities that accompany the building of the product – as well as ongoing Configuration Management and review activities.

Evaluation – Regularly Assess Results

The Iteration Assessment captures the results of an iteration, the degree to which the evaluation criteria were met, the lessons learned and process changes to be implemented.

The Iteration Assessment is an essential artifact of the iterative approach. Depending on the scope and risk of the project and the nature of the iteration, it may range from a simple record of demonstration and outcomes to a complete formal test review record.

The key here is to focus on process problems, as well as product problems: "The sooner you fall behind, the more time you will have to catch up."

Change Requests – Manage and Control Changes

The "essence" of the *Configuration and Change Management workflow* is to manage and control the scope of the project, as changes occur throughout the project lifecycle, while maintaining the goal of considering all stakeholder needs and meeting those, to whatever extent possible.

As soon as the first prototype is put before the users (and often even before that), changes *will* be requested. (One of those certainties of life!) In order to control those changes and effectively manage the scope of the project and expectations of the stakeholders, it is important that all changes to any development artifacts be proposed through Change Requests and managed with a consistent process.

Change Requests are used to document and track defects, enhancement requests and any other type of request for a change to the product. The benefit of Change Requests is that they provide a record of decisions, and, due to their assessment process, ensure that impacts of the potential change are understood by all project team members. The Change Requests are essential for managing the scope of the project, as well as assessing the impact of proposed changes.

User Support – Provide Assistance to the User

At a minimum, this should include a User's Guide, perhaps implemented through online help, and may also include an Installation Guide and Release Notes. Depending on the complexity of the product, training materials may also be needed, as well as a bill of materials along with any product packaging.

In RUP, the "essence" of the Deployment workflow is to wrap up and deliver the product, along with whatever materials are necessary to assist the end-user in learning, using, operating and maintaining the product.

What about Requirements?

Some of you may look at my list of essentials and vehemently disagree with my choices. For example, where do "Requirements" fit into this picture? Wouldn't that be a better choice instead of "Vision"? Requirements are certainly essential, right? Good questions. I would say that's basically what the first essential is – and for those of you who prefer to have "Requirements" on the list, you can put it at this point on your own list! (Remember, each

project team should come up with their own list. My list of ten essentials is only meant as a starting point for further discussion.)

In my experience, though, I sometimes will ask a project team (especially for an internal project), "What are your requirements?" and receive the response, "We don't really have any requirements." At first this surprised me (coming from a mil-aerospace background)! How could they not have any requirements? Then I found out that what they have in mind when they hear the word "requirements" is a set of externally imposed "shall" statements that they "must" conform to or have the project rejected – and they *really* don't have any of those! They may be involved in research and development where the requirements of the product evolve throughout the project.

Now, if I follow up that response with: Okay then, what is the vision for the product? Then, their eyes light up – we easily move through each of the questions in A through G above with no problem – and the requirements follow along thereafter. This is most often true in a collaborative environment, rather than a contractual project, where the requirements are "discovered" more than imposed by edict.

What about Test?

Undoubtedly, some of you keen sorts have also noticed that I didn't include "Test" as one of the "essentials" of RUP. Why is that? Unlike Business Modeling, which I truly believe is optional to an effective software development process, I in no way believe that testing is optional. I view test as an integrated set of activities that accompany the designing and building of the product – much the same way as ongoing Configuration Management and review activities – and, coincidentally, just as the IEEE 1074 software process standard does.

And, if you were very keen, then you noticed that testing is indeed included along with the building of the product (in essential number seven), with the testing results critically important for Verification & Evaluation (essential number eight). The essence of the Rational Unified Process' iterative approach to software development, is to continually build, test and evaluate executable versions of the product in order to flush out the problems and resolve the risks and issues as early as possible.

Summary: Applying the Ten Essentials

So, how can discovery of the "Ten Essentials of RUP" make a difference in my life?

For Very Small Projects

Well, first of all, if someone like my friend, Randy, had in fact come by to ask for advice on a how he could use RUP for building a simple product by himself or with a very small inexperienced team, just learning about process, I now have my own list of "ten essentials" that I can share with him, rather than overwhelm him with the full power of RUP and the Rational Suites of tools!

In fact, these ten essentials can be accomplished with no specialized tool support! A project notebook with one section devoted to each of the ten essentials, is actually a very good starting point for managing the project. And I have found Post-It Notes invaluable when initially managing, prioritizing and tracking change requests for my own personal application of the RUP!

For Growing Projects

Of course, these simple means of applying the ten essentials soon become unmanageable as a project's size and complexity grows, so those of you looking for potential customers for Rational's tools will find ripe prospects in those applying the "Ten Essentials of RUP".

And, of course, they'll eventually want the full RUP, too, as well as support of the Rational tools (if you do your job right!), once they decide to go deeper than just the surface level in each area. I would encourage you, though, to lead with the "Ten Essentials of RUP", alongside our "Best Practices", rather than leading with the tools.

For Mature Project Teams

For more mature project teams, the "ten essentials" can help provide you with a quick method for assessing the balance of the key elements of their process and identify areas where improvement is most beneficial.

In all projects, it is important to explore what will happen if any of these essentials is ignored – what will fail if you do not have it. For example:

> No vision? You may lose track of where you are going and may be easily distracted on detours.
> No plan? You will not be able to track progress.
> No risk list? You may be focusing on the wrong issues now and will explode on an unsuspecting mine 5 months from now.
> No issues list? Without accountability, these are the roadblocks to success.
> No business case? You risk losing time and money on the project, it may be cancelled or go bankrupt.
> No architecture? You may be unable to handle communication, synchronization, and data access issues as they arise; problems with scaling and performance.
> No product (prototype)? Get started writing code; just accumulating paperwork won't get you very far; get something working in front of the customers.
> No evaluation? Don't keep your head in the sand. It is important to face the truth. How close are you really to your deadline? To your goals in quality or budget?
> No change requests? How do you keep track of these?
> No user support? What happens when a user has a question or can't figure out how to use the product? How easy is it to get help?

So, there you have it!

Why "Ten" essentials? No particular reason, except I didn't want *more* than ten (can't count that high!). I would have been happy to have "Nine Essentials" or some smaller number, but we didn't seem to be able to cut any of them out. Guess that qualifies them as essential, right? At least to me! I encourage you to use this as a starting point in your project group: What are your "10 essentials"?

If you're like me, you may want to come up with a little memory hook or acronym to help you remember your list of "Ten Essentials". For example, to get it down to 4 syllables, I use V-PRI-BAPE-CU – and, of course, since I kept the number of essentials to 10, I can slyly use my fingers for counting them without having to refer to the list on my PalmPilot! (Now you know my secret!)

Dual Headquarters:
Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Toll-free: (800) 728-1212
E-mail: info@rational.com
Web: www.rational.com

International Locations: www.rational.com/worldwide

Rational, the Rational logo, Rational the e-development company, Rational Rose, Rational Unified Process are trademarks of Rational Software Corporation. References to other companies and their products use trademarks owned by the respective companies and are for reference purposes only.

Copyright 2000 by Rational Software Corporation.
TP- 177 9/00 Subject to change without notice. All rights reserved

QWE2002 Paper VT1

Mr. Peter Sterck
(ps_testware)

Quality Service Provision

Presentation Abstract

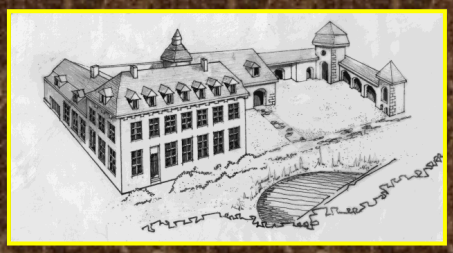
- Introduction
- Why outsource testing?
- QSP approach
- Why ps_testware?
- A business case
- Conclusions

About the Author

Peter Sterck founded ps_testware in 1994. He is CEO and President of the Board of Directors. With 10 years of experience, he is a respected expert in software testing. He graduated as an electronics engineer and gained a Master degree in MBA. He started his career as business analyst and became sales manager in 1992 for Performance Software Ltd., the most important testing tool supplier for UNIX computers.

Quality Service Provision

Quality Week Europe 2002



Peter Sterck,
ps_testware

Copyright © 2002 ps_testware – QSP - Peter Sterck – 1

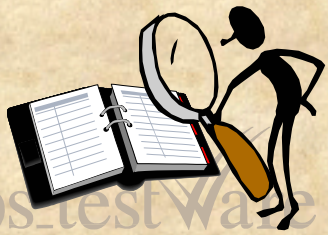
QSP - Agenda

- Introduction
- Why outsource testing?
- QSP approach
- Why ps_testware?
- A business case
- Conclusions

Copyright © 2002 ps_testware – QSP - Peter Sterck – 2

QSP - Agenda

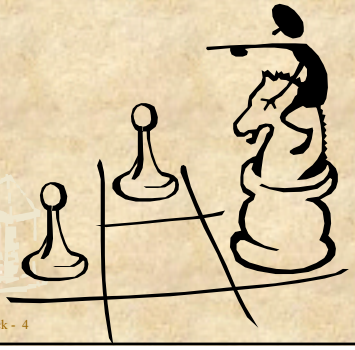
- Introduction
- Why outsource testing?
- QSP approach
- Why ps_testware?
- A business case
- Conclusions



Copyright © 2002 ps_testware – QSP - Peter Sterck — 3

ps_testware
Software Testing Services

Introduction



Copyright © 2002 ps_testware – QSP - Peter Sterck - 4

QSP - Introduction

- The goal of Quality Assurance is :
  - CONFIDENCE in the Result
  - MANAGEABILITY of the Process

Need for information

ps_testware
Software Testing Services

Copyright © 2002 ps_testware – QSP - Peter Sterck — 5

QSP - Introduction

- This information must be :
  - On Time
  - Accurate
  - Clear & Complete
  - Objective
  - Economical
- For all actors

ps_testware
Software Testing Services

Copyright © 2002 ps_testware – QSP - Peter Sterck — 6

QSP - Introduction

- To achieve this, you need a :

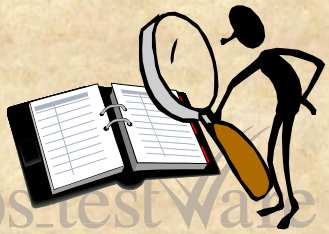
PROFESSIONAL APPROACH

Copyright © 2002 ps_testware – QSP - Peter Sterck — 7

ps_testware
Software Testing Services

QSP - Agenda

- Introduction
- Why outsource testing?
- QSP approach
- Why ps_testware?
- A business case
- Conclusions

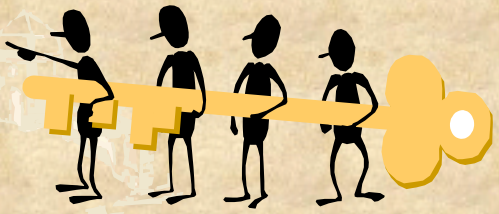


Copyright © 2002 ps_testware – QSP - Peter Sterck — 8

ps_testware
Software Testing Services

ps_testware
Software Testing Services

Why outsource testing?



Copyright © 2002 ps_testware – QSP - Peter Sterck - 9

QSP - Why Outsource Testing?

- **Professional testing is a big advantage for your business :**
  - **Business risks become visible, thus manageable**
  - **Internal manpower/knowledge needed for core business**
  - **Testing is no core business**
  - **Increasing quality needs**
  - **Increase of the diversity of IT solution providers makes IT more complex**
  - **Need to decrease the dependence of IT solution providers**
  - **Need for continuity**
  - **Need for budget control**

Copyright © 2002 ps_testware – QSP - Peter Sterck – 10

ps_testware
Software Testing Services

QSP - Why Outsource Testing?

- **To test professionally :**
  - Specialised knowledge needed
  - Different knowledge is needed in different phases of the process
  - Specialised knowledge not available because of small market and recent recognition of testing
  - Need for speed and on-call availability
  - Need for testers is fluctuating dramatically over time
  - Need for objectivity
  - Need for motivated testers

Copyright © 2002 ps_testware – QSP - Peter Sterck — 11

ps_testware
Software Testing Services

QSP - Why Outsource Testing?

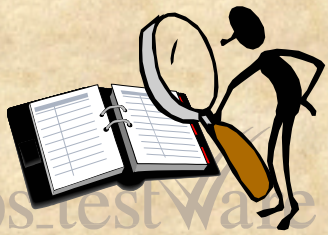
- **Professional testing is a big advantage in the management of your applications :**
  - Dramatically improve the upgrade process
  - Automatic documentation of the user requirements
  - Structuring and managing the requirements
  - Improve the communication between user and supplier
  - Make an investment of testing and not a cost
  - Provide all parties with the necessary information to make decisions

Copyright © 2002 ps_testware – QSP - Peter Sterck — 12

ps_testware
Software Testing Services

QSP - Agenda

- Introduction
- Why outsource testing?
- QSP approach
- Why ps_testware?
- A business case
- Conclusions

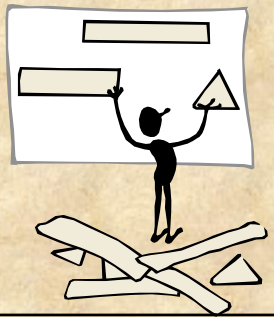


ps_testware
Software Testing Services

Copyright © 2002 ps_testware – QSP - Peter Sterck – 13

ps_testware
Software Testing Services

QSP approach



Copyright © 2002 ps_testware – QSP - Peter Sterck - 14

QSP

- Goal
- Definition
- Scope
- Approach
  - Test plan
  - Test design
  - Test execution
  - Follow up
- Dependencies
- Pricing
- Advantages and risks

Copyright © 2002 ps_testware – QSP - Peter Sterck – 15

QSP - Goal

- The Goal of QSP is :

To assure qualitative information on a software product and its development process by a third party.

Copyright © 2002 ps_testware – QSP - Peter Sterck – 16

QSP

- Goal
- Definition
- Scope
- Approach
  - Test plan
  - Test design
  - Test execution
  - Follow up
- Dependencies
- Pricing
- Advantages and risks

Copyright © 2002 ps_testware – QSP - Peter Sterck – 17

ps_testware Software Testing Services

QSP : Definition

- Quality Services Provision :
  - long-term agreement
  - for end user organisations and IT suppliers
  - to outsource (part of) the quality process
  - including a complete package of testing services
  - tailored to the needs of the customer
  - at a fixed price per year

Copyright © 2002 ps_testware – QSP - Peter Sterck – 18

ps_testware Software Testing Services

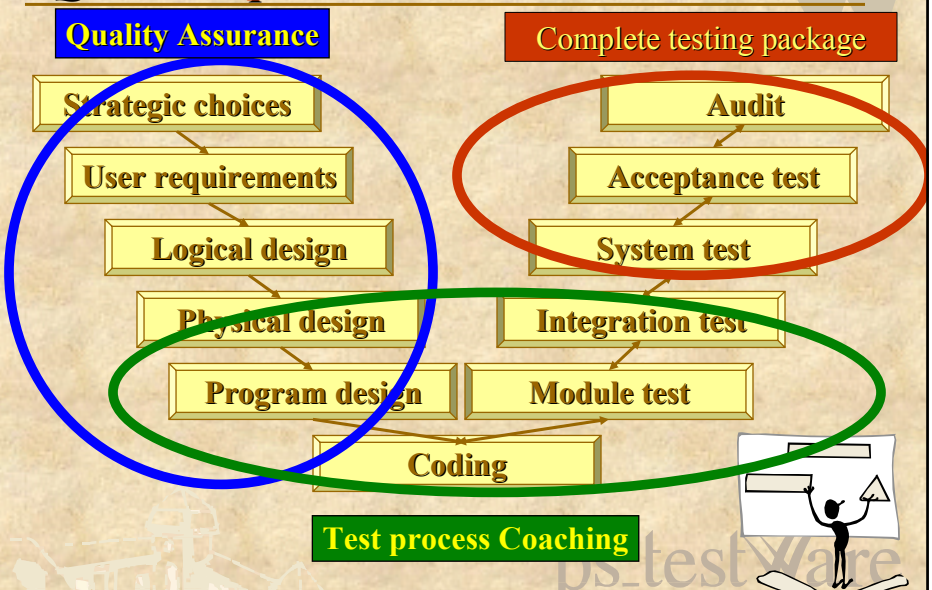
QSP

- Goal
- Definition
- Scope
- Approach
  - Test plan
  - Test design
  - Test execution
  - Follow up
- Dependencies
- Pricing
- Advantages and risks

Copyright © 2002 ps_testware – QSP - Peter Sterck – 19

ps_testware Software Testing Services

QSP : Scope



Copyright © 2002 ps_testware – QSP - Peter Sterck – 20

ps_testware Software Testing Services

QSP : Scope

- Left side of V-Model : Quality Assurance
  - Requirements management
  - Setting standards & procedures
  - Test planning
  - Review processes
  - Documentation
  - …
- Lower side of V-Model : Coaching
  - Test process management
  - Set-up test reference
  - Define reporting
  - Monitor review sessions

Copyright © 2002 ps_testware – QSP - Peter Sterck — 21

ps_testware
Software Testing Services

QSP : Scope

- Upper right side of V-Model : complete test package
  - Test plan
  - Test development
  - Test execution
  - Test reporting
  - Follow-up defects
  - Set up and maintain test environment

Copyright © 2002 ps_testware – QSP - Peter Sterck — 22

ps_testware
Software Testing Services

QSP : Scope

- All kinds of testing
  - Usability
  - Functionality
  - Performance
  - Load & Stress
  - Security
  - Portability
  - …
- Software developed in-house or a package purchased externally

In close cooperation with all parties !

Copyright © 2002 ps_testware – QSP - Peter Sterck — 23

QSP

- Goal
- Definition
- Scope
- Approach
  - Test plan
  - Test design
  - Test execution
  - Follow-up
- Dependencies
- Pricing
- Advantages and risks

Copyright © 2002 ps_testware – QSP - Peter Sterck — 24

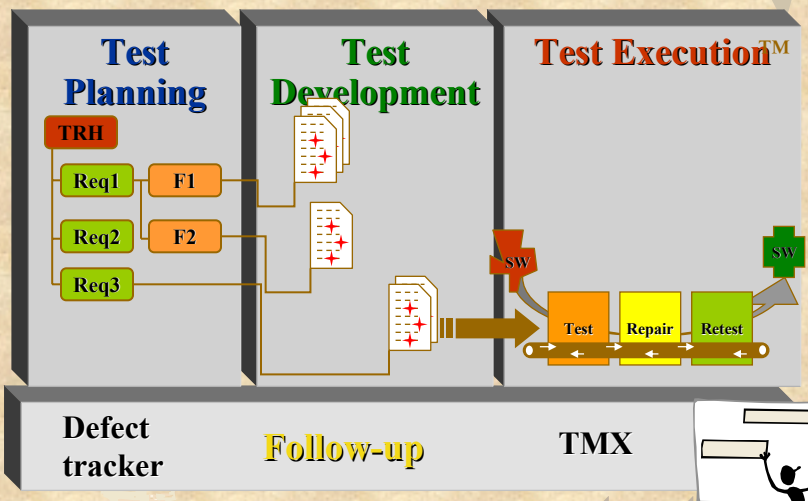
QSP : Approach

- **Test Assessment first**
  - Start from problems to propose the right solution
  - Know what is already in place ; Set up what is needed / missing
  - Put the right test focus (quality – time-to-market – costs)
- **SPI consultancy**
  - Automation (Feasibility study? Needs?)
  - Reporting and metrics
  - Set up coaching for module & integration testing
  - Set up testing for system & acceptance testing
- **I - Model**

Copyright © 2002 ps_testware – QSP - Peter Sterck – 25

ps_testware Software Testing Services

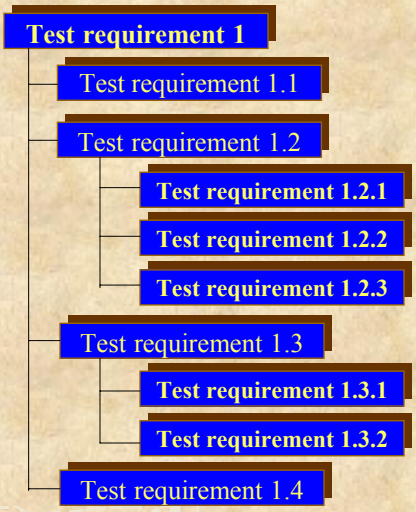
QSP : The Implementation Model



Copyright © 2002 ps_testware – QSP - Peter Sterck – 26

ps_testware Software Testing Services

QSP approach: Test planning



Test Requirements Hierarchy:

- Structured hierarchical set up of test requirements
- Process driven (related to the business)
- Starts with "process" test reqs, results in "function" test reqs
- Trace every possible step to the business goal

Copyright © 2002 ps_testware – QSP - Peter Sterck – 27

ps_testware Software Testing Services

QSP approach: Test Planning

- **Test Against Business Goals…**
  - **Business Process Thinking** TM (WHY ? WHAT? HOW?...)
    - **Business Process driven approach** (goal, transactions…)
    - **Contents driven approach** (coherence, culture…)

Copyright © 2002 ps_testware – QSP - Peter Sterck – 28

ps_testware Software Testing Services

QSP approach: Test planning

- … Faster and More Efficient Enhanced Test techniques
  - Preliminary workshop sessions & meetings
  - Business-oriented TRH
  - Expert knowledge and analytical techniques
- Faster answers
  - Build solutions that fit the market the best
  - Make communication effective and efficient
  - Set up a customised test strategy
  - Prioritisation

Copyright © 2002 ps_testware – QSP - Peter Sterck – 29

QSP approach: Test design

- 1 to n relationship with test requirements
- Use of test techniques
  - "Think green" make documents and templates recyclable (no concession to quality but faster reaction)
  - Limit the deliverables to their strict added value
- Set up and manage test bed
  - Define and manage test environment
  - Define and maintain test data

Copyright © 2002 ps_testware – QSP - Peter Sterck – 30

QSP Approach: Test Execution

- On-site or remote
- Use of testing tools
  - Test automation
  - Data generation
  - Load & stress testing tools
- Regression testing
  - Quick Scan & Intake
  - Based on risk profile of the new version
    - Full-blown retest
    - Specific retest
- Maintenance of testware

Copyright © 2002 ps_testware – QSP - Peter Sterck – 31

ps_testware Software Testing Services

QSP Approach: Test Follow-up

- Effectiveness : Test results
  - Defect tracking and management
  - Daily and weekly reporting
  - Tools
- Efficiency : Managing the test process
  - Prioritisation
  - Milestones
  - Stop criteria based on TRH
  - Weekly and monthly reporting
  - TMX tool

Copyright © 2002 ps_testware – QSP - Peter Sterck – 32

ps_testware Software Testing Services

QSP

- Goal
- Definition
- Scope
- Approach
  - Test plan
  - Test design
  - Test execution
  - Follow-up
- Dependencies
- Pricing
- Advantages and risks

Copyright © 2002 ps_testware – QSP - Peter Sterck – 33

ps_testware Software Testing Services

QSP : Dependencies

- Availability of 'business intelligence'
- Participation in the overall planning (steering committee)
- Mandate to negotiate with suppliers
- Availability of test environments

- Clear company strategy = better
- Up-to-date documentation = better
- Start early = better

Copyright © 2002 ps_testware – QSP - Peter Sterck – 34

ps_testware Software Testing Services

QSP

- Goal
- Definition
- Scope
- Approach
  - Test plan
  - Test design
  - Test execution
  - Follow up
- Dependencies
- Pricing
- Advantages and risks

Copyright © 2002 ps_testware – QSP - Peter Sterck – 35

ps_testware Software Testing Services

QSP : Pricing (1/2)

- Fixed price based upon a long-term agreement
  - Initial investments are waved over the total period
- Use application matrix to determine test effort
  - Platform
  - Software
  - Interface
  - Other parties involved
  - Actual test organisation
  - Complexity of the application
  - Size of the application
  - Risk factors

Copyright © 2002 ps_testware – QSP - Peter Sterck – 36

ps_testware Software Testing Services

QSP : Pricing (2/2)

- Pricing based upon the agreed TRH per application
- Review is possible per period (3 or 6 months)
- The price will be better for a longer term

ps_testware
Software Testing Services

Copyright © 2002 ps_testware – QSP - Peter Sterck — 37

QSP

- Goal
- Definition
- Scope
- Approach
  - Test plan
  - Test design
  - Test execution
  - Follow up
- Dependencies
- Pricing
- Advantages and risks

ps_testware
Software Testing Services

Copyright © 2002 ps_testware – QSP - Peter Sterck — 38

Advantages

- Fixed budget
- No admin/management costs
- No HRM issues like training, absence, hiring, etc.
- Motivated testers
- No peak issues
- No training and less/no facilities cost
- Fast and flexible
- Optimal reporting/metrics and benchmarking
- The results are not depending on one person
- Start being productive at the start
- Pay for the added value only

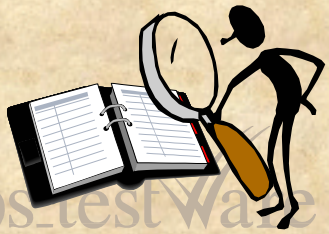
Possible Risks

- Lack of branch / application knowledge
- Supplier reliability / knowledge
- Definition of scope / objectives / R&R / prioritisation

ps_testware
Software Testing Services

QSP - Agenda

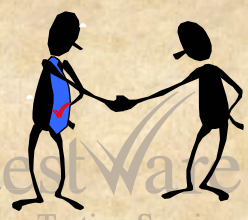
- Introduction
- Why outsource testing?
- QSP approach
- Why ps_testware?
- A business case
- Conclusions



ps_testware
Software Testing Services

Why ps_testware?

- Focus on testing : objectivity and R&D
- Motivated team
- 10 years experience in testing
- Multiple triple A references
- Ready to share the risk/take responsibility
- Unique service QSP: already 3 successful projects/customers

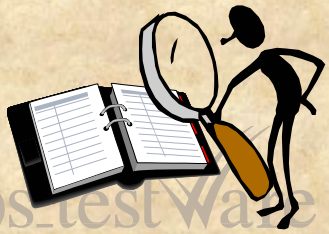


Copyright © 2002 ps_testware – QSP - Peter Sterck – 41

ps_testware
Software Testing Services

QSP - Agenda

- Introduction
- Why outsource testing?
- QSP approach
- Why ps_testware?
- A business case
- Conclusions



Copyright © 2002 ps_testware – QSP - Peter Sterck – 42

ps_testware
Software Testing Services

ps_testware
Software Testing Services

QSP : A Business Case

Copyright © 2002 ps_testware – QSP - Peter Sterck - 43

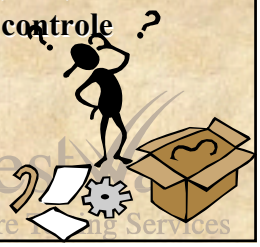
QSP : a Business Case (1/4)

- Financial application
- Custom software developed by a third party
- Phased implementation, 1 release every 3 months
- Different products implemented in a period of 2 years
- At first ps_testware delivered QSP for the supplier
- Later QSP for supplier and customer
- First version delivered to customer with problems

Copyright © 2002 ps_testware – QSP - Peter Sterck – 44

QSP : a Business Case (2/4)

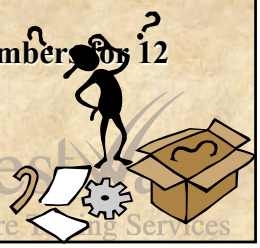
- **Result of the assessment :**
  - Prime problem: lack of quality
  - No test organisation present
  - No test method used (ultimate test was 'gorilla test')
  - Poor documentation and version management
  - Chaotic release management
- **SPI consultancy**
  - Set up of test organisation (90% ps_testware)
  - Introduce methodology and automation (Tool)
  - Set up procedure for version and release controle
  - Set up defect tracking system
  - Set up test process management system
  - Focus on system testing



Copyright © 2002 ps_testware – QSP - Peter Sterck – 45

QSP : a Business Case (3/4)

- **I-Model:**
  - Set up TRH using Workshops and interviews
  - Train developers for basic understanding of testing
  - ps_testware writes test designs
  - ps_testware writes automation scripts
  - a formal communication on quality is initiated with customer :
- **Results**
  - After 4 months the customer awards the supplier with quality excellence
  - The test team consists in average of 3 members for 12 developers
  - Peak resource needs up to 6 testers



Copyright © 2002 ps_testware – QSP - Peter Sterck – 46

QSP : a Business Case (4/4)

- ps_testware organises acceptance testing for customer
- The presence of ps_testware became a guarantee for a good co-operation between customer and supplier

Copyright © 2002 ps_testware – QSP - Peter Sterck – 47

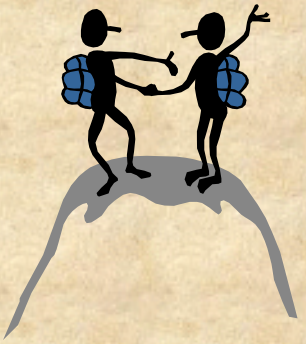
QSP - Agenda

- Introduction
- Why outsource testing?
- QSP approach
- Why ps_testware?
- A business case
- Conclusions

Copyright © 2002 ps_testware – QSP - Peter Sterck – 48

ps_testware
Software Testing Services

Conclusions



Copyright © 2002 ps_testware – QSP - Peter Sterck - 49

QSP : Conclusions (1/2)

- **Will Quality Service Provision reach its goal?**
- **High-quality information needs to be :**
  - **ON TIME**
    - **Well-planned**
    - **Dynamic staffing**
    - **Test process management is under control**
  - **ACCURATE**
    - **Specialised know-how available**
    - **Testers and QA are motivated**
    - **Test reference is set**

Copyright © 2002 ps_testware – QSP - Peter Sterck – 50

ps_testware
Software Testing Services

QSP : Conclusions (2/2)

- **High-quality information needs to be :**
  - **CLEAR & COMPLETE**
    - **TRH is an excellent tool**
    - **Regression testing becomes possible**
    - **Reporting and metrics used are proven technology**
  - **OBJECTIVE**
    - **Third party is independent**
    - **Communication with all parties is under control**
  - **ECONOMICAL**
    - **A well-managed QA process has a positive ROI**
    - **Internal/External resources are better used**
    - **Fixed price = budget under control**

ps_testware
Software Testing Services

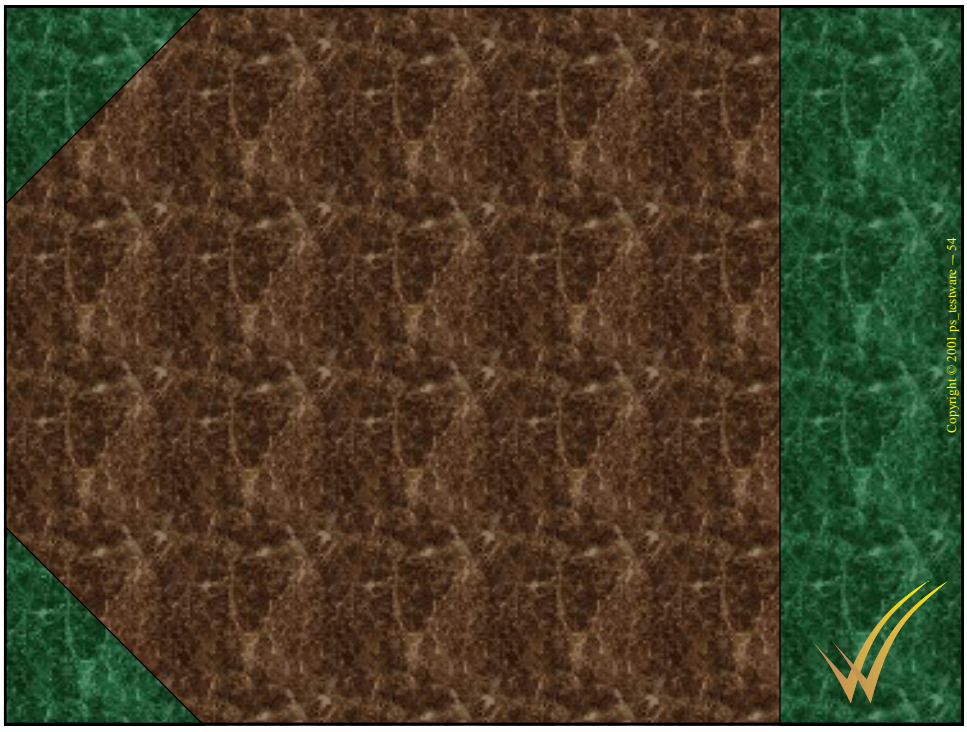
Copyright © 2002 ps_testware – QSP - Peter Sterck – 51

ps_testware
Software Testing Services

Questions

Copyright © 2002 ps_testware – QSP - Peter Sterck - 52

ps_testware
Software Testing Services
Tiensesteenweg 343
B-3010 Leuven
Tel.: +32 (16) 35.93.80
Fax: +32 (16) 35.93.88
e-mail: info@pstestware.com
http://www.pstestware.com
Copyright © 2002 ps_testware - Peter Sterck - QSP - 53


Copyright © 2001 ps_testware – 54

QWE2002 Paper VT2

Mr. Egbert Bouman
(Spherion Technology)

Five years of assesments with the Test Process Scorecard®: the method and the results

Key Points

- What is a Test Process Scorecard®
- How to perform test process reviews
- The value for test process improvement
- Experiences from over 50 test project reviews
- Do's and don'ts

Presentation Abstract

The Test Process Scorecard® method has been used with great success by Spherion since 1997. It enables you to quickly and regularly assess the test process, in projects as well as in test departments. The Test Process Scorecard® approach is designed to avoid lengthy and costly assesments. A typical assesment may be finished within two days, and the results have been proven remarkably valuable and dependable. The basis is a concise and cleverly designed questionnaire, which helps you to keep away from too much detail and to focus on the key success factors of the testing process.

