

11th Dutch Testing Day

Test Automation

**11 November 2005
University of Twente
Enschede, the Netherlands**

Copyright © 2005 by Formal Methods and Tools group, University of Twente

Editorial production by Ed Brinksma and Mariëlle Stoelinga

Program Committee

Egbert Bouman (Maintain)
Ed Brinksma (University of Twente)
Henk van Dam (Collis)
Wan Fokkink (CWI)
Martin Gijsen (LogicaCMG)
Kees van Hee (Eindhoven University of Technology)
Judi Romijn (Eindhoven University of Technology)
Maurice Siteur (Cap Gemini)
Mariëlle Stoelinga (University of Twente)
Jan Tretmans (Radboud University Nijmegen)
Erik van Veenendaal (Improve Quality Services)
Marc Witteman (Riscure)

Local Organization

Axel Belinfante
Machiel van der Bijl
Laura Brandán Briones
Ed Brinksma
David Jansen
Tomas Krilavičius
Joke Lammerink
Ellen Roberts-Tieke
Mariëlle Stoelinga

Printed in the Netherlands

Dutch Testing Day 2005

Table of contents

Preface by Ed Brinksma

Papers

Testing and Test Automation in Telecommunication Carsten Weise	6
Using TTCN-3 to standardize the test automation chain Leon Wolters, Jos van Rooyen, Erik Altena	7
TTCN-3 for Automated Testing Railway Control Systems Stefan Blom, Nicu Goga, Natalia Ioustinova, Jaco van de Pol, Axel Rennoch, Natalia Sidorova	8
Defect registration tools as knowledge base Ed Brandt	10
Proving test coverage at requirements level Han van Gerwen	15
Automatic on-the-fly testing of reactive systems Pieter Koopman	16
Case study AllianzNet: Automated versus manual testing Guido van Leeuwen, Hans van Loenhoud	20
Timed Testing with TorX Axel Belinfante, Henrik Bohnenkamp	22
Test automation in the alpha test of wafer scanners Ivo de Jong, Roel Boumen, Asia van de Mortel-Fronczak, Koos Rooda	24

Company brochures

Collis	26
Centre for Telematics and Information Technology	30
LaQuSo	31
Embedded Systems Institute	32
Four Oaks	33
Harvey Nash	34
LogicaCMG	35
Maintain	36
Mercury	37
ps_testware	38
Qualityhouse	39
Refis	40
Telelogic	41

Preface

Dear Participant,

Since its first gathering in 1995 the Dutch Testing Day has grown from a mostly informal meeting of people from industry and academia with an interest in software testing into one of the larger annual Dutch events of the field. The growing status of the Dutch Testing Day among industrial practitioners and academic researchers alike was convincingly confirmed by last year's tenth anniversary edition in Leiden, which attracted more than 200 participants and enjoyed substantial sponsoring support.

Also this year's interest in the Testing Day promises a continuation of the trend. At the time that these proceedings had to be printed, already well over one hundred participants had registered, and sponsoring support is undiminished. The latter is important, as it is a tradition of the Testing Day that registration is free of charge. Given the growing number of participants it is essential that our financial support grows proportionally to keep up the good work. Therefore, we are most grateful for our sponsors Cap Gemini, Collis, CTIT, Embedded Systems Institute, Four Oaks, Harvey Nash, LaQuSo, LogicaCMG, Maintain, Mercury, ps-testware, Qualityhouse, Refis, Telelogic, and the University of Twente. Without their support the eleventh edition of the Dutch Testing Day would not have been possible.

The program of this day starts with a keynote presentation by a prominent foreign testing expert. Dr Carsten Weise of Ericsson in Aachen will tell us about his experiences with practical testing and test automation in the telecom domain. His topic reflects a trend that also experts from more technical domains, such as telecom and embedded systems, are developing a strong interest in more systematic and automated approaches to testing.

The remainder of the program consists of a careful selection by the program committee of eight presentations from the 31 presentation proposals that were submitted. This large number of submission again indicates the popularity of our event, but unfortunately also means that some quite good proposals had to be turned down. Also, in keeping with the best traditions of the day we have tried to maintain a balance between the industrial and academic submissions. We hope that you will find the program useful and inspiring, and wish you a very instructive, interesting and pleasant Testing Day.

Enschede, November 2005,

Ed Brinksma
Mariëlle Stoelinga

Presentations

Testing and Test Automation in Telecommunication

Carsten Weise
Senior Systems Designer at Ericsson
Email: research@cweise.de

Abstract

In academia, there is a profound pool of knowledge about testing methodologies and automation of test processes. Most industrial work on testing would benefit from application of these methodologies. However, there are also a lot of daily business obstacles which prevent the introduction and application of these methodologies. This is especially true when testing does not only involve software, but also hardware. Based on recent experience in testing of telecommunication software, the talk will discuss these obstacles: what are they, where do they come from, what are their consequences, and are there possible ways to overcome them? The obstacles will be illustrated by real world examples.

About the author:

Carsten Weise got his Diploma and his Ph.D. from Aachen University of Technology, the Diploma was on implementation of real time systems, the Ph.D. thesis on the theory of timed automata. He worked as a post-doc for BRICS (Basic Research Institute for Computer Science) in the real time group of Kim Larsen at the University of Aalborg in the development of the verification tool Uppaal (www.uppaal.com). Carsten Weise then joined Ericsson Research and Development in Aachen, Germany, where he was responsible for the design and development of end-to-end test tools for the telecommunication core network for three years. He then worked for two years as leader of the group supporting the test activities with software production and defect tracking. Since May 2005, he is working in the technical studies group with a main focus on test performance indicators. He is member of the program committee for FATES (see e.g. <http://fates.cs.auc.dk/>).

Using TTCN-3 to standardize the test automation chain

Leon Wolters, Jos van Rooyen, Erik Altena

TTCN-3 is the Testing and Test Control Notation version 3. It is a language that can be used for the specification of all types of reactive system tests over a variety of communication ports. Historically, TTCN has been associated with conformance testing. The domain in which it has been developed and in which it is still mostly used is that of telecommunications. However, its potential application fields exceed these.

The test automation chain stretches from test case generation to report generation with at its heart test execution. The chain – by definition – consists of different steps. Data between these steps must be specified in a format understood by both ends of a step. Different domains – telecoms, embedded, financial, etc. – may have comparable test automation chains in theory, but differ in the realization of their steps.

So, the test automation chains of different domains are in practice not very compatible, while there would be much too gain if they would be more standardized.

Testing systems on the borders of different domains, for instance, could be easier done with a single, standardized approach. Also, with a standardized format in place, the best tools from each domain could be combined to make the optimal chain for each project regardless of domain.

We propose to standardize the different steps of the test automation chain by using TTCN-3. TTCN-3 could not only serve as the language in which specifications are directly written by testers, but also as the intermediate format that applications would use to communicate with each other between the different steps of the chain.

There are already several presentation formats available that map onto the TTCN-3 core notation; this set of formats is extendible. So, at the test specification end of the chain, TTCN-3, or more precisely: its core notation can be the format into which all test specifications are converted. Regardless of how the tester enters it – either some text or graphical format – the notation to link to automated execution can be the same.

The execution and reporting parts of the chain can also be standardized with TTCN-3. There is an architecture defined with codecs and adapters, extendible to all types of protocols, platforms, and interfaces. The standards defined, and being defined, for this architecture can serve as the basis for standardization for execution and reporting tools.

We will show how TTCN-3 and its architecture can be used over the length of the chain for domains others than in which TTCN-3 historically has been applied. We will do this by presenting the results of our efforts to apply it to financial systems. We have added TestFrame Language (TFL) as a presentation format and we integrated interface modules to GUI applications into the architecture. We will conclude with our suggestions for the next steps toward standardization of the test automation chain.

Leon Wolters

Leon Wolters is a senior software engineer. Since 1998 he has been developing test tools at LogicaCMG, starting with the TestFrame Engine. He has been architect and lead engineer of numerous software testing applications that are used in various industrial domains. Leon has written and spoken extensively on the subjects of testing and test automation.

Jos van Rooyen

Jos van Rooyen is a senior test manager / test advisor at LogicaCMG. He works for major clients in different branches. He has been involved in testing since 1990. He is experienced in multiple disciplines of software testing, such as test management, test automation, and testing of packaged software. He has published several articles and has spoken about test topics on seminars. Jos teaches courses on test management. On a regular base he lectures on several universities.

Erik Altena

Erik Altena is a test specialist at LogicaCMG. Operating in different roles, he gained a broad testing experience; from test analysis and automation to test management and consulting. He has worked on a variety of projects in financial services and public sector. Erik teaches many test courses on testing, TestFrame and testing techniques.

TTCN-3 for Automated Testing Railway Control Systems *

S. Blom¹, N. Goga^{2,4}, N. Ioustinova², J. van de Pol^{2,4}, A. Rennoch³, N. Sidorova⁴

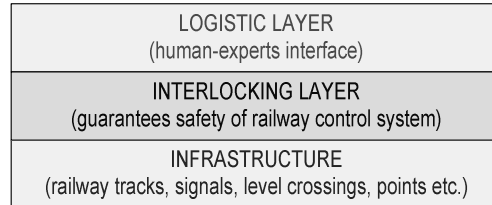
1 University of Innsbruck, Stefan.Blom@uibk.ac.at

2 Centrum voor Wiskunde en Informatica, ustin@cwi.nl, Jaco.van.de.Pol@cwi.nl

3 Fraunhofer FOKUS, axel.rennoch@fokus.fhg.de

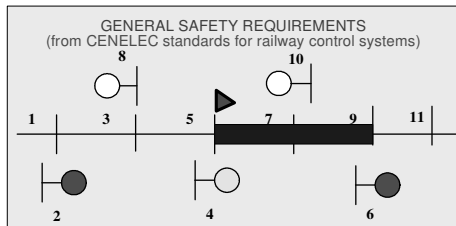
4 Eindhoven University of Technology, n.goga@TUE.nl, n.sidorova@tue.nl

Railway control systems are safety-critical, so we have to ensure that they are designed and implemented correctly. Testing these systems is a key issue. Prior to system testing, the software of a railway control system is tested separately from the hardware. The interlocking is a layer of railway control systems that guarantees safety. It allows to execute commands given by a user only if they are safe; unsafe commands are rejected.



Railway Control System

Railway interlockings are central to efficient and safe traffic management for railway infrastructure managers and operators [1]. In the European railway sector, the current target is to increase the proportion of railway transportation by 100-150% within a short period [2]. European integration requires new standards for specification and testing interlockings. Here we propose an approach to *testing interlockings* with TTCN-3. Together with engineers from ProRail, we have applied this approach to testing the interlocking of Hoorn-Kersenboogerd station.



A general safety requirement



Concrete infrastructure of Hoorn-Kersenboogerd station

Starting from the general safety/reliability/maintainability etc. requirements (CENELEC standards EN 50126/50128/50129) for railway control systems, we have developed a test suite for testing the interlocking of Hoorn-Kersenboogerd station. The standard requirements are formulated for a station with a general configuration. All requirements are of the form: initial situation, action, expected results. To develop test cases for the Hoorn-Kersenboogerd station, we had to (1) map the general configuration to a particular configuration of the station; (2) map the initial situation to the stimuli for the SUT; (3) map the final situation to the output values expected from the SUT; (4) define default values for objects of the station that are not involved in the tested situation but still can influence it; (5) formulate time requirements for tested actions.

We specified the test cases in TTCN-3. TTCN-3 is a language for specification and *automated* execution of test suites. The language, its operational semantics, the general structure and interfaces of a TTCN-3 test system are standardized by ETSI [3,4,5,6]. Originally developed for testing telecommunication system, TTCN-3 supports real-time testing. A TTCN-3 test executable has predefined standard interfaces [5,6] that allow to offer TTCN-3 solutions that do not depend on the implementation details of a system under test (SUT). In TTMedal project [7], we applied TTCN-3 to testing railway interlockings.

The software part of the interlocking is a program that consists of a large number of guarded assignments. The program defines a *control cycle* that is repeated by the system. The length of the control cycle is fixed by design. Although the environment of the system changes continuously, the system sees only snapshots of the environment made at the beginning of each control cycle. Thus the environment is discrete from the system's point of view. The system is *timed*, delays are used to guarantee safety.

