04371 Abstracts Collection
Perspectives of Model-Based Testing
— Dagstuhl Seminar —

Ed Brinksma[1], Wolfgang Grieskamp[2] and Jan Tretmans[3]

[1] Univ. of Twente, NL
brinksma@cs.utwente.nl
[2] Microsoft Research, Redmond, US
wrwg@microsoft.com
[3] Univ. of Nijmwegen, NL
tretmans@cs.kun.nl

Abstract. From 05.09.04 to 10.09.04, the Dagstuhl Seminar 04371 "Perspectives of Model-Based Testing" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Model-based testing, software testing, formal methods, automatic test generation

04371 Summary – Perspectives of Model-Based Testing

Jan Tretmans (Univ. of Nijmwegen, NL)

The aim of the seminar *Perspectives of Model-Based Testing* was to bring together researchers and practitioners from industry and academia to discuss the state of the art in theory, methods, tools, applications, and industrialization of model-based testing, and to identify the important open issues and challenges.

Keywords: Model-based testing, software testing, formal methods, automatic test generation

Joint work of: Brinksma, Ed; Grieskamp, Wolfgang; Tretmans, Jan

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/364

Symbolic Testing in TorX, with an Applicaton in Timed Testing

Axel Belinfante (University of Twente, NL)

We present TorX, which is both a flexible, open architecture for conformance testing, and an on-the-fly testing tool implementation of that architecture, which has been used successfuly for testing of software components in an industrial setting.

With on-the-fly we refer to the integration of test generation and test execution, such that the test-generation is done in a demand-driven way, driven by the test execution.

We describe the mechanisms which have been introduced into TorX to facilitate Symbolic Testing, such that we can have (typed) variables and constraints over these in the generated test steps (possible stimuli and expected observations).

We show how we choose data values for the free variables in the stimuli,and how we update (the free variables in) the model from the parameters in the actual interactions (stimuli applied, observations received) with the System Under Test.

As an application of the symbolic testing capabilities of TorX we describe a symbolic "explorer" component which takes a Timed Automaton as input and provides access to a labelled transition system, enhanced with constraints on the absolute time that an action of the respective transition can happen.

We show that the mechanisms in TorX for symbolic testing are powerful enough to allow Timed Testing in a similar way as proposed by Brandan Briones and Brinksma.

Joint work of: Belinfante, Axel; Bohnenkamp, Henrik

Timed Testing in TorX

Henrik C. Bohnenkamp (University of Twente, NL)

We present TorX, which is both a flexible, open architecture for conformance testing, and an on-the-fly testing tool implementation of that architecture, which has been used successfuly for testing of software components in an industrial setting.

With on-the-fly we refer to the integration of test generation and test execution, such that the test-generation is done in a demand-driven way, driven by the test execution.

We describe the mechanisms which have been introduced into TorX to facilitate Symbolic Testing, such that we can have (typed) variables and constraints over these in the generated test steps (possible stimuli and expected observations).

We show how we choose data values for the free variables in the stimuli,and how we update (the free variables in) the model from the parameters in the actual interactions (stimuli applied, observations received) with the System Under Test.

As an application of the symbolic testing capabilities of TorX we describe a symbolic "explorer" component which takes a Timed Automaton as input and provides access to a labelled transition system, enhanced with constraints on the absolute time that an action of the respective transition can happen.

We show that the mechanisms in TorX for symbolic testing are powerful enough to allow Timed Testing in a similar way as proposed by Brandan Briones and Brinksma.

Joint work of: Bohnenkamp, Henrik C.; Belinfante, Axel

Test for quiescent real-time systems

Laura Brandán Briones (University of Twente, NL)

We present an extension of Tretmans' theory and algorithm for test generation for input-output transition systems to real-time systems.

Our treatment is based on an operational interpretation of the notion of quiescence in the context of real-time behaviour. This gives rise to a family of implementation relations parameterized by observation durations for *quiescence*.

We define a nondeterministic (parameterized) test generation algorithm that generates test cases that are sound with respect to the corresponding implementation relation. The test generation is also exhaustive in the sense that for each non-conforming implementation a test case can be generated that detects the non-conformance.