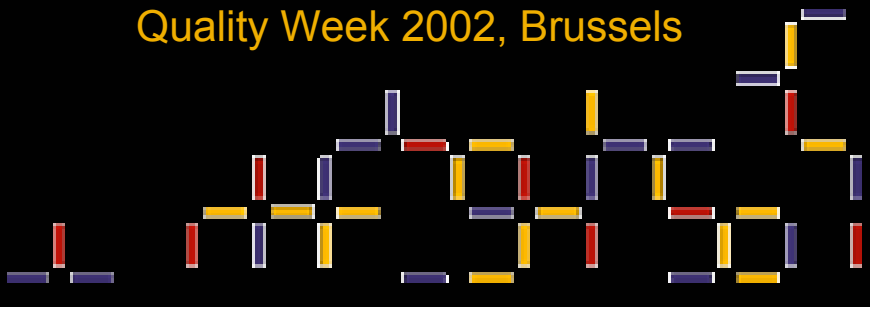
About the Author

Egbert Bouman is a test consultant and test manager at Spherion-IQM. He was assigned at customer sites at Telecom, Banking and Insurance companies. He was responsible for the Test Process Improvement part of the New Project Development methodology at KPN Telecom. He developed the Test Architecture for the ISI datawarehouse at ING Bank. Additionally, he is responsible for business development within Spherion IQM, and working as instructor in various courses, workshops and seminars on inspection, test management and requirements management. He has a broad experience in Test Process Scorecard Assesments with various customers as ING Bank and Delta Lloyd Insurance Group. Before joining Spherion, Egbert was involved in systems and acceptance testing since 1986.

Five years of assesments with the Test Process Scorecard®.

The method and the results

Quality Week 2002, Brussels

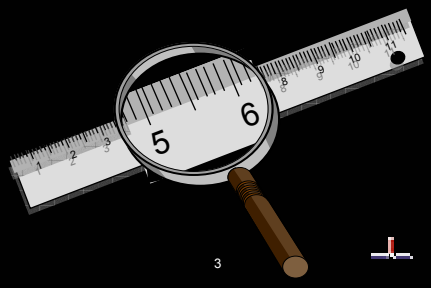


Keypoints

- What is a Test Process Scorecard®
- How to perform Test Process Reviews
- What is the value for Test Process Improvement®

What is a TPS

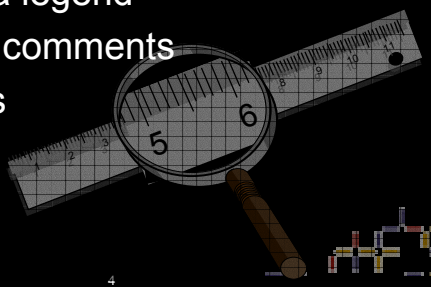
- An easy way to measure risks of individual test projects
- An instrument for measuring the test process maturity of an organization



Quality Week 2002 Brussels, Belgium 3

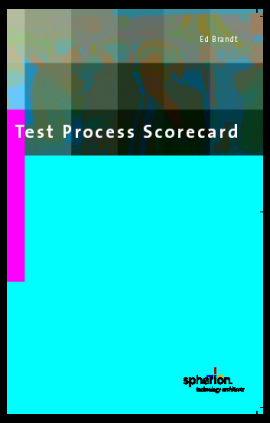
What is a TPS

- A checklist consisting of
  - 38 checkpoints or key process indicators
  - in 7 different subject areas
  - including a legend
  - additional comments
  - and charts

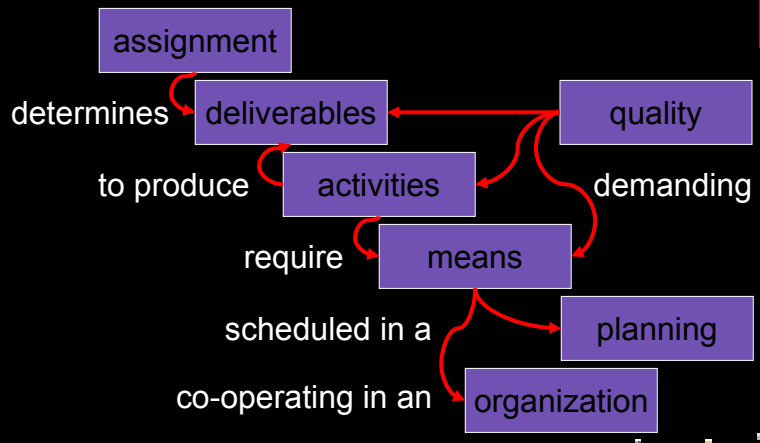


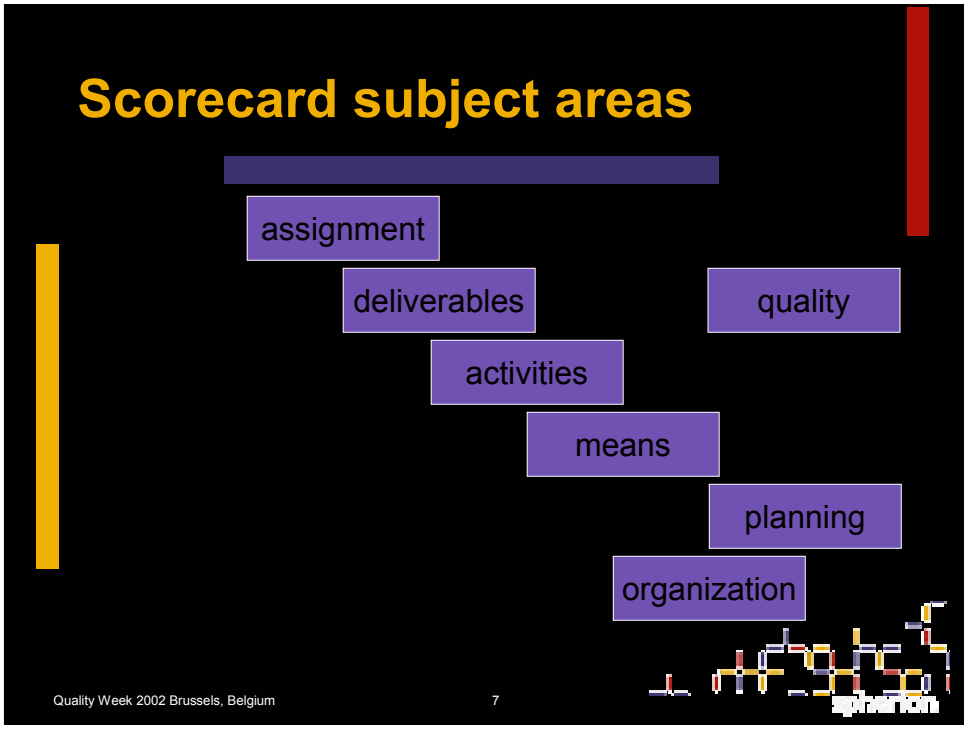
Quality Week 2002 Brussels, Belgium 4

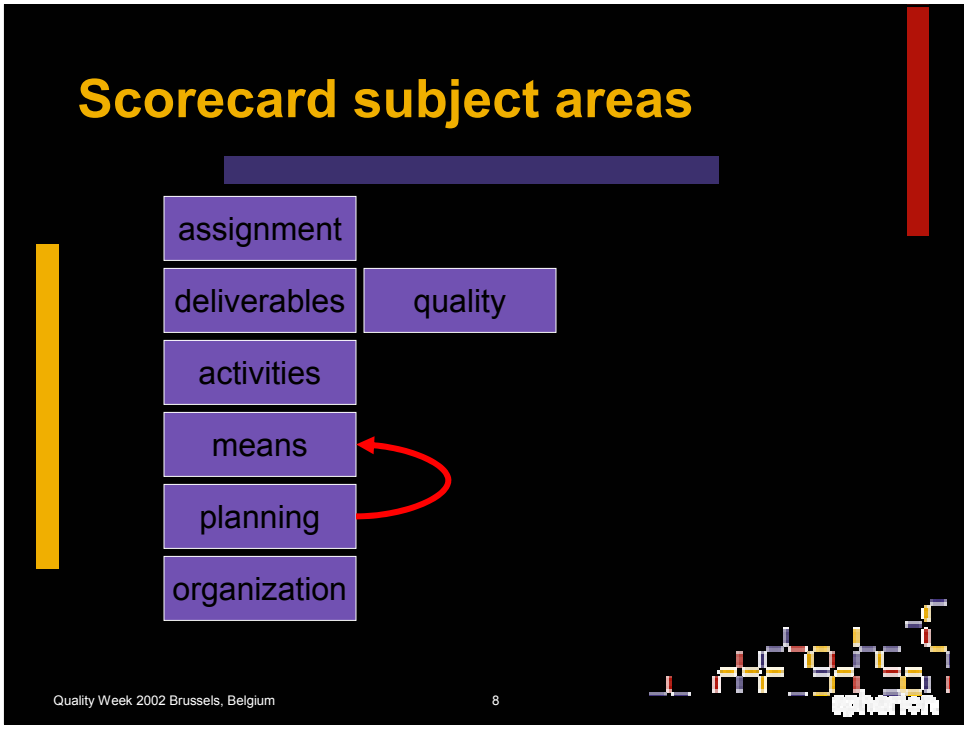
The Book

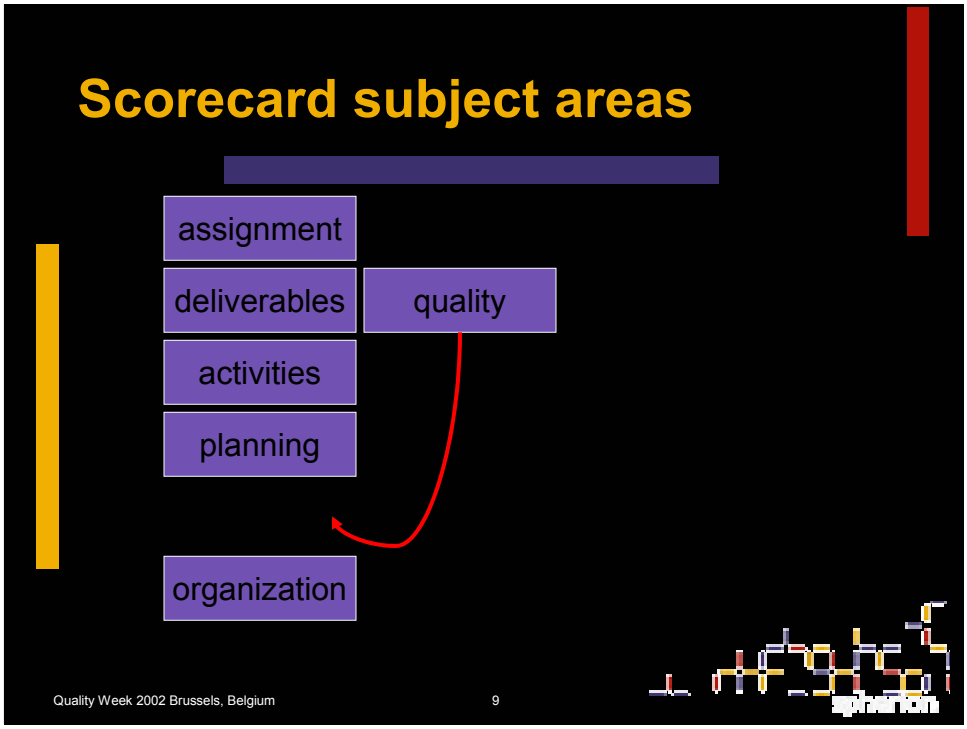


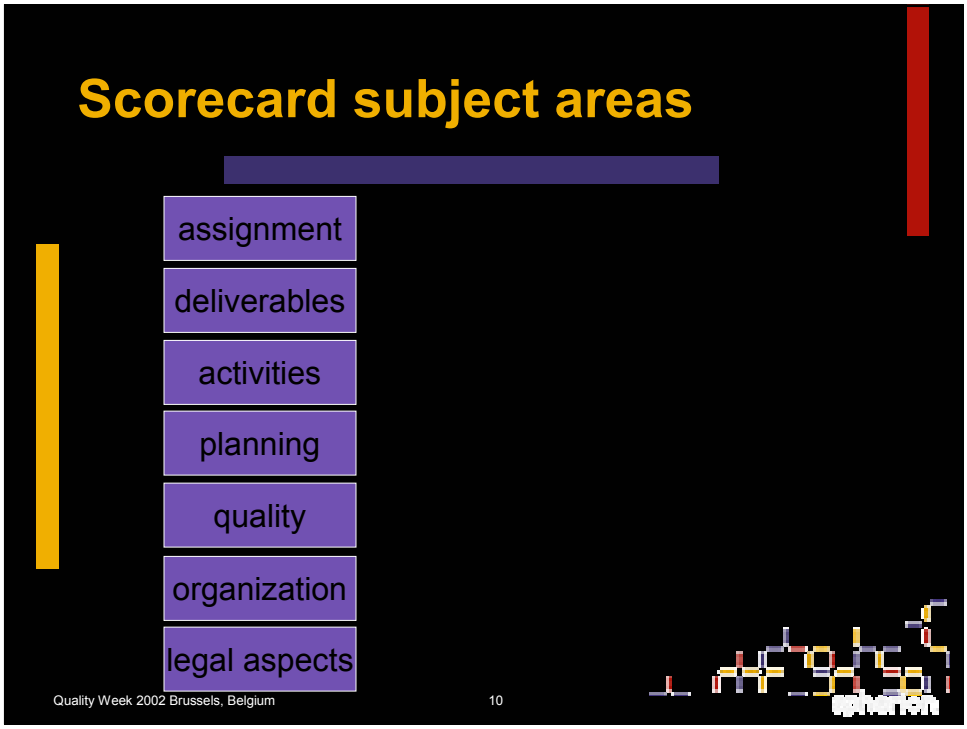
Context




Scorecard subject areas
assignment
deliverables
quality
activities
means
planning
organization
Quality Week 2002 Brussels, Belgium
7


Scorecard subject areas
assignment
deliverables
quality
activities
means
planning
organization
Quality Week 2002 Brussels, Belgium
8


Scorecard subject areas
assignment
deliverables
quality
activities
planning
organization
Quality Week 2002 Brussels, Belgium
9


Scorecard subject areas
assignment
deliverables
activities
planning
quality
organization
legal aspects
Quality Week 2002 Brussels, Belgium
10

Scorecard checkpoints

- assignment
  - •Goal of test project
  - •Assignment
  - •Contract partner
  - •Test manager
  - •Limitations
  - •Basics
  - •Critical Success Factors
- planning
- quality
- organization
- legal aspects

Scorecard checkpoints

- deliverables
  - •Overall-testplan
  - •Detail-testplan
  - •Acceptance criteria
  - •Test design
  - •Test cases
  - •Test procedures/scripts
  - •Test results
  - •Test infrastructure
- planning
- quality
- organization
- legal aspects

Scorecard checkpoints

- assignment
- activities
  - •Define test stages
  - •Determine testgoals
  - •Intake testbase
  - •Define testobjects
  - •Translate to physical testscript
  - •Set up test infrastructure
  - •Execution of testcases
  - •Delivery and testware management
- quality
- organization
- legal aspects

Scorecard checkpoints

- assignment
- deliverables
- planning
  - •Human resources
  - •Money
  - •Cost/benefits
  - •Infrastructure
  - •Elapsed time
  - •Milestones
- organization
- legal aspects

Scorecard checkpoints

- assignment
- deliverables
- activities
- planning
- **quality**
  - •Required level of test quality
  - •Specific quality improvement actions
  - •Review by independent authority
- organization
- legal aspects

Scorecard checkpoints

- assignment
- deliverables
- activities
- planning
- quality
- **organization**
  - •Structure
  - •Tasks & responsibilities
  - •Internal communication
  - •External communication
- legal aspects

Scorecard checkpoints

- assignment
- deliverables
- activities
- planning
- quality
- organization
- legal aspects
  - •Service Level Agreement
  - •Warranty

How to perform a review

- Fill in the scorecard….

	RI	++	+	0	-	--	n.a.
Testplan	3		x				
Test design	2				x		
Test results	3	x					

How to perform a review

- Fill in the scorecard….
- Using a reference model….
  - company standards
  - TMap®, TIMELESS™, CAST, etc.
  - common sense & experience

How to perform a review

- Fill in the scorecard….
- Using a reference model….
  - After studying the current situation….
    - Reading Testplans, designs, results, etc.
    - Interviewing keyrole players

How to perform a review

- Fill in the scorecard….
- Using a reference model….
  - After studying the current situation….
  - And discuss the results with the responsible manager(s).

What is the value for TPI®

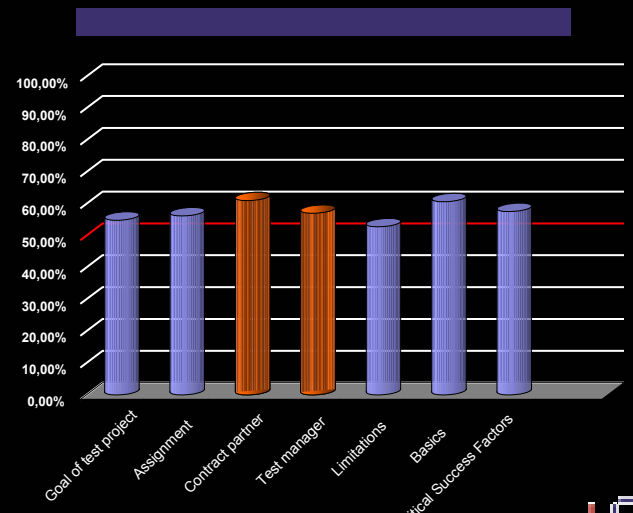
Or any other improvement model….

Totals from a series of scorecards illustrate current maturity or performance and show trends.

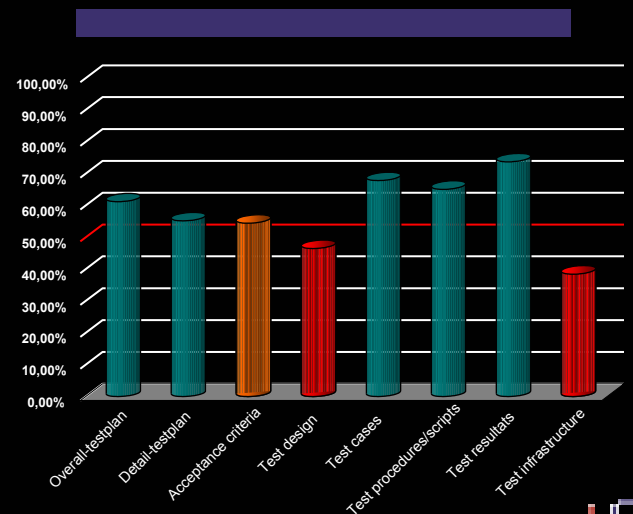
For example:

The following results are gathered from 34 reviews in 4 different organizations:

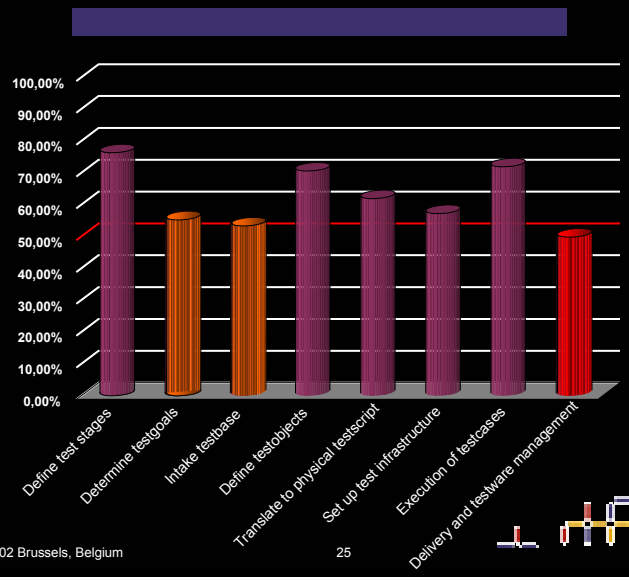
Assignment



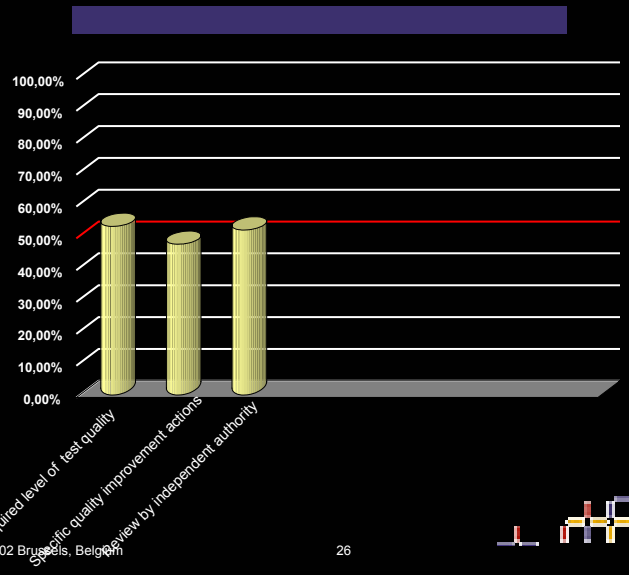
Deliverables



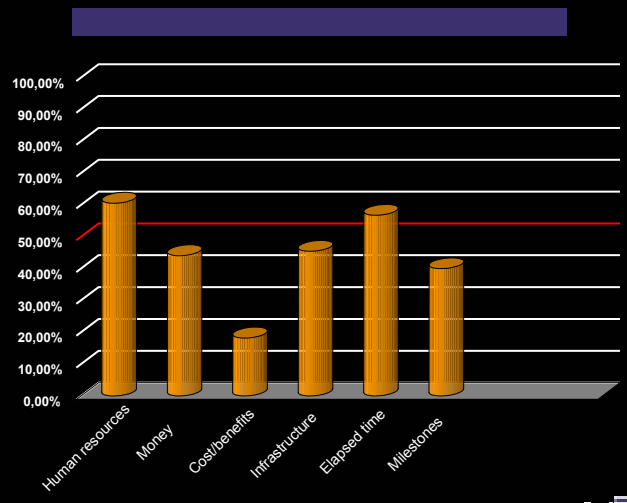
Activities



Quality

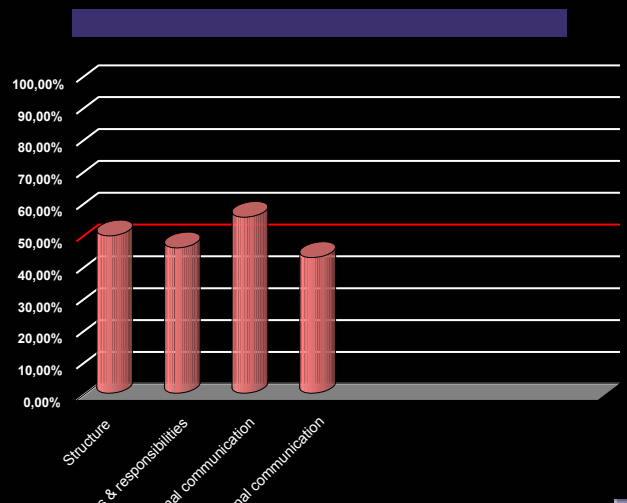


Planning



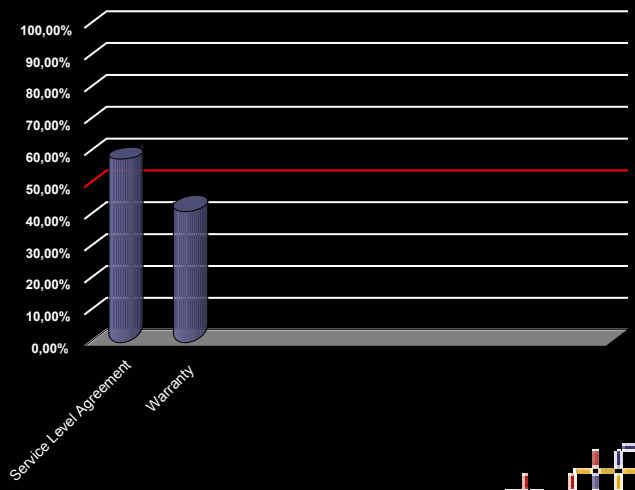
Quality Week 2002 Brussels, Belgium 27

Organization

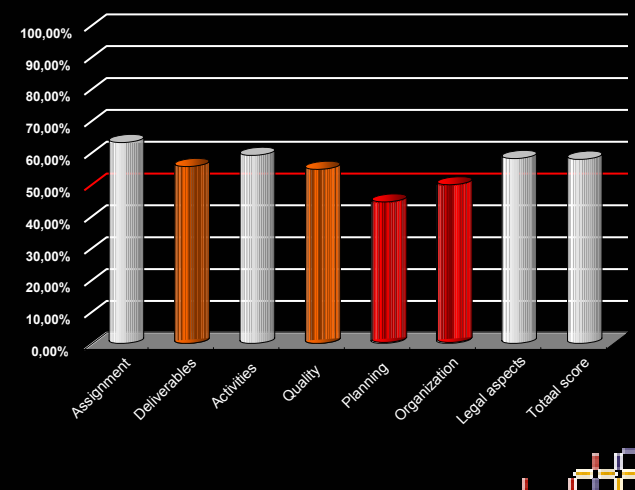


Quality Week 2002 Brussels, Belgium 28

Legal aspects



Total scores



Pitfalls:

- DON'T focus on %

Lessons learned from this example:

- What are we good at:
  - Define test stages and test objects
  - Execute test cases
  - Deliver test cases, scripts & results

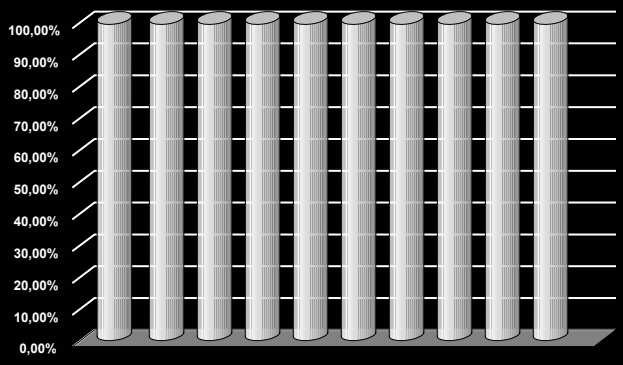
Lessons learned from this example:

- First priority improvements:
  - Deliver logical test design and infrastructure
  - Testware management
  - Planning & organization
- Second priority improvements:
  - Testmanager responsibility
  - Acceptance criteria & testgoals
  - Intake testbase

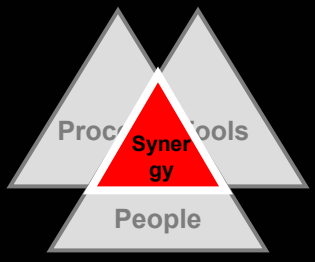
Test Process Scorecard®

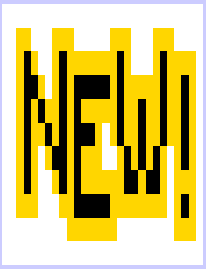
- Cons:
  - Experience required
  - Somewhat subjective
  - Does not include a reference model
- Pro's:
  - Easy to use
  - Quick results

And what do you think of it?



It all depends on synergy





QWE2002 Paper VT3

Mr. Kris Dugardyn
(Rational Software)

Develop High Quality e-business Applications - Test Smarter, Finish Faster

Presentation Abstract

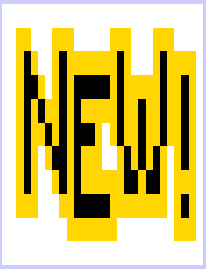
The Software Development Paradox describes the contradiction between delivering software on time, on budget and software that meets the expected quality. Delivering software on time allows your company a competitive advantage, but if this impacts the quality, the advantage of delivering on time will be lost.

How can we overcome this paradox? Iterative development is a solution, but only changing the way of developing software is not enough, it involves changing your current processes including your approach to testing software. Many obstacles and frustrations can impede testing productivity and potentially derail a project's success. Common problems include the ubiquitous issue of developers not being able to reproduce bugs; unintentional failure to test important business requirements; and testers working in frustrating, isolated environment where changes are difficult to assimilate and frequent rework is needed.

Find out how you can remove the impediments to successful testing and help testers focus on what really matters; assuring the quality if the software being developed.

About the Author

Kris Dugardyn - Technical Representative at Rational Software, focuses primarily on the testing domain for Belgium and Luxemburg. Prior to joining Rational in July 2000, he has spent over 10 years within ICT; both as a consultant and later responsible for Quality Assurance working on building business applications within both the Financial and Logistics industry.



QWE2002 Paper VT4

Mr. Martin Gijsen
(CMG)

ConTest: the Automation Architecture for the TestFrame Method

Presentation Abstract

While test automation offers very interesting benefits, making it work often proves a (very) significant challenge. The TestFrame method for test automation and the supporting ConTest architecture and tools have proven themselves as flexible solutions to achieve high quality products when time-to-market is essential. They have been applied successfully in hundreds of projects, including very large ones, for GUI, web, embedded and distributed systems. This presentation will explain why they have been so successful and how they can work for other (= your) systems as well.

About the Author

Originally a software engineer, Martin Gijsen soon developed an interest in test automation, where software engineers can use their skills to support testers. In his five year professional career he has been a software engineer, a test engineer (or test automator) and a test analyst, all for (very) large systems. He holds a degree in Computer Science, has published several articles on test automation and has given talks at both national and international conferences.

ConTest – the Automation Architecture for the TestFrame Method

Martin Gijsen
Martin.Gijsen@cmg.nl

Goal

- Describe the TestFrame method
- Describe the flexible ConTest architecture
- Show the wide applicability of TestFrame/ConTest

CMG

Agenda

- TestFrame method
  - Why automate?
  - Strategic context
  - Approach
- ConTest architecture
  - Basics
  - Some example configurations
- Recap

3

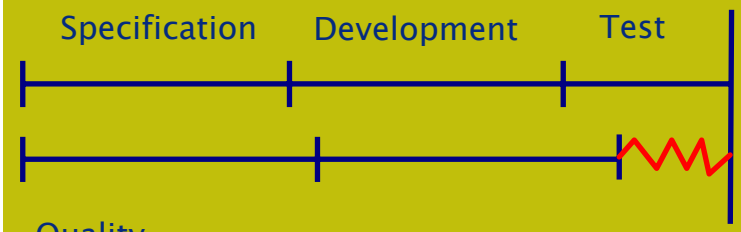
CMG

TestFrame The CMG approach to testing

CMG

TestFrame: why automate?