We test the interlocking's software without the corresponding hardware. Therefore, we *simulate the code* of interlockings during the test execution. To ensure repeatability of testing results, we need to establish *control over time* in the SUT and in the test system. We have to guarantee that the SUT and the test system agree on time. We propose a solution with *simulated time* where time is modeled as a discrete clock. Since TTCN-3 was originally developed for real time testing, we have to make extra efforts to implement simulated time in the TTCN-3 framework. Our solution for testing with simulated time [8] is based on an extension of Dijkstra's well known algorithm for distributed termination detection [9].

Applying our approach to testing the interlocking of Hoorn-Kersenboogerd station, we have covered the whole test-process starting from the development of test cases, proceeding with the implementation of the test system, and finally with the *automated execution* of tests and the interpretation of results. The approach allows detecting violations of safety requirements in interlocking software. The solution for testing with simulated time in TTCN-3 is also applicable to other systems with similar characteristics.

We showed that TTCN-3 can be used as a standard language to specify test suites for railway applications. It is suitable to provide standard reusable platform-independent test suites for railway interlockings. Clear semantics and high maintainability of TTCN-3 allow not only to provide high-quality test suites but also to reduce costs for testing and maintenance of test systems on the long run.

This work was done within TTMedal project (Test and Testing Methodologies for advanced languages) where TTCN-3 was also applied by a team of Daimler Chrysler to test automotive systems and by teams from NetHawk, Nokia and VTT to test modern telecommunication systems. More information about TTCN-3 and using it for various domains is available on the TTMedal web-page [7]. We thank FOKUS (www.fokus.fraunhofer.de), Testing Technologies (www.testingtech.de) and Conformiq (www.conformiq.com) for providing us with TTCN-3 tools and training in this project.

References

- [1] European Standards for Railways Interlocking Systems, www.euro-interlocking.org
- [2] Safe technology for Rail Transport, www.sintef.no
- [3] ETSI ES 201 873-1 V2.2.1 (2003-02). Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. www.etsi.org.
- [4] ETSI ES 201 873-4 V2.2.1 (2003-02). Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 4: TTCN-3 Operational Semantics.
- [5] ETSI ES 201 873-5 V1.1.1 (2003-02). Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI).
- [6] ETSI ES 201 873-6 V1.1.1 (2003-02). Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 6: TTCN-3 Control Inter-face (TCI).
- [7] TT MEDAL: "Test and Testing Methodologies for Advanced Languages (TT-Medal)" www.tt-medal.org.
- [8] S. Blom, N. Ioustinova, J. van de Pol, A. Rennoch and N. Sidorova, Simulated time for testing railway interlockings with TTCN-3, To appear in Proc. of 5th International Workshop on Formal Approaches to Testing of Software, FATES 2005, Lecture Notes in Computer Science, ©Springer-Verlag, 2005. (see www.cwi.nl/~ustin/stime.htm for more details)
- [8] E. W. Dijkstra, W. H. J. Feijen, and A. J. M. v. Gasteren. Derivation of a termination detection algorithm for distributed computations. *Information Processing Letters*, 16(5):217-219, June 1983.

“Defect registration tools as knowledge base”

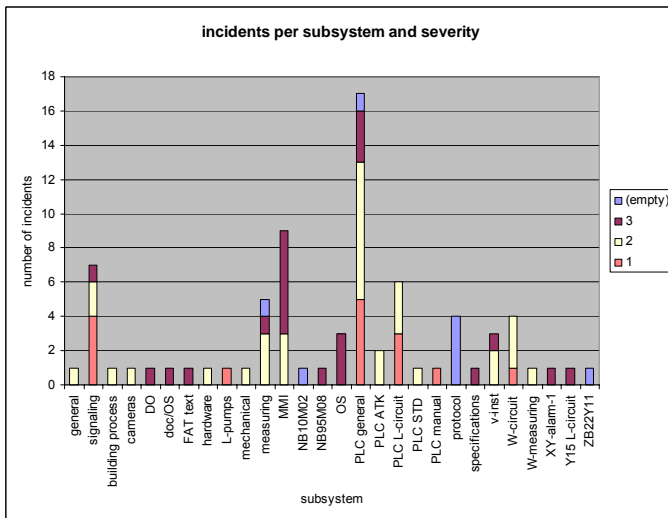
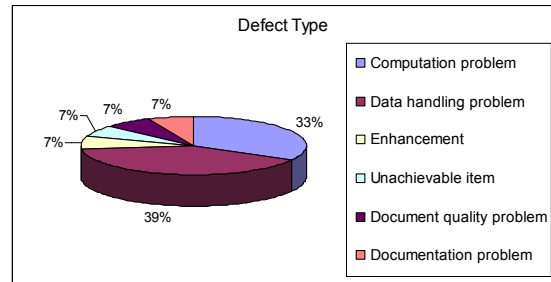
By Ed Brandt (Refis System Reliability Engineering)

Following the (automatic) execution of test cases, defect registration and tracking is a feature of test tools that is commonly used. This article shows the importance of this feature, its potential use and challenges. A database with defects and incidents is more than just a registration and tracking tool. In fact it's a knowledge base for improvement of test and development processes. The article will show how to use the data for various valuable metrics and management information. But it will also show that the value could even be bigger if we were able to link this information to failures that occur in production. How can tools support us there?

The benefits of a well structured test incident and defect database are obvious. Besides accurate logging of incidents, most tools provide some sort of work flow management for defect repair and management information on status and progress. But the incident database is in fact also a knowledge base. It shows in which parts of a system defects are concentrated, what kind of defects are common and the effort that is needed to fix them. Reliability assessments can be applied to quantify the failure intensity of single software products. And metrics like “defect removal rate” can help us to identify possible areas of improvement in testing and development.

Analysing defects

From an enhancement project on a travel expenses system for a large bank, the next figure shows the different types of defects recorded. Data handling problems occur more often than computational problems. This type of basic graphics can be produced by almost any registration tool, provided the data is actually captured.



The figure on the left comes from a process control system for an object of Rijkswaterstaat. It looks like the previous one, but more detailed and therefore better helps to recognize the weak parts of the system as a whole. Not only contains PLC-general the most defects, it also contains the most severe defects. Again, this information can be derived from almost any tool, provided the appropriate data is captured during the analysis process.

You would expect analysis of defects like with these two simple and basic examples to be standard in every organization. But it's not. We tend to focus on defect detection and repair only, without bothering about the lessons learned for development. Or actually the lessons that we therefore do not learn!

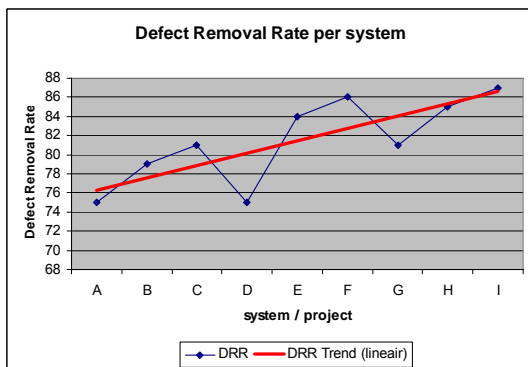
Trends in process improvement

Test process efficiency and improvement have been buzz words for the past five years or so. A little more sophisticated, but still available from standard defect registration tools, is the following set of metrics that can be very helpful to visualize trends in that area. The first one is called “Defect Removal Rate¹” (a.k.a. Defect Removal Efficiency). It shows the ratio of the number of defects found in one test phase to the number of defects found in the next phase (test or production).

In the example on the right, 19 defects were injected into the product during the definition study phase, 10 of which are detected during that phase. The others occurred during following phases; 2 at functional design, 1 at programming, 4 during test and 2 after deployment of the actual system, thus giving a 53% removal rate at the definition study phase.

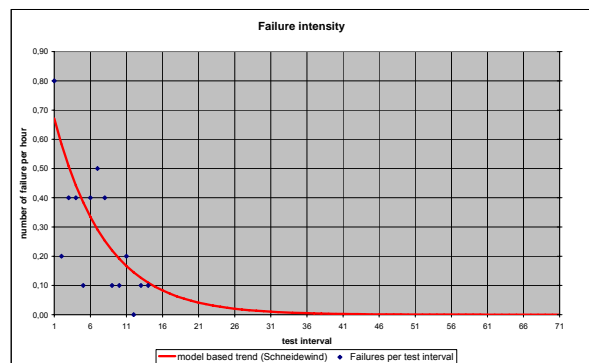
phase of defect detection	phase of defect injection				total defects detected	Defect Removal Rate
	definition study	functional design	technical design	programming		
definition study	10				10	53%
functional design	2	9			11	29%
technical design	0	4	4		8	22%
programming	1	4	5	8	18	27%
testing	4	9	1	20	34	69%
production	2	3	0	10	15	
total defects injected	19	29	10	38	96	84%

In the testing phase 34 defects were detected. This is 69% of the total of 96 defects minus the 47 defects that had already been removed in previous stages. In total 96 defects were injected into the system at the various development stages. 81 of them were detected prior to deployment. The overall Defect Removal Rate for this particular project or system therefore is 84%.



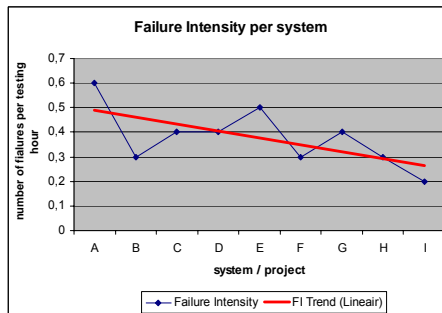
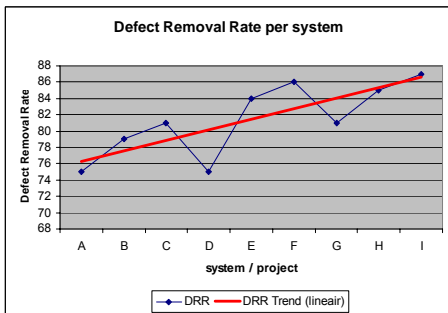
When this rate increases over subsequent projects, it indicates an increasing effectiveness of the quality control measures like inspections, reviews and testing, during system development stages. Using the Defect Removal Rate for benchmarking enables organizations to compare the effectiveness of the process against industry standards. The average US software defect removal rate is about 85 percent¹.

The second metric to observe in this respect, is the Failure Intensity. This is the number of defects occurred during a certain period of test execution time. It can be calculated during test execution (e.g. per test interval as shown in figure) and per project or system. Note that this metric also requires the hours spent on test execution. The metric may be an indicator for the quality of the system, provided testing quality is equal. And that is where the DRR comes in. Although DRR and FI metrics have a significant meaning by themselves, the power lies in the combination of the two. Tree of the possible combinations are described below:



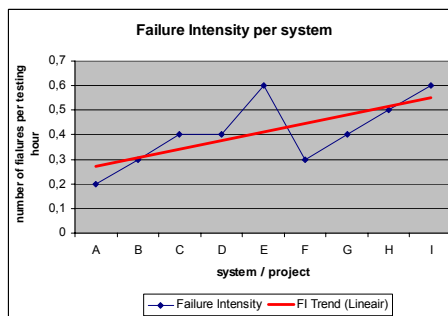
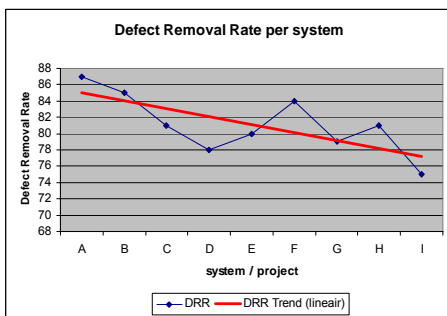
¹ Note that in some cases in literature Defect Removal Rate is also referred to as the number of defects found per hour of testing, i.e. the same as Failure Intensity.

