Testnet –Abstract for conference - preliminary

Model-Based Testing: Models for Test Cases

Jan Tretmans, Embedded Systems Institute, Eindhoven

Abstract:

Systematic testing of software plays an important role in the quest for improved software quality. Testing, however, turns out to be an error-prone, expensive, and time-consuming process. Model-based testing is one of the promising technologies to meet the challenges imposed on software testing. In model-based testing a system under test (SUT) is tested against a formal description, or model, of the SUT's desired behaviour. First, such a model can be used for analysis and checking of the design. Second, the model serves as a precise and complete description of what the SUT should do, and, consequently, is a good basis for testing. Third, a model can be processed by tools, in particular by a test generation tool, so that large quantities of test cases can be automatically generated. This allows effective test automation beyond the automatic execution of test scripts. And if the model is valid, i.e. expresses precisely what the SUT should do, then all these tests are valid too.

In this presentation we discuss the basic ideas and principles of model-based testing, together with its advantages, pitfalls, and perspectives. It is shown how testing research is turned into useful and applicable techniques and tools by showing some academic and industrial examples of how model-based testing can be used for `on-the-fly', automatic generation, execution, and analysis, of test cases consisting of hundreds of thousands of test events.

Biography:

Jan Tretmans is researcher at the Embedded Systems Institute (ESI) in Eindhoven, and part-time associate professor at the Radboud University Nijmegen, both in The Netherlands. He is working in the areas of software testing, and the use of formal methods in software engineering; in particular, he likes to combine these two topics: testing based on formal specifications, also referred to as model-based testing. In this field he has several publications, and he has given numerous presentations at scientific conferences as well as for industrial audiences.

Jan Tretmans holds a degree in Electrotechnical Engineering and a PhD. in Computer Science, both from the University of Twente in The Netherlands. He spent some time as a post-doctoral researcher in Norway, Greece and Germany, and for a couple of years he was in involved in the academic-industrial transfer of formal methods technology. Before joining ESI he was at the University of Twente, and full-time at the Radboud University Nijmegen. Currently he is involved in the Dutch research projects Atomyste, Stress, and Poseidon, and in the EU projects Tarot and Quasimodo, which all address some aspect of the theory, tools and applications of model-based testing based on formal methods, and in which industrial-academic collaboration plays an important role.

Building a Software Test Team

Bruce Benton, Microsoft Denmark

Abstract:

Designing and building a software test organization can largely be broken down into a set of decision categories and an understanding of the environmental factors which influence those decisions.

This paper summarizes the three key strategies which must be addressed (automation strategy, organizational strategy and test strategy) and discusses how various environmental factors such the product domain, the problem domain and the business domain influence those strategies.

This paper is based on experience generated building test organizations in a large international software company spanning several technologies and market segments. Additionally, this paper provides general recommendations for each of these decisions and the information required to tailor those decisions to the specific needs of the reader's organization.

Test Automation

Carsten Weise, Cluster of Excellence at Aachen University at the Embedded System Labs.

Abstract:

With quickly increasing sizes of test suites, test automation becomes a crucial success factor in the validation and verification of modern software systems. The talk will look at the complete test automation process, starting from test specifications to test case development, test case databases, test reports and test statistics. Test automation cannot be done without a robust test tool chain. We will look at the different kind of tools needed within the test processes, with a special focus on tools accessing the system under test. These access tools are of high complexity when the system under test is a combined soft-/hardware system. Often proprietary design of test tools is required. In-house development of test tools easily leads to a catch-22: who tests the test tools?

Biography:

Carsten Weise got his Ph.D. from Aachen University. His expertise is in the area of embedded systems, where he has been working both on the theory of timed automata and hybrid systems and also practically in the implementation of real-time systems.

He has been working in the German test center of Ericsson for eight years, where he was responsible for the test tool development and worked as technical coordinator in several world-wide test projects. Since 2007, he is working in the UMIC (Ultra High Speed Mobile Information and Communication) Cluster of Excellence at Aachen University at the Embedded System Labs. As part of his job, he is involved in test activities in several companies in automotive and railway systems.

Model-driven Test and Verification of Real-time and Embedded Systems

Kim Guldstrand Larsen, CISS, University of Aalborg

Abstract:

The presentation explains how a model-driven development can improve the testing and verification of real-time and embedded systems. The presentation will explain how the behaviour of real-time and embedded systems can be modelled and how real-time requirements can be specified. Given a design model and system requirements, the model can now be automatically (exhaustively) checked against the requirements using the state-of-the-art model-checking tool Uppaal. The main part of the presentation emphasizes the use of real-time models for test generation for actual system implementations. Test generation may either be offline where test sequences are generated and later executed, or it may be online, where generation and execution are integrated. The talk will contain demonstrations of the Uppaal and Uppaal-TRON tools. Also some commercial results will be presented.

Biography:
Prof. Kim Guldstrand Larsen obtained his PhD from Edinburgh University in 1986. He is Professor at Aalborg University, Denmark, and Industrial Professor at Twente University, The Netherlands. He is also the Director of the Center for Embedded Software Systems, CISS. Kim Guldstrand Larsen became Honorary Doctor (Honoris causa) at Uppsala University in 1999 for his outstanding contributions to the popular verification tool UPPAAL. In 2005 he received the Danish Citation Laureates Award, Thomson Scientific, as the most cited Danish computer scientist in the period 1990-2004. In 2007 he became Knight of the Order of Dannebrog . In 2007 he became Honarary Doctor (Honoris causa) at Laboratoire Specification et Verification, Ecole Normale Superieure Cachan, France. Kim Guldstrand Larsen is member of the Royal Danish Academy of Sciences and Letters, Copenhagen, and is member of the Danish Academy of Technical Sciences. For a period of seven years he served as member of the Danish Natural Science Research Council. Since 1987 Kim Guldstrand Larsen has written and/or edited 10 books, published 27 papers in international journals, and approximately 130 papers in international reviewed conferences within the areas of concurrency theory, model checking, real-time and embedded systems. Kim has coauthored 6 software-tools, holds one patent and is prime investigator in the real-time verification tool UPPAAL (www.uppaal.com). Kim Guldstrand Larsen has given invited talks and course all over the work, including North-America, China, India, and most European countries.