- Time: planning vs. reality



- Quality
- Money
- Happy testers

5

CMG

TestFrame: a strategic context

6

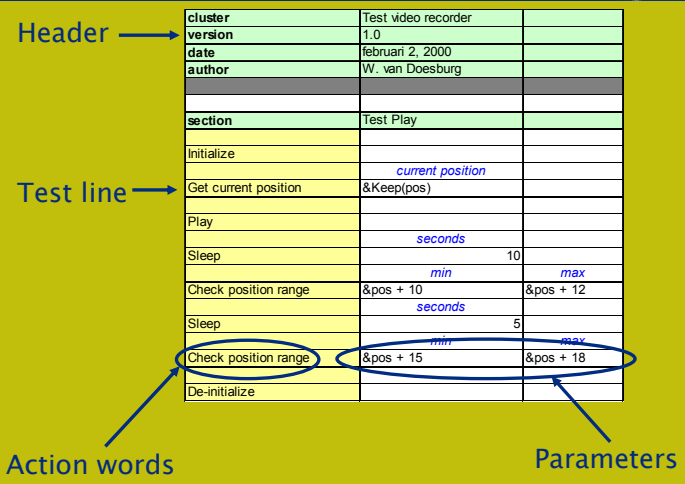
TestFrame: key elements

- Automate test execution when beneficial
- Develop tests as early as possible
- Use a flexible process

- 'Action words':
  - make automated tests
    easy to read, write and maintain
  - separate test definition and execution
  - limit maintenance sensitivity

7

Test specification: the cluster



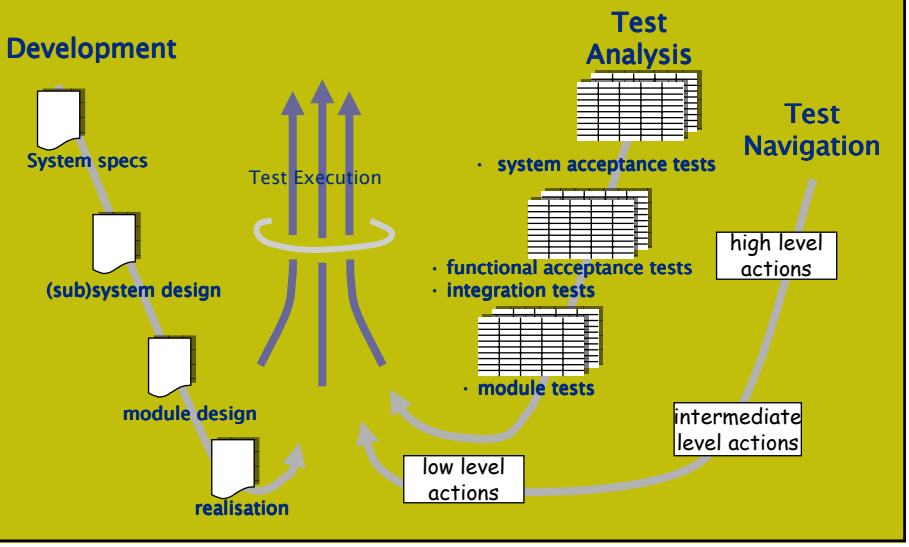
cluster	Test video recorder	
version	1.0	
date	februari 2, 2000	
author	W. van Doesburg	
section	Test Play	
Initialize		
	current position	
Get current position	&Keep(pos)	
Play		
	seconds	
Sleep	10	
	min	*max*
Check position range	&pos + 10	&pos + 12
	seconds	
Sleep	5	
	min	*max*
Check position range	&pos + 15	&pos + 18
De-initialize		

8

back

CMG

The TestFrame approach



9

CMG

ConTest the basics

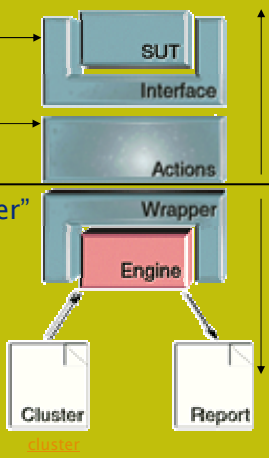
ConTest architecture

Communication with the System Under Test

ConTest module and/or project specific navigation

ConTest Engine and "glue layer"

Test specification & reporting



ConTest suites/ Record & Playback

ConTest core

11

The power of ConTest

- Action word based automated testing
  - Easy to use, low maintenance, …
- Standard interfacing solutions for many kinds of systems
  - Graphical User Interfaces
  - APIs
  - Protocols
  - Embedded systems
  - SAP
- Flexibility to combine existing modules
  - A SAP system sending out SMS messages can be tested end-to-end using a single test system
- Freedom to build your own modules
  - NAPI specification and Engine interface are open

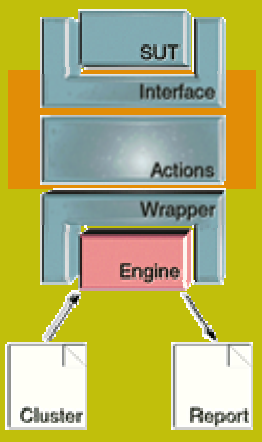
12

CMG

ConTest example configurations

CMG

Application – the principles



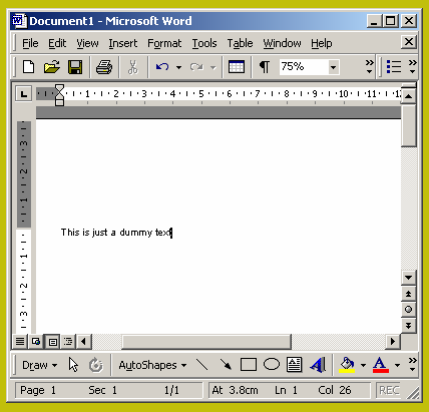
- Same architecture for all applications
- Interface and Action layers are replaced for different systems
- Increasing number of interfacing modules available off-she-shelf

14

Application – Graphical User Interface

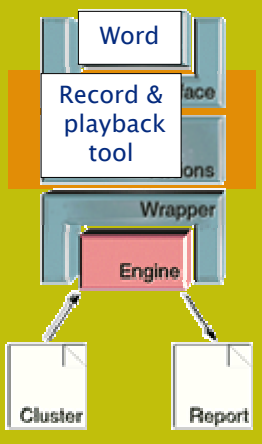
Case: I want to test an application that is controlled via a GUI

- Example MS Word
- The interface uses buttons, menus, listboxes, …



15

Application – Graphical User Interface



Strategy: integration

- Interface and language for action layer are provided by Record & Playback tools (WinRunner, QARun, Robot, …)
- ConTest Engine adds support for the TestFrame approach

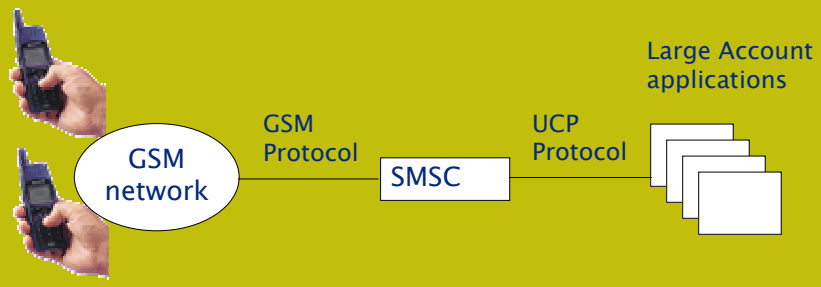
Your gain:
- Early start of test development
- Readable tests and reports
- Low test script maintenance

16

Application – Telecommunications

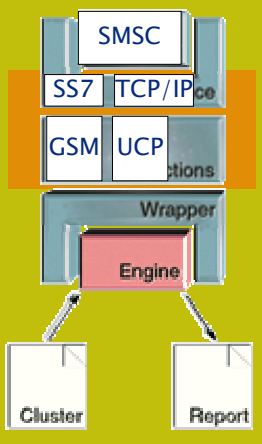
Case: I want to test a telecommunications system

- Example: Short Message Service Centre (SMSC)
- The system uses protocols (GSM, UCP, SMPP, …) using hardware connections (SS7, TCP/IP)



17

Application – Telecommunications



Strategy: Datacom Suite

- Interface and action words for protocol testing are provided by ConTest Datacom
- ConTest Engine adds support for the TestFrame approach

Your gain:

- Off the shelf actions and interfaces
- Readable tests and reports
- Low test script maintenance

18

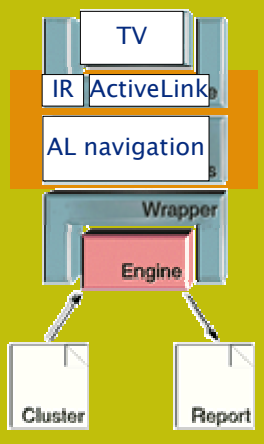
Application – Embedded software

Case: I want to test a device with embedded software

- Example: a television set
- The system responds to stimuli from remote control and has internal feedback interface (RS/232)

19

Application – Telecommunications



Strategy: ActiveLink Suite

- ActiveLink provides host – target communication and peripheral control
- ConTest Engine adds support for the TestFrame approach

Your gain:

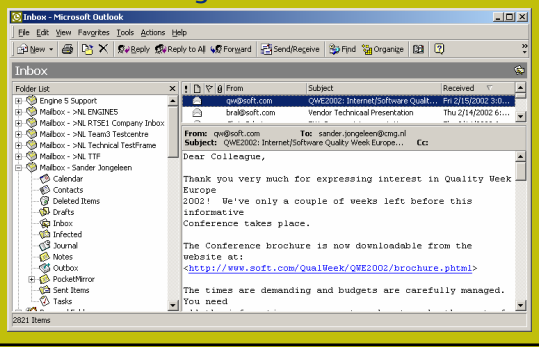
- Off the shelf link for both simulator and real device
- many OS's supported
- Readable tests and reports
- Low test script maintenance

20

Application – Complex systems

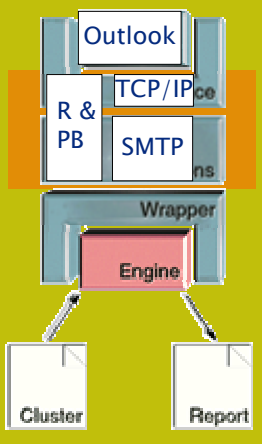
Case: I want to test a system via both GUI and protocol

- Simple example: MS Outlook sending SMTP messages
- The interface uses buttons, menus, listboxes, …
- The application sends out SMTP messages



21

Application – Complex systems



Strategy: Combination

- Interface and action words for protocol testing are provided by ConTest Datacom, the GUI interface by a Record and PlayBack tool
- ConTest Engine adds support for the TestFrame approach

Your gain:

- Complex systems can be tested using all interfaces
- Off the shelf actions and interfaces
- Readable tests and reports
- Low test script maintenance

22

Recap

- TestFrame helps achieve the benefits of test automation
- The ConTest architecture supports TestFrame
- The basic architecture is the same for any system
- Different types of interfaces of a system can be combined in one test environment
- ConTest provides the Engine and off-the-shelf modules for an increasing number of interfaces

23

Questions?

24

QWE2002 Paper VT5

Dr. Martin J. Wieczorek
(SQS Group)

Setting up a Test Competence Centre

Presentation Abstract

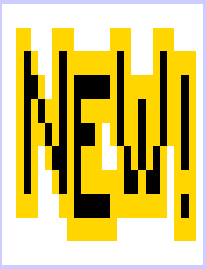
We all know the situation: new software systems are planned and the basic conditions concerning costs and time are predefined. As a project leader you can't say no, so close your eyes and do the job. After some period project management determines time and budgets are exceeded. Reasons for this could be:
- different components cannot be integrated, because one did not adhere to interface specifications,
- unexpectedly many errors in the system test occur, because the developers test could not be executed,
- unexpected performance problems during rollout occur, as stress and performance test were not feasible due to the complexity of the total system.

The basic idea to solve such quality assurance problems is to introduce an independent Test Competence Centre which will provide the certification of the software to be released. By using experiences, solutions, and tools from other projects effectiveness and efficiency will be increased. At the end, the project using TCC is able to deliver its product on time, in budget, and with high quality.

In this talk, goals, functions, processes, organisation, tools, and basic conditions for setting up a Test Competence Centre will be presented according to experiences from different projects and customers.

About the Author

Martin Wieczorek joined the SQS AG in 1995. As Senior Executive he is responsible for the business areas IT Services, Telecommunication, (e-)Commerce, Logistics, Public, Airports, Postal Services, Transportation, and Energy. He has many years of experiences in project management. His current interests are Test Competence Centres, Software Engineering, Specification, Risk Management, and Test Management.



QWE2002 Paper VT6

Mr. Brett Hodges
(Sitraka)

Opening the J2EE Black Box with PerformaSure: Identifying and Eliminating Performance problems for Java and J2EE

Presentation Abstract

Your team has been developing a critical J2EE-based system. When you deploy it, you find that it only meets 10 per cent of your capacity needs. You're not alone, and help is available. This talk will present approaches to ensuring maximum performance of these systems. It will look at how cross-functional teams can diagnose performance problems by looking at database performance, network effects, scheduling priority, and memory availability. It will then suggest remedies that might include adding memory or disk space, processors, or boxes, or making adjustments to scheduling priorities. It will also focus on the things you can do to avoid these problems in the first place, such as analyzing performance earlier in the development cycle, and later in the context of the full production and the production-mirror environments.

About the Author

Brett Hodges is an Applications Engineer at Sitraka. As a member of the Client Services team, based in the European Business Unit, he plays a key role in the success of our customer's adoption of Java technology, by helping our customers to identify and resolve performance issues in their J2EE development efforts, using Sitraka's performance assurance products. Brett possesses strong Java and J2EE development skills, has experience in the deployment of large scale J2EE projects, as well as knowledge of N-tier application architectures.

He is a consultant and a developer. He is well versed in the development process on Linux, Solaris, IRIX and Windows, offering a strong understanding of the challenges in bringing quality software products to the market. On top of building a high technical knowledge, he has gained valuable experience in technical consultation.

Opening the J2EE Black Box with

sitraka PerformaSure™

Identifying and Eliminating Performance Problems for Java and J2EE

Brett Hodges, Applications Engineer

sitraka

Sitraka: Proven Leader in J2EE Performance

Our focus	**J2EE Performance Assurance**
Our customers	**Fortune 500 customer base**
Our partners	**BEA, IBM, Sun, Mercury Interactive**
Our reach	**Global**

New Category in Performance Diagnosis

PerformaSure:

- Is a **transaction-centric diagnosis tool** for multi-tiered J2EE applications
- Helps companies **to measure, analyze and maximize performance**
- Is used during load testing and before costly problems occur in production

PerformaSure is **unique**:

- Not a production monitor tool, not BMC Patrol or Topaz
- Not a load simulator, not Mercury Interactive LoadRunner
- Not a profiler, not Sitraka JProbe

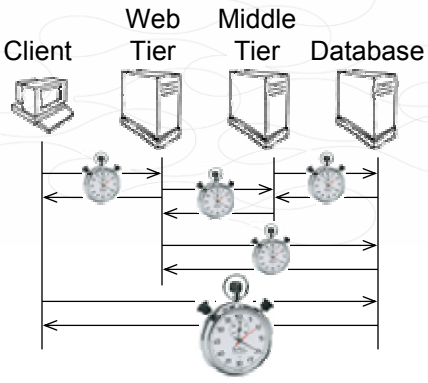
J2EE Performance: Hard Realities

- 90% of J2EE applications still in development
- Performance is mission critical
- Performance requirements not in specification
- System performance 20% of expectation
- Companies cutting production capacity; "more with less" philosophy

sitraka

What does Performance Mean?

Complex N-tier Clustered Architecture



Bandwidth?

Memory usage?

Database throughput?

CPU usage?

Component efficiency?

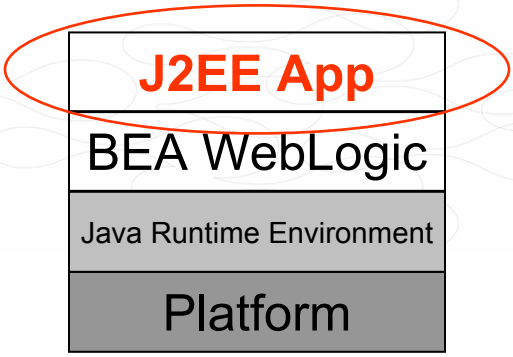
sitraka

Today, Performance Means

- End-user response time
- Doing more with less
- Resolving problems quickly

sitraka

An Integrated View of J2EE Performance



sitraka

The Performance Puzzle

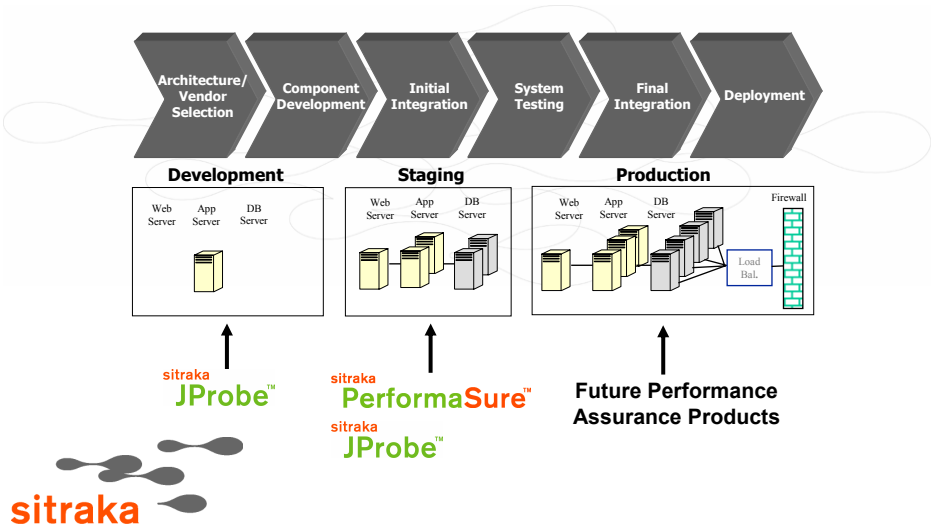
- Development tools provide “microscope”
- Load testing tools are “black box”
- Enterprise systems management tools are production; “black box”
- Production monitors … insufficient depth

Missing piece: Deep diagnostics in a system-wide view with low overhead.

PerformaSure Defines New Category

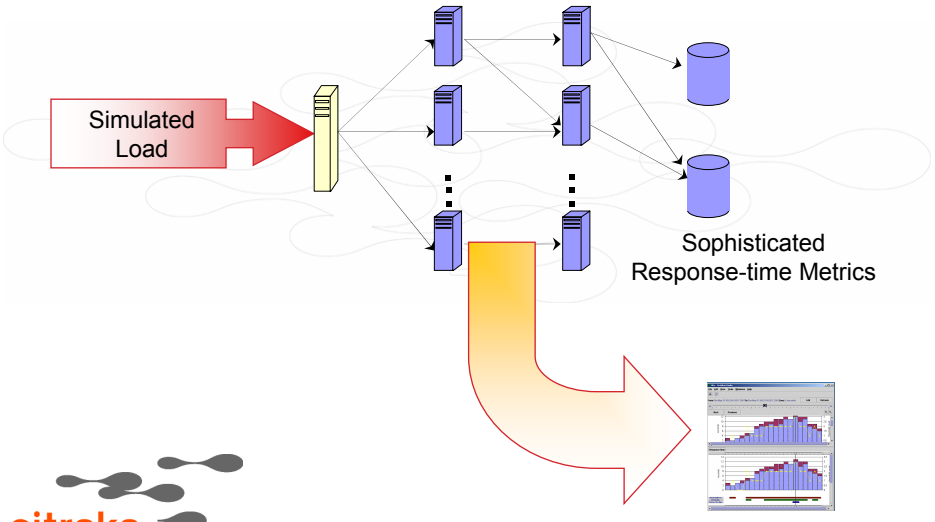
Product	Deep Application Diagnostics	System-Wide View	Low Overhead
Profilers	✓	✗	✗
Load Testers	✗	✓	✓
Production Monitors	✗	✓	✓
PerformaSure	✓	✓	✓

PerformaSure Targeted at Pre-production Testing



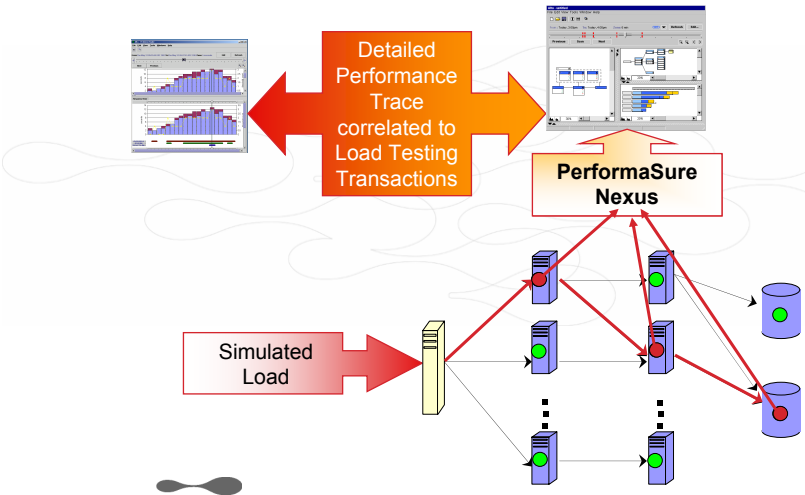
sitraka

Black-box Load Testing



sitraka

PerformaSure: J2EE Intelligence



sitraka

Diagnosing Performance Problems

- Focus on end-user transactions
- Identify the time window where the problem occurs
- Identify the problematic transaction
- Narrow the problem to specific server
- Isolate the component
- Call in the *right* expert

sitraka

PerformaSure Features

1. **Deep application diagnostics**
   - Breaks down transaction timing by component, server or cluster to show end-user point of view (unique Tag-and-Follow technology)
   - Collects and organizes a range of vital performance metrics for breadth and depth of J2EE diagnosis
   - Lets analysts drill down to transaction and method in question to find root cause

2. **System-wide view**
   - Shows how application code interacts with real-world infrastructure (application server configuration, databases, network, operating system, hardware, etc.)
   - Rich graphical user interface and views guide you through performance data
   - Powerful time navigation: fast-forward, rewind, zoom in and out (Zonar technology)

3. **Low overhead**
   - Configurable investigation options allow focused, low-overhead analysis
   - Agents installed on each component can be enabled and disabled
   - Non-intrusive Agents: no code changes, setup or recompilation necessary

What Beta Customers Are Saying

We were amazed when Sitraka PerformaSure validated some suspected performance issues with our J2EE application, clarified performance issues and, ultimately, helped us to dramatically improve our application performance. PerformaSure gives us a clear understanding of the performance subtleties and dependencies prior to deployment. This is critical to our success and is a key reason why we rely on PerformaSure.

Dan Hoogterp, executive vice president and CTO
enamics

sitraka

PerformaSure™

DEMO

sitraka

The Test Application

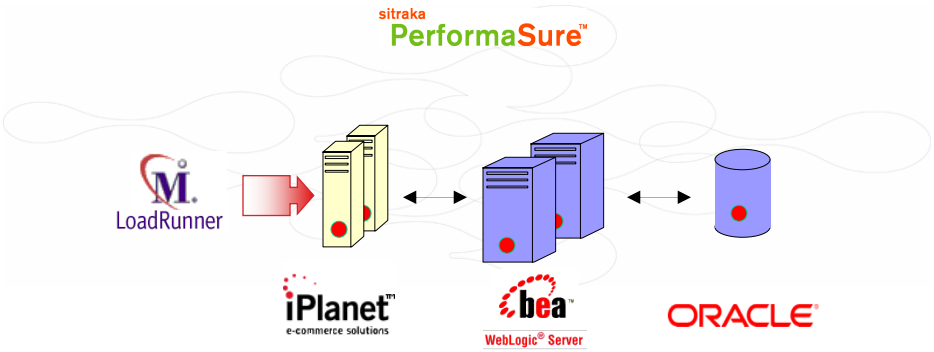
Presentation Tier

Business Logic Tier

Database Tier

sitraka

Test Lab Setup



sitraka

Questions?

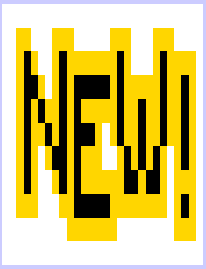
Brett Hodges
Applications Engineer
Sitraka Europe

Tel: +31 20 510 6729
Email: brett@sitraka.com

Quality Week Booths 307-308

www.sitraka.com

sitraka



QWE2002 Paper VT7

Mr. Raynald Korchia
(inQA.labs)

Census - Issue Tracking

Presentation Abstract

- inQA.labs
- MetaQuest Software, inc
- CENSUS® Web Issue tracking
- Demo

About the Author

Raynald Korchia has been working in the IT industry for more than 12 years, in the Software development, integration and Software Quality Assurance fields. He collaborated with the French Defense Administration and with companies like the Société Européenne de Propulsion or Alcatel in France and in Spain. He is one of the funders of inQA.labs a leading test laboratory and outsourcing company based in Spain. www.inqalabs.com

census web

inQA.labs
Software Testing

Census®

Issue Tracking

Raynald Korchia – inQA.labs

inQA.labs
Software Testing

Agenda

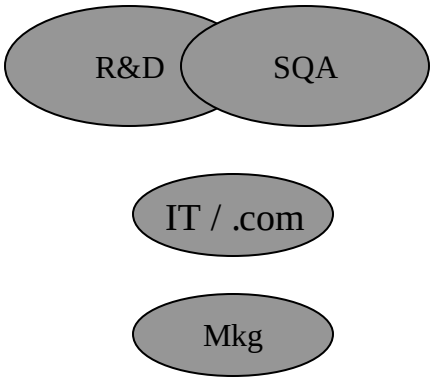
1. inQA.labs
2. MetaQuest Software, inc
3. CENSUS® Web Issue tracking
4. Demo

inQA.labs

To provide best in class Software Quality Outsourcing Services for helping Software Industry to release quality products

- Software Test Outsourcing
- Software QA Consultancy
- Test Automation
- QA Solutions
- E-testing
- Training

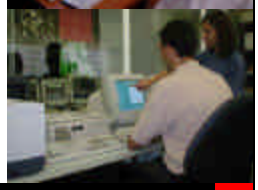
Helping the computer industry



- To develop Software
  - Testing
  - SQA Activities
- To deploy Software
  - Scalability
  - Reliability
- To market Software
  - Benchmarks
  - Localization Testing

Key Data

- Founded in 1999
- HQ in Barcelona, Spain
- Present in France and Spain
- SQA Professionals from 7 countries (Technical and Superior degrees only)
- Structured Auditable methods
- 24 hours guarded building
- Laboratory split in individual modules with restricted access and alarms
- FTP+HTTP servers dedicated to customers
- Flexibility and Cost effective
- More than 20,000 hours testing for worldwide



A Modern Laboratory

- Hardware
  - PC's (PIII, PIV), Macintosh, Unix WS
  - Servers: W2K, WNT, Novell, Linux
  - Routers: Cisco + HP connectivity
- Software
  - MSDN, ADC, HP-UX, Solaris, Linux...
  - Microsoft, Adobe, Quark, ACAD, ...
- Internet
  - ADSL 2Mb, Cable 1Mb, ISDN

Test Outsourcing Services

- Functional Testing
- Integration / System Testing
- Compatibility Testing
  - Platforms (Windows, MacOS, Unix)
  - Software (Application, browsers, …)
  - Networking (TCP/IP, AppleTalk, Novell,…)
- Localization Testing (Western Europe, CE, ME, Asia)
- Peripheral & Printer Labs
  - Driver testing,
  - Embedded controler testing,
  - Competitive intelligence,
  - pre-certification
- E-testing
  - Functional and load tests
  - On site and through the Internet

Software Quality Consultancy

- SQA Process
  - Test Planning,
  - Test Automation,
  - Inspection Plan
  - ...
- Training (English, French, Spanish)
  - Software Testing Essentials
  - Software Testing for non testers and management

QA Solutions

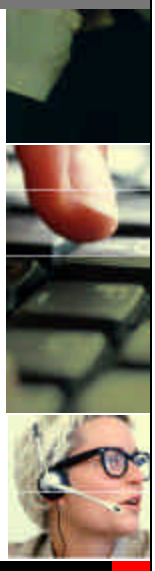
- MetaQuest® CENSUS Issue Tracking

METAQUEST SOFTWARE

METAQUEST census web

METAQUEST SOFTWARE

- Who is MetaQuest Software?
  - MetaQuest is a privately owned Software Publisher (ISV)
  - Located in Montreal, Canada
  - Founded in February 1996
  - Founders are
    - Ralph Akhras
    - Tony DiCaprio
  - Each with over 20 years software development and management experience



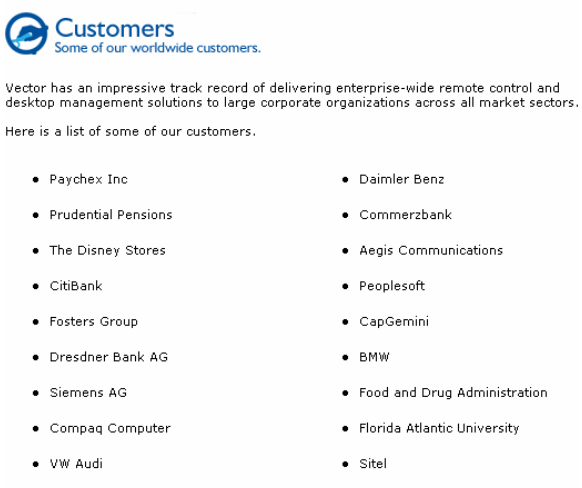
MetaQuest and Vector

- MetaQuest was recently acquired by **Vector Networks**
- Vector is a leading provider of PC Desktop and Management solutions

Vector Networks

- Founded in 1991
- 100 employees
- Vector products are distributed in over 35 countries
- Vector products are installed on over 2,000,000 PCs worldwide
- Offices in
  - Atlanta (Head Office + North America sales of Vector products)
  - UK (R&D + Marketing and distribution outside North America)
  - Russia (R&D)
  - Montreal (MetaQuest)

Vector Customers



Customers
Some of our worldwide customers.

Vector has an impressive track record of delivering enterprise-wide remote control and desktop management solutions to large corporate organizations across all market sectors.

Here is a list of some of our customers.

- Paychex Inc
- Prudential Pensions
- The Disney Stores
- CitiBank
- Fosters Group
- Dresdner Bank AG
- Siemens AG
- Compaq Computer
- VW Audi
- Daimler Benz
- Commerzbank
- Aegis Communications
- Peoplesoft
- CapGemini
- BMW
- Food and Drug Administration
- Florida Atlantic University
- Sitel

METAQUEST SOFTWARE

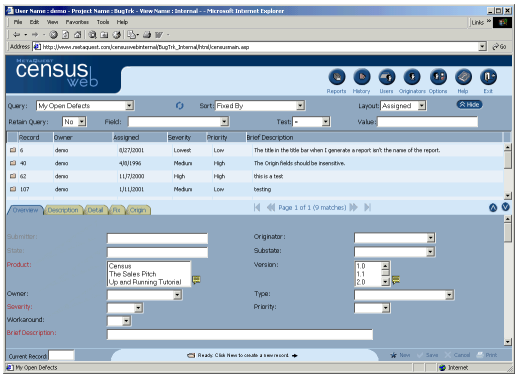
- MetaQuest Products Include
  - Census®
    - A Web-based Issue Tracking System
  - Triage®
    - An eSupport solution for streamlining the process of resolving technical support issues

Census® Customers

Census® Overview

- Census Issue Tracking
  - Automates the recording, tracking and management of issues such as
    - Software defects
    - Enhancement requests
    - Helpdesk calls
    - Tasks
  - Web-based
    - Supports distributed teams - access from anywhere, anytime, and any platform
    - Easy to deploy - no client setup

CENSUS®



Looks like the Web, but feels like Windows

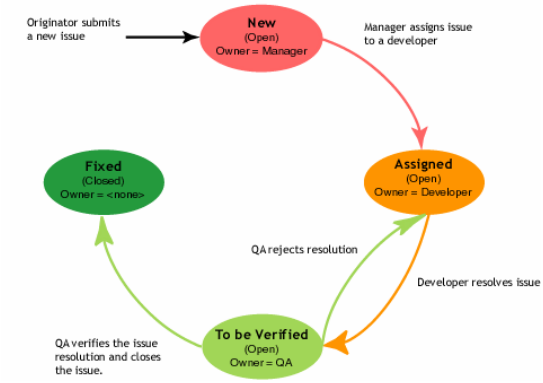
- Easier to use
- Easier to adopt

CENSUS® Features

Custom Web Views

- Make the entire database visible, or expose only the relevant issues, fields and functionality.