1. Increasing DRR and decreasing FI over subsequent projects;



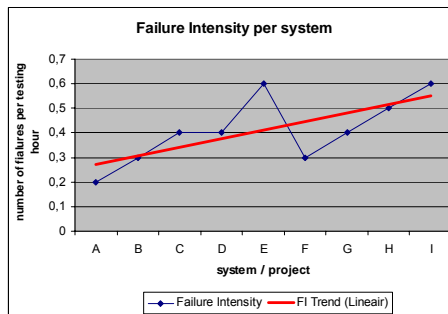
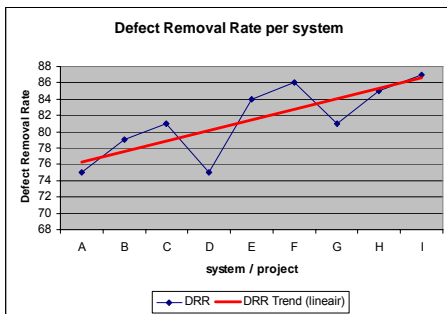
The best of all possible combinations: while the effectiveness of testing is increasing the number of defects found per hour of testing is decreasing. It means improvement in both testing and development processes.

2. Decreasing DRR and increasing FI:



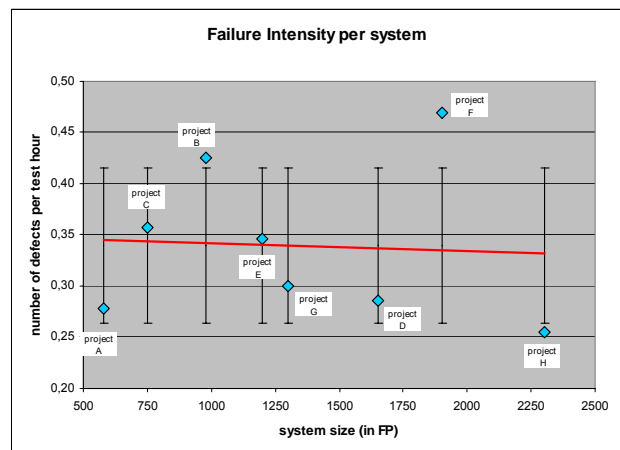
And this is no doubt the worst case scenario: even though the effect of testing is decreasing we still find an increasing number of defects per test hour. Both test and development processes need serious attention.

3. Both DRR and Failure Intensity are increasing:



The number of defects found per test hour is increasing caused by the increasing effectiveness of testing. The implementation risk for the organization is declining, but the quality of the development process still needs to be looked after.

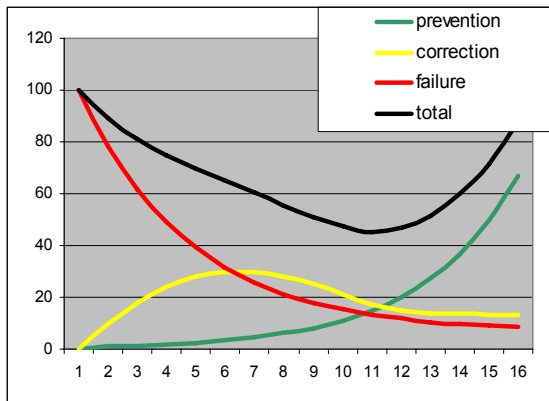
Since Failure Intensity may be influenced by the complexity and size of the system, the next addition may support previous metrics. As shown in the figure, the metric is extended with the size of the system (e.g. Function Points, SLOC). In this example the standard deviation is used to determine exceptional projects both in positive as in negative way.



Matching incidents from production with test results

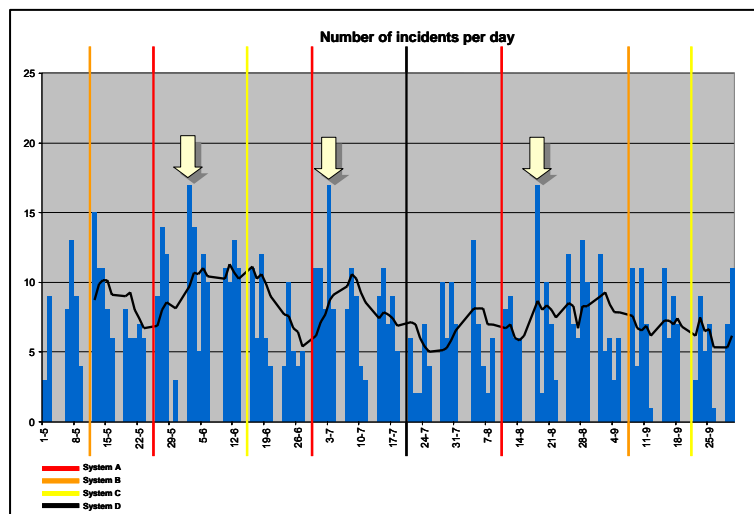
But it gets even better, when defects that occur in production could be matched to the appropriate development projects and test results. That would open a whole new range of information and shared knowledge between development and maintenance. Then we could really learn from our mistakes.

The figure “Incidents per subsystem and severity” from the first page, could be expanded with information from a helpdesk, showing the actual incidents from the production phase. It would show the gap that testcases has left in the system. And that would enable us to improve techniques for designing testcases. It would also allow us to verify and improve the predictions of system reliability and chance of failure.



And finally we would be able to really break down quality costs, determine the optimum and show the return on investment of testing. Quality costs in this case are determined by costs of production failure and costs of prevention and correction in development phase. The method for calculating the optimum is this is based on the following assumption (see figure on the left); costs of production failure decreases when more is spent on prevention and correction. At least to certain extend. Costs of correction can be limited when more is spent to prevent defects from being injected. Also here, at least to certain extend. The total costs of quality shows an optimum where the total of all tree ingredients is low.

The matching however of defects found in production with the prior development and test process, is difficult. The figure on the right shows the actual number of incidents per day and (in vertical lines) the implementation dates of major releases of the four most important systems of a large Dutch non-profit organization. It is obvious that only in some cases the increase of incidents can be related to prior releases. And this is only at high level.



In most organizations it appears to be virtually impossible to link an individual defect in production to its origin in a software engineering project. This is

mainly caused by the fact that incident management is aimed at service level maintenance. The origin of a defect is of less importance. But another problem is that even if we would want to capture the information, the available tools do not support this. Should defect registration within test tools also be made suitable for ITIL-like incident and problem management processes? Or should we use helpdesk and service level management tools as defect registration base during test phase?

For a recent website project of a city council, test incidents were registered in HP Openview. The tool served its primary objective: capturing data for managerial purposes like workflow management for defect repair. No surprise of course since that's one of the main objectives of a helpdesk tool. Capturing data analysis and relationships between defects appeared to be less obvious. But since the tool includes a link to the configuration management base, it should be easy to relate future incidents to previous test activities. Again, provided the data is captured during the process of incident management.

Challenge for the future

Making sure all data is available is probably the biggest challenge. As stated before, most tools offer many ways to capture data and turn it into information and even knowledge. But we can only benefit from this huge knowledge base, if we feed it with the necessary data. To teach people to capture and register all relevant data is therefore essential. And it would be helpful if tools were designed to support that. Or maybe the ultimate solution would be a next generation of tool suites; covering the entire system life cycle?

About the author

Ed Brandt is working in information technology since 1982. His experience extends from software engineering to project management and consultancy in both national and international projects. He specialised in software testing since 1996 and is the author of various articles and papers on this subject. He is the co-founder and owner of Refis System Reliability Engineering. Refis is specialized in metrics for quality systems and reliability analysis on software. Refis' client base consists of major companies and organisations in the Netherlands like Rijkswaterstaat, ABN AMRO Bank, Delta Lloyd Insurances and ING Bank. (see www.refis.nl)

ⁱ Capers Jones, "Software Quality", International Thomson Computer Press, 1997

Proving test coverage at requirements level

Han van Gerwen

In many projects, the progress of test execution is measured and reported as the percentage of test cases that are completed. The customer however is not primarily interested in the progress of the test cases, but he wants to know which features or requirements are qualified and ready to be delivered. Especially in case of an incremental delivery process, this is important information, because for each increment it must be clear which functionality is ready for delivery.

To be able to provide this information we defined and installed a process at one of our customers in a large project. This process links all test cases and its execution results to the requirements, using the tools DOORS from Telelogic and TestDirector from Mercury. With this process we were able to:

- Show that all requirements were sufficiently covered by tests
- Show which requirements were sufficiently qualified and ready for delivery
- Show the progress in percentage of accepted requirements
- Determine which test cases must be executed to validate only the changed functionality.
- Show which requirements were covered by subsystem tests and need no additional tests at system level.

This provided a lot of insight in the actual quality of the product features. Furthermore it made testing more efficient, because less tests were required at system level and for subsequent releases where only part of the functionality had changed

The presentation addresses the following topics:

- The process to get the information
- Implementation in the tools
- Examples of reports that were used

Profile

Han van Gerwen is a consultant working for Ordina, with more than 20 years experience in different phases of the software and system development process in the area of technical automation. Since 8 years he specialised in integrating and testing systems, both embedded products and industrial systems. His current focus is on the definition and implementation of integration- and test-processes and the introduction of tools to support these processes.

LaQuSo

automatic on-the-fly testing
of reactive systems

Pieter Koopman
Radboud University Nijmegen
The Netherlands

the scene

- **testing:**
 - planned experiments with implementation to determine quality aspects
- **functional testing:**
 - focus on behaviour of (software) systems
 - relation between input and output
- **reactive systems:**
 - reaction depends on input and state
 - state determined by the history

2

specification of reactive systems

- based on FSM
 - e.g.: machine produces Coffee or Tea after making a choice by pressing a button and inserting a coin

3

nondeterminism

- nondeterminism needed if
 - system is really nondeterministic
 - not the entire state is known in the specification
 - machine produces coffee if there is water and beans, but specification does not no if this condition is met

4

observable nondeterminism

- the output tells with transition in used
 - output Coffee: system is in state Idle
 - output Coin: System remains in S_coffee

5

nonobservable nondeterminism

- outputs are also identical
 - based on the input/output we do not know which transition is chosen
 - after inserting a coin we discover the state

6

ESM: Extended State Machines

- state, input and output can be any type
 - in particular there can be arguments

$Can\ c \mid c \leq n / n=c; \text{ repeat } c\ Coffee$
 $n=0$
 $Coin / n+=1; []$
 $Button \mid n>0 / n-=1; [Coffee]$

otherwise: stay in state; no output

- transition table becomes infinite (or too large)
- specification by function (algorithm)


```
spec :: Int x Input -> ( Int, [ Out ] )
spec (n, Can c) = if (c <= n)
  then (n-c, repeat c Coffee)
  else (n, [])
```

...

testing

- SUT: System Under Test
- assumptions: SUT is black box state machine
 - apply input, observe output
 - behaves as extended state machine
 - input enabled
 - each input can be applied in any state

conformance

- allowed specifications
 - nondeterministic extended state machine
 - can be partial
 - noting defined for some states and inputs
- implementation
 - black box
 - input enabled state machine
- conformance relation:
 - if the specification does not cover an input for some state, anything is allowed
 - otherwise, only the specified transitions are allowed

example of conformance

- SUT right is conform to specification left

state	input	output	state
Idle	Butcoffee	[]	Scoffee
Scoffee	Coin	[Coffee]	Idle
Idle	Butcoffee	[]	Scoffee
Scoffee	Coin	[Coffee]	Idle
etc.			

can be repeated forever, but we do not know that

counterexample of conformance

- SUT right is *not* conform to specification left

state	input	output	state
Idle	Butcoffee	[]	Scoffee
Scoffee	Bang	[]	Scoffee
Scoffee	Coin	[Coffee]	Idle
Idle	Butcoffee	[]	Scoffee
Scoffee	Bang	[]	Scoffee
Scoffee	Coin	[Tea]	Scoffee

no new state: error found

manual testing

- manual testing
 - human supplies input and checks output
- advantages
 - simple
 - flexible
- disadvantages
 - error prone
 - test are slow
 - dull
 - especially for regression tests

always needed!

test script

- description of inputs and allowed outputs
 - branches for nondeterministic specifications
 - can be executed automatically

state space explosion

- state space explosion:
 - we have to consider an unbounded number of states and/or transitions
 - occurs with parameterized states and labels