Automatic Generation of Tests from MSC Specifications with Data

Simon Burton (DaimlerChrysler AG, D)

Automotive telematics features are becoming increasingly more complex and can only be realised by the correct interaction between a number of devices. As a result, the task of testing these features is becoming increasingly more important as well as more difficult and time consuming. The interaction between devices in a telematics system is specified using Message Sequence Charts. Due to the complexity of the systems involved and the size of the specifications, automation is essential to ensure consistency of the test cases and to allow for sufficient coverage of the specifications during test. This presentation describes the problems involved in automatically generating executable test cases from abstract MSC specifications and amongst other relevant topics, discusses how the classification tree method (a systematic method of test design based on the idea of equivalence classes) can be used to select appropriate data points to be used within the tests.

Keywords: Message Sequence Charts, test generation, classification tree method

Can Model-based Testing Scale to Large Systems?

Colin Campbell (Microsoft Research - Seattle, USA)

Can model-based testing be used for industrial-scale systems? This talk discusses some of the challenges and then shows some techniques developed at Micrsoft Research to deal with this kind of complexity.

Keywords: Model-based testing, software engineering, software contracts

Systematic Testing of Embedded Automotive Software - The Classification-Tree Method for Embedded Systems (CTM/ES)

Mirko Conrad (DaimlerChrysler Research - Berlin, D)

The software embedded in automotive control systems increasingly determines the functionality and properties of present-day motor vehicles. The development and test process of the systems and the software embedded becomes the limiting factor. While these challenges, on the development side, are met by employing model-based specification, design, and im-plementation techniques [KCF+04], satisfactory solutions on the testing side are slow in arriving. With regard to the sys-tematic selection (test design) and the description of test scenarios especially, there is a lot of room for improvement. Thus, a main goal is to effectively minimize these deficits by creating an efficient procedure for the selection and description of test scenarios for embedded automotive software and its integration in the model-based development process.

The realization of this idea involves the combination of a classical software testing procedure with a technology, prevalent in the automotive industry, which is used for the description of time-dependent stimuli signals. The result of this combination is the classification-tree method for embedded systems, CTM/ES.

The classification-tree method for embedded systems complements model-based development by employing a novel ap-proach to the systematic selection and description of the test scenarios for the software embedded in the control systems. CTM/ES allows for the graphic representation of time-variable test scenarios on different levels of abstraction: A problem-oriented, compact representation, adequate for a human tester and containing a high potential for reusability, is gradually being transformed into a solution-oriented technical representation which is suited for the test objects' stimulation. The CTM/ES notation facilitates a consistent representation of test scenarios which may result from different test design tech-niques. The test design technique which this method is primarily based on, is a data-oriented partitioning of the input do-main in equivalence classes. Secondary test design techniques are, for instance, the testing of specific values (or value courses) or requirement-based testing.

A domain-specific application pragmatics in the form of agendas supports the methodical execution of individual test activi-ties and the interaction of different test design techniques. The methodology description leads up to an effective test strat-egy for model-based testing, combining the classification-tree method for embedded systems with structural testing on the model level, and accommodating the different forms of representation of the test object during model-based development.

Systems which have been developed in a model-based way can be tested systematically and efficiently by means of the CTM/ES and the tools based thereon, such as the classification-tree editor for embedded systems CTE/ES, as well as the model-based test environment MTest.

Keywords: Model-based Testing, Classification-tree Method for Embedded Systems (CTM/ES), test design technique, test notation

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2005/325

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/325

Another Approach to Symbolic Test Generation

Lars Frantzen (Radboud University of Nijmegen, NL)

Classical state-oriented testing approaches are based on simple machine models such as Labelled Transition Systems (LTSs), in which data is represented by concrete values. To implement these theories, data types which have infinite universes have to be cut down to finite variants, which are subsequently enumerated to fit in the model. This leads to an explosion of the state space.

Moreover, exploiting the syntactical and/orsemantical information of the involved data types is non-trivial after enumeration.