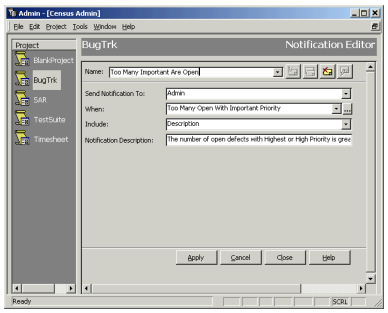
CENSUS® Features



Workflow

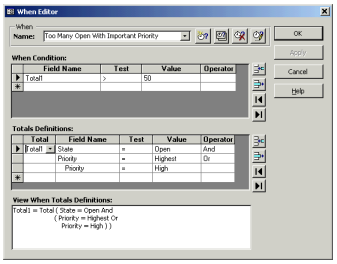
- Define issue lifecycle.
- States, possible transition between states and assign permissions for state transitions based on user roles.

CENSUS® Features



E-mail Notification

- E-mail notifications inform team members of changes to issues.
- Use GUI editors to define business rules for notifications.



CENSUS® Features

Built-in Reporting

- Generate reports, charts, and graphs.

CENSUS® Features

- Predefined queries
- Free-form keyword searches
- Attach files to issue
- Branding (logo, logon page, etc.)
- Customizable (add/remove fields, set drop-down choices, etc.)
- Import (over 10 formats)
- Export (RTF, XLS, HTML, and text)
- Change history (audit trail)
- Role-based permission (groups)
- Security (logon required)

CENSUS® Features

- User + Group Security
- Concurrent licensing
- Attachments
- Visual Source Safe Integration
- Scalability
  - MS Access® out of the box to SQL Server®

CENSUS®

- Demo
- Q&A

Special offer for Quality Week 2002 Delegates and Visitors
5% rebate on CENSUS® licenses acquired before April 30th 2002

More infromation...

- Contact:
  - +33 (0)1 47 56 20 33
  - +34 932 917 632
  - contact@inqalabs.com
- www.metaquest.com

MetaQuest census web

www.inqalabs.com

QWE2002 Paper VT8

Dr. Prakash Mutalik
(RelQ)

Testing to Improve Projects' Productivity

Presentation Abstract

Testing software is no longer a case of 'sitting at the terminal and banging a few keys' activity. Thorough testing is one of the key necessities for developing good quality software and this requires much more than just 'banging the keys...'. The need for quality testing is more acutely felt as the software complexity increases or when software is mission critical.

Thorough testing requires test technology and domain specific skill / technology within the test team. Suitable test models, technology for optimal test case generation, test automation and test effort estimation are critical in making testing effective. The productivity improvements brought by these technologies are clearly seen in software development.

In this session, the speaker discusses some of the models / technologies he has employed in some of projects and will discuss few case studies. He will also outline the concept of the 'Test knowledge engine' which would further boost testing productivity.

About the Author

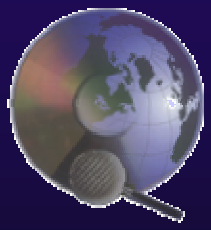
Dr. Mutalik is a co-founder and President of RelQ Software, a global company specialized in Independent Validation & Verification. RelQ is a Bangalore based software services company with operations in USA, Europe and Asia Pacific. RelQ has been successful in this niche and has a reputed set of clients world over.

Dr. Mutalik has over 26 years of experience in the software industry in India and USA and has served corporations such as Hewlett Packard and Wipro.

Dr. Mutalik has been a Quality consultant since 1992 to several reputed software companies such as Verifone, Honeywell, and Cadence, helping them in building Quality Management Systems to meet ISO 9000 standards. Twelve of his clients have been successfully certified for ISO 9001. He has conducted several corporate and public workshops in Software Engineering topics such as Project Management, Software Estimation, Software Testing, Reviews, and Configuration Management.

Through successful assignments and experience, he has developed specific technologies in the Quality Assurance area to help his clients develop better Quality software.

Testing to Improve Projects' Productivity
RelQ's Experience



QWE2002

By **Dr. Prakash Mutalik**
Group President & CEO
RelQ Software

Dr. Prakash Mutalik

- Co-founder of RelQ
- Over 26 years in Software Industry
- Worked in HP, Wipro
- Quality & process consultant to over 50 companies such as Honeywell, Cadence, Verifone
- Conducted several training in Software Engineering and Quality Assurance topics

RelQ Software -- Snapshot

- RelQ is a global software services company that offers software testing and related QA services.
- Over 4 years old; 300 strong.
- Completed over 185 testing projects.
- Areas of expertise: Internet, Real Time / Embedded Systems, Banking/Insurance, Safety / Mission Critical Applications.
- Clients include HP, Motorola, General Motors, Vodafone, Deloitte Consulting, Citibank, Siemens, Hitachi, Cadcentre, Infosys and Baxter.

Need of the hour

- **Today, the emphasis is on COST reduction and NOT …… fancy things**
- **Quality does NOT mean ISO or CMM but as perceived by your customer**

Agenda

- Road Blocks for improving Testing Productivity
- Testing Life Cycle
- Ways to increase Testing Productivity
- Increase Testing Productivity-a holistic approach.
- Test Productivity Summary

Testing Effort Vs Development Effort

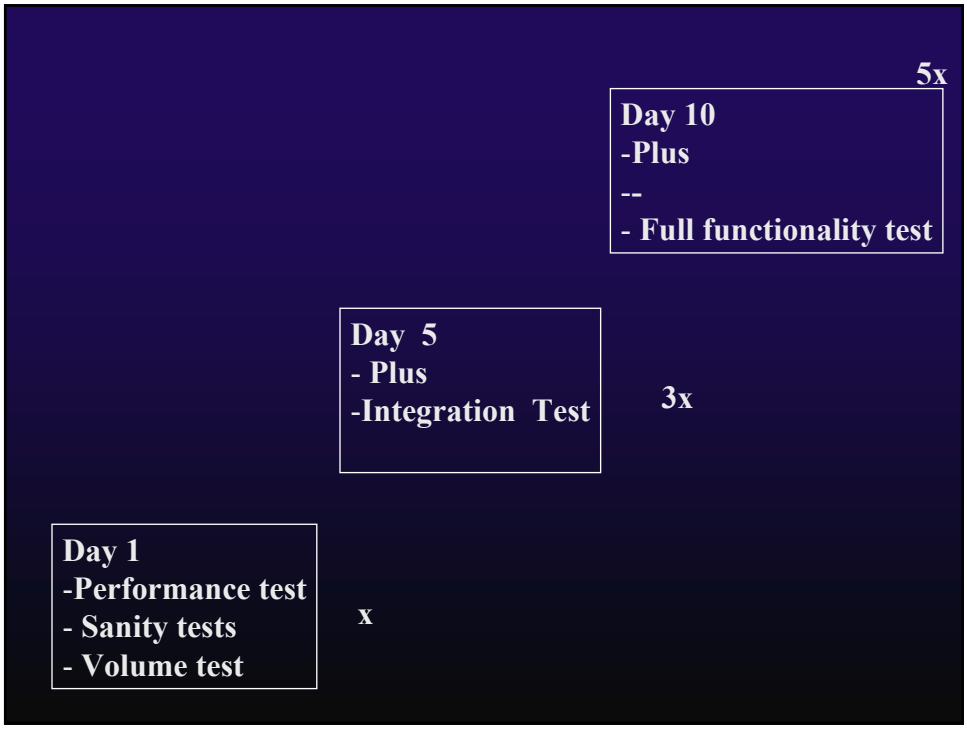
Type : Web Based Applications			
Projects	Development Effort (Person Months)	Testing Effort (Person Months)	Test effort as % of Dev eff
Project 1	122	52	43%
Project 2	84	15.5	18%
Project 3	180	56	31%
Project 4	24	6	25%
Project 5	42	12	29%
Project 6	6	3	50%
Project 7	27	7.5	28%
Project 8	76	12	16%
Average Test Effort As a % of Development Effort			**29%**

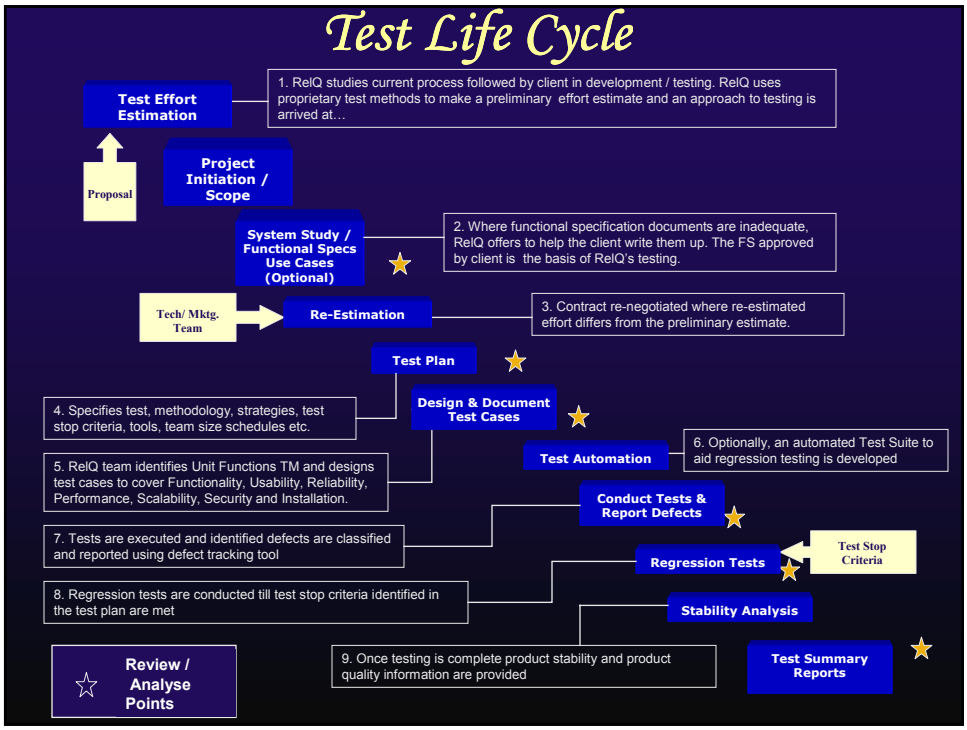
Road blocks for improving Test Productivity

- Unclear Specs / Scope
- Improper Life Cycle
- Inadequate Effort estimation, Test Planning
- Manual / Laborious Test Case generation
- Tools for Life Cycle discipline
- Manual Testing
- Not using the past project experience

Unclear Specs / Scope

- We are aware of importance of defining requirements for a sw project (ISO 9001 – contract review…)
- Over 40 % of the sw we have tested, did not have specs useful / good for testing .
- Many times customer is not clear what all needs to be tested !

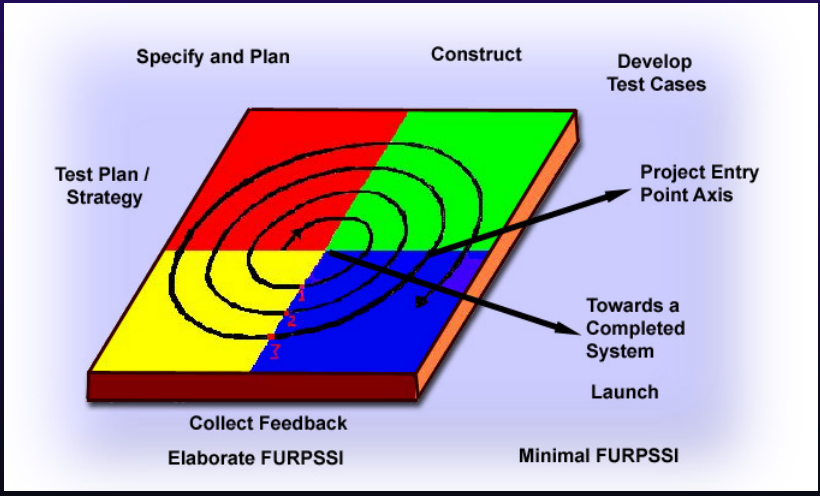

5x
Day 10
-Plus
--
- Full functionality test
Day 5
- Plus
-Integration Test
3x
Day 1
-Performance test
- Sanity tests
- Volume test
x


Test Life Cycle
Test Effort Estimation
1. RelQ studies current process followed by client in development / testing. RelQ uses proprietary test methods to make a preliminary effort estimate and an approach to testing is arrived at...
Proposal
Project Initiation / Scope
System Study / Functional Specs Use Cases (Optional)
2. Where functional specification documents are inadequate, RelQ offers to help the client write them up. The FS approved by client is the basis of RelQ's testing.
Tech/ Mktg. Team
Re-Estimation
3. Contract re-negotiated where re-estimated effort differs from the preliminary estimate.
Test Plan
4. Specifies test, methodology, strategies, test stop criteria, tools, team size schedules etc.
Design & Document Test Cases
6. Optionally, an automated Test Suite to aid regression testing is developed
Test Automation
5. RelQ team identifies Unit Functions TM and designs test cases to cover Functionality, Usability, Reliability, Performance, Scalability, Security and Installation.
Conduct Tests & Report Defects
7. Tests are executed and identified defects are classified and reported using defect tracking tool
Test Stop Criteria
Regression Tests
8. Regression tests are conducted till test stop criteria identified in the test plan are met
Stability Analysis
Review / Analyse Points
9. Once testing is complete product stability and product quality information are provided
Test Summary Reports

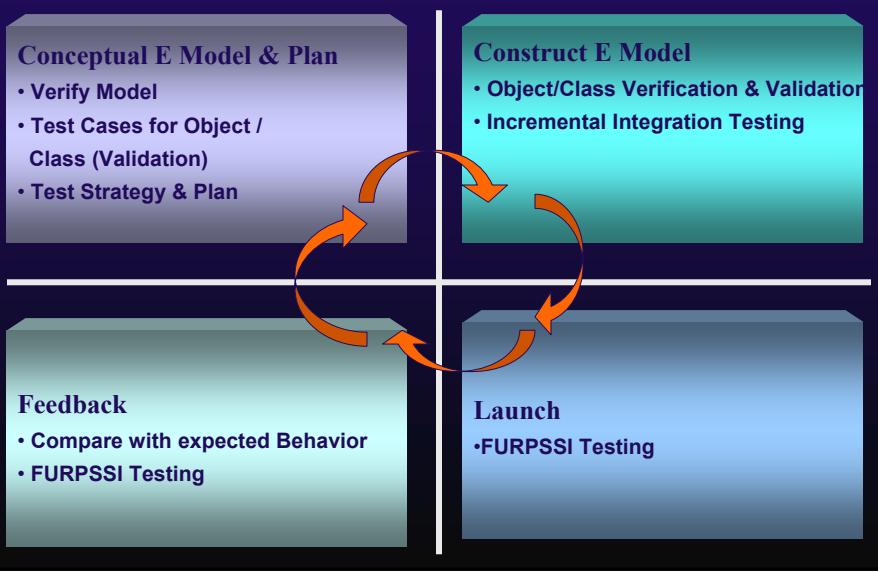
Tailoring Test Life Cycle

- Just as in Development where we choose/tailor the life cycle (std… 12207), we need to choose a suitable test life cycle to suit the project
- Case study

RelQ Rapid Web Testing Model



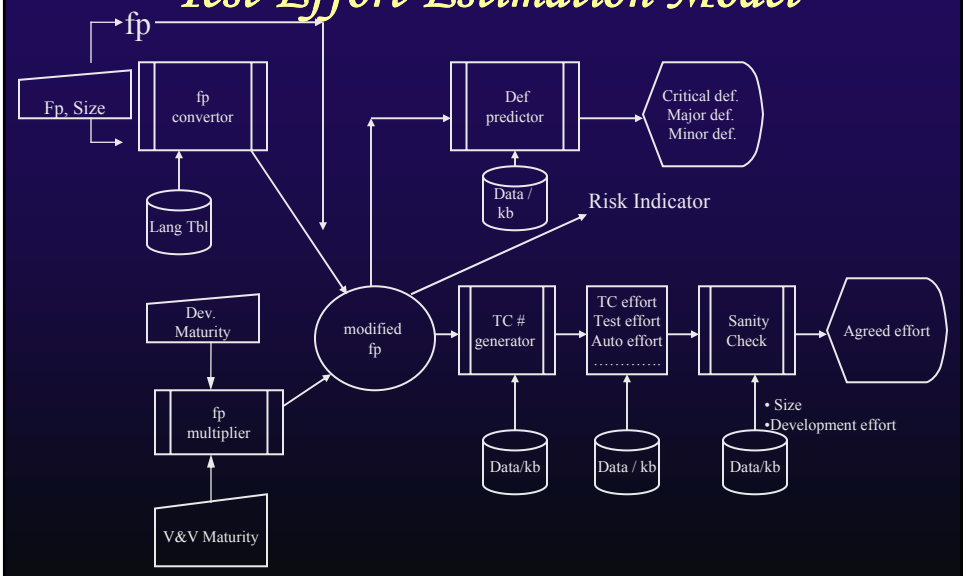
V & V Model for E-engineering



Plan Test Effort for productivity

- Low estimation - testing does not complete in time - overshoots expected deadlines.
- High estimation - Increase in cost - undue foreseeing delays of product release.
- Estimation tools add to productivity- bring in actual effort,resources - probable defects

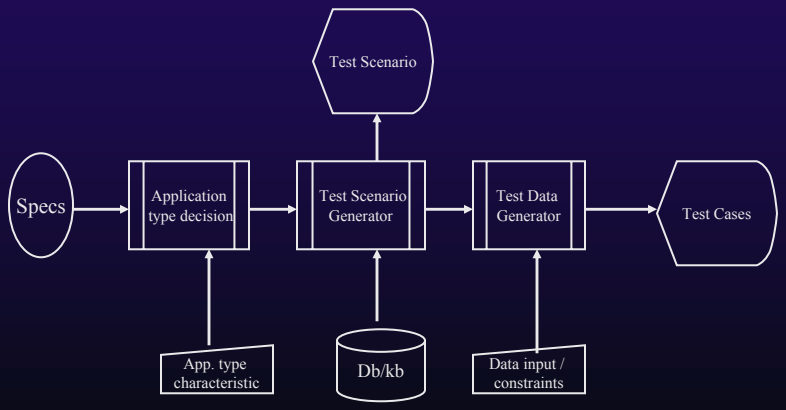
Test Effort Estimation Model



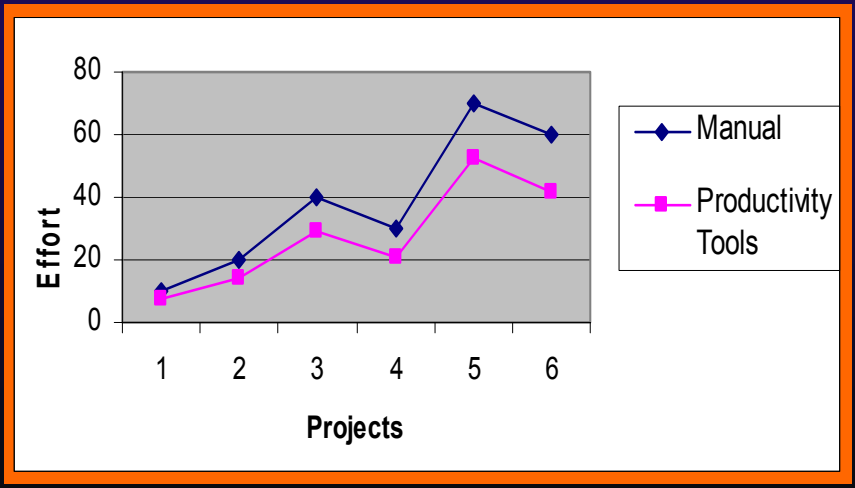
TC generation productivity

- Manually consumes 40% – 45% of the entire Testing Effort
- Testing efficiency is determined by Test Case efficiency and completeness.
- Productivity Tools generate TCs, increase completeness, reduces time and effort.
- Test Cases are generated from details of Unit functions (from LLD) and/or also from the knowledge base of Test cases already available for similar projects.
- Test cases modified or New test cases need to be added which are specific to the application only.
- Can reduces TC development time by 50 – 70%

Test Case Generation Model



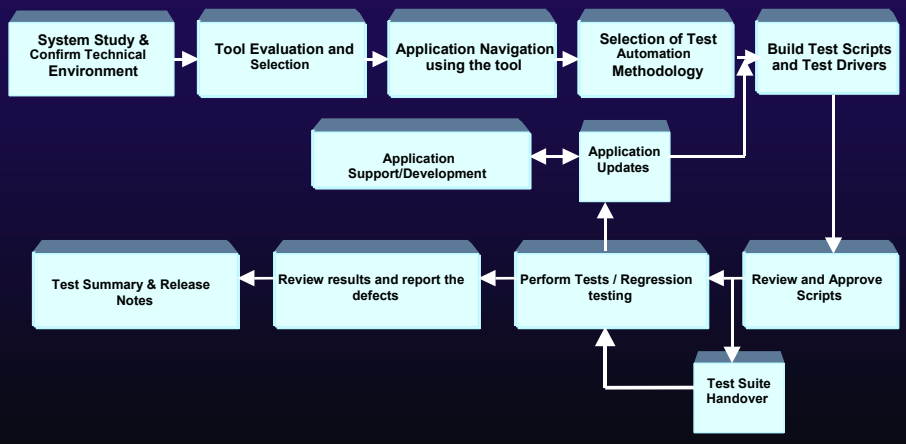
Tool based test case generation & Test Productivity



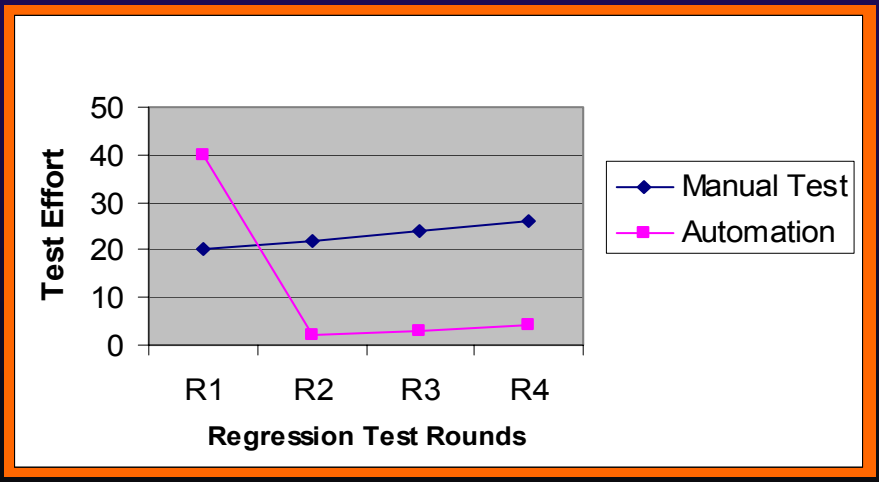
Productivity through automation

- Test Execution consumes 40-50% of the overall test effort.
- Regression rounds may require full round of test execution.
- Significant amount of effort and time goes in manual test execution for every regression round.
- More the application changes during maintenance, more is the number of Regression Rounds.
- Automating Test Execution by help of productivity tools can significantly bring down test effort and time.
- The more the number of regression rounds automated, the more would be the productivity factor.

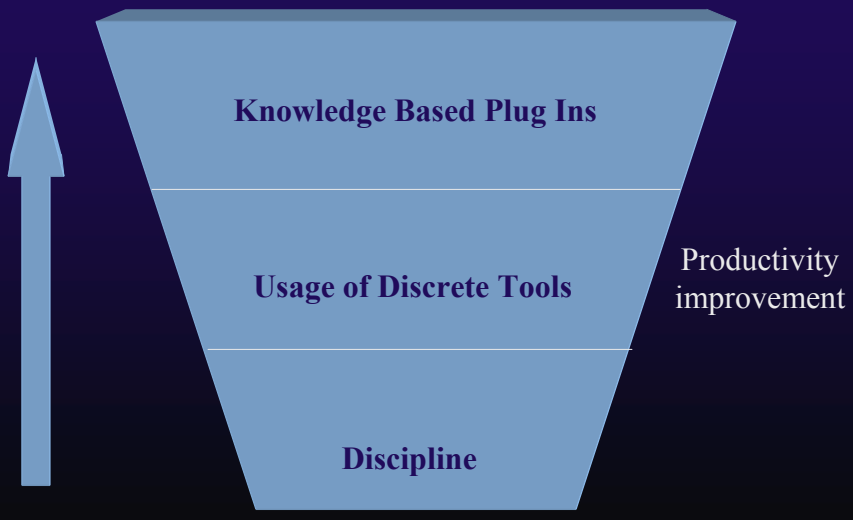
RelQ Test Automation Model



Productivity by Test Automation



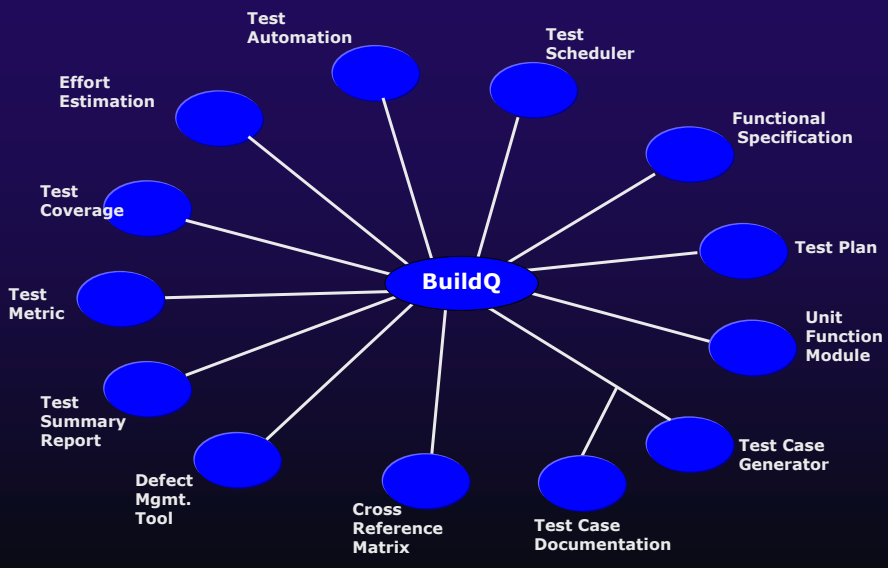
Holistic Approach for Test Productivity Improvement



Productivity improvement through discipline

- Enforces a sequence of thinking and test process.
- Test Engineers cannot deviate from best standards and good practices brought out by the test framework and tools.
- Builds close co ordination among teams, ensures everybody is in synergy with a common understanding of all testing related issues and processes.
- Test Engineers need to follow standards for preparing Test Artifacts, documentation and communicating.
- Tools will enforce timely response as per requirements and also help contributions from each Test Engineer.

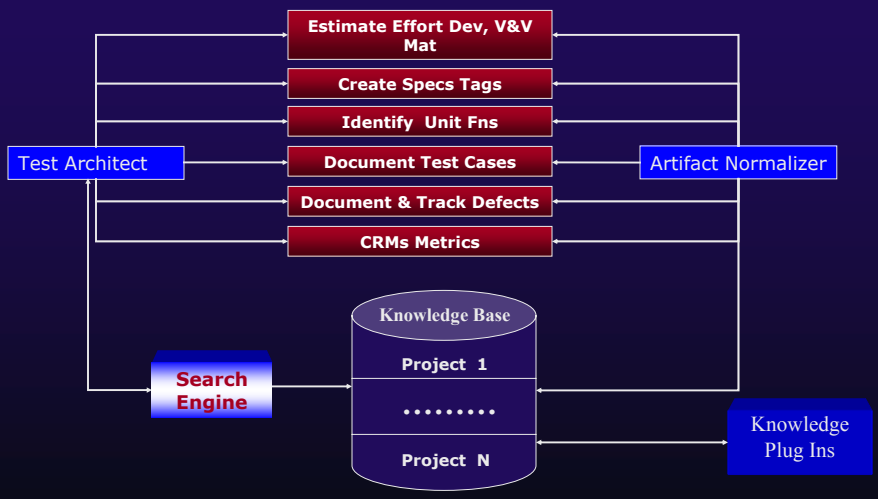
BuildQ: Productivity Edge



Productivity through knowledge base

- RelQ has worked on more than 185 projects, on various domains and environments.
- We have taken full advantage of our experience by creation of a comprehensive Knowledge base.
- Knowledge mining engines will assist in generating Test Plan, Estimating Test Effort, developing Test Scenarios and Test Cases from the Knowledge base.
- Knowledge productivity tools would avoid, repeating the same effort spent earlier on a similar kind of application.
- This help increasing productivity by reducing time spent on overall test effort by at least 25%.

Fully Integrated Knowledge Base



Productivity Summary

Effort Estimation	Effort estimator	Near to accurate resource, infrastrucutre requirements and realistic project schedules
Test Plan & Schedule	Test Planner	10-15%
Test Case Generation	Test Case Generator	25-30%
Test Execution	Automation Tools	75-150%
Overall Test Lifecycle	Knowledge management	25-40%

Summary

- Software not tested by formal V&V methods can prove disastrous.
- Software Testing adds another 30% to 50% of the development effort.
- Technology and Tools help you
  - To plan efficiently,
  - Reduce resource, infrastructure and other costs
  - Enforce testing discipline and processes
  - Increase your testing efficiency
  - Increase product stability
  - And ……..……...

Will Add Wings to your

Testing Productivity

Thank You

QWE2002 Paper VT9

Mr. Peter Theunis
(Integri)

Inter-Operability: the testing issue

Key Points

- Inter-operability Testing: Why?
- Inter-operability Testing: How?
- Inter-operability Testing: How to organise?
- Examples.

Presentation Abstract

The objective of attaining systems inter-operability is to ensure that the information systems and applications of organisations form an integrated whole, allowing all applications to exploit the data resources in different information systems. Inter-operability also means that the products in question have passed some form of external conformance test. The cost benefits related to this inter-operability are significant as they impact upon, and encourage common investments, economies of scale, an open supplier market and ultimately lower product price.

This presentation shows how inter-operability testing can be organised, what the pre-requisites are and real world examples.

About the Author

Peter Theunis is currently a Managing-Director of Integri Professional Services in Belgium. Integri Professional Services is an independent testing organisation active in the field of electronic payment systems and telecommunication networks. Prior to Integri Professional Services, he was a Competence Group Manager at AtosOrigin, Belgium. Since 1994 he has been directly involved in software quality and electronic payments. His interests are in electronic commerce, electronic payments, test design, and test automation for security software. He holds a degree in Computer Science from the PHHI of Hasselt, Belgium.