Can c | c ≤ n / n=c; repeat c Coffee

Button | n>0 / n=1; [Coffee]

script based testing

- script can be executed automatically
- advantages
 - testing is documented
 - testing is fast
 - reliable
 - easy to repeat
- disadvantages
 - scripts needed
 - generated by human or tool
 - can be out of date!
 - state space explosion
 - scripts are used partially
 - better testing requires more scripts

on-the-fly testing

repeat N times:

- select input allowed in current state;
- apply input to SUT and observe output;
- if output allowed by specification compute new state;
- else report error;

- advantages
 - testing is fast and reliable
 - no state space explosion
 - better testing by changing a parameter
 - testing always corresponds to current specification
- on-the-fly testing tools:
 - Torx, Spec Explorer, T-Uppaal, Gast, ..

needed for automatic on-the-fly testing

- detailed specification to
 - determine allowed inputs
 - compute new states given current state, chosen input and observed output
- interface to System Under Test
 - apply input, obtain output
 - reset: bring SUT to initial state

Can c | c ≤ n / n=c; repeat c Coffee

Button | n>0 / n=1; [Coffee]

what specifications are suited

- requirements
 - determine allowed inputs for current states
 - check output for applied input
 - compute new states
- finite state machines
 - transition tables
- extended state machines
 - statecharts
 - Harel
 - UML
 - transition functions
 - an algorithm computing output and target state
 - ...

inappropriate statecharts

- not every specification can be used for on-the-fly testing
 - inputs allowed must be clear
 - transitions must be completely specified
 - other statecharts can express an informal idea, but contain not enough information

Pieter Koopman test404 2005 19

UML models

- specification can be depicted by a statechart
 - statecharts are part of UML
 - but, not every statechart is a good specification
- UML is widely used
 - wanted tool suite:

Pieter Koopman test404 2005 20

some applications with Gast

- FSM in industrial context
 - over 300 states and 11,000 transitions in C++
 - specification generated from table in MS-Excel
 - *error found although it was proven to be correct*
- Java-card electronic purse
 - extended state machine, too large for a table
 - errors found in all 25 mutants (each within 1 sec)
- Web-server
 - under development, first results look fine
- usually errors are found fast
 - typically within seconds
 - but it is possible to construct errors that are hard find

Pieter Koopman test404 2005 21

conclusion

- manual testing is good
 - gives impression of the software quality quickly
 - flexible
- script based testing is better
 - executing tests is fast and accurate
 - easy to repeat tests
- model based on-the-fly testing is best
 - tests always conform the current specification
 - use a small specification instead of large test suites
 - number of tests controlled by parameter
- simple formal specification needed
 - generation from UML is feasible

Pieter Koopman test404 2005 22

future work

- applications
 - webservers
 - large embedded systems
 - Dutch biometric passport
 - contains smart card with biometric data
 - ..
- theory/tool development
 - using UML specifications
 - determine quality of tests
 - determine quality of system after tests
 - time in specifications and tests
 - composition of specifications
 - interaction with model checker
 - asynchronous communication in model
 - ..

Pieter Koopman test404 2005 23

Case study AllianzNet: Automated versus manual testing

by Guido van Leeuwen and Hans van Loenhoud

The AllianzNet project is a major project from Allianz Nederland Schadeverzekeringen, adding a web-based front-end to the general insurance system as a replacement for the original mainframe front-end application. With this new Internet application brokers and agencies can communicate directly with the general insurance system of Allianz. This results in higher quality, faster handling and lower costs.

240 man months of IT effort have been spent on the design, development and implementation of the first phase of this project, resulting in the on-line offering and acceptance of some 15 different insurance types. More than 20% of the total IT effort was spent on testing, acceptance testing by the end users not being included.

The general insurance system of Allianz is very complex, as a consequence of its flexibility and customer intimacy. It contains more than 30 years of knowledge on tailor-made insurance solutions, offered to both the commercial and the personal market. The complexity of the system is, of course, reflected in the complexity of testing the system, both on the side of the Internet application and on the interface to the mainframe system.

Senior management realised that testing would be a critical success factor to the project. Right from the start, they assigned a dedicated test team, which prepared an extensive testing programme based on a Master Test Plan. In this Master Test Plan, a significant role was attributed to automated testing, which eventually used up 20% of the total testing effort.

Automated testing was deployed during system and integration testing in three areas:

- Premium comparison
Due to the complexity of business rules on insurance premiums, manual output prediction for premium calculations is very labour-intensive. Therefore, an automated test script calculated insurance premiums on the original mainframe system. By definition, this yields the right figures, which were subsequently compared to the calculations with the same input on the Internet application.
- Regression testing of new releases and bug fixes
During the design and build phase of the system, a growing regression test set has been developed that is used after each delivery of a new build.
- Performance testing
As this was, within the Dutch Allianz organisation, the first implementation of a major interactive Internet application, performance expectations were unknown and therefore performance was explicitly tested.

The test team discerned almost 1500 logical test cases, based on 250 different test conditions. About 300 test cases were subjected to automated tests. Altogether, a total of 700 findings has been recorded, some 10 % of these found by automated testing. As these 10 % of the findings consumed 20% of the testing effort, automated testing may seem to be a costly affair.

However, this applies only to the initial testing phases, where 40% of the effort was on automated testing. As testing proceeded, the relative effort gradually decreased to less than 15%.

The following lessons learned were derived from our experiences:

- In the beginning of the project, during system testing, the system under test was still unstable and many architectural and functional changes occurred. Automated test scripts built at that stage had to be updated time and time again. In this period, manual testing proved to be cheaper and more effective than automated testing. In retrospect, a later start of automated testing would have been better.
- Later on, during the integration test phase, the system gradually became more (functionally) stable. Bug fixing, resulting from integration and user acceptance testing, required frequent cycles of regression testing. In that phase, automated testing quickly proved its value. Especially at the end of the project, when the externally committed implementation date approached, regression testing of bug fixes could not have been done manually in the time available.
- Performance testing was used to simulate the future load of the system with up to 1000 concurrent users. Without test automation this could never have been accomplished. In this matter, manual testing is not a feasible option.

In the AllianzNet project, automated testing made a valuable, even indispensable contribution to the final success. Initial costs were rather high, in fact twice as much as manual testing. This can be seen as an investment, that pays back during the next stages of the project, by means of the reuse of the regression test set developed.

Guido van Leeuwen is a consultant of INQA, specialised in quality management of ICT processes. In the AllianzNet project he has the role of configuration manager and project secretary to the test team. His responsibilities include the inventory of the test objects, the control of the test environments, the co-ordination with the developers and the administration of the findings and the other project data.

Hans van Loenhoud is managing partner of INQA, a specialised management and consulting firm on quality in ICT. He is a senior test manager, well known within the Dutch testing community as vice-president and secretary of TestNet, the Dutch special interest group on software testing.

In the AllianzNet project he developed the testing strategy, wrote the Master Test Plan and introduced automated testing. He gives advice to Allianz senior management, both ICT and business, on quality and testing issues.

Timed Testing with TorX

Abstract

Axel Belinfante^a and Henrik Bohnenkamp^b

^a University of Twente, 7500 AE Enschede, The Netherlands

^b University Aachen (RWTH), 52056 Aachen, Germany

TORX is a testing tool that implements the *ioco* testing theory. Whereas *ioco* testing is purely functional, TORX was recently extended to allow the testing of real-time properties. Real-time testing means that the decisions whether an IUT has passed or failed a test-case is not only based on which outputs are observed, given a certain prior accepted trace, but also *when* the outputs occur, given a certain prior *timed trace*. Our approach uses safety-timed automata as specification formalism. A distinguishing feature of our approach from earlier work in the literature is that the timed automata are allowed to be non-deterministic. Another feature is that we deal with *quiescence*, i.e. the detection of *absence* of output.

Safety-Timed Automata

TorX' testing algorithm expects labelled transition systems (LTS) with input- and output-labels as specification formalism. In order to incorporate safety-timed automata into TORX we derive zone-automata [1] from the original TA. Nondeterminism in the original TA is translated to non-determinism in the underlying zone automaton. TORX interprets the zone automata as LTS. Transparent to the user, an additional clock is considered in the zone computations, which keeps track of the time since system start. The valuations of this clock are time intervals which describe when certain actions can be taken. This information is used by TORX to determine the legality of event occurrence times.

Occurrence of events determine the valuation of the absolute clock, and an *instantiation* carried out by TORX takes care that relevant clock zones in the zone automaton are updated according to the event occurrence times.

Quiescence

An implementation is considered quiescence if it will never, for all times, produce an output without prior input. The concept of quiescence can thus be interpreted as a real-time property. Quiescence is actually a property that is not detectable, since it would require to wait for an output for all times. In practice it is therefore necessary to work with an approximation to quiescence, using timeouts.

We assume that an implementation is quiescent, if it does not emit an output for at least M time units, counted from the last event. M is an appropriately chosen constant. This assumption makes quiescence detection possible [3]. In order to do so, we add another clock to the zone computations, a clock which is reset to zero whenever an input or output has occurred. Only when this clock has a value larger than M , quiescence is allowed to be signalled. In the implementation, it is not really necessary to use a clock (which makes zone computations more expensive). The timeout M can be measured independently from the zone computations.

Implementation in TorX

The architecture of TORX comprises roughly four components: explorer, primer, driver, and adapter. The explorer is responsible to provide the LTS representation of the specification. The primer implements the test-derivation and execution. The adapter is implementation-specific and provides a standard interface between implementation under test (IUT) and TORX. The driver is the broker between primer and adapter.

In order to implement timed testing, a new explorer had to be implemented. It takes care of the zone computations and quiescence detection. The primer algorithm is basically the same as before, except that the on-the-fly handling of quiescence has to be switched off. The driver remained unchanged. Adapters have to be extended to provide mechanisms for time-stamping and timely application of inputs.

A special component, the *instantiator*, connected to the primer, had to be implemented. Inputs are under the control of TORX, and are in general enabled for a certain time interval. The instantiator is responsible to choose a concrete time at which to apply the corresponding input to the IUT.

Conceptually, dealing with time turned out to be a special case of dealing with symbolic data. TORX had already machinery implemented to deal with symbolic data, which made the incorporation of timed testing straightforward.

Related Work

This work has been strongly inspired by the work of Brandán Briones and Brinksma [3] and Mikucionis et al.[4]. A paper about this has been presented on *Formal Methods Europe* [2].

Acknowledgements

This research has been supported by the Senter/ESI project TANGRAM.

Authors

Axel Belinfante works as a Researcher at the University of Twente. He is the main Developer and maintainer of the testing tool TORX.

Henrik Bohnenkamp is a Researcher, formerly at the University of Twente, as of September 2005 at the University of Aachen, Germany.

References

- [1] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets.*, volume 3098 of *LNCS*, pages 87–124. Springer-Verlag, 2004.
- [2] H. Bohnenkamp and A. Belinfante. Timed testing with TorX. In John Fitzgerald, Ian Hayes, and Andrzej Tarlecki, editors, *Formal Methods 2005*, volume 3582 of *LNCS*, pages 173–188. Springer-Verlag, 2005.
- [3] Laura Brandán Briones and Ed Brinksma. A test generation framework for quiescent real-time systems. In J. Grabowski and B. Nielsen, editors, *Formal Approaches to Testing of Software (FATES '04)*, volume 3395 of *LNCS*, pages 64–78, 2005.
- [4] M. Mikucionis, B. Nielsen, and K. G. Larsen. Real-time system testing on-the-fly. In K. Sere and M. Waldén, editors, *15th Nordic Workshop on Programming Theory*, number 34, pages 36–38. Abo Akademi, Department of Computer Science, Finland, 2003.