Verification & Validation of Critical Software

Jørgen Bundgaard, ROVSING A/S

Abstract:
Several industries, including space and aerospace, automotive and railway, have standards for software development where the required amount of software verification and validation (V&V) activities are governed by the criticality of the software.

For example, highly critical software may require investigation of worst case load scenarios, covering robustness of the software with respect to injections outside or the specified boundaries.

According to those standards a criticality level is assigned to software primarily on the basis of the likelihood and consequences of a system failure caused by software.

Specifically, the European Space Agency (ESA) software development standard requires that critical software is subjected to Independent Software Verification & Validation (ISVV). This is an engineering practice intended to improve the quality and reduce the costs of software products as well as reducing development risks by independent analysis and review on the specification, design and code of a software product.

Rovsing A/S is a Danish company which delivers electronics and software to the European Space Industry and which is an internationally recognised specialist in ISVV.

One of the resent projects carried out by Rovsing A/S concerned ISVV of the on-board software for the European space shuttle, the Automated Transfer Vehicle (ATV).

The ATV is a 20 ton unmanned “space truck” which brings water, oxygen, food, fuel and other supplies to the International Space Station (ISS). The ATV is fully computer controlled and since a software fault could result in a fatal collision with ISS the software is rated as “Category A”, the highest criticality level in manned spaceflight.

On the 3rd of April, 2008 the first ATV spacecraft, named “Jules Verne”, performed a flawless, automated docking manoeuvre with the ISS.

The presentation will show how software criticality is evaluated, how the criticality is used to determine the required V&V activities, and finally, we will explain the techniques and methods Rovsing A/S used to independently verify and validate the ATV software, along with some of the results obtained.

Biography:
Jørgen Bundgaard is R&D Director Software & ISVV in ROVSING A/S, Denmark.

Applying Six Sigma Methodologies to Software and Systems Test Management

James Joseph Waskiel, Motorola, Denmark

Abstract:
Six Sigma is a rigorous, focused and highly effective implementation of proven quality principles and techniques with boundless application. Taking the Six Sigma Black Belt ”tool box” into the world of software and system test we are able to methodically approach optimisation of test management and continuous product quality prediction.

This presentation will first briefly focus on the basic elements of the Six Sigma ”tool box” to set a framework to build upon. We will then use this framework to demonstrate the application of these methods by walking through two major projects that have applied SS and discuss their results. The first project is at a unit software test level where interface complexities are at a minimum. The second project will look at the testing of ”system of systems” where interface complexity increases exponentially.

We will examine the Six Sigma techniques that were applied for both of these projects for the optimisation of testing to maximize coverage, minimize effort, and focus on high risk areas of poor quality as well as continually providing quantifiable predictions of product quality throughout the lifecycle. The intent of this presentation is to provide improvement ideas and the tools to implement the ideas in the test management environment.

Biography:
James Joseph Waskiel, Motorola, Denmark is Certified Six Sigma Black Belt and Certified SEI CMMI V1.2 Instructor

The Role of the Scientific Method in Improving Software Testing

Les Hatton, Kingston University and Oakwood Computing Associates Ltd., London

Abstract:
Software development in general and software testing in particular do not make use of scientific principles in improving their performance. The scientific method whereby progress occurs as a result of careful measurement, repeatability and the elimination of models of behaviour which cannot be supported by experiment is largely absent. It is sometimes said that this is because software is 'rapidly moving'. In fact it is only rapidly moving in the sense of a pan of boiling water. The water bubbles away merrily but the pan doesn't go anywhere.

This talk will describe a number of recent careful experiments often with surprisingly non-intuitive results. Depending on time available, these will include at least some of the following:

- The equilibrium state of a software system appears to conserve defect,
- There is strong evidence in quasi-equilibrated systems for xlogx growth in defects where x is a measure of the lines of code,
- Software measurements, (also known rather inaccurately as metrics) are effectively useless in determining the defect behaviour of a software system,
- Most such measurements, (including the ubiquitous cyclamate complexity) are almost as highly correlated with lines of code as the relationship between temperature in degrees Fahrenheit and degrees Centigrade measured with a slightly noisy thermometer. In other words, lines of code are just about as good as anything else when estimating defects,
- 'gotos considered irrelevant'. The goto statement has no obvious relationship with defects even when studied over very long periods. It probably never did,
- Checklists in code inspections appear to make no significant difference to the efficiency of the inspection,
- When you find a defect, there is an increasing probability of finding another in the same component. This strategy is effective up to a surprisingly large number of defects in youthful systems but not at all in elderly systems,
- Overlapping delivery with testing can remove around half of all defects before the customer gets chance to see them,
- Test engineers underestimate long maintenance tasks and overestimate short maintenance tasks. They also often get the nature of a short task, (corrective, adaptive or perfective), completely wrong.

Biography:

Les Hatton is professor of Forensic Software Engineering, Kingston University and Oakwood Computing Associates Ltd., London.