Interoperability - The Testing Issue

Peter Theunis
Integri Professional Services nv
Leuvensesteenweg 325
B-1932 Zaventem - Belgium
email: ptheunis@integri.com
url: www.services.integri.com

© Copyright Integri 2000
(www.integri.com)

INTEGRI PROFESSIONAL SERVICES

Outline

- Introduction
- Inter-Operability Testing : Why ?
- Inter-Operability Testing : How ?
- Inter-Operability Testing : How to organise ?
- Examples
- Conclusion

© Copyright IPS 2002
(www.services.integri.com)

INTEGRI PROFESSIONAL SERVICES

Inter-Operability

Introduction

© Copyright Integri 2000
(www.integri.com)

INTEGRI PROFESSIONAL SERVICES

What means inter-operability?

- Pre-Requisites
  - Open, unambiguous specifications
  - ”Association" or company seeking for open supplier market
    - issuing specifications
    - maintaining specifications and support
    - registration of members
    - security
    - certification and accreditation
  - A need in the market (business case)

© Copyright IPS 2002
(www.services.integri.com)

INTEGRI PROFESSIONAL SERVICES

What means inter-operability?

- Pre-Requisites
- Target
  - cost
    - common investments
    - economies of scale
    - open supplier market
    - lower cost price
  - growth
    - facilitate cross-community
    - facilitate new applications

© Copyright IPS 2002
(www.services.integri.com)

What means inter-operability?

- Pre-Requisites
- Target
- Testing = instrument to reach this target
  - assuring conformance to specifications
  - assuring security ("trust")
  - assuring robustness, scalability, etc…
  - "quality label"

© Copyright IPS 2002
(www.services.integri.com)

INTEGRI PROFESSIONAL SERVICES

What means inter-operability?

- Pre-Requisites
- Target
- Testing = instrument to reach this target
- Certification versus Accreditation
  - Certification :
    - evaluation and testing process of a component
    - by the association or any other accredited body
  - Accreditation :
    - evaluation of certification tools, certification bodies or testing laboratories
    - by the association

© Copyright IPS 2002
(www.services.integri.com)

Inter-Operability

Testing : Why ?

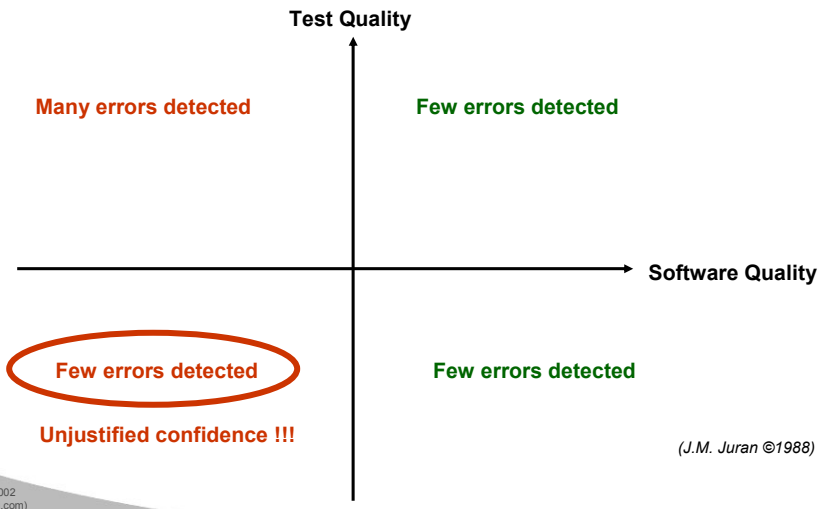
© Copyright Integri 2000
(www.integri.com)

Why Testing?

- Improve quality
- Reduce uncertainty
- Security Assessment
  - Reduce Risk
  - Common Criteria
- Cost of not testing
  - Direct Cost
    - Loss of revenues
    - Fraud
    - Maintenance cost
    - Repair cost
    - Damage claims
  - Indirect Cost
    - Loss of credibility
    - Loss of customer confidence
    - Loss of market share

© Copyright IPS 2002
(www.services.integri.com)

Test quality versus software quality



(J.M. Juran ©1988)

© Copyright IPS 2002
(www.services.integri.com)

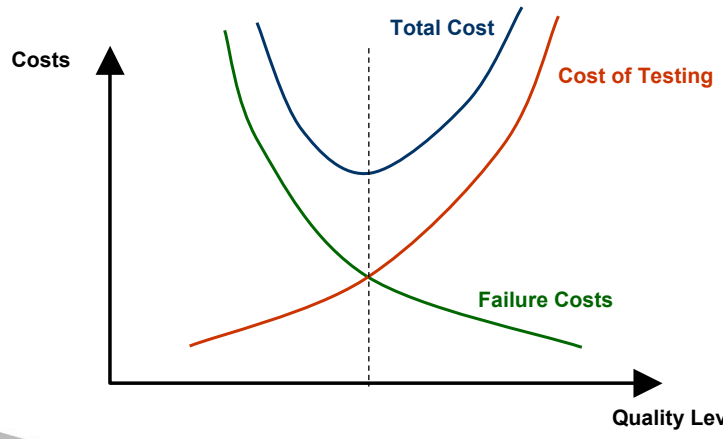
Cost aspects of testing

- testing costs time and money
- testing is never exhaustive
- testing costs depend on type of application (% of project budget)
  - 10% low-cost consumer products
  - 30-50% industry average
  - 80% spaceships, nuclear reactors, etc...
- trade of between failure cost and test cost
- => management decision

© Copyright IPS 2002
(www.services.integri.com)

Cost aspects of testing

- costs versus quality



(J.M. Juran ©1988)

© Copyright IPS 2002
(www.services.integri.com)

Quality is more than testing

- Quality covers
  - specification methodology and standards

© Copyright IPS 2002
(www.services.integri.com)

Quality is more than testing

- Quality covers
  - specification methodology and standards
  - design style

© Copyright IPS 2002
(www.services.integri.com)

INTEGRI PROFESSIONAL SERVICES

Quality is more than testing

- Quality covers
  - specification methodology and standards
  - design style
  - coding standards

© Copyright IPS 2002
(www.services.integri.com)

Quality is more than testing

- Quality covers
  - specification methodology and standards
  - design style
  - coding standards
  - inspection methods
    - desk checking : check systematically using checklists
    - peer review : desk checking by someone else
    - walkthroughs : explanation to group
    - review : discussion in group
    - inspection : formal review

© Copyright IPS 2002
(www.services.integri.com)

Quality is more than testing

- Quality covers
  - specification methodology and standards
  - design style
  - coding standards
  - inspection methods
  - static analysis
    - source code analysis
    - formal verification

© Copyright IPS 2002
(www.services.integri.com)

Quality is more than testing

- Quality covers
  - specification methodology and standards
  - design style
  - coding standards
  - inspection methods
  - static analysis
  - debugging

© Copyright IPS 2002
(www.services.integri.com)

INTEGRI PROFESSIONAL SERVICES

Quality is more than testing

- Quality covers
  - specification methodology and standards
  - design style
  - coding standards
  - inspection methods
  - static analysis
  - debugging
  - **TESTING**

© Copyright IPS 2002
(www.services.integri.com)

Quality is more than testing

- Quality covers
  - specification methodology and standards
  - design style
  - coding standards
  - inspection methods
  - static analysis
  - debugging
  - **TESTING**
- "quality must be built, not tested"

© Copyright IPS 2002
(www.services.integri.com)

INTEGRI PROFESSIONAL SERVICES

Quality is more than testing

- Quality covers
  - specification methodology and standards
  - design style
  - coding standards
  - inspection methods
  - static analysis
  - debugging
  - **TESTING**
- "quality must be built, not tested"
- testing is only 1 aspect to improve quality

© Copyright IPS 2002
(www.services.integri.com)

INTEGRI PROFESSIONAL SERVICES

Inter-Operability

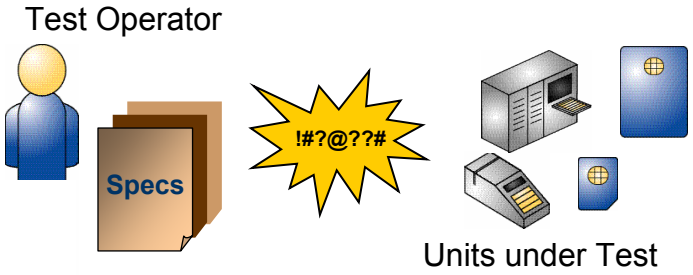
Testing : How ?

© Copyright Integri 2000
(www.integri.com)

INTEGRI PROFESSIONAL SERVICES

Inter-operability Testing

- Problem
  - –testing ?
  - –where do we start ?
  - –where do we end ?



© Copyright IPS 2002
(www.services.integri.com)

What to test?

- Interfaces
  - – inter-operability specifications are strongly interface oriented
  - – different interfaces
    - lower layer communication interfaces
    - application communication interfaces
      - – smartcard-terminal interface
      - – host interfaces
      - – batch interfaces
    - man-machine interface
    - embedded API interfaces
  - – application is tested via the interface

© Copyright IPS 2002
(www.services.integri.com)

What to test?

- Interfaces
- security
  - physical aspects
  - logical aspects

© Copyright IPS 2002
(www.services.integri.com)

What to test?

- Interfaces
- Security
- Robustness

© Copyright IPS 2002
(www.services.integri.com)

What to test?

- Interfaces
- security
- Robustness
- Capacity

© Copyright IPS 2002
(www.services.integri.com)

Why is it difficult to test?

- ambiguous specifications
  - not always available
- dynamic system
  - multiple interfaces
  - event driven
  - unit changes during the test (state)
- security
  - common criteria
- features implemented but not yet used

© Copyright IPS 2002
(www.services.integri.com)

Component testing versus integration testing

- component testing
  - description
    - tested from outside against functional specifications (blackbox testing)
    - UUT (unit under test) put in a test environment
    - trigger UUT with all possible situations and check behavior
  - (dis)advantages
    - + : tests normal and ABNORMAL behavior
    - + : can be automated
    - + : test scope can be very high
    - - : simulated environment

© Copyright IPS 2002
(www.services.integri.com)

Component testing versus integration testing

- component testing
- integration testing
  - description
    - system is built together
    - "real" transactions are executed
  - (dis)advantages
    - + : trust
    - + : live environment
    - - : manual
    - - : poor robustness testing
    - - : poor security testing
    - - : poor stress testing

© Copyright IPS 2002
(www.services.integri.com)

Component testing versus integration testing

- Component testing versus integration testing
  - component testing
  - integration testing
  - conclusion
    - both are needed

© Copyright IPS 2002
(www.services.integri.com)

INTEGRI PROFESSIONAL SERVICES

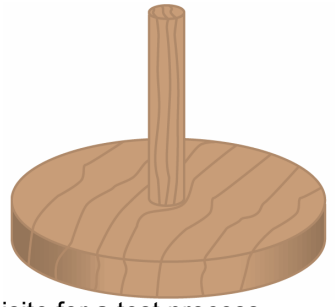
Inter-Operability

Testing : How to Organise ?

© Copyright Integri 2000
(www.integri.com)

INTEGRI PROFESSIONAL SERVICES

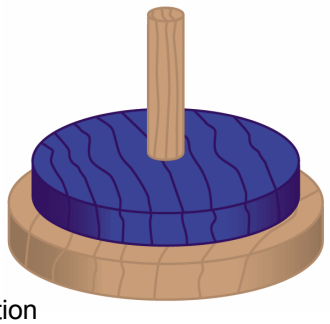
Test Specifications



- definition
  - pre-requisite for a test process
  - bible for testing
- difference
  - official versus proprietary
  - open versus closed

© Copyright IPS 2002
(www.services.integri.com)

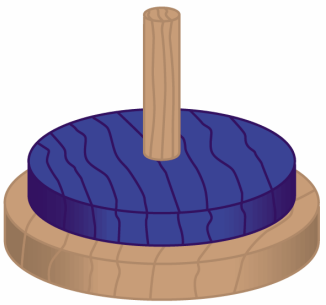
Test Tools



- Provider
  - Association
  - Open Market (independent test tool providers)
  - Open Market + accreditation by the "association"

© Copyright IPS 2002
(www.services.integri.com)

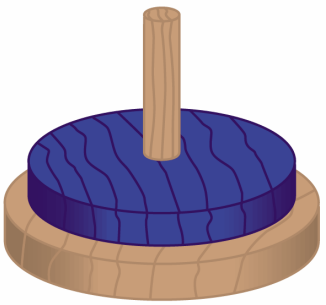
- Test Tools



  - Provider
  - Approach
    - Software/hardware package
    - Used during development
    - used during factory and client acceptance testing
    - “Self-Certification

© Copyright IPS 2002
(www.services.integri.com)

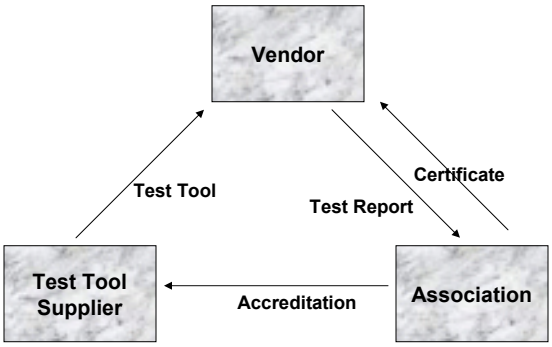
- Test Tools



  - Provider
  - Approach
  - Target
    - Unit Testing
    - normal and abnormal behavior (robustness)
    - stress testing

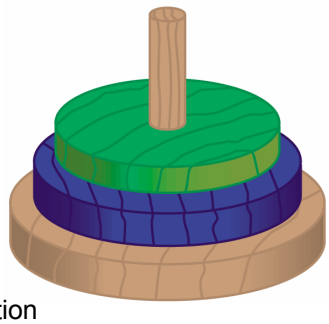
© Copyright IPS 2002
(www.services.integri.com)

- How to organise testing



© Copyright IPS 2002
(www.services.integri.com)

- Test Services

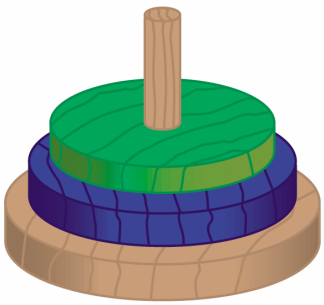


  - Provider
    - Association
    - independent test companies)

© Copyright IPS 2002
(www.services.integri.com)

How to organise Testing

- Test Services

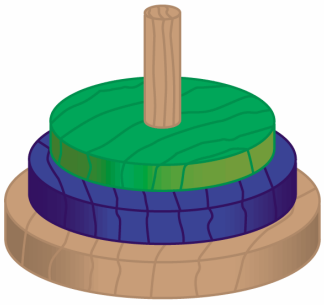


  - Provider
  - Approach
    - Co-operative
    - Pragmatic

© Copyright IPS 2002
(www.services.integri.com)

How to organise Testing

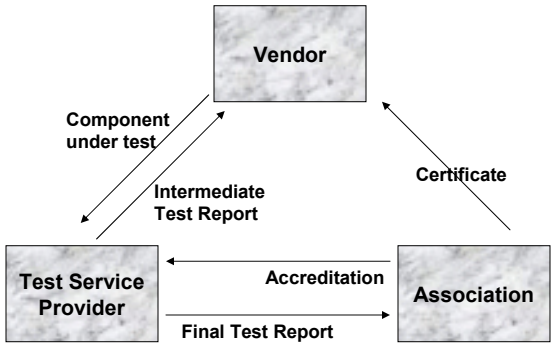
- Test Services



  - Provider
  - Approach
  - Target
    - help to assure quality and conformance
    - outsourcing part of the quality assurance

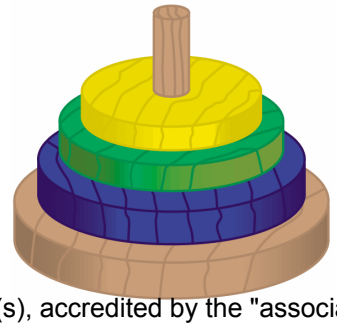
© Copyright IPS 2002
(www.services.integri.com)

How to organise testing



© Copyright IPS 2002
(www.services.integri.com)

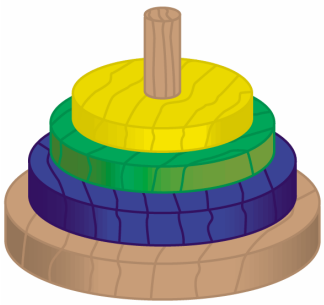
Test Lab



- Provider
  - test lab(s), accredited by the "association"
  - almost "legal" status

© Copyright IPS 2002
(www.services.integri.com)

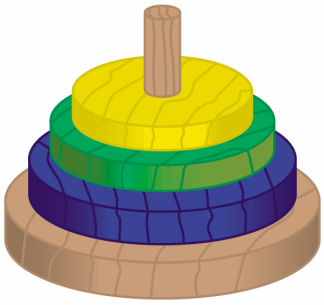
- Test Lab



  - Provider
  - Approach
    - very strict rules (based on ISO standards)
    - pass/fail

© Copyright IPS 2002
(www.services.integri.com)

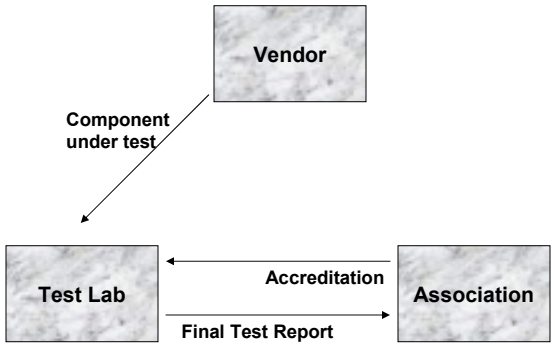
- Test Lab



  - Provider
  - Approach
  - Target
    - accreditation
    - no debugging, no support

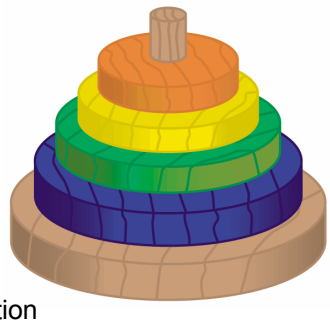
© Copyright IPS 2002
(www.services.integri.com)

- How to organise testing



© Copyright IPS 2002
(www.services.integri.com)

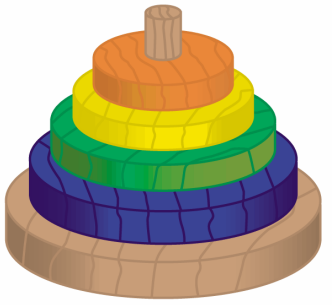
- Inter-Operability Demonstrator



  - Provider
    - Association

© Copyright IPS 2002
(www.services.integri.com)

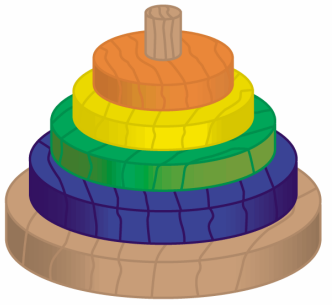
- Inter-Operability Demonstrator



  - Provider
  - Approach
    - hard/soft available from different suppliers
    - collaborative between all players

© Copyright IPS 2002
(www.services.integri.com)

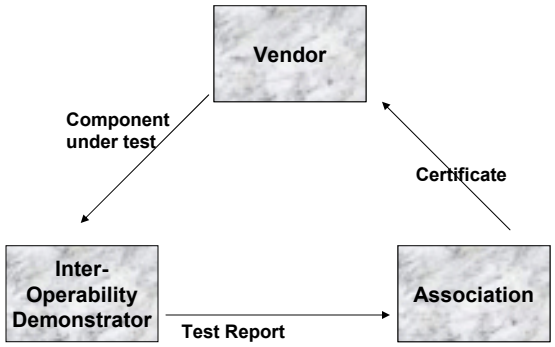
- Inter-Operability Demonstrator



  - Provider
  - Approach
  - Target
    - NOT unit testing, NOT robustness
    - integration testing
    - end-to-end testing

© Copyright IPS 2002
(www.services.integri.com)

- How to organise testing



© Copyright IPS 2002
(www.services.integri.com)

INTEGRI PROFESSIONAL SERVICES

Inter-Operability

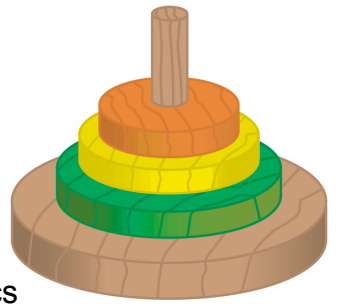
Examples

© Copyright Integri 2000
(www.integri.com)

INTEGRI PROFESSIONAL SERVICES

Real World examples

- POS Terminal Certification by EPCI

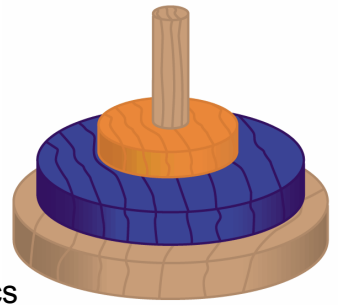


  - Test Specs
  - Test Services
    - Pre-certification
  - Test Lab
    - Certification Testing
  - Inter-Operability Demonstrator
    - Non-disturbance testing

© Copyright IPS 2002
(www.services.integri.com)

Real World examples

- Forecourt certification by IFSF

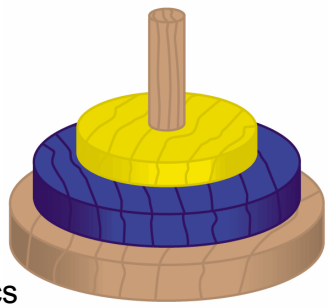


  - Test Specs
  - Test Tools
    - Self-Certification testing
  - Inter-Operability Demonstrator
    - End-to-end testing

© Copyright IPS 2002
(www.services.integri.com)

Real World examples

- EMV

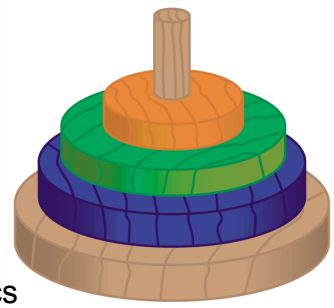


  - Test Specs
  - Test Tools
    - Pre-certification tools
  - Test Labo
    - Certification testing in accredited labo

© Copyright IPS 2002
(www.services.integri.com)

Real World examples

- Europay Authorisation and Clearing Interfaces



  - Test Specs
  - Test Tools
    - Simulators for functional testing
  - Test Services
    - Acceptance Testing
  - Inter-Operability Demonstrator
    - End-to-end testing

© Copyright IPS 2002
(www.services.integri.com)

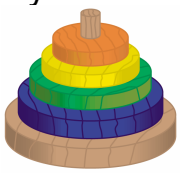
Inter-Operability

Conclusion

© Copyright Integri 2000
(www.integri.com)

Conclusion

- testing is critical to obtain inter-operability
  - security
  - confidence and trust
  - costs
- test methodology
  - component testing (blackbox, test environment)
  - system testing (integrated)
- different layers



  - each of the layers fulfills a given need
  - certification and accreditation issue

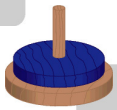
© Copyright IPS 2002
(www.services.integri.com)

Integri Professional Services Service Portfolio

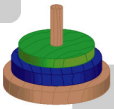
Consulting & Training

TARGET MARKET

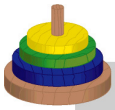
- Electronic Payment
- Smart Card Applications
- Mobile Application
- Security
- Forecourt Devices



Test Tool Development



Test Services



Certification Test Lab

Inter-operability Test Centre

© Copyright IPS 2002
(www.services.integri.com)

INTEGRI PROFESSIONAL SERVICES

Questions?

© Copyright Integri 2000
(www.integri.com)

INTEGRI PROFESSIONAL SERVICES

QWE2002 Paper VT10

Mr. Michel Kalis
(Ubizen Aethis)

How to monitor the Quality of Service of your business infrastructure

Key Points

- To be supplied.

Presentation Abstract

During this presentation different kind of performance and availability issues from which networked systems and infrastructure suffer will be exposed through a high-level overview of encountered problems and the description of concepts and technologies used to pinpoint and solve them.

This will be illustrated by examples and products.

About the Author

To be supplied.

Ubizen AETHIS

How to monitor the

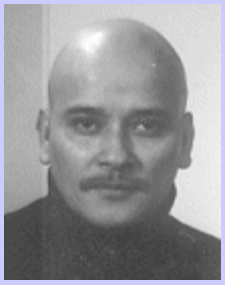
Quality of Service

of your business infrastructure

Abstract

During this presentation different kind of **performance** and **availability** issues from which networked systems and infrastructure suffer will be exposed through a high-level overview of encountered problems and the description of **concepts** and **technologies** used to pinpoint and solve them.

This will be illustrated by examples and products.



QWE2002 Paper VT12

Mr. Eddy Lentze
(SQS)

Configuration Management Improvement Method (CMIM)

Presentation Abstract

-The importance of CM in existing improvement methods
Reach level 2/3 SEI and/or level 2/3 Bootstrap.

-CM in CMM (SEI)
Configuration Management specific goals & practices in CMM(I)

-CM in Bootstrap (Bootstrap-Institute)
Configuration Management specific goals & base practices in Bootstrap

-CM aspects in generic or management goals & practices
CM aspects in the generic practices of CMM(I)
CM aspects in the management practices of Bootstrap

-Types of (CM) assessments
CM Self Assessments
CM Quick Assessment
CM Focussed Assessment
Complete Assessment (CMM / Bootstrap)

-The CM Assessment & Improvement Method (CMIM): An independent and broad method to assess & improve CM processes & functions. The method is able to link to any SPI method.

About the Author

- Experienced software engineer (since 1987)
- Former IT Manager Dutch Ministery of Defense
- Implemented, controlled and improved several IT Service management processes
- Seven years experience in implementing, controlling and improving CM
- Certified Bootstrap Assessor
- Development of the (Software) CM Assessment & Improvement Method (CMIM)

Eddy Lentze

Configuration Management Improvement Method (CMIM)

Quality Week Europe
Brussels
March 2002

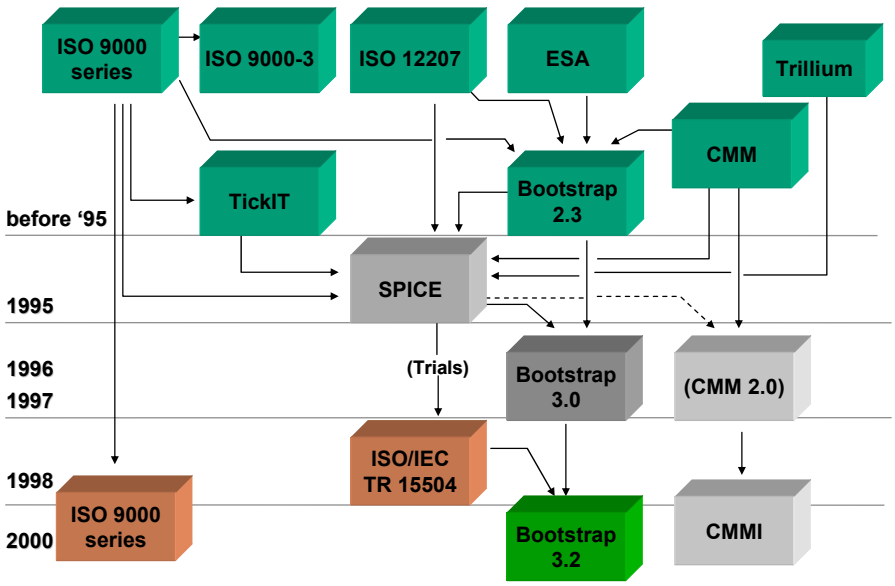
Agenda

- **Focus on improvement**
- **Improvement & Assessment methods**
- **CM Importance for SPI methods**
- **The basis of improvement projects: Assessments**
- **CMIM: CM Improvement Model**
- **CMIM Assessment model**
- **CMIM Assessment scheme**
- **CMIM Calculated profiles**

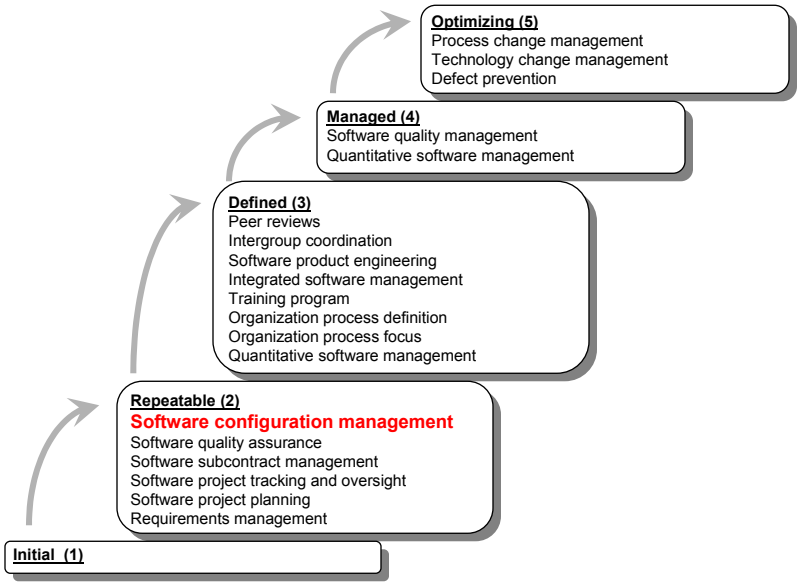
Focus on improvements

- **CM is a "hot" topic**
- **Companies focussed on CM**
  - Software Process improvement projects
    - Reach CMM level 2/3 process maturity
    - Reach Bootstrap level 2/3 process capability
  - Time to market improvement
  - Product quality improvement

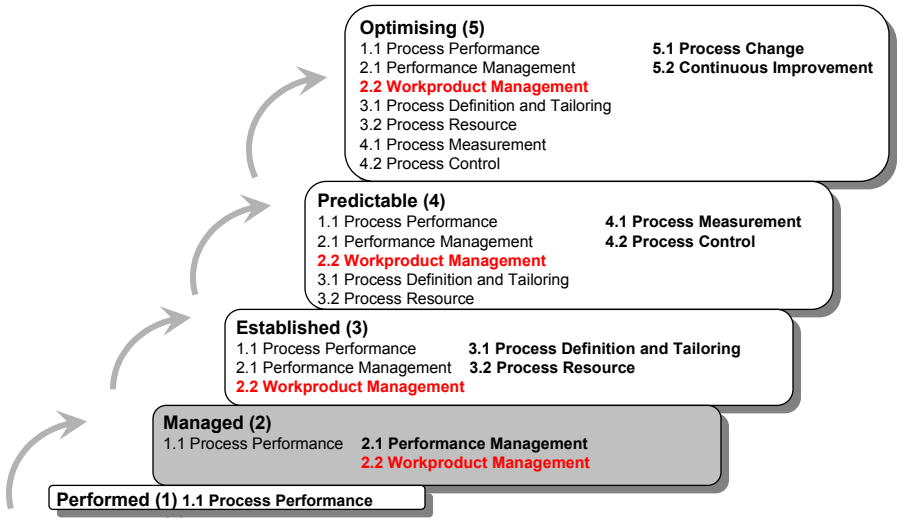
Improvement & assessment methods



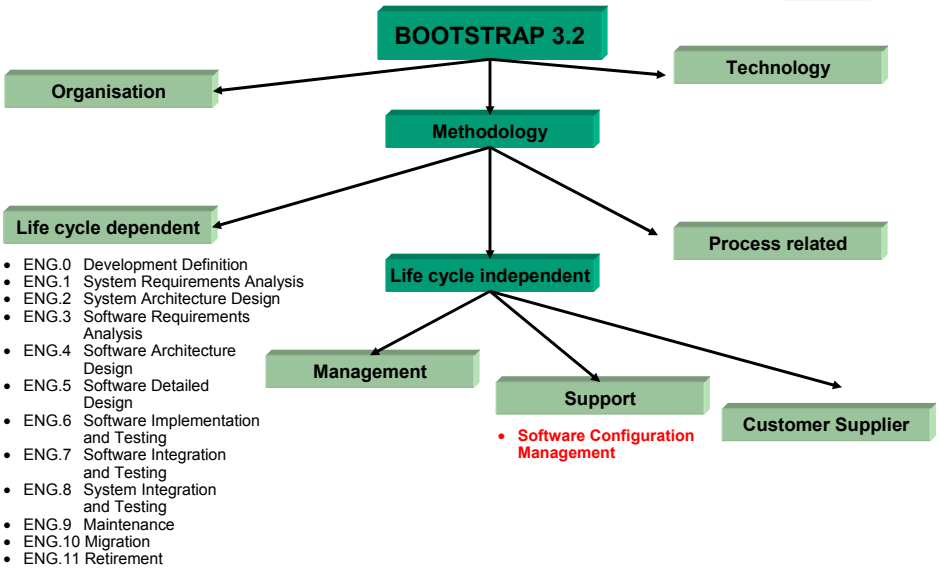
CMM Key Process Areas



Bootstrap Management Practices



The Bootstrap Process Model V3.2



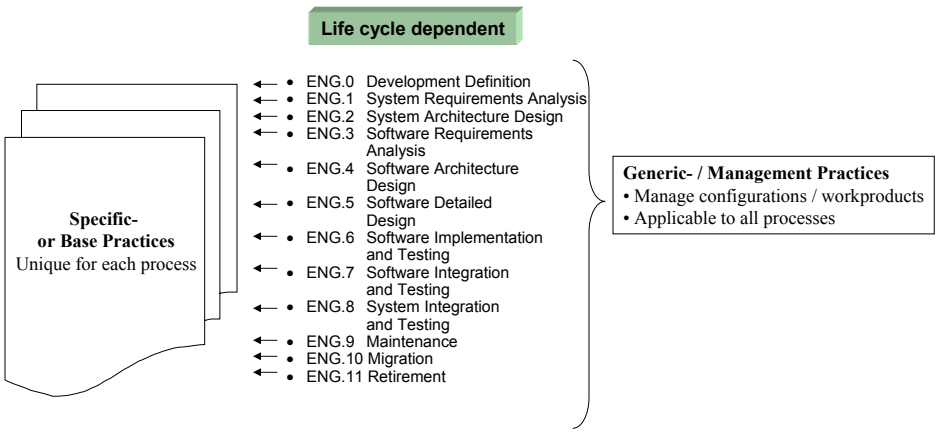
- ENG.0 Development Definition
- ENG.1 System Requirements Analysis
- ENG.2 System Architecture Design
- ENG.3 Software Requirements Analysis
- ENG.4 Software Architecture Design
- ENG.5 Software Detailed Design
- ENG.6 Software Implementation and Testing
- ENG.7 Software Integration and Testing
- ENG.8 System Integration and Testing
- ENG.9 Maintenance
- ENG.10 Migration
- ENG.11 Retirement

- Software Configuration Management

SCM Importance for assessment methods

- **Importance of the (S)CM process is high (level 2)**
- **(S)CM functions are embedded in generic goals and practices:**
  - CMM(I): Manage your configurations
  - Bootstrap: Workproduct quality is planned and managed
- **SCM is related to all Software Engineering Processes.**

Assessing Software Engineering Process

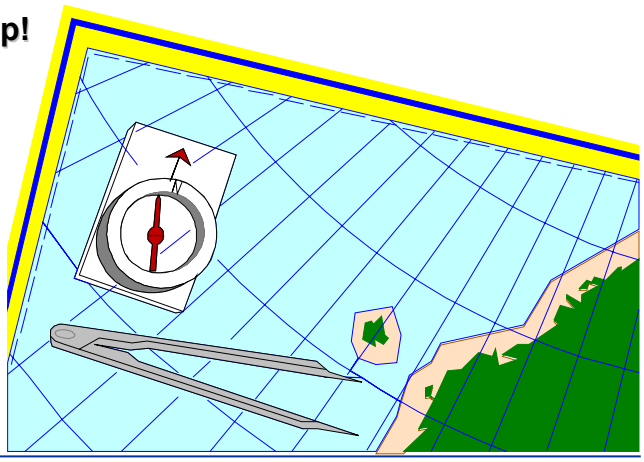


Software Configuration Management is the backbone of the Software Development Process. *(Software Technology Support Center, US Airforce)*

Finding the way to improve

If you don't know where you are ..

a map won't help!