Test automation in the alpha test of wafer scanners¹

Ivo de Jong², {Roel Boumen, Asia van de Mortel-Fronczak, Koos Rooda}³
ivo.de.jong@asml.com

In general, the goal of test automation is to reduce test time, to decrease the amount of manual labor and to increase the test coverage. In a time-to-market environment, like ASML, the primary goal of test automation is to minimize the *total test duration*. The *total test duration* is the time between test start and the moment that the system is fault free, so including diagnosing and fixing faults. Therefore, the *total test duration* is not only depending on the test case duration, also the diagnosis and fix duration (the *fix-loop*) influences the *total test duration*. Time to market can also be reduced by using a parallel test and fix strategy. In this strategy, available fixes are integrated on the system under test as soon as possible. Applying fixes as soon as possible results in fewer test cases failed in the remainder of the test phase, however the fixes may introduce new faults as well. The influence on *total test duration* of both applying test automation and applying a parallel test and fix strategy has been investigated.

In the presentation the influence on the *total test duration* of combining a parallel test and fix strategy and test automation is demonstrated. This effect is demonstrated using a discrete-event simulation model of the test process on an ASML case study. The test set in this case study consists of 54 functional test cases with an average duration of 122 minutes for each test. The order in which tests are executed is tester dependent and therefore considered to be random. This test phase is part of the software alpha test phase. The duration of the *fix-loop* in the alpha test phase is around 6 hours (360 minutes).

Test automation is simulated by reducing the test duration of each test case with a factor 10, 100 and 1000. Table 1 shows that the resulting *total test duration* is reduced to resp. 49%, 45% and 44% (the duration of the *fix-loop* is $t_{(D+F+AF)}=360$).

	Factor of the test case duration			
$T_{(D+F+AF)}$ [min]	1	10	100	1000
360	100%	49%	45%	44%
180	78%	27%	22%	22%
90	67%	17%	12%	11%

Table 1 The effect of reducing test duration and decrease of $t_{(D+F+AF)}$ using random test selection.

The slight reduction of *total test duration* at a test automation factor of 100 and 1000 is caused by the *fix-loop*, which is now the bottleneck in the process. To illustrate this, the *fix-loop* duration is also varied. The duration of the *fix-loop* is decreased to 180 and 90 minutes. Again, the test process is simulated for all four test case duration factors (1,10,100 and 1000). The combination of reducing test case duration with a factor 10 and reducing the duration of the *fix-loop* to 90 minutes results in a reduction of *total test duration* to 17%. Again the reduction of the test case duration by a factor 100 or more does not result in significant reduction in *total test duration*.

Conclusion: The duration of the *fix-loop* must be taken into account when test automation is applied and a parallel diagnosis and fix strategy is used. For this case study the combination of test automation and *fix-loop* duration reduction is most beneficial.

¹ This work has been carried out as part of the TANGRAM project (www.esi.nl/tangram) which is partly sponsored by the Dutch Ministry of Economic Affairs under grant TSIT2026.

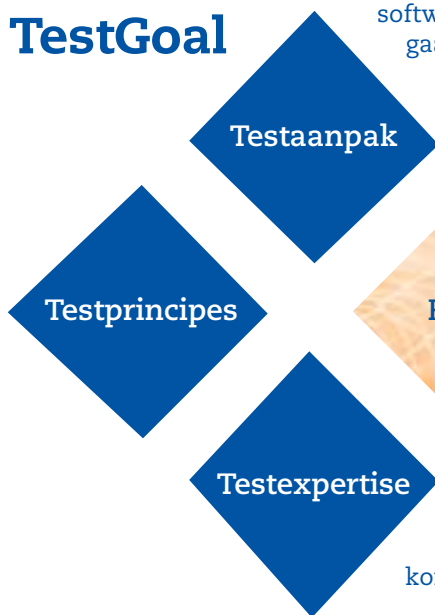
² ASML and TU/e – Systems Engineering

³ TU/e – Systems engineering

Company Brochures

TestGoal, result driven testing

TestGoal



De toegevoegde waarde van testen

De toepassing van software dringt steeds verder door in onze samenleving. Goede software leidt tot processen die efficiënter verlopen en tot kwalitatief hoogwaardige producten. Ondanks het grote belang van goede software en de enorme potentie gaat er bij de totstandkoming van software helaas nog veel mis. Vaak wordt software te laat opgeleverd, kost meer dan begroot en zijn uiteindelijk functionaliteit en performance niet conform verwachtingen. Hierdoor is het rendement van software niet altijd voldoende. Bij de realisatie van software is testen een essentieel instrument om eerdergenoemde problemen te voorkomen.

Professioneel testen van software biedt u concrete handvatten om de kwaliteit van software daadwerkelijk te verbeteren. De kwaliteit van software wordt met testen stap voor stap inzichtelijk gemaakt. Dit geeft u vertrouwen en zekerheid over uw ICT-systeem. Daarnaast kunt u bij de uitrol van ICT-systemen, op basis van de met testen opgedane kennis over gerealiseerde functionaliteit en performance, een goede aansluiting maken met de operationele processen. Kortom, testen levert u veel toegevoegde waarde.

TestGoal

Collis richt zich op de kwaliteitsverbetering van ICT-systemen en heeft veel kennis van en ruime ervaring in het werkveld van software kwaliteit en testen. Collis is goed op de hoogte van ICT-technologie, -ontwikkelingen en de business domeinen waarin geopereerd wordt. Collis consultants zijn vakmensen. Met opleiding, training, kennisdeling en intensief field-management professionaliseert Collis voortdurend haar dienstverlening.

Collis geeft u altijd inzicht en overzicht in voortgang en status van het testwerk en de kwaliteit van het onderhanden zijnde testobject. Met een no-nonsense instelling, een sterke betrokkenheid en een optimale focus op resultaat. Met deze gerichtheid en de ruime vakkennis en ervaring voegt Collis veel waarde toe aan ontwikkeling en gebruik van ICT-systemen.

De Collis testfilosofie en -aanpak is gebundeld onder de naam TestGoal. Met TestGoal biedt Collis een concrete testoplossing, gebaseerd op krachtige testprincipes en volledig gericht op het toevoegen van waarde voor de klant: *result driven testing*.

TestGoal heeft tien testprincipes als fundament. De TestGoal testprincipes zijn een korte en krachtige formulering van dat waar Collis voor staat.

TestGoal omvat daarnaast een concrete testaanpak met een gestructureerde procesgang, waarin de focus op resultaat optimaal wordt ondersteund.

TestGoal testprincipes



Focus op resultaat

Een goede tester heeft altijd de focus op het resultaat. Met als doelstelling het bijdragen aan een betere software kwaliteit.

Het oplossen van met testen gevonden fouten verbetert de kwaliteit van software. Hierdoor worden risico's geminimaliseerd en wordt gebouwd aan vertrouwen in uw ICT-systeem. Met testen wordt bovendien goed inzicht verkregen in de feitelijke werking. Uw ICT-systeem kan met dit inzicht beter worden ingepast in de bedrijfsprocessen en is daarmee 'fit for purpose'.



Bouw aan vertrouwen

Door intensief te testen en de gevonden problemen op te lossen worden risico's geminimaliseerd

tot een acceptabel niveau. Testen is niet alleen het zoeken van fouten in andermans werk. Het is een activiteit van positief kritisch zoeken naar het goede in het systeem. Dat creëert vertrouwen in het ICT-systeem en verschaft zekerheden met betrekking tot de in productienaam.



Neem verantwoordelijkheid

Goede testers nemen hun verantwoordelijkheid. Ze behartigen het testdomein en zijn sterk betrokken

bij de gehele ICT-lifecycle; van de wens en het idee tot en met het gebruik en beheer. Een goed testproces is effectief en efficiënt ingericht. Zodat u niet wakker hoeft te liggen van potentiële problemen die zich zouden kunnen voordoen in de productie. Aan het einde van het testproces geven goede testers een vrijgaveadvies waar ze 'de handen voor in het vuur durven te steken'.



Testen is een vak

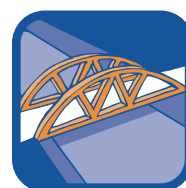
Testen is geen kwestie van 'knoppen drukken'. Het is een vak dat erop gericht is om aannemelijk te maken dat een ICT-

systeem goed functioneert. Een goede tester is ervaren en heeft aantoonbare kennis van alle mogelijke teststandaarden, -methodieken en -technieken. Daarbij is het belangrijk om in de praktijk de beste elementen van deze methodieken te gebruiken, waardoor een optimaal resultaat wordt verkregen.

Een goed testtraject is goed voorbereid én gestructureerd. Er zijn duidelijke en werkbare testplannen, testspecificaties en testscripts, die natuurlijk wel ruimte bieden aan de creativiteit van de tester. Er wordt bijvoorbeeld bij de sterk in opkomst zijnde agile ontwikkelmethodieken meer en meer een beroep gedaan op deze creativiteit.

Tot de bagage van een goede tester behoort naast testkennis en ervaring in de breedte ook het beheersen van specialismen. Zoals bijvoorbeeld testautomatisering, security testen en load-, stress- en performance testen.

Een goede tester heeft ook veel kennis van ICT in het algemeen en systeemontwikkeling in het bijzonder. Daarnaast is bekendheid met de businessdomeinen waarin gewerkt wordt een vanzelfsprekendheid.



Sla bruggen

Goed testen doe je door met één been in de business te staan en met één been in de techniek; sla bruggen tussen ontwikke-

laar en gebruiker, tussen het project en de lijn en tussen vernieuwing en 'going concern'. Tegelijkertijd dien je met testen snel te anticiperen op ontwikkelingen en wijzigingen in technologie en business. Daarbij dien je continu rekening te houden met 'bewegende' requirements, risico's en verwachtingen.



Test gefaseerd

De ervaring leert dat gefaseerd testen de beste software kwaliteit oplevert. Per testfase moeten allerlei testen gedaan

worden. Dit gebeurt niet lukraak maar risico gebaseerd en gestructureerd. Bepaal welk soort testen gedaan worden en waar de nadruk ligt. Geef aan welke risico's met de testen gereduceerd worden. Test bij voorkeur eerst de componenten, vervolgens de integratie hiervan en de aansluiting met de bedrijfsprocessen.

Verdeel de kwaliteitsattributen over de verschillende testsoorten. Test de functionaliteit grondig en goed, maar vergeet vooral zaken als performance, interoperabiliteit, gebruiksvriendelijkheid en beveiliging niet.

Hanteer een duidelijke projectaanpak, plan en faseer de testactiviteiten.

Met gefaseerd testen wordt het stap voor stap duidelijk of het te testen systeem aan de juiste verwachtingen voldoet. Zo wordt het naar het einde van het testtraject steeds aannemelijker gemaakt dat een systeem in productie kan worden genomen.



Testers faciliteren de gehele ICT-lifecycle

Testen is een onlosmakelijk onderdeel van de ICT-ontwikkelketen en -lifecycle.

Met testen stel je eisen aan voorgaande en volgende schakels in de keten. Hierop moet je regie voeren, daarbij dien je anderen op hun verantwoordelijkheden te wijzen. Testen doe je altijd professioneel, niet alleen tijdens de ontwikkeling van ICT, maar ook tijdens het gebruik en beheer.



Geef overzicht en inzicht

Bij testen kijk je naar veel aspecten vanuit verschillende invalshoeken. Daarbij is het zaak om zowel 'van

bovenaf' naar het te testen ICT-systeem en testproces te kijken, als de details goed te kennen. Zorg als tester voor het hebben en houden van overzicht en inzicht. Door een transparante werkwijze te hanteren kun je dit overzicht en inzicht makkelijk ontsluiten en delen met alle overige betrokkenen. Dat helpt bij het halen van gezamenlijke doelen en het boeken van resultaten.



Zorg voor herbruikbaarheid

Een goede testaanpak is zodanig ingericht dat opgeleverde zaken herbruikbaar zijn. Dit komt de efficiency van

vervolgtrajecten ten goede en voorkomt dat wielen meer dan eenmaal worden uitgevonden. Bijvoorbeeld door duidelijke testspecificaties op te stellen. Of door het automatiseren van testgevallen ten behoeve van regressietesten.



Testen is leuk

Testen is een mooi en leuk vak. Het is uitdagend om samen met alle betrokkenen een kwalitatief goed en renderend ICT-

systeem tot stand te brengen en te houden. Daarbij vervul je als tester een voortrekkersrol.