To overcome these problems, we lift the family of testing relations $ioco_F$ to the level of Symbolic Transition Systems (STSs).

We present an algorithm based on STSs, which generates and executes tests on-the-fly on a given system. It is sound and complete for the $ioco_F$ testing relations.

Joint work of: Frantzen, Lars; Tretmans, Jan; Willemse, Tim

Unfolding Symbolic Transition Systems

Marie Claude Gaudel (Université Paris Sud, F)

Testing based on symbolic transition systems raises several problems due to infiniteness of the underlying model and the possible unfeasibility of some traces.

This talk focuses on selection strategies of finite test sets based both on the topology of the symbolic system and the properties of the data types used in guards and actions.

It revisits the notion of transition coverage, introducing:

- Weak symbolic transition coverage, where the transition is covered once only;
- Stronger symbolic transition coverage, where the hidden sub-cases coming from guards and actions are unfolded on the basis of the data types description, and covered.

Some implementation is under development, based on randomised constraint solving, narrowing (the LOFT system), and various unfolding strategies.

Joint work of: Gaudel, Marie Claude; Lestiennes, Grégory

The AGEDIS Model Based Testing Tools

Alan Hartman (IBM - Haifa, IL)

We describe the tools and interfaces created by the AGEDIS project, a European Commission sponsored project for the creation of a methodology and tools for automated model driven test generation and execution for distributed systems. The project includes an integrated environment for modeling, test generation, test execution, and other test related activities. The tools support a model based testing methodology that features a large degree of automation and also includes a feedback loop integrating coverage and defect analysis tools with the test generator and execution framework. Prototypes of the tools have been tried in industrial settings providing important feedback for the creation of the next generation of tools in this area.

Keywords: Automated test generation, UML modeling, test execution framework, coverage analysis, defect analysis

Joint work of: Hartman, Alan; Nagin, Kenneth

Testing with Functions as Specifications

Pieter Koopman (Radboud University of Nijmegen, NL)

In this paper we show that mathematical functions and logical expressions can very well be used as, partial, specifications. Reactive systems can be modelled by powerful extended state transition systems, that can be nondeterministic and can handle parameterized and infinite types for the inputs, outputs and states. These specifications can very concisely and directly be stated in a modern functional programming language. The test tool Gast is able to generate test data based on these specifications, execute the associated tests, and make a verdict fully automatically. Test data can be generated fully automatically, but can also be tailored in various high level ways, if that is desired. Advantages of this approach are that one specifies properties instead of instances of these properties, test data are automatically derived instead of manually, the tests performed are always up to date with the current specification, and testing is automatic (and hence fast and accurate).

Keywords: Automatic testing, model based testing, specification based testing, functions

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/324

Applying Model Based Testing in Different Contexts

Victor Kuliamin (Academy of Sciences - Moscow, RUS)

We describe ISP RAS experience in applications of model based testing in various areas. The two different examples are considered - UniTesK test development tools aimed at software component testing and OTK tool intended to be used in test development for complex structured text processors, the main example of which is compilers. The suprising fact is that the two methods used in the tools have different prerequisites for successful applications in industrial software development.

Keywords: Software component testing, compiler testing, testing based on software contracts, testing based on finite automata, test data generation

Joint work of: Kuliamin, Victor; Petrenko, Alexander

Testing Applied to Analysis of Model Relations

Mass Soldal Lund (University of Oslo, N)

The vision of model driven development demands sound methods for handling and analyzing models. We propose the use of testing techniques in combination with a formalization of UML models within an MDA framework as a way of supporting and strengthening the vision.

Online Testing of Real-Time Systems using UPPAAL: Status and Future Work

Brian Nielsen (Aalborg University, DK)

We present TUPPAAL — a new tool for online black-box testing of real-time embedded systems from non-deterministic timed automata specifications. We describe a sound and complete randomized online testing algorithm, and describe how to implement it using symbolic state representation and manipulation techniques. We propose the notion of relativized timed input/output conformance as the formal implementation relation. A novelty of this relation and our testing algorithm is that they explicitly take environment assumptions into account, generate, execute and verify the result online using the UPPAAL on-the-fly model-checking tool engine. A medium size case study shows promising results in terms of error detection capability and computation performance.