Basis of improvement projects

- **Conducting an assessment**
  - Self-assessments
  - Quick SCCM assessment
  - Focussed SCCM assessment
  - Complete assessment (ISO, CMM or Bootstrap)

Self assessments

- **Self-assessments**
  - Performed by internal personnel
  - Using self-assessment guidelines and/or tools

Quick assessments

- **Quick SCCM assessment**
  - Performed by SCCM assessors
  - Scope of the assessment:
    Software Producing Unit and / or one or more projects
  - Provides raw overall profiles (ISO, CMM, Bootstrap)
  - First step in SCCM process assessment and improvement

Focussed assessments

- **Focussed SCCM assessment CMIM**
  - Performed by SCCM assessors
  - Provides accurate profiles of the SCCM process/functions in all Software Engineering Processes (ISO, CMM, Bootstrap)
  - Scope of the assessment:
    Software Producing Unit and /or one or more projects
  - Performed
    - together with or in continuation of a complete assessment
    - for monitoring the success of improvement actions

Complete assessments

- **Complete assessment (ISO, CMM or Bootstrap)**
  - Performed by certified assessors
  - Provides an independent assessment that covers all main life-cycle processes
  - Scope of the assessment:
    Software Producing Unit and / or one or more projects

CMIM: CM Improvement Method

- **Objectives CMIM:**
  - Corporate, process and tool level
  - Process & generic assessments
  - Compliant with all standards (SPICE, ISO / TR 15504, CMM, Bootstrap,etc)
  - Calculates SPICE, CMM & Bootstrap Capability levels

CMIM: Assessment Scheme

DEMO CMIM Database

CMIM: Assessment Reference Structure

DEMO CMIM Database

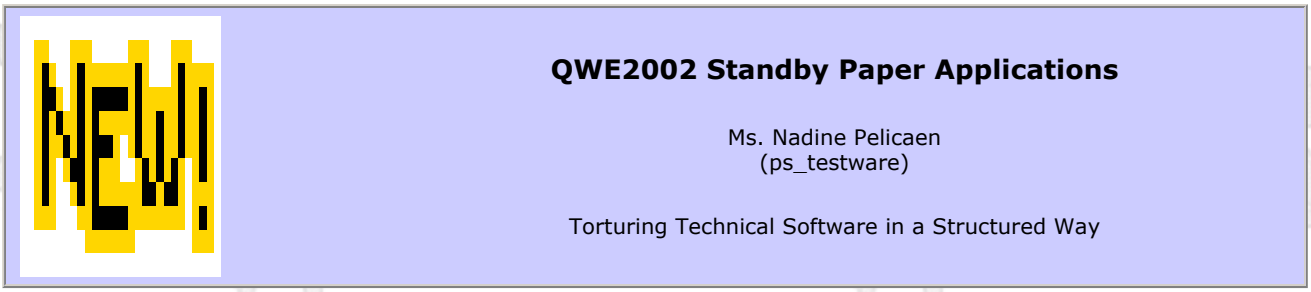
CMIM: Calculated Profiles

DEMO CMIM Database

The End

Configuration Management Improvement Method (CMIM)

Quality Week Europe
Brussels
March 2002



QWE2002 Standby Paper Applications

Ms. Nadine Pelicaen
(ps_testware)

Torturing Technical Software in a Structured Way

Key Points

- Changes in test methodology for testing technical software.
- Embedded systems, security software, real-time systems, telecom and networking software.
- Experiences, tools, techniques and standards for testing technical software.

Presentation Abstract

When testing technical software, you have to be aware of certain difficulties that come with the territory. In this paper, we reflect on the lessons learned when testing software in the area of embedded systems, real-time systems, networking and telecom software, and security issues. When trying to pinpoint what makes it hard, we talk about the special nature of the software under test and the problems that you, as a tester, have to cope with. Our own experiences in this field lead us to an adaptation of the test methodology, where non-intrusive testing, additional quality attributes and special tester profiles are of major importance. Hints and tips are given on how to implement structured software testing in high-tech environments, where there is lack of control and often zero-defects is demanded. Some tools and techniques are discussed, as well as testing against specific standards, especially for telecom and security.

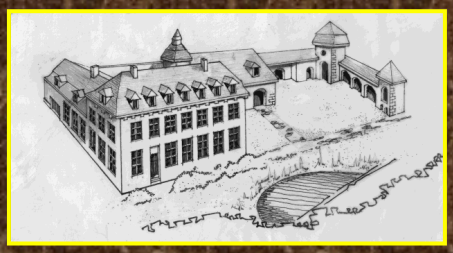
About the Author

Nadine Pelicaen is active as a Test Consultant for structured software testing at ps_testware. She has experience with performing Test Assessments for large projects and she is involved in doing research for the ps_testware competence center of test methodology. Nadine has a university degree in informatics and more than 12 years experience in the IT sector. She has worked in different functions in technical IT environments: software developer, QA engineer and manager of a software department.

Torturing technical software in a structured way

Quality Week Europe 2002

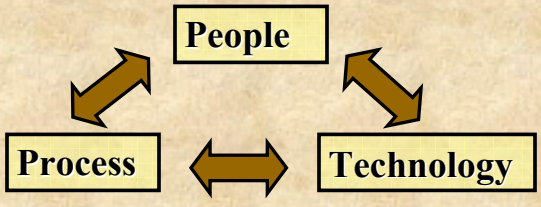
Copyright © 2002 ps_testware



Nadine Pelicaen
ps_testware

Agenda

- Introduction
- Process
- Technology
- People
- Conclusions



Copyright © 2002 ps_testware - Nadine Pelicaen — 2

ps_testWare
Software Testing Services

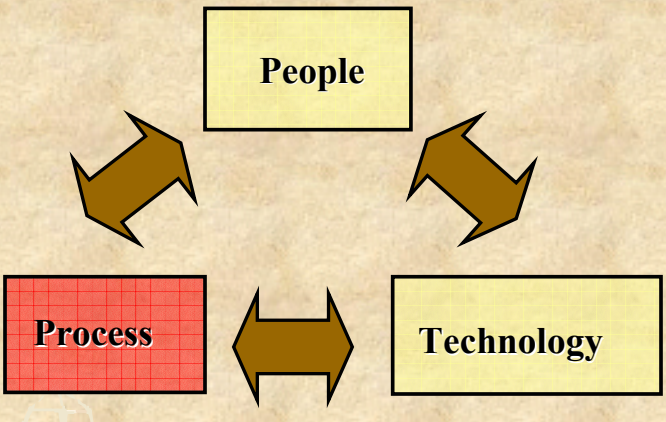
Introduction

- **Scope**
  - Embedded systems
  - Real-time systems
  - Networking and telecom
- **What makes it so special?**
  - High tech environment
  - Zero tolerance
  - Lack of control
  - Heisenberg principle
  - Intensive cooperation with developers
  - Simultaneous development of HW and SW

Copyright © 2002 ps_testware - Nadine Pelicaen --- 3

ps_testware
Software Testing Services

Agenda



Copyright © 2002 ps_testware - Nadine Pelicaen --- 4

ps_testware
Software Testing Services

Process: Quality attributes (1/3)

Classic Quality Attributes	Functionality, performance, volume, usability…
Response time	Real-time
Scalability	Hardware + software
Graceful degradation	e.g. traffic light
Robustness	Always specified behavior
Footprint	Code size limitations
Endurance	Tests for hours or days
Security	Vulnerabilities, exposures,…

Copyright © 2002 ps_testware - Nadine Pelicaen — 5

ps_testware Software Testing Services

Process: Quality attributes (2/3)

Conformance to standards	Telecom, networking, …
Interoperability	Working together
Recoverability	Time to get it running again
Safety	Hazard prevention
Installability	How easy is it?
MTBF	Benchmark failure rate
Configuration	Run under different configurations

Copyright © 2002 ps_testware - Nadine Pelicaen — 6

ps_testware Software Testing Services

Process: Quality attributes (3/3)

Multithread safety	Deadlocks
Documentation	Technical manuals
Availability	Uptime
Upgradability	Hooks to handle upgrades
…	…

Copyright © 2002 ps_testware - Nadine Pelicaen — 7

ps_testware Software Testing Services

Process: Software life cycle

- **RAD, Iterative Incremental, RUP,…**
- **Flexible Test Planning**
  - **No luxury situation**
  - **Testing process: Quickly and efficiently**
- **Risk assessment**
  - **Put priorities**
- **Test automation**
  - **Regression testing vs Maintenance**
- **Test exit criteria**
- **Develop a killer test**

Copyright © 2002 ps_testware - Nadine Pelicaen — 8

ps_testware Software Testing Services

Process: Prototyping

- Proof of concept testing
- Check out the promises
- Test environment
- Test strategy

Copyright © 2002 ps_testware - Nadine Pelicaen — 9

ps_testWare
Software Testing Services

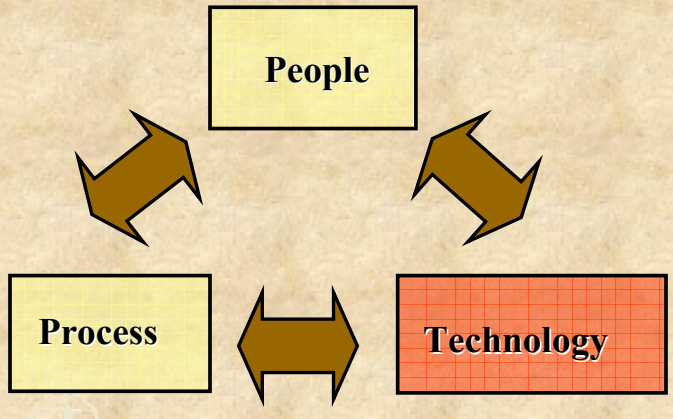
Process: Test strategy

- Incorporate all the quality attributes
- Test planning
  - Flexible
  - Early integration planning
  - Emphasis on unit and modular testing
- Benchmarking
- White box testing
- Structured approach + guerrilla testing
- Testing is not enough
- Acceptance test: FAT + FIT

Copyright © 2002 ps_testware - Nadine Pelicaen — 10

ps_testWare
Software Testing Services

Agenda

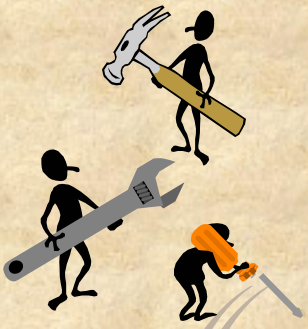


Copyright © 2002 ps_testware - Nadine Pelicaen — 11

ps_testWare
Software Testing Services

Technology : Environment

- Dedicated hardware
- Simulators
- OS's
- Languages for development
- Languages for testing
  - Perl
  - Tcl/Tk
  - Expect
- Lack of control



Copyright © 2002 ps_testware - Nadine Pelicaen — 12

ps_testWare
Software Testing Services

Technology : Test development (1/3)

- Open source tools and environments
  - Testers need some basic knowledge
  - Development environment
  - Free stuff for testers
    - Security: manuals, tools, ...
    - Networking: RFC's, test suites, ...
- Memory analysis tools
- Code coverage tools

Copyright © 2002 ps_testware - Nadine Pelicaen — 13

ps_testware
Software Testing Services

Technology : Test development (2/3)

- Software development needed to do the testing
- Automation
  - Commercial tools
  - Own Test Automation Framework
- Bootstrap problem
  - Simultaneous development
  - Creative solutions: stubs…

Copyright © 2002 ps_testware - Nadine Pelicaen — 14

ps_testware
Software Testing Services

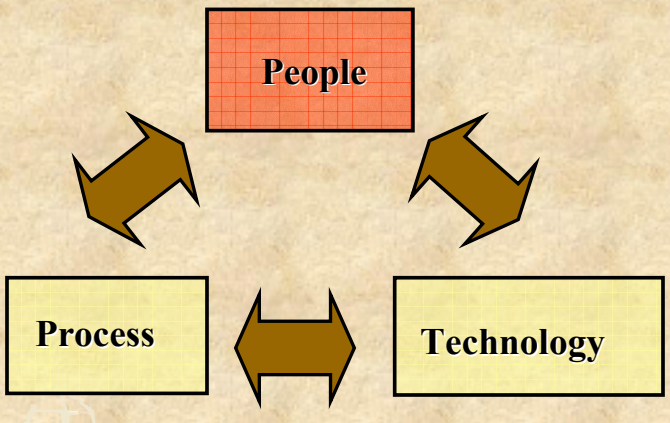
Technology : Test development (3/3)

- Intrusiveness
  - Influence synchronization, interrupts, …
  - No impact: timing, performance, extra code
- Special testing techniques
  - E.g. FSM, closed loop, evolutionary testing, …

Copyright © 2002 ps_testware - Nadine Pelicaen — 15

ps_testWare
Software Testing Services

Agenda



Copyright © 2002 ps_testware - Nadine Pelicaen — 16

ps_testWare
Software Testing Services

Tester : role

- Close relationship with developers
- Team work, but independence
- Roles and responsilibities
- Own testing resources
- Configuration management
- Staffing ratio
- Problem: tester retention

Copyright © 2002 ps_testware - Nadine Pelicaen — 17

ps_testWare
Software Testing Services

Tester : skills

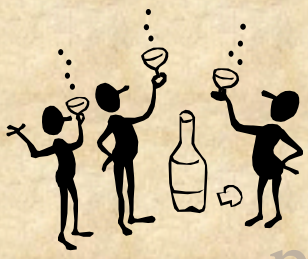
- Expected know-how
  - Product knowledge
  - Technical background
  - New technologies
- Pragmatic
- Team players
- Handling change
- Get the big picture
- Destructive attitude
- Accuracy

Copyright © 2002 ps_testware - Nadine Pelicaen — 18

ps_testWare
Software Testing Services

Conclusions

- Changes in standard test methodology
- Keep an eye on available tools and techniques
- Get the right people
- Close synergy between process – people - technology



ps_testWare
Software Testing Services

Copyright © 2002 ps_testware - Nadine Pelicaen — 19

ps_testWare
Software Testing Services

Questions

Copyright © 2002 ps_testware - Nadine Pelicaen — 20

ps_testware
Software Testing Services
Tiensesteenweg 343
B-3010 Leuven
Tel.: +32 (16) 35.93.80
Fax: +32 (16) 35.93.88
e-mail: info@pstestware.com
http://www.pstestware.com
Copyright © 2001 ps_testware – Eurostar - Nadine Pelicaen — 21

QWE2002 Standby Paper Management

Mr. Olivier Denoo
(ps_testware)

Weight Before It Is Too Late!

Key Points

- Metrics, risk management.
- Progress and process management.
- Efficiency, effectiveness, improvement.

Presentation Abstract

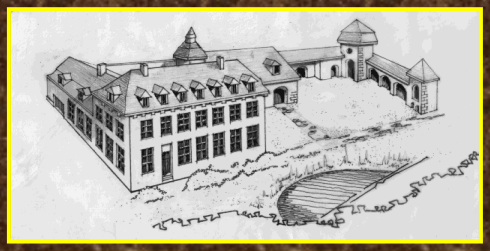
Whats wrong with most of the test processes we have to deal with?
The answer is simple and invariably the same : the absence of metrics! Testing is measuring! Professional testers keep on saying that for years. But why measuring? Measuring what? And measuring how? We will illustrate these concepts and answer these questions from a simple example we had to face a while ago at one of our large (telecom) customers.
Putting in place quality metrics as well as measuring efficiency and effectiveness of the complete process helped them a great deal in understanding their lacks, managing their process and restoring confidence in their major supplier.

About the Author

Olivier Denoo is a consultant active in the Business of Structured Software Testing since 1997. He was the key-developer of ps_testware's Y2K testing techniques. Nowadays, he still is a highly respected trainer of this methodology. Continuously looking for new challenges, Olivier started to investigate the possibilities of website and e-commerce testing and became in charge of ps_testware's e-commerce/WWW-testing knowledge base. Currently, he is working in a project at one of the largest Belgian Telecom companies.

Copyright © 2002 ps_testware – Wieght Before it is too Late – Olivier Denoo - 1

Weight Before it is too Late!



Tracking & Managing the Test Process

ps_testWare
Software Testing Services

Weight Before it is too Late

How to measure the progress of a test project in order to manage it

Olivier Denoo
Sr Test consultant

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 2

Agenda

- Introduction
- Basis for measuring
  - A test process is similar to a production process
  - Managing a production process by using the Theory of Constraints (TOC)
  - Translation of the TOC for measuring a test process
- Implementation of TOC
  - What should be measured in order to manage a test process ?
  - What should be registered in order to measure a test process ?
- Introduction to ps_testware

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 3

ps_testWare
Software Testing Services

Introduction

Measure the progress ?

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 4

A Real Case Study

- Context
  - Large TelCo operator
  - Third party (core billing) application
  - Close but very damaged relationship with the supplier (a big integrator)
  - Very constrained time boxes
  - Perception was : quality is very poor!
  - Fast-moving ever-changing Requirements
  - Wrong focus and large test overlap due to lack of confidence

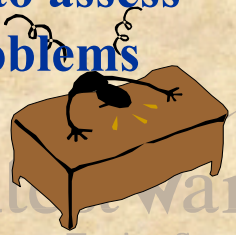
Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 5

ps_testware
Software Testing Services

A State of (bad) Practice

- No reference was defined
- No figures were available
  - Coverage was not analysed
  - Effectiveness was not measured
- Estimated figures were used

Management was unable to assess progress or evaluate problems properly.

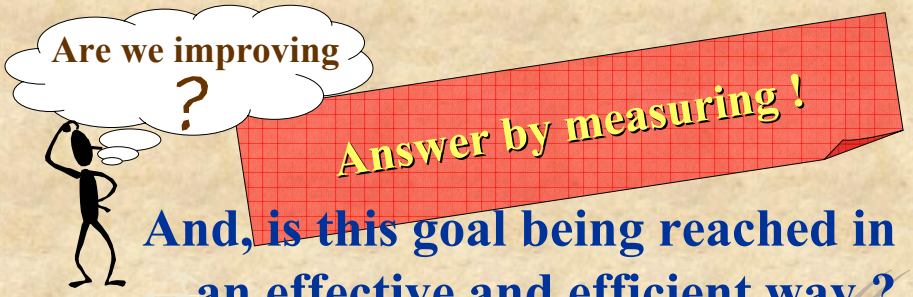


Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 6

ps_testware
Software Testing Services

The Goal of Testing ?

Improve the quality of the system under test by solving found defects



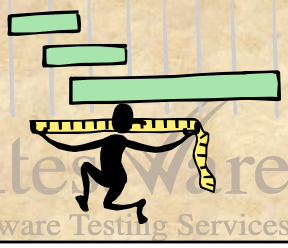
And, is this goal being reached in an effective and efficient way ?

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 7

ps_testware Software Testing Services

Why Measuring ?

- **To assess whether the Goal is achieved, by comparing the present situation with a measurable reference.**
- **To be able to manage and steer the testing process!**



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 8

ps_testware Software Testing Services

What Do You Need to Know ?

- How much of the test goal is reached ?
- Is the test goal reached as planned ?
- Could the project be done more efficiently ?
- What is our test quality ?
- Are the found defects solved ?
- What did we learn ?

Not measuring = Not knowing

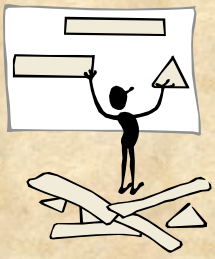
ps_testWare
Software Testing Services

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 9

ps_testWare
Software Testing Services

Basis for Measuring

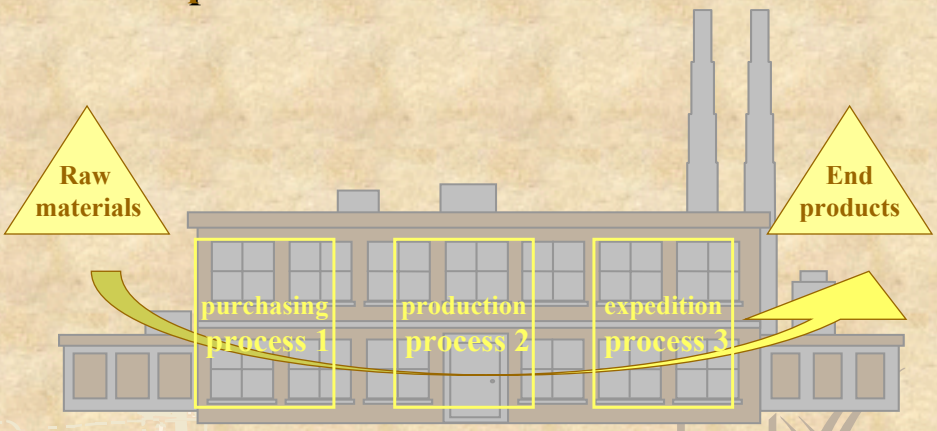
A test process is similar to a production process



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 10

A production process model

- A company is a chain
- Each department is a link in the chain



ps_testWare Software Testing Services

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 11

Focussing Process of Throughput World

- Bottleneck theory:
  1. Locate the bottleneck(s)
  2. Determine how to exploit the bottleneck(s) to the limit
  3. Make all other activities subordinate to the above solution
  4. Elevate the bottleneck(s)
  5. When the bottleneck(s) is(are) removed, go back to point 1

ps_testWare Software Testing Services

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 12

TOC Measures a Process Using 3 Dimensions

- **Inventory:**
  - The work-in-process in between each production process
- **Operational Expense:**
  - The cost of converting inventory into throughput
- **Throughput:**
  - The pace at which the system produces end products

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 13

ps_testware Software Testing Services

How to Optimise a Production Process ?

By reducing operational expense and reducing inventory while increasing throughput

Inventory
Operational Expense

Throughput

At the same time !

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 14

ps_testware Software Testing Services

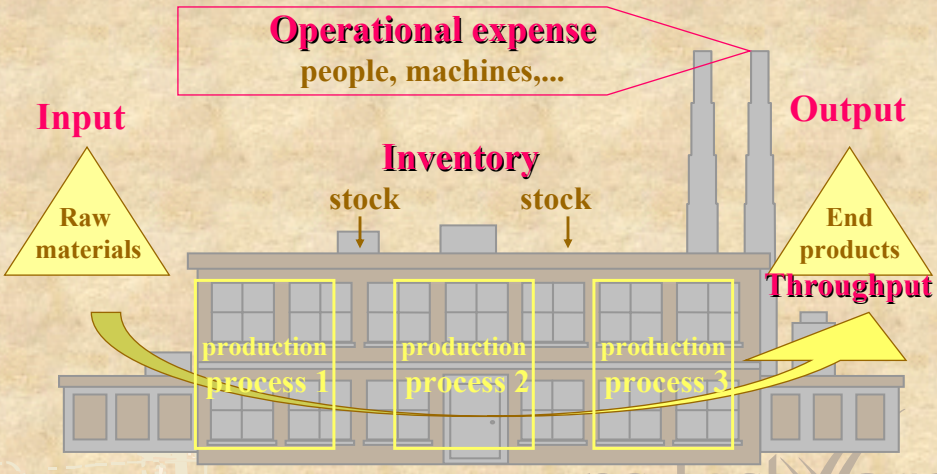
ps_testWare
Software Testing Services

Basis for Measuring

Translation of the TOC for measuring a test process

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 15

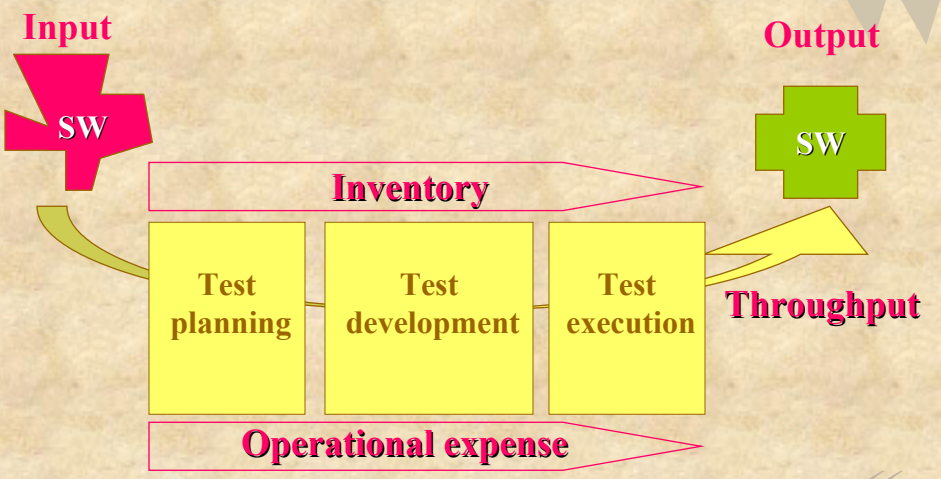
A Production Process



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 16

ps_testWare
Software Testing Services

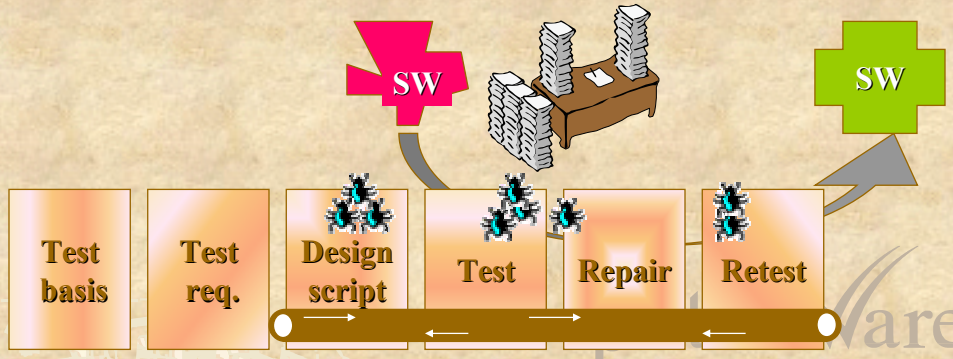
A Test Process



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 17
ps_testWare
Software Testing Services

Inventory

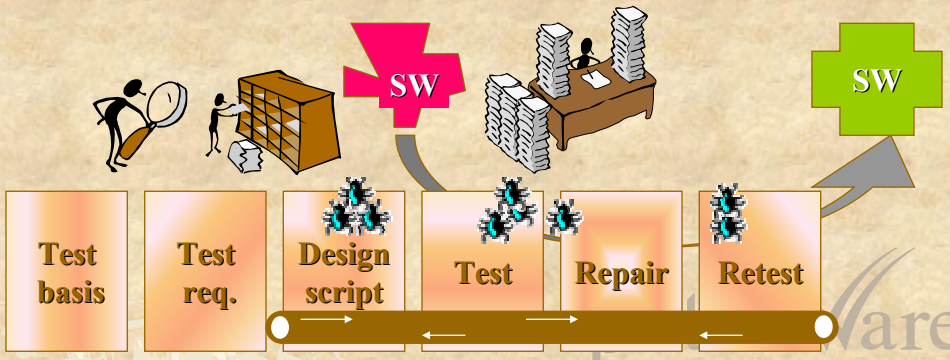
The number of defects found in the system under test.



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 18
Software Testing Services

Operational Expense

The cost of converting found defects into solved defects.

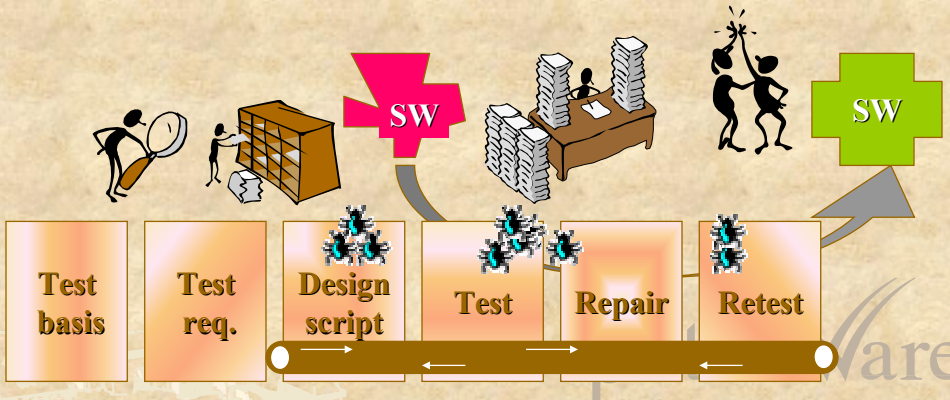


Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 19

Software Testing Services

Throughput

The pace at which the test process generates solved defects.