De TestGoal testprincipes van Collis zijn gebaseerd op de klantgerichte houding, het vakmanschap en de zeer rijke ervaring van de Collis consultants. Zij acteren bij alle dienstverlening conform bovenstaande principes. Daar kunt u op rekenen en hen altijd op aanspreken.

TestGoal, onze resultaatgerichte aanpak

De TestGoal testaanpak heeft de testprincipes als uitgangspunt en is complementair aan de gangbare testmethoden en technieken. Collis vervangt deze methoden niet, maar richt zich binnen deze methoden nog meer op de toegevoegde waarde voor de klant. Daarbij testen we niet om het testen en gebruiken we geen groter kader dan nodig is.

TestGoal is een concrete, op maat te snijden aanpak waarmee op een doelgerichte, efficiënte manier de kwaliteit van het te testen systeem kan worden vastgesteld. De aanpak wordt ondersteund met een tool met 'ready to use' templates, concrete checklists en een krachtige webbased rapportage portal. Via de portal krijgt u op elk gewenst moment inzicht in de kwaliteit van het in test zijnde ICT-systeem en actuele voortgang en status van het testproces.

Diensten en expertise

Alle diensten van Collis zijn gebaseerd op de omvangrijke kennis van en ervaring op het gebied van software kwaliteit en testen in brede zin. Van procesinrichting, teststrategie en testmanagement tot en met testuitvoering. Dit doet Collis in de vorm van consultancy, resourcing van projecten, maatwerkprojecten en training.

Consultants van Collis werken op basis van de TestGoal testprincipes en de TestGoal testaanpak. Als professional zijn we daarnaast zeer vertrouwd met de meest voorkomende standaarden en testmethodieken. De kracht van TestGoal is dat deze standaarden en methodieken eenvoudig en zonder restricties toe te passen zijn, met behoud van de toegevoegde waarde van TestGoal.

De testconsultants van Collis zijn gecertificeerd volgens de internationaal breed erkende ISEB norm op het gebied van software testen. De senior consultants beschikken over het ISEB Practitioner certificaat.

Collis laat u scoren!

Collis is opgericht in 1997 en geldt als een sterk groeiende en succesvolle organisatie binnen de nationale én internationale ICT-branche. Met als core business het bieden van hoogwaardige oplossingen op het gebied van software kwaliteit & testen en e-Business.

Collis is ISO 9001:2000 gecertificeerd. Dit geeft extra waarborg op hoge, herhaalbare kwaliteit. Onze processen zijn aantoonbaar ingericht op maximale klanttevredenheid en het voortdurend verbeteren hiervan.



De Heyderweg 1
2314 XZ Leiden
Nederland

T +31 71 581 36 36

F +31 71 581 36 30

E info@collis.nl

I www.collis.nl

Collis, het logo van Collis en TestGoal zijn geregisteerde trademarks van Collis BV.



CTIT (Centre for Telematics and Information Technology) of the University of Twente is the largest academic ICT research institute in the Netherlands. It conducts research on the design of advanced ICT systems and their application in a variety of application domains. Over 400 researchers actively participate in the CTIT programme and the budget amounts to 23 M-€. CTIT closely cooperates with many public and private organizations.

Disciplines and research themes

Major scientific, economic and societal challenges are to be expected in areas where the interests of various disciplines overlap. The awareness that ICT has an impact on daily life, on business operations, on government, and on science is growing. The challenges are there both for research on ICT and for the various fields in which ICT is used.

Integration of technology-based research and its application in specific domains is our clear focus. Therefore, the spectrum of research groups involved ranges from primarily technology oriented towards highly application oriented. Groups from the Electrical Engineering, Mathematics & Computer Science department focus on topics such as security, embedded systems & ambient computing, mobile & wireless communication and services, next generation Internet, performance, multimedia & virtual reality, and information & software engineering. These groups interact intensively with disciplines which are oriented more towards applications research, such as Business, Public Administration & Technology (production and logistics, e-commerce, e-governance, e-health), Engineering Technology (traffic and transport systems, product development, virtual reality) and Behavioural Sciences (web-based learning, knowledge management, contextual design and use).

The strength of CTIT is bringing together a sufficiently large number of researchers from different disciplines around a limited number of Strategic Research Orientations (SROs). An SRO is a large scientific programme combining research of at least five groups into a genuinely multidisciplinary programme, providing excellent opportunities for international top-level research. Within a SRO, research on certain aspects of ICT is combined with specific application areas to enhance the exchange of ideas about the ICT requirements of applications and the possibilities ICT can provide for these applications. Our six Strategic Research Orientations are:

- A-Services Internet (in the area of Telematics),
- Building Blocks for Ubiquitous Computing and Communication (in the area of Embedded Systems),
- Integrated Security and Privacy in a Networked World (in the area of Security),
- Natural Interaction in Computer-mediated Environments (in the area of Virtual Reality and Multimedia),
- eHealth (in the area of ICT and Healthcare), and
- eProductivity (in the area of ICT and Productivity).

Co-operation and innovation

CTIT maintains an extensive international network of academic and industrial partners, participating in active collaboration with ICT and manufacturing companies, universities and research institutes, health care organizations, financial institutes, governmental organizations, and logistics service providers.

CTIT also has special connections with high-tech start-up companies, often spin-offs of the university: HOMA Software, Recore Systems, Ambient Systems, Realise b.v., MAG Productions, TalkingHome, Utellus, Carp Technologies, Arkia, iGear, B-SIM, Imotech b.v., Aemics b.v., Borbala Online Conference Services, Travel Service International, FLEX, A&R Management Consultants, Atsmart, Keypoint Consultancy, Technomatics, FACT, OpenFortress, CAALMA b.v., Cintus, Forward Thinking, Index Society, Neuro-Fuzzy Centrum, Sightline, Controllab Products b.v., C2V, Yucat b.v., Emaxx, Internet & Database Technology (I&DT), and Cohesys.

Software Quality

The quality of software becomes more important every day. Since the development of software is expensive and users are becoming more sophisticated and demanding, companies should be able to deliver high quality software. The quality of software can be controlled by carefully managing the software development process and (thoroughly) testing the software product.

Since the two methods of controlling software quality do not prevent software products from containing errors, research on software design, verification methods and their practical application, to achieve a high standard of software quality, has to be done. Therefore the Mathematics and Computers Science Department of the Technische Universiteit Eindhoven and the Science, Mathematics and Computing Science Department of the Radboud University Nijmegen have joined forces and initiated LaQuSo. LaQuSo's mission is to become recognized as one of the leading scientific institutes for quality software in Europe. LaQuSo works together with partners from industry, government and science and focuses on verification (formal) methods, techniques and tools; validation (empirical) methods, techniques and tools and software system certification.

Certification

If an organization wants certainty about or confidence in a software artifact a LaQuSo certificate can be requested.

Certification is a check that the artifact fulfills a well-defined set of requirements. These requirements are defined by the customer or a third party; LaQuSo will do the check. The certificate will always refer to the requirements that were used to check the artifact against. An important issue of certification is that the evaluator should be independent. As part of a university, LaQuSo is able to perform this role.

Each certification project has its own goals. This means that a certification plan (steps to take and techniques to apply) has to be made for each project. LaQuSo can perform different kinds of checks:

- Error detection in the artifact: failure of user defined properties or failure of generic system properties (e.g. absence of deadlock, proper termination of transactions);
- Reliability estimates;
- Check if the product is compliant with a standard (e.g. C++ coding conventions, FDA);
- Benchmark the product with peers for certain properties;
- Determination of the quality attributes of the artifact (e.g. portability, maintainability).

The certification goal is defined based on the customer questions and requirements and the norms and standards that are applicable in that area. The outcome is a decision to grant a certificate yes or no.

address: Den Dolech 2
5612 AZ Eindhoven

phone: +31 40 247 2526
fax: +31 40 247 4252

email: info@laquso.com
web: www.laquso.com

Way of Working

The ESI is one of the few research institutes in the world that address embedded systems design at the multidisciplinary systems level. The ESI distinguishes itself through its positioning between academia and industry and through the strong industrial role in its research projects. As far as we know, the ESI is the only embedded systems institute in the world that is thus positioned. It is the explicit goal of the ESI to develop into a center of expertise with international, technological leadership in embedded systems design. The research projects are the means to achieve that position. To make that possible the kernel of the ESI has a research staff to transform and consolidate the results of the research projects into tangible knowledge and expertise.



Network Institute

The ESI consists of a kernel and a network. In the kernel the ESI builds up its expertise in embedded systems design. The research staff in the kernel is especially focused at the multidisciplinary systems level. Together with its partners the ESI carries out the research projects. The partners in the network provide the necessary disciplinary contributions and product/market expertise in these projects.

At the end of 2004 the ESI kernel counts 17 people (in FTE) and the ESI network 60. The researchers in the network come from 13 different academic and industrial organizations in the Netherlands. The kernel and the network are expected to grow to 35 and 150 FTE, respectively. These numbers are based on available budgets until 2011.

Industry-as-Laboratory

The research projects are carried out in an industrial context. This context is essential for three reasons. First, it is a means to get access to the vast body of (usually implicit) design knowledge among engineers and architects in industry. Second, it is the way the ESI uses to acquire practical design experience. Research in design methods cannot be done without actually building embedded systems⁽¹⁾. The ESI uses the system provided by the Carrying Industrial Partner (CIP) as such. Third, it provides a laboratory setting to test and verify the applicability of the methods developed at the ESI. Leading European industries act as CIPs; thus far Philips, ASML, and Océ. All projects are 'one roof' projects; they can be carried out at the ESI, at the CIP, or at one of the other partners involved. The ESI is responsible for all project management.

Multiple Market/Application Domains

In order to test the genericity of the design methods developed at the ESI, the methods are tried in various domains. The choice of the actual domains is opportunity driven. The current domains are Professional Devices (office equipment, semiconductor manufacturing equipment, health care imaging equipment) and Consumer Electronics (digital television). Automotive is a domain the ESI certainly wants to enter as well.

(1) The experience is that the time spent on reflection and abstraction is only a fraction of the time spent on actual design

FOUROAKS

Four Oaks

Four Oaks is gespecialiseerd in het testen van informatiesystemen en het inrichten van testprocessen.

Wij streven naar het beheersbaar maken van de ontwikkelprocessen van onze klanten door het verder optimaliseren van testprocessen.

Testen volgens Four Oaks is:

'Het managen van de risico's en verwachtingen rond het ontwikkelen en implementeren van informatiesystemen'.

In de visie van Four Oaks anticipeert een goed ingericht testproces zowel op de veranderingen in de organisatie als op veranderingen in ontwikkelmethoden.

Onze dienstverlening

Iedere organisatie is uniek. Teststrategie en testuitvoering zijn daarom maatwerk voor iedere organisatie en ieder project.

In de aanpak van Four Oaks is de noodzakelijke flexibiliteit ingebouwd om de testorganisatie in te richten die past bij de vraag van de klant.

Four Oaks heeft ruime ervaring met de gangbare testmethoden, - technieken en tools en kan deze op maat inzetten.

Four Oaks zorgt er daarnaast voor dat testen geen eenmalige aangelegenheid is.

Door grondig te documenteren en kennis over te dragen maken wij iedere test herhaalbaar.

De dienstverlening van Four Oaks bestaat uit:

- Testconsultancy
- Testmanagement
- Testuitvoering

Onze medewerkers

De medewerkers van Four Oaks hebben ruime ervaring in het testvak.

Zij vormen een enthousiast team, waarin continue aandacht bestaat voor uitwisseling van kennis en vakinhoudelijke ontwikkeling.

Al onze medewerkers zijn ISEB gecertificeerd, het kwaliteitskeurmerk voor testers.

Testspecialisten van Four Oaks kenmerken zich door:

- Professionaliteit
- Betrokkenheid
- Teamgeest
- Degelijkheid
- Improvisatievermogen

Meer weten?

Wilt u weten hoe Four Oaks u kan ondersteunen bij het optimaliseren van uw testprocessen? Neem contact met ons op.