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 20

Software Testing Services

ps_testWare
Software Testing Services

Implementation of TOC

What should be measured in order to manage a test process ?

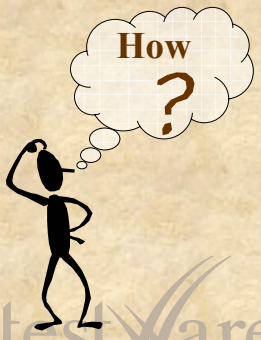
Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 21

What do we Need to Measure ?

Measure

=

Compare with a Reference



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 22

ps_testWare
Software Testing Services

Define your Reference

Test Requirement Hierarchy

Basis

Test Coverage Reference (TCR)

 # Defined test requirements (requirements we want to test) used as a reference against which we measure

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 23

ps_testware Software Testing Services

Measure Goal Achievement

Test Coverage Reference (TCR)

%

Test Design Coverage

Test Execution Coverage

Passed Product

Snapshot of Figures

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 24

ps_testware Software Testing Services

Measure Progress

Work Product Trend

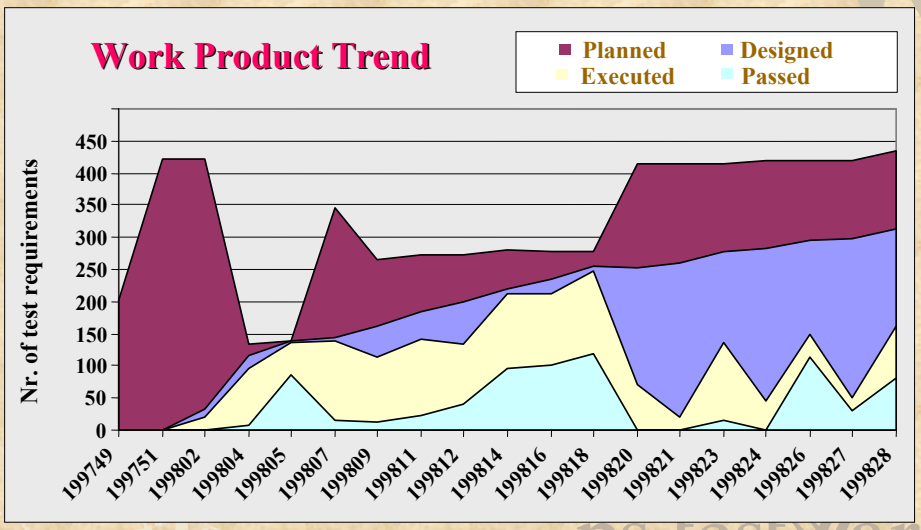
 # Test requirements planned, designed, executed & passed in time

Figures in Time

ps_testware Software Testing Services

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 25

Analyse Measurement



ps_testware Software Testing Services

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 26

Measure Effectiveness

Are we doing the right thing ?

Operational Expense

%Test Overlap

Actual hours spent ⟺ Planned hours per test activity

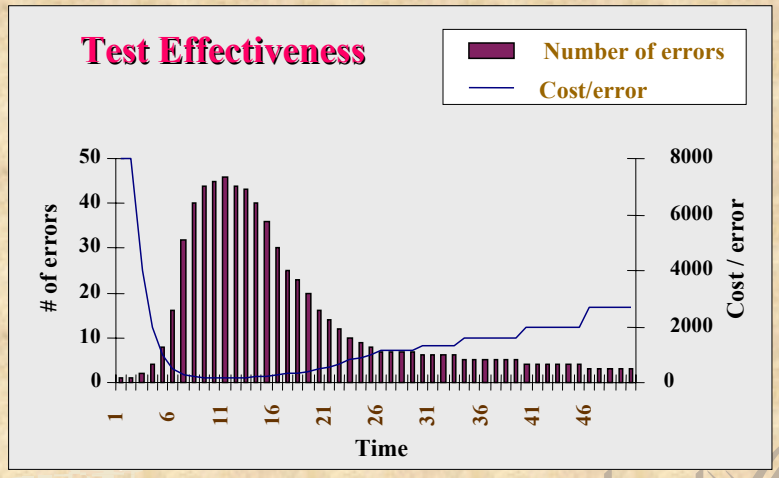
Defects Found

Defect Removal Effectiveness (DRE)

Solved defects ⟺ # Found defects in a release

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 27

Analyse Measurement



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 28

Measure Efficiency

Are we doing the things right ?

- Defect Detection Rate
- Nb of changed Reqs.
- Ratio at which the process finds defects
- Nb of defects timed out
- Test Maintainability
- Test Waste
- Defect Throughput
- Nb of late Reqs.
- Ratio at which the process produces solved defects

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 29

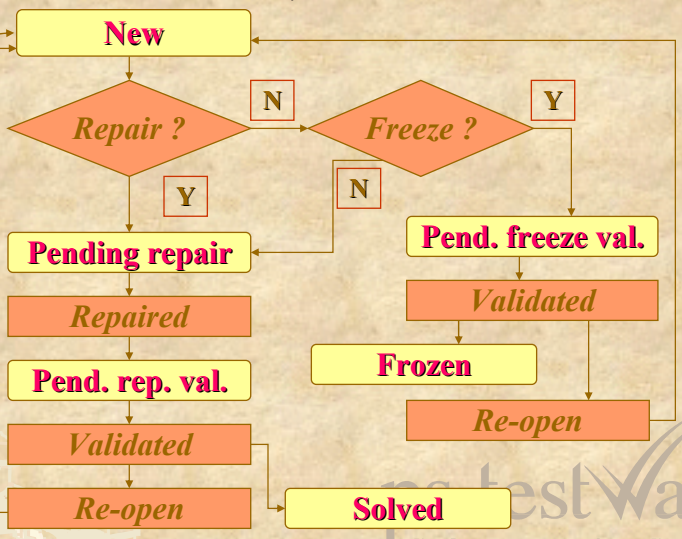
Software Testing Services

Locate bottlenecks

Status Action Workflow

Basis

e.g.



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 30

ps testWare
Software Testing Services

Locate bottlenecks

Status Action Workflow

Basis

Defect Inventory

All not-solved defects

Defect Distribution

 # Defects per status

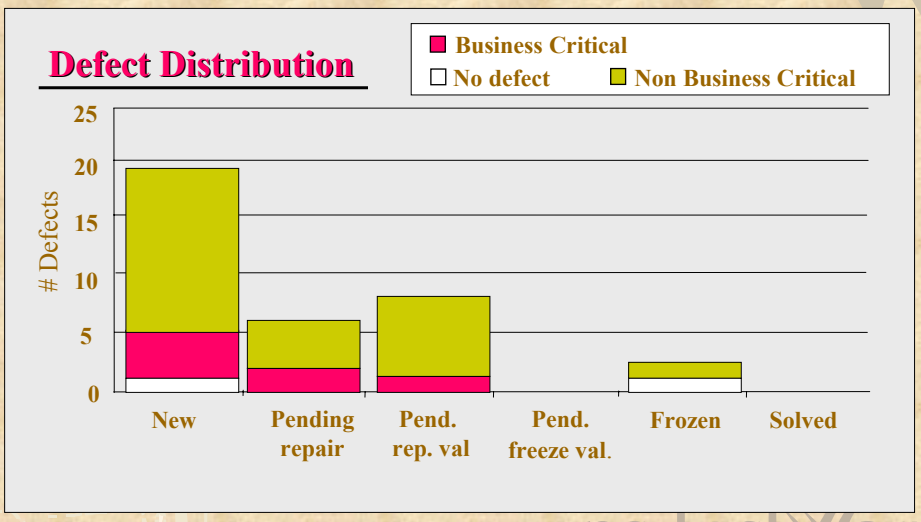
Defect Trend

 # Found and Solved Defects in time

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 31

ps_testWare
Software Testing Services

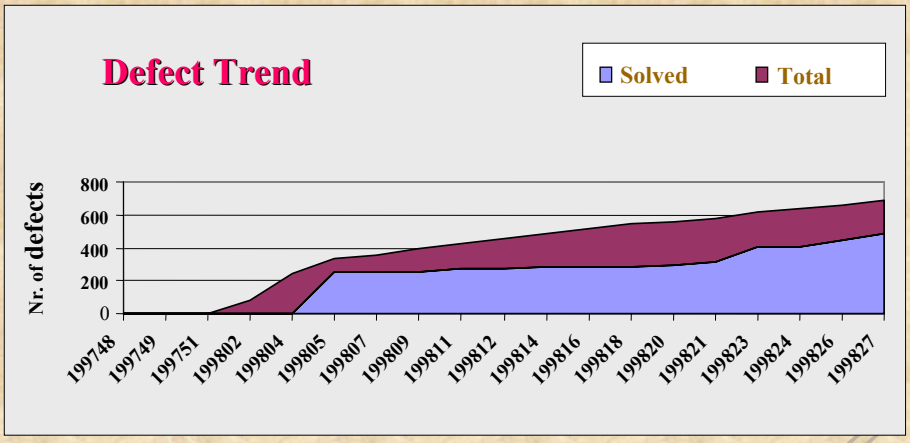
Analyse measurement



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 32

ps_testWare
Software Testing Services

Analyse measurement



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 33

ps_testWare
Software Testing Services

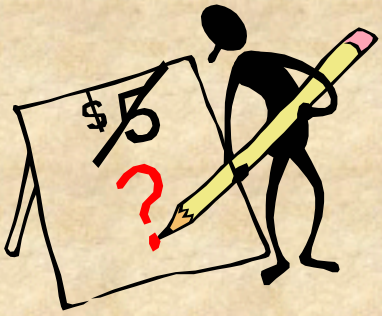
Implementation of TOC

What should be registered in order to measure a test process ?

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 34

Register what ?

- Work Product Tracking
- Defect Tracking
- Resource Tracking



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 35

ps_testWare
Software Testing Services

Work Product Tracking

How much of the system have we tested ?

- # Test requirements
- % Test requirements covered by the planned test procedures
- # Test procedures (weekly):
  - Planned
  - Designed
  - Executed
  - Passed

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 36

ps_testWare
Software Testing Services

Defect Tracking

How many defects have we found ?
How many found defects have we solved ?

 # Defects found

- **Status**
  - New
  - Pending repair
  - Solved ...
- **Severity**
  - Business Critical
  - Non Bus. Critical
  - No defect

Defect Throughput

ps_testware Software Testing Services

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 37

Resource Tracking

How much effort/time have we spent on the process ?

- **# Hours per Test Activity (daily):**
  - Test preparation
  - Test planning
  - Test design
  - Test environment
  - Test execution
  - Test maintenance
  - Test repair
  - Test reporting
  - Test overhead

Operational Expense

ps_testware Software Testing Services

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 38

A Real Case Study: How did it Help?

- Metrics confirmed what was seen in our assessments (supplier and customer sides)
- Management could steer actions based on real facts and figures (no more perception mode)
- Management was able to understand and locate (underground) problems
- Solving organisational problems helped a lot
  - shared views
  - stabilised pace
  - better follow-up
  - Better focus
- Metrics were only a (big) part of the complete solution

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 39

ps_testWare
Software Testing Services

Questions

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 40

References:

- Critical Chain, Eliyahu Goldratt
- Theory of Constraints (TOC) developed by Eliyahu Goldratt
- The Goal, Eliyahu Goldratt & Jeff Cox
- Goal directed Project Management, Andersen, Grude, Haug & Turner

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 41

ps_testWare
Software Testing Services

ps_testWare
Software Testing Services

Tiensesteenweg 343
B-3010 Leuven
Tel.: +32 (16) 35.93.80
Fax: +32 (16) 35.93.88
e-mail: info@pstestware.com
http://www.pstestware.com

Copyright © 1999 ps_testware - Geert Pinxten - <Name Presentation> - 42

ps_testware

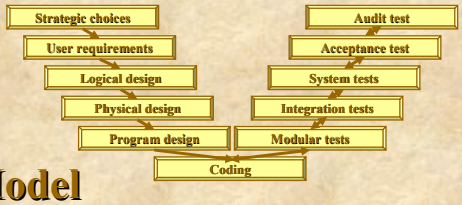
- 6 • Started in 1991
- 5 • 1993: Tools & Technical Services
  - Tool Training
  - Coaching
- 6 • 1995: Methodological Services
  - Consultancy
- 15 • 1996: Software Testing Services
  - Test Assignment
  - Test Plan
  - Test Report
- 25 • 1997: Software Testing Services Suite
  - Test Assessment
  - Y2K Training
- • 1998: PSTI
  - Office @NL
  - Total Outsourcing
  - Partnerships
- 33 • 1999:Testing Hill™ Campus
  - …

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 43

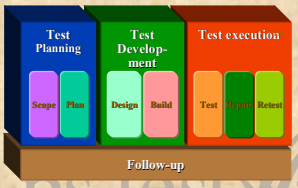
ps_testware Software Testing Services

Our Business

- **Structured Software Testing**
- Methodology



- Implementation Model



Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 44

ps_testware Software Testing Services

Our Services

- Training (see ps_testware institute)
- Coaching
- Consultancy
- Outsourcing (now also Total Outsourcing)

Provided by:

- Test Engineers
- Test Consultants
- Management Consultants

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 45

ps_testWare
Software Testing Services

Our Products

- Test Assessment
- Test Assignment
- Test Plan
- Test Report
- Test Advice
- Test Audit
- Test Pack™ New
- Test Laboratory New
- *Tools* New

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 46

ps_testWare
Software Testing Services

References

- Kredietbank
- Barco Graphics
- Exact Maatwerk
- ING Bank
- Bank Card Company
- Janssen Pharmaceutica
- Tessa
- Europese Raad
- Lernout & Hauspie
- Origin
- Specs
- Gemeentekrediet
- Siemens
- ING 2
- Yokogawa
- Link
- Alcatel Bell
- Mobistar

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 47

Credo

ps_testware's first responsibility goes to the customers who use our services. Our services must be of high quality and must be a reference for our customers. In line with our primary business, Structured Software Testing, we may not indulge in pressure, quantity or quick profit.

We are responsible to our members, the men and women who work with us. Every member must be respected as an individual and must be rewarded personal and fair. We must support our members via a competent management, an adequate working environment and proper working conditions. Our members must have the means to provide and receive feedback, allow them and the organisation to learn continuously. We must support our members in their family responsibilities. Our actions must be just and ethical.

Our final responsibility is to our stockholders. Our business must make a sound profit. We must innovate and continuously improve our methods and techniques. We must develop new services and implement them effective and efficient. We must create reserves to provide for adverse times. Our stockholders must receive a fair return on their investments.

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 48

The Mission

To offer the <u>best solution</u> to quality problems of computer systems by using its <u>test expert knowledge</u> in a <u>professional</u> way.

***<u>Best solution</u>**: the solution that provides the highest contribution.*

***<u>Test expert knowledge</u>**: the intellectual asset of ps_testware, a profound and complete knowledge regarding verification and validation (testing).*

***<u>Professional</u>**: the courage to really provide what has been promised.*

Copyright ©2002ps_testware – Weight Before it is too Late – Olivier Denoo 49

ps_testWare
Software Testing Services

ps_testWare
Software Testing Services

Copyright ©2002 ps_testware – Wight Before it is too Late- Olivier Denoo - 50

QWE2002 Standby Paper Management

Mr. Michael J. Hillelsohn
(Software Performance Systems)

Organizational Performance Engineering: Quality Assurance For The 21st Century

Key Points

- Multi-disciplinary, pro-active Organizational Performance Engineering is the future of quality
- Effective Quality Assurance facilitates the next step in a process instead of waiting to correct
- Fresh approaches to quality assurance are needed to apply sound software engineering principles

Presentation Abstract

In too many organizations, quality assurance is relegated to verification and validation activities and is separate from training and software process improvement organizational elements. Proactive quality assurance is actually a multi-disciplinary set of activities that combine these three organizational elements into organizational performance engineering. The approach being advocated uses techniques from quality assurance and control, training, program management, business process reengineering, and software process improvement disciplines to analyze the organization's practices and behaviors and initiate in-process interventions. The philosophy of organizational performance engineering is based on the notion that people really do like to do things correctly and really hate to go back and rework what they have already completed. It is a pro-active approach that emphasizes preventing problems and defects from occurring.

About the Author

Michael J. Hillelsohn is a Director, Product Assurance at Software Performance Systems (SPS) in Arlington, VA. SPS builds secure e-commerce, case management, and network solutions for government and industry clients. Mr. Hillelsohn is a certified quality professional with more than thirty years of experience doing development, management and performance improvement in software and systems development environments. His multi-disciplinary approach combines quality systems and training expertise to improve the performance of organizations and individuals. Mr. Hillelsohn's process-oriented, performance engineering methods facilitate adoption of external frameworks (CMM, ISO, Baldrige) to improve the quality of organizational products and services.

Quality Week Europe 2002

Organizational Performance Engineering:

Quality Assurance for the 21st Century

Michael J. Hillelsohn

SOFTWARE *PERFORMANCE* SYSTEMS

2011 Crystal Drive Suite 710
Arlington, VA 22202
(703) 797-7707 mhillelsohn@goSPS.com

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Objectives

- Expand your definition of quality assurance to include all sorts of interventions that improve the quality of the products and services that your organization delivers.
- Design an engineering approach to determining how you are going to be most effective in your organization.
- Think about an implementation strategy for pro-actively impacting the performance of people and processes in your organization.

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Quality Assurance Definitions

- (1) *A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or (software work) product conforms to established technical requirements.* (2) *A set of activities designed to evaluate the process by which (software work) products are developed or manufactured.*

IEEE Std 610.12-1990

( ) SW CMM

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Quality Assurance Definitions

- *All the planned and systematic activities implemented within the quality system (organizational procedures, processes, and resources needed to implement quality management) and demonstrated as needed to provide adequate confidence that an entity will fulfill requirements for quality.*

ANSI/ISO/ASQC A8402-1994

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Quality Assurance Definitions

- *The set of support activities (including facilitation, training, measurement, and analysis) needed to provide adequate confidence that processes are established and continuously improved in order to produce products that meet specifications and are fit for use*

Quality Assurance Institute's CQA Study Guide Version 2

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Where We Would Like To Be

- PERFORMANCE - *The activities performed by individuals, groups, and organizations to produce the things they deliver to customers.*

Michael

- ENGINEERING - *The application of a systematic, disciplined, quantifiable approach to structures, machines, products, systems or processes.*

IEEE Std 610.12-1990

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Underlying Principles

People prefer to do a good job rather than a poor job

&

People hate to go back and correct things that they have already completed

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Planning Activities

- Define quality policy and goals
- Participate in preparing management and development plans
- Support life cycle tailoring
- Write Quality Assurance Plan/Quality Plan
- Personnel and task management
- Tracking and oversight of resources, schedule, activities, products

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Policies, Standards, Guidelines

- Write and maintain quality policies
- Research current standards & methods
- Adapt standards/methods to project /organization
- Write implementation guidelines
- Facilitate definition of work instructions
- Train/consult users on implementation
- Tailor standards… as required

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Compliance Verification

- Documents
  - Review standard with developer
  - Conduct in-process reviews
  - Participate in inspections
  - Final compliance check
  - Sign off on deliverable/product
  - Compile metrics

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Compliance Verification

- Design Diagrams/Code
  - Publish standards that affect design/code
  - Conduct training on walkthroughs/ inspections/ peer reviews
  - Coordinate & attend reviews
  - Compile results of reviews
  - Follow up on action items
  - Verify & validate test activities
  - Review test results

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Process Improvement

- Analyze "as-is" environment
- Conduct "how to define a process" training
- Facilitate analysis of process improvements
- Define processes
- Plan implementation of improvements
- Audit the process(es)

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Consult & Train

- Determine & anticipate life-cycle training requirements
- Design event driven learning (EDL) sessions
- Provide “just-in-time” EDL
- Maintain training records
- Exercise a consulting role on the project team

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Process Teams

- Identify issues to be addressed (data-driven)
- Help form the team
- Train/facilitate/coach the team
- Support analysis of issue
- Facilitate definition of solutions
- Provide methodology for implementation pilot
- Generate standards/guidelines for implementation planning

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Conduct Assessments

- Train organization on expectations
- Review compliance/non-compliance criteria
- Establish the baseline culture
- Determine compliance with internal & external standards
  - **Perform gap analysis**
  - **Suggest method to reach compliance**
  - **Facilitate development & implementation of action plan**
- Perform in process audits

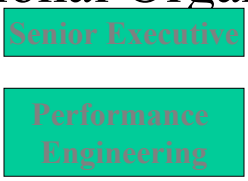
© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Reporting

- Track progress against goals
- Define relevant metrics
- Gather metrics from other disciplines
- Analyze data
- Report results
- Recommend action(s)
- Conduct special studies

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

A Functional Organization



Product Assurance

- **Quality Assurance/Systems**
- **Process Improvement**
- **Requirements Analysis**
- **Verification & Validation**
- **Configuration Management**

System Services

- **Training**
  - **Internal (non-HR)**
  - **External**
- **Technical Publications**
- **Customer Service(s)**

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

An Example - PE

- **Brief/Tailor RM Process & Standards**
- **How to Facilitate Reqs Gathering**
- **How to Write Reqs**
- **Coordinate/Facilitate Walkthrough**
- **Gather Data**
- **Review SRS Standard**
- **Quality Assurance Review**

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Successful Performance Engineering Is...

- Inter-disciplinary
- Insidious
- Flexible
- Cooperative
- Versatile
- Pervasive
- Systematic
- Based on facts
- Team oriented
- Communication
- Practical
- Focused on internal & external customers
- Supported by senior management

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

Summary

- Proactive quality assurance entails a multi-disciplinary set of activities
- Performance Engineering is assessing and enhancing the performance of individuals', groups' and the organization's products and services
- Success means making the developers' job more effective and efficient
- Result = achieving quality goals!

© 2002 Michael J. Hillelsohn SOFTWARE *PERFORMANCE* SYSTEMS

www.cmg.com

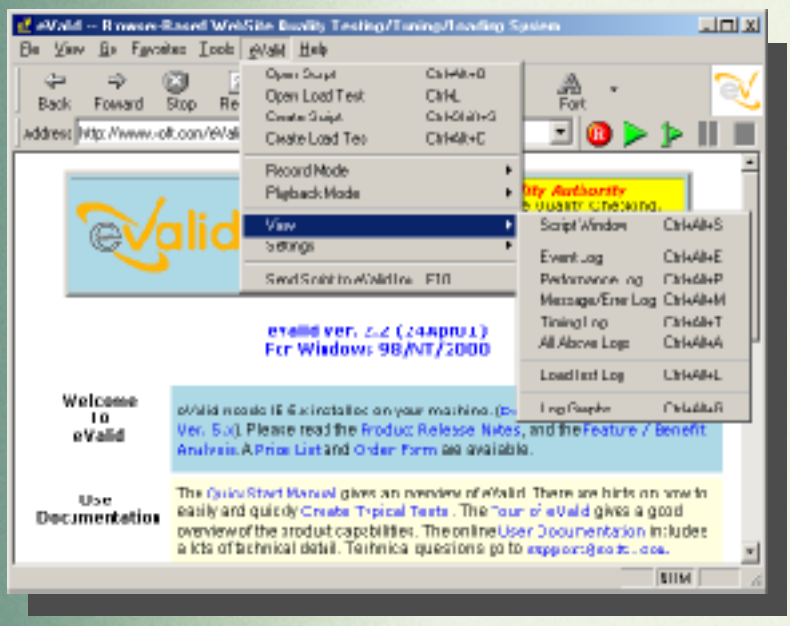
Time-to-market and quality-to-market are the prime motivators in ICT development. Good testing should not conflict with these principles. TestFrame helps to resolve this dilemma. Imagine faulty invoices being sent, and crucial communication systems failing, or other primary business processes being at a standstill. With TestFrame you can ensure good testing within reasonable time, control of risks and costs, and delivering products and services of proven quality.

For further information please contact the CMG TestFrame Research Centre,
Telephone +31 (0)348 45 40 00 or E-mail info@testframe.com.
or take a look at www.testframe.com

CMG
when IT really matters

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

gitek®

interaction through software

Gitek nv

Gitek nv was founded in 1986 and is in addition to the design and development of software, specialised in software testing. Gitek employs more than 120 people, with 40 professional software Test Engineers. Its customers include the pharmaceutical industry, the financial industry, the telecom industry and insurance companies.

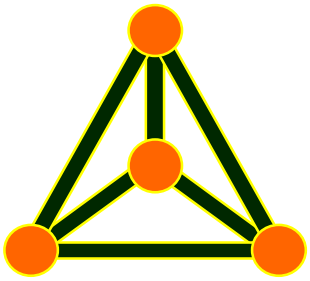
The difference

Gitek is an enthusiastic team of professionals, specialised in customised IT solutions and structured testing. A personal approach forms the basis of our success. This ensures better-adapted services to the customer's specific requirements, and a committed and motivated team ensuring optimal efficiency.

Structured Testing

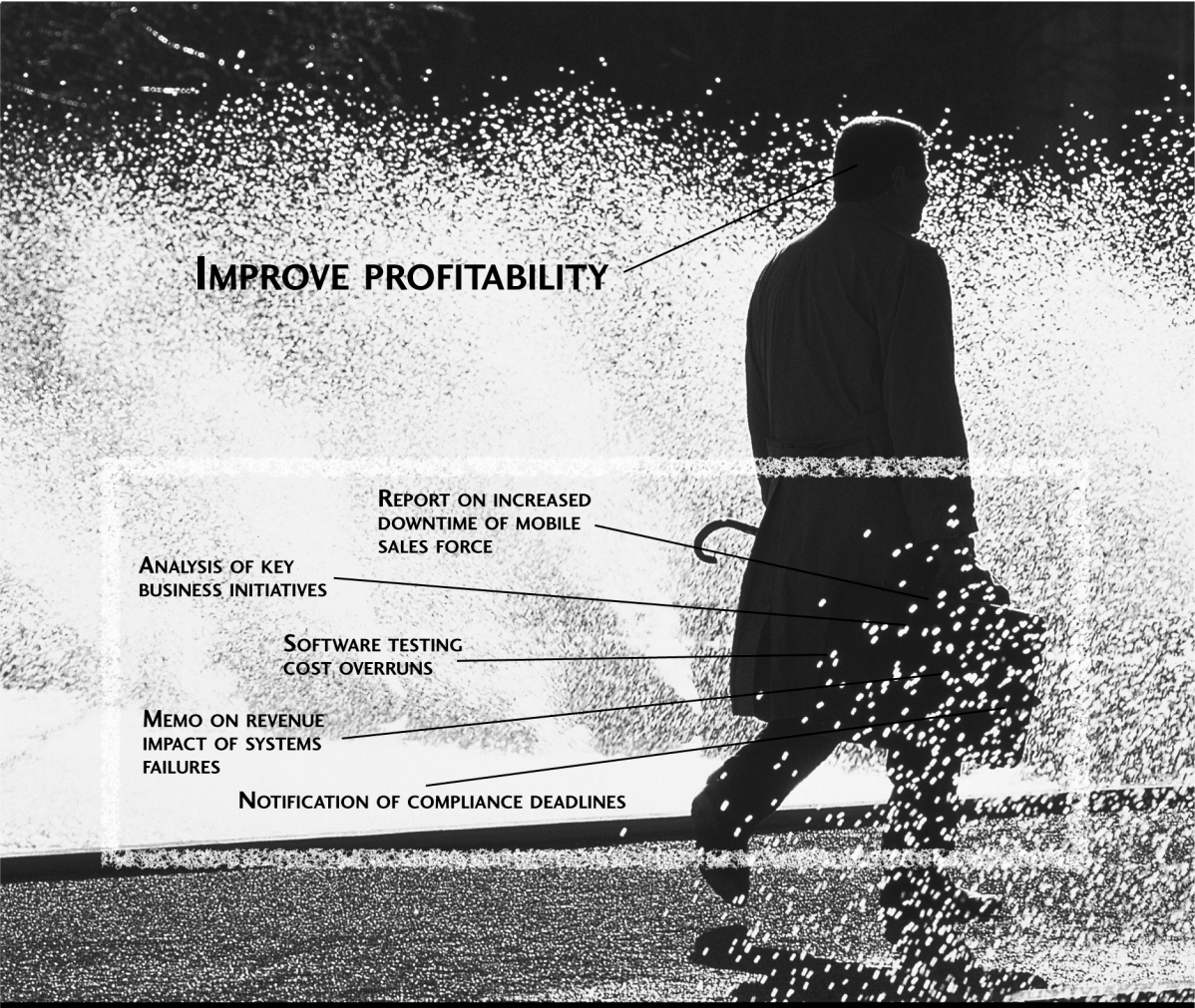
Gitek is exclusive distributor of TMap® (Test Management Approach), TPI® (Test process Improvement) and TAKT© (Knowledge of Testing, Automation and Tools) in Belgium.
Gitek provides services that offer a complete solution for testing:

- Participation in the operational test process
  - Test planning and test management
  - Test design and test execution
- Complete test projects and fixed price projects
- Test advice and support
- Defining and implementing a structured test process
- Selection and implementation of test tools
- Improvement of the test process
- Training and coaching in testing



Contact us

Gitek nv
Sint Pietersvliet 3
B-2000 Antwerp (Belgium)
http://www.gitek.be
e-mail: gitek@gitek.be



YOU THINK OUT OF THE BOX. WE'LL TAKE CARE OF THE REST.

Justifying technology investments in today's economy is an uphill battle. How can you concentrate on core business matters such as productivity and profitability, and still have time to manage day-to-day demands? Fortunately, there is a company that will help you deliver on all fronts. Whether you need full life-cycle or stand-alone technology solutions. Managed services, consulting, vertical industry solutions or staff augmentation. Who is this company? It's the one Fortune **1000** companies have turned to for the past **30** years for proven technology solutions.

spherion℠
Recruitment•Outsourcing•Technology

©2002 Spherion Pacific Enterprises LLC All Rights Reserved
Spherion is a registered trademark and the Spherion logo is a service mark of Spherion Pacific Enterprises LLC

VeriTest.®
Got a Gap?
Your IT department is good. But do you have the resources to ensure the highest levels of software reliability? Scalability? Interoperability?
The bar is higher for regulated industries that can't afford IT failures or downtime. VeriTest's worldwide product testing services close the gap between you and scalable, manageable,reliable, and cost-effective applications.
As the only lab authorised by Microsoft® to provide desktop-to-datacenter certification for Windows® 2000 applications, VeriTest has set the standard for both commercial and proprietary software.
We'll get you smiling again.
Call +33-6-86-43-10-54 today or visit http://www.veritest.com/Testers'Network/ and get our ROI Calculator.
Certified For
Microsoft® Windows®
Windows 2000 Datacenter Server
Windows 2000 Advanced Server
Windows 2000 Server
Windows 2000 Professional
Certification Labs
VeriTest is a business unit of Lionbridge Technologies, Inc.

Computer Associates™

VISION

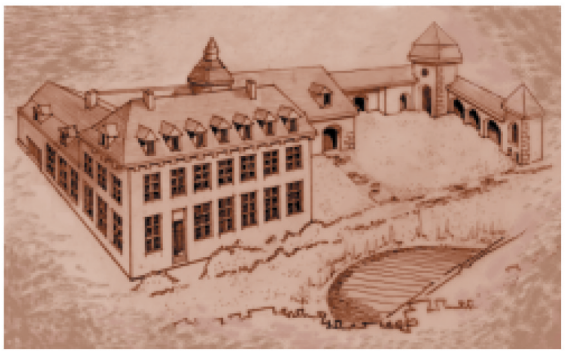
AUTONOMIC

Managing eBusiness Development

ca.com

Have a taste of Quality

ps_testware is a privately held company with headquarters in Belgium and a branch office in The Netherlands, specialised in Structured Software Testing. Our company offers methods and techniques to test software applications in a structured and an economically sound way. It has developed an overall concept of testing services in the shape of Training, Consultancy, Outsourcing, Coaching and TestLab and as such can be involved in test execution as well as in high-level project management.