Four Oaks
Boulevard Heuvelink 62
6828 KS Arnhem
Tel: 026 - 355 46 52
Fax: 026 - 355 47 34
E-mail: info@fouroaks.nl
Internet: www.fouroaks.nl

Harvey Nash IT Recruitment

Over ons

Opgericht in 1988 in Engeland en opererend in Amerika, Australië, Azië en Europa heeft Harvey Nash een naam gevestigd door duizenden opdrachten succesvol uit te voeren. Harvey Nash beschikt inmiddels over 26 kantoren en ca. 400 medewerkers. Ons succes is gebaseerd op het leveren van duidelijke, op de wensen van onze cliënten afgestemde oplossingen.

Harvey Nash richt zich met name op het leveren van projecten aan ICT professionals op tijdelijke basis. Wij werken met een omvangrijk netwerk van hoog gekwalificeerde ICT professionals die werken als Freelancer of op basis van Tijdelijke Arbeids Overeenkomsten of in dienst zijn bij derden.

In deze hoedanigheid fungeert Harvey Nash vaak als leverancier naast preferred suppliers. Daar waar een preferred supplier zich primair richt op de inzet van eigen personeel richt Harvey Nash zich op het inzetten van de benodigde specialist.

Ten behoeve van snelle en effectieve bemiddeling proberen we ook ons netwerk van niche-partijen constant uit te breiden.

Wij zoeken jou

Voor diverse eindklanten in verschillende branches (Retail, Energie, Industrie, Bank& Verzekeringen, Telecom/ICT, Entertainment, Overheid en Transport) zijn wij regelmatig op zoek naar ervaren Testers. In de rol van Tester kunnen wij diverse functies vergeven zoals Technisch Tester, Functioneel Tester, Testanalist, Test Engineer, Testadviseur, Testcoördinator en Testmanager.

Uw werkzaamheden zullen o.a. bestaan uit het:

- Het in teamverband alsmede zelfstandig uitvoeren van testopdrachten
- Het dragen van verantwoordelijkheid voor de resultaten van het testen en bewaking van de interne kwaliteitszorg.
- Opstellen en bewaken van testplannen

Gewenste Kennis en ervaring:

Minimaal één jaar kennis van en ervaring met Methodieken zoals:

- Test Management Approach (TMap®);
- TestFrame;
- Test Process & Performance Improvement (TPPI);
- (D)SDM;

Kennis van en ervaring met Tools zoals:

- TestDirector;
- LoadRunner;
- WinRunner;
- QA Run

Geïnteresseerd?

Indien u interesse heeft in dergelijke projecten of als u meer wilt weten aangaande het uitvoeren van projecten op tijdelijke of freelance basis, dan kunt u contact opnemen met Yvonne Wittendorp.

U kunt natuurlijk ook uw CV sturen naar maarsen@harveynash.com
o.v.v. YVW/Test.

Harvey Nash plc headline sponsor of
CBI interactive
conference
business at its best 2005

**HARVEY
NASH**

www.harveynash.com

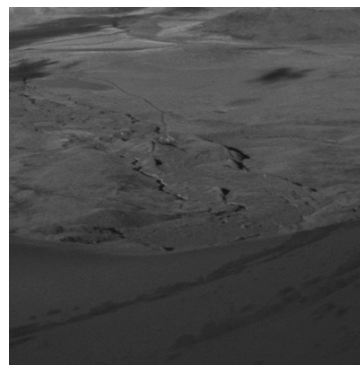
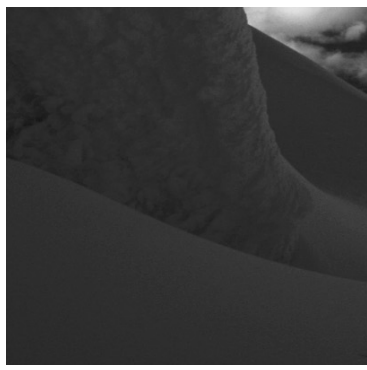


Testing & Quality Management

LogicaCMG offers a comprehensive range of consultancy, tools, methodologies and specialist expertise for testing and quality management. Our test solutions are not just an effective way to define the quality of IT related solutions, they are also used more and more to define the quality of complete processes and services. For instance, LogicaCMG's **TestFrame** is the leading standard for structured testing of applications, with dedicated solutions for specialist areas such as ERP, embedded systems and infrastructure testing. Our **risk assessment and test management** services ensure testing remains focused on clear business objectives. Beside this, LogicaCMG has a strong position in Business Acceptance Management. We can help our customers with testing and accepting third party products. Companies can enhance their in-house test capability through LogicaCMG's extensive training curriculum, or we can manage the entire test function on an outsourced basis. Whichever service you choose, you can expect the kind of pragmatic, innovative solutions that have made LogicaCMG a global force in testing and quality management with over 2.500 professional testing staff.

About LogicaCMG

LogicaCMG is a major international force in IT services and wireless telecoms. It provides management and IT consultancy, systems integration and outsourcing services to clients across diverse markets including telecoms, financial services, energy and utilities, industry, distribution and transport and the public sector. The company employs around 21,000 staff in offices across 35 countries and has more than 40 years of experience in IT services. Headquartered in Europe, LogicaCMG is listed on both the London and Amsterdam stock exchanges (LSE: LOG; Euronext: LOG). More information is available from www.logicacmg.com



Slimme menselijke schakel tussen ICT en Business

Dat testen belangrijk is behoeft – zeker in dit gremium – nauwelijks nog toelichting. Maar er is testen en testen.

Testers behoren allereerst volstrekt onafhankelijk te zijn. Zij zijn immers bondgenoot (en het geweten) van drie partijen: opdrachtgevers, ontwikkelaars en gebruikers. Ook als de opdrachtgever en de ontwikkelaar onderdeel zijn van hetzelfde bedrijf, kunnen ze maar beter enigszins uit elkaar worden gehouden. Op die manier kan de tester zijn belangrijke taak ongehinderd en effectief vervullen.

Slim testen

Maintain heeft haar ruime ervaring op testgebied ‘gecanoniseerd’ in de SmarTEST-aanpak, die is beschreven in het gelijknamige boek. Niet alles hoeft op elk moment te worden getest; de testdoelen moeten heel precies en uitgekiend worden gekozen.

De SmarTEST-methode is gebaseerd op de principes:

- Strategisch
- Mensgericht
- Adaptief
- Risicogebaseerd
- Transparant

SmarTEST combineert het beste uit de gangbare methoden voor gestructureerd testen met de nieuwste inzichten uit moderne, iteratieve methoden van systeemontwikkeling. Want een kwaliteitspartner moet degelijk en betrouwbaar zijn én helemaal bij de tijd.

Onafhankelijk testen

Maintain is als testspecialist volledig onafhankelijk, en ziet dit als een van haar kerncompetenties. Geen systeemontwikkeling dus, omdat de functies produceren en testen volgens deze filosofie niet in één hand mogen liggen. Recent heeft Maintain nauw aan het testproces verwante disciplines als requirements engineering en opleidingen omgevormd tot zelfstandige bedrijven, die – samen met Maintain – onderdeel uitmaken van Valori. Maintain positioneert zich daarmee nog duidelijker als dé specialist in ‘slim testen’, terwijl ook over aanverwante competenties kan worden beschikt. In Talanto, onze ‘academy’ voor talent, krijgen o.a. pas afgestudeerden een zware opleiding en certificering in het test- en kwaliteitsvak.

De mens centraal

In de combinatie Mens – Methode – Middelen staat volgens Maintain de mens centraal. Dit is gevisualiseerd in het logo van Maintain, waarin de factor ‘mens’ wordt uitgebeeld door de stip in het midden. Middelen en methoden mogen dan onmisbaar zijn; het zijn altijd mensen die de uiteindelijke kwaliteit bepalen.

Maintain werkt onder andere voor;

Delta Lloyd, Interpolis, ASML, VGZ, Randstad I-bridge, NOB, CJIB, RaboBank, Gasunie, Interpay, ING Groep



OPTIMIZE APPLICATIONS

- *Kunt u erop vertrouwen dat u nog steeds aan de wensen en behoeften van uw organisatie kunt voldoen wanneer een nieuwe applicatie in gebruik wordt genomen?*
- *Weet u zeker dat de systeemprestaties niet zullen afnemen en dat het systeem flexibel genoeg is om zowel normaal gebruik als piekbelastingen aan te kunnen?*
- *Bent u zich bewust van de eventuele compromissen die moeten worden gesloten?*
- *Kunt u een goed onderbouwde beslissing nemen of en op welk moment een applicatie in gebruik kan worden genomen?*

Hebt u één van de bovenstaande vragen met 'NEE' beantwoord, dan hebt u behoefte aan een oplossing waarmee u ervoor kunt zorgen dat uw applicaties gegarandeerd de gewenste kwaliteit en prestaties leveren, vóórdat ze in gebruik worden genomen. Met andere woorden, een strategie waarmee u de waarde van de IT-afdeling voor uw bedrijf kunt maximaliseren. In dat geval kan Mercury uitkomst bieden.

Mercury Application Delivery

Mercury Application Delivery biedt een geïntegreerde aanpak om gefundeerde 'go-live' besluiten voor applicaties te nemen. Dankzij Mercury's Optimization Centers for Quality and Performance - inclusief bekende oplossingen zoals LoadRunner en TestDirector - kunt u onderbouwde beslissingen nemen over het moment waarop applicaties in gebruik kunnen worden genomen. Verder kunt u fouten in de software reduceren, de tijd en kosten van het inzetten van nieuwe software en upgrades verminderen en ervoor zorgen dat softwareapplicaties daadwerkelijk de beoogde bedrijfsresultaten opleveren. Hoe krijgt u dit voor elkaar? Door het complete kwaliteitsproces te standaardiseren en te beheren. Met andere woorden, door op basis van bedrijfsgegevens real-time inzicht te verschaffen in alle kernprestatie-indicatoren (KPI's) en door uw IT-afdeling in de loop der tijd uit te laten groeien tot een Center of Excellence in testen.

Business Technology Optimization (BTO)

Mercury is wereldwijd marktleider op het gebied van BTOsoftware en -services. Onze BTO-oplossingen helpen klanten bij het beheren van IT en bij het optimaliseren van de kwaliteit, de prestaties en de beschikbaarheid van applicaties. Mercury stelt IT-organisaties in staat om hun focus te verplaatsen van het beheer van IT-projecten naar het optimaliseren van bedrijfsresultaten. Global 2000-bedrijven en overheidsinstanties over de gehele wereld vertrouwen op Mercury voor het verlagen van IT-kosten; het beperken van de risico's en het optimaliseren van groei; het ontplooiën van strategische IT-initiatieven; en het optimaliseren van enterprise-applicatie-omgevingen zoals J2EE, .NET en ERP/CRM.

Meer informatie?

Wilt u meer informatie ontvangen over onze oplossingen en hoe u kwaliteit en prestaties van bedrijfsprocessen en applicaties kunt optimaliseren?

Mercury Nederland
Van der Hooplaan 241
1185 LN Amstelveen
Tel.: +31 (0)20 545 2000
Fax: +31 (0)20 545 2001
E-mail: info.nl@mercury.com
www.mercury.com/nl



ps_testware als onafhankelijke partner in softwarekwaliteit

Problemen met de kwaliteit van je software? Hulp nodig bij het structureren van je requirements of testactiviteiten? Inzicht nodig in wat gestructureerd testen en validatie voor je kan betekenen en hoe je dit implementeert? ps_testware kan hiervoor een oplossing bieden.

ps_testware is een Belgisch/Nederlands consultancy bedrijf een met kantoor in Gorinchem voor de Nederlandse markt en is al meer dan 10 jaar gespecialiseerd in softwarekwaliteit (gestructureerd software testing en computer system validation). Een 60-tal consultants van ps_testware werkt in Nederland, België en Frankrijk aan de verbetering van softwarekwaliteit bij grote klanten (vooral bank-, verzekerings-, farmaceutische en energie-sector). Deze medewerkers zijn stuk voor stuk ISEB gecertificeerd en passen een bewezen methodologie toe (gebaseerd op het I-Model), waarbij de diverse fases in het test- en validatieproces op een gestructureerde wijze worden doorlopen om zo tot een kwaliteitsvol softwareproduct te komen en in één klap ook kostbare tijd te winnen.