Quality Service Provision

ps_testware offers you a long-term co-operation through a complete package of testing services tailored to your needs at a fixed price.

Training

ps_testware has created a training programme to introduce people to the different aspects of Software Testing.

Consultancy

When you want to know where your company stands concerning Structured Software Testing, a Test Assessment could be your answer. We will determine your strengths and weaknesses (in testing) and provide you with a detailed approach so that you can improve your processes.

Outsourcing

Our Test Consultants integrate in your organisation and apply their knowledge and skills to test your software. Outsourcing is done on three levels: Test Management, Test Development and Test Execution.

Coaching

When you need a hand to guide you through the complexity of structured software testing but you don't need a full-time consultant, coaching could be the solution for you.

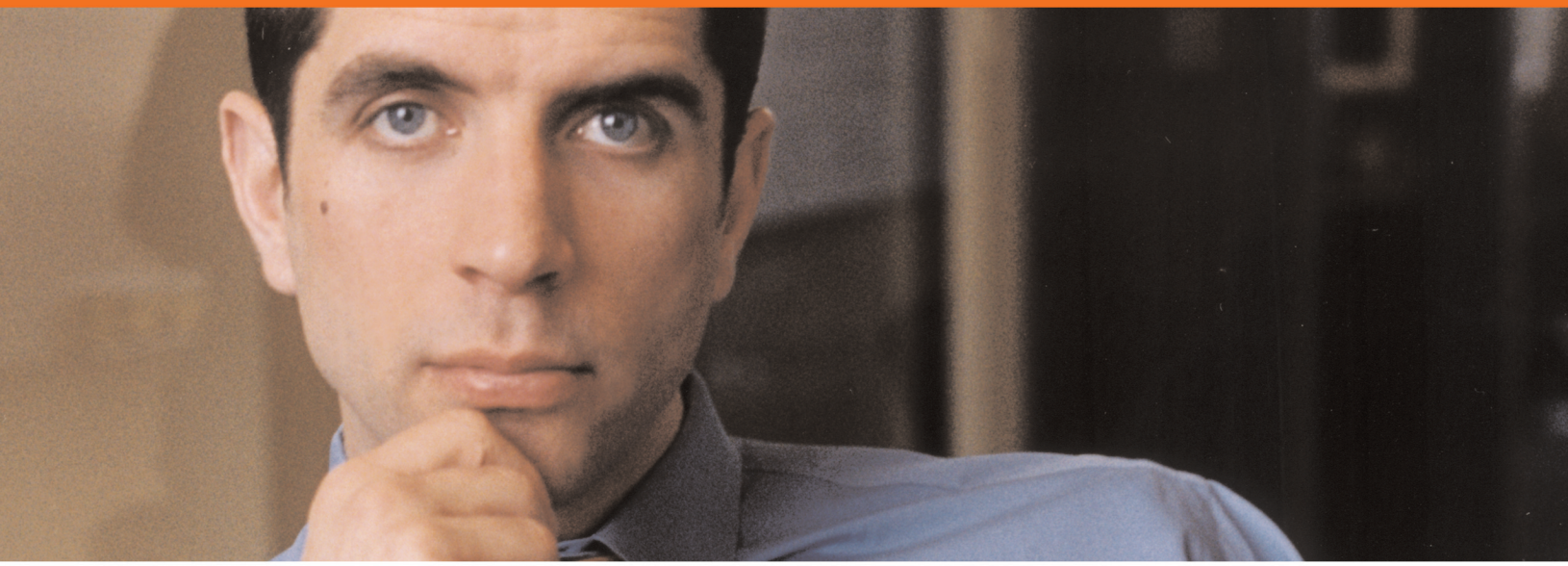
TestLab

You are able to benefit from the advantages of Structured Software Testing without any inconvenience through our professional and fully equiped TestLab.

ps_testware
Software Testing Services

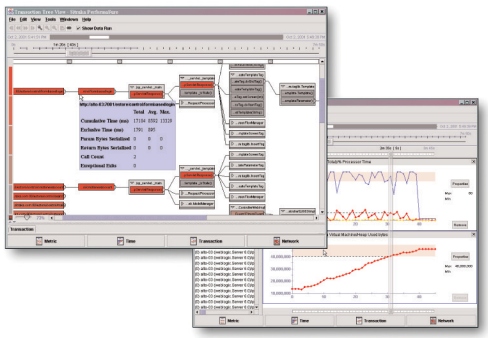
Tiensesteenweg 343 - 3010 Leuven (Belgium)
Tel.: +32 (16) 35 93 80 - Fax +32 (16) 35 93 88
e-mail: info@pstestware.com - http://www.pstestware.com

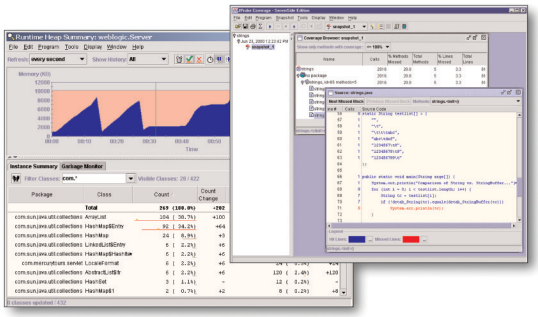
Innovation in J2EE performance.



sitraka
PerformaSure™

From the makers of JProbe, and leaders in J2EE performance assurance, Sitraka PerformaSure is a diagnosis tool that helps performance teams measure, analyze and maximize J2EE application performance from a round-trip transaction perspective during load testing. Arm yourself with PerformaSure for wide-angle diagnostic intelligence. And isolate poorly performing components before they create costly problems in production.





sitraka
JProbe™

Often the culprit behind performance problems lies within the application code itself. Award-winning Sitraka JProbe gives you a microscopic view of your Java application, right down to the offending line of code. The premier toolkit for performance tuning, JProbe provides deep J2EE diagnosis tools for profiling, memory debugging, thread analysis and code coverage.

For more info visit:
www.sitraka.com/performance

sitraka
the Java™ advantage

© Copyright 2002 by Sitraka Inc. All rights reserved. Sitraka, the Sitraka logo, JProbe and PerformaSure are trademarks of Sitraka Inc. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in US and other countries. All other products are trademarks or registered trademarks of their respective companies.

ABOUT CSE...

The Centre for Software Engineering (CSE) in Dublin-Ireland is a leading technology support centre in Europe for the industrialisation of software. The CSE provides advanced skills training, consultancy and evaluation services to software intensive organisations, typically in the ICT industry sectors. A comprehensive range of services addresses Software Process Improvement, Component Based Development, Engineering High Quality Products, and Project Management and People Issues. http://www.cse.dcu.ie

ABOUT Software Technology Forum...

The Software Technology Forum is a professional network, which consists of about 75 Danish companies. The forum organises ten experience exchange groups in areas like software quality, testing, metrics, object oriented methods, broadband communication, multimedia technology and software process improvement (the Danish SPIN-group). http://www.datatekniskforum.dk

ABOUT ESI...

The European Software Institute is one of the world's leading independent authorities on software process improvement. ESI is a non-profit-making organisation driven by the demands of European industry. It is supported by the European Commission, the Basque Government and through company membership. ESI's work is centred on products and services that are tied directly to core business objectives such as reducing costs and increasing predictability of results among others. ESI's headquarters are in Bilbao, Spain.
(http://www.esi.es)

ABOUT GI...

The German Informatics society (GI) is the association of about 21,000 men and women who, by their work, take part in the progress of informatics or who are interested in the development of informatics. The work of the GI ist mostly done in a decentralized way in divisions, expert committees, special interest groups, working groups, regional groups, advisory councils and in user groups. (http://www.gi-ev.de)

ABOUT OCG...

The objective of OCG is the comprehensive and interdisciplinary promotion of information processing, with due regard to its effects on people and society, including technical as well as educational and legal aspects. In fulfilling this objective, the Society performs five primary functions: serves as an umbrella organization; represents Austria in IFIP, CEPIS, IMIA and ERCIM; provides members with services, information and consultation; promotes research and development projects; and it is an affiliate member of ACM (Association of Computing Machinery) and of the IEEE Computer Society. (http://www.ocg.at)

ABOUT SAST...

SAST is a non-profit and independent organization open for all persons interested in software testing in Sweden. There are several goals for SAST: First of all to create a way for members to exchange experiences and knowledge related to testing. Secondly, to increase the status of testing by spreading information and knowledge about software testing. Also, creating a network among testers is important, both within Sweden and abroad. SAST was formed in 1995 by an initiativ from Anna Eriksson and Sigrid Eldh. Today, 6 years later, SAST has approximately 800 members and continues to grow, but the vision is the same - to increase the knowledge about Software Testing.
(http://www.sast.net/)

ABOUT Soft-ED Ltd...

Soft-ED Ltd
Description to be supplied. (http://www.soft-ed.net/)

ABOUT SAI...

SAI (Studiecentrum voor Automatische Informatieverwerking) is a non-profit organization whose purpose is to promote the knowledge of Information. As far as the theoretical as practical aspects of information knowledge, the organization takes a stand on social questions that has to do with automation of information as far as it applies. The activities of the organization are focused on people who are specialists in information. SAI organizes discussions, meetings, seminars, workshops and a magazine called "Informatie". (http://www.sai.be/)

ABOUT TESTNET...

The aim of TestNet is the professionalization of testing IT products, and an increase in the awareness and importance of testing as a profession in its own right. TestNet stimulates the exchange of professional knowledge and practical experience amongst testers, and stimulates research, from a scientific standpoint as well as from a practical perspective. (http://www.testnet.org)

ABOUT TI...

The Technologisch Instituut is active in all professional fields where technology and science play a central part. These include a wide range of activities, ranging from civil engineering, infrastructure, process technology, electrical engineering, telecommunications, computer technology, food and agricultural technology, the study of materials to general technologies such as management techniques, energy, safety and environmental technology (34 different sections). (http://www.ti.kviv.be)

YOUR WAY OUT OF THE e-MAZE

A diversely skilled team of Information Technology professionals supports you, and your IT department, in your search for quality. We offer guidance from Requirements Analysis to the choice of technologies, and we provide the design, the development and the implementation of your projects, ensuring the training of your IT people and end users.

Let's go down the path of efficient computing together

These days, your company won't be competitive, nor will your services be quality-based or your products lauded as excellent and successful unless you are capable of handling and promptly implementing the new information technologies.

Let CEDITI instruct you and help you set up and use the solutions that are best adapted to your needs.

Strategic analysis and Consulting services

Defining and implementing the software architecture, integrating the new technologies into the existing (EAI), and choosing the right ones, and the right tools are all processes of growing complexity that have to take into account an ever-increasing number of technical, economic and humans factors.

CEDITI supplies you with the very latest techniques, ensuring that all you have to worry about is how successful your project is going to be.

GRAIL, a software tool developed by CEDITI, uses KAOS Requirements Analysis methodology to integrate into one global model all your needs, operating processes and constraints involved in your IT projects, in order to be able to draft consistent specs and check the validity of the solution for as long as it continues to be implemented.

Net solutions

CEDITI realizes the development and implementation of NET solutions and transfers its know-how to your teams.

Awards received in 2001

"Warobase de Cristal 2001", a contest organized by the Regional Government of Wallonia, won for the design and implementation of the Internet/Intranet System of Alpha Repartition, the main pharmaceutical distributor in Wallonia – the system links Alpha product data bases to the drugstores.

SIS, the Sports Information System developed for the press group IPM (Dernière Heure – Les Sports, La Libre Belgique) and the newspaper group Vers l'Avenir. This achievement has been rewarded by the "ICT Business Award 2001", as a result from a contest organized by Keesing Business Media in Belgium.

Learn more about our experience and our reference list while visiting our Web site http://www.cediti.be/

Cediti S.A.
Avenue Georges Lemaître 21
B-6041 Charleroi (Belgium)
Phone : +32 71 259 404
Fax : +32 71 372 064
E-mail : info@cediti.be
URL : http://www.cediti.be

CMG plc (www.cmg.com) is an international services and solutions Group that applies information and communication technologies to create added value for its customers. Constantly at the forefront of business change, our goal is to deliver enhanced operational efficiency and competitive advantage. We are at the heart of our customers' critical activities, combining industry knowledge, technical expertise and practical innovation. We believe in long-term relationships built upon the trust that comes from a consistent track record of delivery.

Information & Communications Technology (ICT)

CMG is a full service provider through management consultancy, systems development & integration and the outsourced management of targeted business processes. This constantly evolving skills base is focused through vertical market knowledge in both public and private sectors including finance, telecommunications, transport, petrochemicals, utilities, manufacturing, retail, media and government. Today our solutions extend to the creation of differentiated electronic and mobile business models (e&mBusiness).

Test Management & Consultancy

One of CMG's many specialisations is testing. CMG has developed TestFrame (www.testframe.com), a very successful method for structuring of the test process and automation of the test execution. CMG's full service concept in testing covers test management, test consultancy and test managed services.

About Computer Associates

Computer Associates International, Inc. (NYSE: CA) delivers The Software That Manages eBusiness. CA's world-class solutions address all aspects of eBusiness management through industry-leading brands: Unicenter for infrastructure management, BrightStor for storage management, *e*Trust for security management, CleverPath for portal and business intelligence, AllFusion for application life cycle management, Advantage for data management and application development, and Jasmine for object-oriented database technology. Founded in 1976, CA serves organizations in more than 100 countries, including 99 percent of the Fortune 500 companies.

The continual emergence of new business models require solutions that extend beyond an organization's boundaries to include applications and processes of business partners. Integrating diverse processes across multiple partners and complex computing environments is critical to meeting unique eBusiness challenges. Organizations must adopt new strategies and create effective solutions within a narrow window of opportunity.

CA offers AllFusion, our family of comprehensive solutions for building, deploying and managing eBusiness applications. Our best-of-breed modeling solutions and proven change and configuration management solutions enable organizations to model, develop and deploy applications enterprise-wide. These solutions, coupled with efficient process and project management, and CA's own Best Practices, can help any organization quickly respond to change, accelerate time-to-market and remain competitive in today's evolving marketplaces.

CA Services, the post-sales support organization of CA, delivers technology solutions for customers in the areas of eBusiness process management, information management, and infrastructure management. Services include product support, consulting and education

For more information, visit http://ca.com.

eValid™ -- The Internet Quality Authority™
Client-Side Browser-Based WebSite Quality Checking,
Testing, Validation, Tuning, Loading, 24x7 Monitoring
Training, Consulting, Seminars
© Copyright 2001 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.

eValid, Inc.
901 Minnesota Street
San Francisco, CA 94107 USA
www.e-valid.com

Phone [+1] 415.550.3020
FAX [+1] 415.550.3030
info@soft.com.

Gitek nv

Gitek nv was founded in 1986 and is in addition to the design and development of software, specialised in software testing. Gitek employs more than 120 people, with 40 professional software Test Engineers. Its customers include the pharmaceutical industry, the financial industry, the telecom industry and insurance companies.

The difference

Gitek is an enthusiastic team of professionals, specialised in customised IT solutions and structured testing. A personal approach forms the basis of our success. This ensures better-adapted services to the customer's specific requirements, and a committed and motivated team ensuring optimal efficiency.

Structured Testing

Gitek is exclusive distributor of TMap® (Test Management Approach), TPI® (Test process Improvement) and TAKT© (Knowledge of Testing, Automation and Tools) in Belgium.
Gitek provides services that offer a complete solution for testing:

- Participation in the operational test process
  - Test planning and test management
  - Test design and test execution
- Complete test projects and fixed price projects
- Test advice and support
- Defining and implementing a structured test process
- Selection and implementation of test tools
- Improvement of the test process
- Training and coaching in testing

Contact us

Gitek nv
Sint Pietersvliet 3
B-2000 Antwerp (Belgium)
http://www.gitek.be
e-mail: gitek@gitek.be

I2B

COMPANY DESCRIPTION

I2B

TERVUURSESTEENWEG 163

B-3001 LEUVEN, BELGIUM

TEL: 32 -9 -384-86-27 FAX: 32-16-84-59-49

WWW.I2B.BE

Idea to Business (I2B) is a Belgian consultancy firm. We help Clients tackle issues that rise when they want to **radically innovate**.

We provide novel ways to master the standard business skills such as business strategy, project management, change management, ICT strategy and Management. We help our Clients in acquiring new competences such as creativity or product usability, both very essential in an innovative context.

At Quality Week, I2B is represented by its ICT consultants. These consultants have significant history in Software Quality Assurance and testing. I2B has developed new ways of dealing with software testing, using Emotional Intelligence or Swarm Intelligence. We look for solutions to reduce testing to a bare minimum whilst reducing risk to a minimum. I2B provides advice in the "soft skills" designed especially for Software Quality Assurance Personnel. We teach software testers to become "Trusted Advisors", a key success factor to becoming a good tester. In short, I2B helps people using and deploying ICT in a radical different way. In all our actions we aim at giving you only the best solution.

Integri is a company specialised in providing test tools (INQ), oriented towards testing systems interacting with each over clearly defined transactional interfaces. The INQ platform is being used for testing host real-time and batch interfaces (ISO8583, EDIFACT, X.509, …), for testing smartcards (EMV, CEPS, GSM, …) and terminals (debit and credit terminals, EMV terminals, forecourt devices, …). For international standards, Integri can provide a fully features test tool including all test scripts. However the INQ platform is open and can easily be adapted to the client's specific requirements.

Integri Professional Services (IPS) helps its clients in test project. Integri organises and executes acceptance and certification tests for its clients, acting as an independent test company. The client provides IPS with detailed functional specifications of a project. From these functional specifications, IPS derives the tests, executes them on the unit under test and issues a test report. Integri Professional Services also prepares and operates the inter-operability test lab of inter-industry associations. Customers Integri and IPS are large national and international payment system providers, smartcard and payment terminal manufacturers, and GSM operators. Typical systems for which Integri's test technology and methodology is suited for are: smartcard masks, payment terminals, smartcard devices, security systems, authorisation host interfaces, clearing and settlement systems.

inQA.labs are Spain's leading independent Software Quality Engineering and Test Outsourcing company. Their international team of quality assurance professionals offers world-wide the full range of QA services, including software quality consulting, test planning, test organization, and process improvement.

inQA.labs also provide complete software validation through structured testing, automatization and simulation to bring you a true and accurate picture of the quality of your product.

Among the key services offered are functional, reliability, performance, load, interoperability, and localization testing, plus benchmarking.

For e-business, inQA.labs offer original load testing in-house at the client's site. This enables systems to be dimensioned correctly by simulating thousands of virtual users, and eliminates costly test tools.

inQA.labs also distributes CENSUS solutions, from MetaQuest inc, the first truly web based Issue Tracking System with Windows look & feel, which enables teams to track and manage project-related issues.

You can find out more about inQA.labs at www.inqalabs.com.

1 Consulting Services

ITeeVision's Consulting Services are performed by highly motivated and skilled professionals, who are continually trained to further their qualifications, career, and value to your projects. This has resulted in an invaluable knowledge capital, from which ITeeVision can draw, and which it can render available to its customers.

To date, ITeeVision has established several practices, geared towards structuring its Consulting Services around different technologies:

- **ERP / CRM**
- **CLIENT / SERVER DEVELOPMENTS**
- **INTERNET DEVELOPMENTS**
- **QUALITY CONTROL**

  To support the Quality Control process, ITeeVision has built a strong and close partnership with Segue Software. Segue Software delivers the industry's premier e-business reliability products and solutions to help you address the risks and complexities that can affect the performance and quality of your e-business system.

2 E-Testing

The E-Testing (testing of all relevant aspects of E-Business applications) Practice at ITeeVision revolves around control of ***time***, ***money***, and ***people*** - all of which are an integral part of successful E-Initiatives. Our E-Testing experts have created a unique E-Testing Framework, based on Requirements and Risks management. This framework eliminates the excessive risks that prevail in E-Business developments, by addressing the following issues: functionality, reliability/availability, usability, performance, and security. This is organized around the following services:

- **CONSULTING SERVICES**
- **OUTSOURCED SERVICES**
- **PRODUCTS RESELLING (SEGUE PRODUCTS)**
- **TRAINING SERVICES**

Pearson Education, part of Pearson PLC, publishes many educational, reference and professional titles and employs over 10.000 people in 40 countries around the world. Pearson PLC (www.pearson.com) is an international media company with market leading businesses in education, strategic business information and consumer publishing. Pearson businesses include the Pearson Education, the Financial Times Group and the Penguin Group, and all command rich content, powerful brands and true international reach.

Pearson Education Benelux (www.pearsoneducation.nl) was established in 1998, following the merger of a number of publishing houses, including Addison-Wesley Longman and the education division of Simon & Schuster. Pearson Education Benelux handles the distribution of Pearson's English language titles, the translation of books by well-known foreign writers, as well as the publishing of original Dutch manuscripts.

Our books fall into the professional and Higher Education category, which again is subdivided into Information Technology and business. Pearson Education Benelux creates books for "absolute beginners", as well as books for experienced specialists and professionals, who wish to keep up to date with the latest developments.

ps_testware is a privately held company with headquarters in Belgium and a branch office in The Netherlands, specialised in Structured Software Testing. Our company offers methods and techniques to test software applications in a structured and an economically sound way. It has developed an overall concept of testing services in the shape of Training, Consultancy, Outsourcing, Coaching and TestLab and as such can be involved in test execution as well as in high-level project management.

Quality Service Provision

ps_testware offers you a long-term co-operation through a complete package of testing services tailored to your needs at a fixed price.

Training

ps_testware has created a training programme to introduce people to the different aspects of Software Testing.

Consultancy

When you want to know where your company stands concerning Structured Software Testing, a Test Assessment could be your answer. We will determine your strengths and weaknesses (in testing) and provide you with a detailed approach so that you can improve your processes.

Outsourcing

Our Test Consultants integrate in your organisation and apply their knowledge and skills to test your software. Outsourcing is done on three levels: Test Management, Test Development and Test Execution.

Coaching

When you need a hand to guide you through the complexity of structured software testing but you don’t need a full-time consultant, coaching could be the solution for you.

TestLab

You are able to benefit from the advantages of Structured Software Testing without any inconvenience through our professional and fully equiped TestLab.

QA-Systems - the software quality company - is a provider of software development solutions that improve the quality of software applications and the productivity of software development teams. Our solutions, a combination of consultancy, tools, training and support focus on the area's of

1. software requirements management
2. software configuration management
3. automated software quality testing (static analysis)

QA Systems represents the products of Programming Research (QA-C, QA-C++), QA Systems Technologies (QStudio for JAVA) and Starbase Corporation (Caliber-RM and Starteam) in Austria, the Benelux and Germany.

QA Systems International BV
Amersfoortsestraat 124
3769 AN SOESTERBERG
contact: Dirk Giesen
dirk.giesen@qa-systems.com

About Rational Software

Rational Software provides a software development platform that improves the speed, quality, and predictability of software projects. This integrated, full life-cycle solution combines software engineering best practices, market-leading tools, and professional services. Ninety-six of the Fortune 100 rely on Rational tools and services to build better software, faster. This open platform is extended by partners who provide more than 500 complementary products and services. Founded in 1981, Rational is one of the world's largest software companies, with revenues of $751.4 million in its twelve months ended December 31, 2001, and over 3,700 employees worldwide. Rational is a component of the Nasdaq-100 IndexR. Additional information is available on the Internet at www.rational.com/benelux

Rational Software NV
Interleuvenlaan 15d
3001 Leuven
Tel: +32 (0)16 397 600
Fax: +32 (0)16 397 610
www.rational.com/benelux

RelQ is a global Independent Software Testing and QA Company offering services to organisations involved in software development.

Currently RelQ have over 300 personnel worldwide and have successfully completed more than 150 testing projects. Our test engineers are fully trained in best practice testing methodologies and have expertise in a wide range of domains, including, Banking & Finance, Real-Time / Embedded Software, Avionics, EDA and Web Based Applications.

With offices throughout Europe, USA and Asia Pacific we are able to assist software development companies, in most geographies, in delivering reliable and quality software on time.

Our full range of services covers Functional, Usability, Reliability, Performance, Scalability, Security and Installation Testing. RelQ's Tools Group have knowledge and experience of most testing tools from vendors such as Mercury, Segue, Rational and Parasoft allowing automated testing of customer's products.

RelQ's flexible testing services offer our customers a package commensurate with their individual needs. Testing can be on-site, offshore, or in a dedicated Test Lab specifically built to meet the customer's needs. In each case testing is strictly managed by RelQ and staffed by test engineers with specific skill sets matching the customer's requirements.

Contact: Dave Arden - davearden@relq-europe.com

Sitraka Company description

About Sitraka
A proven leader in J2EE Performance Assurance, Sitraka delivers advanced diagnostic solutions that help companies to pinpoint and eliminate performance hazards in mission-critical J2EE applications. Offering a system-wide view of performance, Sitraka PerformaSure is a transaction-centric diagnosis tool that enables e-business performance teams to measure, analyze and maximize performance of multi-tiered J2EE applications during load testing and before costly problems occur in production. For a highly detailed performance solution, developers use Sitraka JProbe to rapidly identify errors and inefficiencies in Java code for maximum application performance and reliability. Sitraka JClass is an award-winning family of integrated Java components, and DeployDirector is a Java application provisioning and management solution that helps IT organizations to deploy, manage and update rich clients, maximizing uptime of core systems while dramatically reducing time-to-repair.

Through industry partnerships with companies such as IBM, BEA Systems, Sun Microsystems and Mercury Interactive, Sitraka ensures that our products integrate seamlessly with the latest development environments and platforms. Sitraka products are sold and supported directly through our North American headquarters in Toronto, European headquarters in Amsterdam and global network of resellers. Visit Sitraka on the Web at www.sitraka.com.

Sitraka Europe
Derkinderenstr. 22, Trivium 6th floor
NL-1062 DB Amsterdam
The Netherlands
Tel. +31 (0)20 5106700
Fax +31 (0)20 4700326
www.sitraka.com
email: eurosales@sitraka.com

Contact Sales: Christophe Bazin, christophe.bazin@sitraka.com
Contact Marketing: Martha Blassnigg, martha.blassnigg@sitraka.com

TestWorks

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

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

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.

WWW.SOFT.COM

Spherion Technology Group

Technical issues do not solely determine the quality of your information systems. First of all the information demand is business driven. The applied solution can only be determined by users, management and IT-specialists together. Spherion recognizes the importance of this and bridges the gap between these key players. We offer services and solutions in information management, service management, risk analysis, testing, test management and training. Spherion can help you maintain the required level of service and quality, so you can focus on the matters that are really important: your business objectives.

With a staff of highly qualified professionals, Spherion is capable of forging a durable partnership and maintaining high-quality service. This quality and the recognition it has gained in the market by our customers, accounts for the strong and steady growth experienced by Spherion Technology Group during the past years.

Worldwide, Spherion Corporation (NYSE:SFN) operates in the field of recruitment, outsourcing and technology in the broadest sense. From over 1000 branches throughout the world Spherion offers tangible improvements to the competitive position of its clients. The Technology Group of Spherion operates in North America, Europe, Asia and Australia with more than 5500 consultants.

About SQS Software Quality Systems AG – www.sqs.de

SQS is an independent software consulting company that has been enjoying success with its consulting services, products (e.g. the SQS-TEST/Professional tool) and tutorials on quality management and testing for software since 1982.

In 2001 the SQS Group, which has 570 employees (End of 2000: 370), will make DM 140 million (2000: DM 78 million). The Group consists of SQS AG Cologne (Germany) and the majority-owned subsidiaries SQS Software Quality Systems S.A. in Bilbao (Spain), SQS Software Quality Systems Ges.mbH in Vienna, SIM Group in Woking (England), AQpro B.V and AQsoft B.V in Zaltbommel (The Netherlands) and Cologne-based FTT Future Technology Team GmbH.

SQS' main customers are multinational companies in the finance sector (e.g. Deutsche Bank AG), telecommunications (e.g. Deutsche Telekom AG), industry (e.g. Alcatel SEL, DaimlerChrysler), public administration (e.g. the Federal Administration Office) and commerce (e.g. EDEKA) but it has also been working with small and medium-sized enterprises for several years.

For further information please contact Bianca Hammacher on +49 2203 91 54 18 or email sales@sqs.de

About SIM – www.simgroup.com

SIM are the UK's leading independent testing service provider and part of the SQS group of companies – Europe's largest testing group - that commands a 30% market share of the outsourced testing marketplace.

SIM supply and manage testing teams that provide expert consultancy and execution of testing strategies to help companies manage their business risks. The SIM philosophy is to add value to the business development process by understanding the business case for testing and by providing metrics to prove the effectiveness of our service. As well as independent advice, we also supply testing teams that perform to the highest standards of efficiency and effectiveness. With proven skills in test management, test asset management, healthchecks, performance and load testing and automation, SIM is able to provide significant cost and time savings on all projects and provide an effective return on investment.

SIM don't supply tools. SIM recommends the most appropriate tools to provide an independent and objective view of its clients current test processes.

For further information please contact Dominique Grant on +44 (0)1483 733 119 or email enquiries@simgroup.com

Ubizen AETHIS

Ubizen AETHIS provides tools and services to measure and monitor the **Quality of Service** (QoS) of IT networks, applications and **e-business** infrastructure.

Ubizen AETHIS solutions allow the customer to insure
- **performance** (the application must respond fast enough),
- **availability** (the application must always respond) and
- **security**

of his applications and infrastructures.

The large set of high-level tools (best of breed policy) distributed by Ubizen AETHIS allow:
- 24x7 monitoring of the e-business infrastructure (website monitoring, transaction simulation, streaming, …) from outside or inside the firewall
- 24x7 monitoring of the internal network (simulation of transactions, detection of bottlenecks, root cause analysis…)
- Load (stress) testing of e-business infrastructures
- Diffusion of alarms (SMS, mail, voice, etc…) with escalation and acknowledgement
- Development and installation of SNMP agents (including secure SNMP V3)
- Simulation of SNMP agents and networks

In terms of its network security activities, Ubizen AETHIS combines consulting, deployment and support services with best of breed 3rd party products to provide **complete security solutions**.

Typical customers of Ubizen AETHIS are large organisations having to insure high availability and security of their infrastructures and applications.

Some products: Keynote, HP OpenView (NNM and Internet products), InfoVista, ProactiveNet, WebTrends, Adlex, SNMP Research, SimpleSoft, Telamon, CheckPoint, Nokia.

Ubizen AETHIS is a daughter company of Ubizen and is considered as European competence centre for Quality of Service tools.

Ubizen AETHIS
Rue du Bosquet, 7
B-1348 Louvain-la-Neuve
Belgium
http://www.aethis.com
info@aethis.ubizen.com
Tel. : + 32 10 456 130
Fax : + 32 10 456 155

VeriTest is the leading provider of comprehensive product testing, certification, and strategic QA consulting services for the IT developer community. Since 1987, clients have leveraged VeriTest's experience, partnerships, and scalable resources to take the uncertainty out of product quality. We help product developers reduce the costs of testing, while speeding time to market and ensuring interoperability and reliability. VeriTest is the sole authorized certification lab for applications running on Microsoft Windows 2000 Professional, Server, Advanced Server and Datacenter Server.

Service Offerings

- Enterprise Interoperability Testing
- Accessibility Testing
- Functionality and Compatibility Testing
- Globalization Testing
- Performance Testing
- Application Migration Testing
- Product Certification

VeriTest's in-depth skills, experienced testing staff, hardware and software infrastructure, and best practices have been developed throughout our 15 years in the industry, providing customers with the solutions-based test services they need.

Our team of career test engineers and project managers deliver the know-how required to execute complex test missions under the most demanding schedules. We help clients launch higher quality products faster.

VeriTest is the testing division of Lionbridge Technologies, a provider of solutions for worldwide deployment of technology and content.

For more information, see www.veritest.com