Een belangrijk aspect van onze methodologie is het werken met een URH (User Requirements Hierarchy). Hierbij wordt het mogelijk de gebruikers- en testvereisten op een eenvoudige manier te beheren en als basis te gebruiken voor zowel de bouwfase als testontwikkeling en -uitvoering. De methode die hiervoor wordt gebruikt, noemen we RUM™ (Risk-Assessed User Requirements Management).

ps_testware is enkel tevreden als de klant tevreden is. Wat de klant wenst, is ook wat wij willen en zo komen we tot een gemeenschappelijk doel: het opleveren van een kwaliteitsvol product volgens een logisch gestructureerd en kwaliteitsvol proces. ps_testware levert test- en validatiediensten onder diverse vorm: consultancy, coaching, training, co-ordinatie, management en outsourcing. Onze methodologie passen we zowel toe op manuele testprocessen als testautomatisering (regressietesten en performantietesten) en we testen allerhande software (administratief, embedded, logistiek...).

ps_testware durft slecht nieuws te brengen als dit nodig is. Advocaat van de duivel kunnen we enkel zijn omdat we optreden als onafhankelijke (test)partner om kwaliteit op een hoger niveau te brengen.



GESPECIALISEERD IN TESTEN

Qualityhouse is een bedrijf dat zich volledig heeft toegelegd op de kwaliteitszorg en procesverbetering in de ICT. Het bouwen van kwalitatief goede software vraagt om een goede beheersing van het ontwikkelproces. Om te beoordelen of het ontwikkelde systeem inderdaad aan de verwachtingen en aan alle vooraf gestelde eisen voldoet, is een gedegen teststrategie nodig. Daarop richt de dienstverlening van Qualityhouse zich. Testen van software, inrichten van kwaliteitssystemen, introduceren van inspectie- en testmethoden en verzorgen van opleidingen op al deze gebieden, is ons specialisme. (www.qualityhouse.nl)

Dit leidt tot het volgende overzicht van onze diensten en producten:

Testdetachering:

Afhankelijk van de aard en het niveau van de testwerkzaamheden kunnen op locatie van de opdrachtgever testers, testcoördinatoren en testmanagers worden ingezet. Op ieder niveau heeft Qualityhouse de juiste testspecialist. Het merendeel van onze testspecialisten is ISEB-gecertificeerd. Bovendien worden al onze medewerkers regelmatig bijgeschoold.

Testprojecten:

Om verschillende redenen kan het handig zijn om een testtraject uit te besteden. Qualityhouse kan complete testtrajecten uitvoeren of op locatie bij de opdrachtgever of in ons projectenbureau in IJsselstein. Een groot voordeel van het outsourcen van het testtraject is dat er afspraken gemaakt kunnen worden over fixed price, fixed date en/of fixed quality.

Kwaliteitszorg:

Het testen van software en kwaliteitszorg binnen de ICT zijn nauw met elkaar verbonden. Onze kwaliteitsconsultants kunnen u begeleiden in CMM-trajecten en het implementeren en verbeteren van testtrajecten door middel van het Verification and Validation Maturity Model (V2M2). Daarnaast kunnen zij u helpen met auditing van bestaande methoden en implementatie of verbetering van testmethoden.

Trainingen:

Op alle hierboven genoemde gebieden verzorgt Qualityhouse diverse standaard trainingen die op maat kunnen worden aangepast. Deze trainingen worden gegeven in ons trainingscentrum in IJsselstein maar kunnen desgewenst bij de opdrachtgever of op locatie worden gegeven.

Research & Development:

Onze afdeling Research & Development is voortdurend bezig nieuwe producten en diensten te ontwikkelen. Daarnaast is zij continu bezig met het verbeteren van het huidige scala aan diensten en producten. Hiervan profiteert u als opdrachtgever aangezien onze producten en diensten voortdurend up-to-date worden gehouden en omdat onze medewerkers continu op de hoogte worden gehouden van de nieuwste ontwikkelingen.

“Onderzoek, ontwikkeling, advies en opleiding op het gebied van betrouwbaarheid van informatiesystemen”.

Refis houdt zich bezig met de kwaliteit van geautomatiseerde systemen in de meest brede zin van het woord. Of het nu gaat om betrouwbaarheidsanalyses, metrieken en meetsystemen, of het testen van informatiesystemen, Refis adviseert, ontwikkelt en participeert in alle aspecten van kwaliteitsmeting en –verbetering.

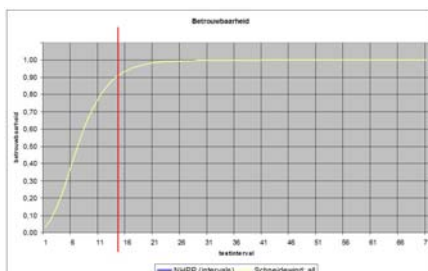
➤ **Onderzoek**

In nauwe samenwerking met universiteiten, opdrachtgevers en collega bedrijven werkt Refis aan een brede inzetbaarheid van bestaande betrouwbaarheidsmodellen, nieuwe hulpmiddelen en een verspreiding van kennis en ervaring.

➤ **Advies**

Refis adviseurs zijn betrouwbare partners in ontwikkel-, test- en kwaliteitszorg trajecten. Als projectmanager, testmanager of adviseur. Hands-on ervaring en inzicht in bedrijfsprocessen levert concrete ideeën die ook werkelijk bijdragen tot verbetering.

“Bespaar kosten in ontwikkeling, testen en exploitatie”.



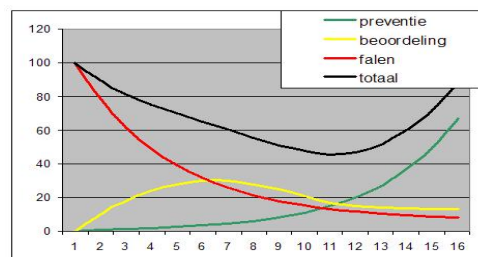
➤ **Ontwikkeling**

In nauwe samenwerking met haar opdrachtgevers, ontwikkelt en implementeert Refis meetsystemen waarmee de opdrachtgever continue inzicht heeft in de performance van de eigen auto-matiseringsprocessen en -producten.

➤ **Opleiding**

Refis trainingen op het gebied van testen, kwaliteitszorg en systeembetrouwbaarheid onderscheiden zich door het praktische karakter en de directe toepasbaarheid van de lesstof.

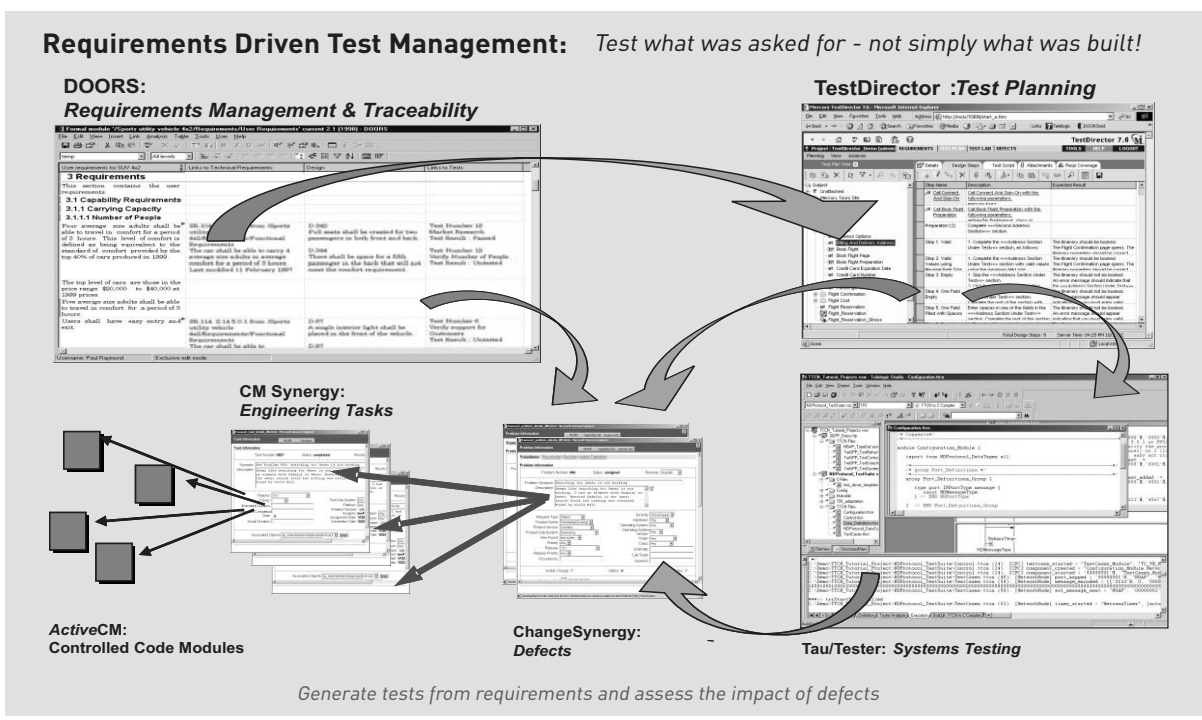
“Verhoog de betrouwbaarheid van informatie- en procesbesturingssystemen”.





Requirements-Driven Testing

Integrated requirements, test and defect management



Test for what customers actually want.

According to leading industry analysts The Standish Group, over 50% of the reasons that make a project successful can be attributed to good requirements management practices. It's no surprise then that a solid requirements management process is often the driving force for successful developments.

Many organisations recognise this fact but concentrate on requirements only at the beginning of their projects. Yet, requirements management is a discipline that impacts every stage of development from initial specification to final testing. And that's why requirements-driven testing is so important.

After all, if requirements don't drive testing how can you demonstrate that the system delivers exactly what the customer demands? And how do you trace defects back to requirements in order to understand how to correct them? What's more, when requirements change either through error corrections or enhancements to subsequent releases, without requirements-driven testing how can you keep multiple tests 'in sync'?

Telelogic's requirements-driven test solutions combine leading test management (e.g. Mercury's TestDirector) and test execution environments (e.g. Telelogic TTCN-3 TAU Tester environment) with the market leader for requirements management, Telelogic DOORS and the technology leader for change- and configuration management, Telelogic SYNERGY. The result? Development processes that are smooth, effective and as error-free as possible.

About Telelogic

Telelogic® is a leading global provider of solutions for automating and supporting best practices across the enterprise - from powerful modeling of business processes and enterprise architectures to requirements-driven development of advanced systems and software. Telelogic's solutions enable organizations to align product, systems and software development lifecycles with business objectives and customer needs to dramatically improve quality and predictability, while significantly reducing time-to-market and overall costs.

For more details please visit: www.telelogic.com or call us direct.

Telelogic Netherlands, Rijnzathe 7 B-3, 3454 PV De Meern.

Phone: +31 30 666 5530. Email: info.netherlands@telelogic.com



Program

10:00 Registration and coffee

10:25 Opening (Ed Brinksma)

10:30 Keynote speaker: Carsten Weise (Ericsson)
Testing and Test Automation in Telecommunication

11:30 Leon Wolters, Jos van Rooyen, Erik Altena (LogicaCMG)
Using TTCN-3 to standardize the test automation

Stefan Blom (University of Innsbruck), Nicu Goga, Natalia Ioustinova, Jaco van de Pol (CWI), Axel Rennoch (Fraunhofer FOKUS), Natalia Sidorova (Eindhoven University)
TTCN-3 for Automated Testing Railway Control Systems

12:30 Lunch

14:00 Ed Brandt (Refis)
Defect registration as knowledge base

Han van Gerwen (Ordina)
Proving test coverage at requirements level

Pieter Koopman (Radboud University)
On-the-fly generation of test suites for automatic testing

15:30 Coffee

16:00 Guido van Leeuwen, Hans van Loenhoud (INQA)
Case study AllianzNet: Automated versus manual testing

Axel Belinfante (University of Twente), Henrik Bohnenkamp (RWTH Aachen)
Timed Testing with TorX

Ivo de Jong (ASML), Roel Boumen, Asia van de Mortel-Fronczak, Koos Rooda (Eindhoven University)
Test automation in the alpha test of wafer scanners

17:30 Reception