Keywords: Online testing,black-box testing, real-time systems, embedded systems, symbolic state representation, relativized timed input/output conformance, model-checking,

Joint work of: Larsen, Kim G.; Mikucionis, Marius; Nielsen, Brian

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/326

The State of the Art of Model-Based Testing with Markov Chain Usage Models

Stacy Prowell (University of Tennessee, USA)

Statistical testing of software is an approach which treats software testing as a statistical experiment, in which a sample (test cases) is drawn, and performance on the sample (success or failure in operation) is used to make inferences about important quantities, such as reliability, time to failure, and the like. In order to apply these techniques, a model of the population (all test cases of interest) is needed. Markov chains provide such a model.

Markov chain usage models have been used in industry for over a decade, and have been used to test a variety of different kinds of systems, including embedded and distributed systems, medical device software, scientific computing codes, GUI applications, networking applications, and combined hardware and software systems.

In the last few years, there have been many advances in the use of Markov chain usage models.

These have included the introduction of standard notations and modeling languages specifically suited to describing Markov chain usage models, the use of mathematical programming techniques to determine transition probabilities, more powerful test case generation through the use of concurrency operators, stopping criteria based on cost-benefit analysis, and improved reliability models. Three generations of tools to support the use and analysis of Markov chain usage models have been developed, with the latest generation (the JUMBL toolkit) providing support for a variety of notations and test case execution methods.

This talk gives a quick overview of statistical testing using Markov chain usage models, presents some of the more recent developments listed above, and concludes with a look at future tools and research topics.

Keywords: Statistical testing, usage-based testing, model-based testing, automated testing

Testing practices inside France Telecom's R&D labs - An example of automatic test generation

Yves-Marie Quemener (France Télécom, F)

In my talk, I treat two somewhat disjoint subjects.

First, I give the results of a survey done inside the R&D division of France Telecom at the end of 2003, about the testing practices in this entity. The results of the survey show the need for automating test processes, and the importance of automating test execution.

Second, I give an example of use of automatic test generation for validating a messaging platform.

Keywords: Test, test processes, industrial practices, telecommunications, automatic test execution, automatic test generation

Generating Efficient Test Sets with a Model Checker

John Rushby (SRI - Menlo Park, USA)

It is well-known that counterexamples produced by model checkers can provide a basis for automated generation of test cases. However, when this approach is used to meet a coverage criterion, it generally results in very inefficient test sets having many tests and much redundancy. We describe an improved approach that uses model checkers to generate efficient test sets. Furthermore, the generation is itself efficient, and is able to reach deep regions of the statespace. We have prototyped the approach using the model checkers of our SAL system and have applied it to model-based designs developed in Stateflow. In one example, our method achieves complete state and transition coverage in a Stateflow model for the shift scheduler of a 4-speed automatic transmission with a single test case.

Papaer, SAL scripts, and examples at
http://www.csl.sri.com/users/rushby/abstracts/sefm04

Joint work of: Hamon, Gregoire; de Moura, Leonardo; Rushby, John

Keywords: Model checking, test generation, Stateflow, SAL

Verification and symbolic test generation for safety properties

Vlad Rusu (INRIA Rennes, F)

We present a combination of verification and conformance testing techniques for the formal validation of reactive systems. A formal specification of a system - an input-output automaton with variables that may range over infinite domains - is assumed. Additionally, a set of safety properties are given under the form of observers described in the same formalism. Then, each property is verified on the specification using automatic techniques (e.g., abstract interpretation) that are sound but not complete for the class of sytems/properties considered here.

Next, for each property, a test case is generated from the specification and the property and is executed on a black-box implementation of the system. If the verification step was successful, that is, it has established that the specification satisfies the property, then the test execution may detect the violation of the property by the implementation and the violation of the standard IOCO conformance relation [Tretmans] between implementation and specification. On the other hand, if the verification step did not conclude (i.e., it did not allow to prove or to disprove the property), the test execution may detect violations of the property by the specification.

The informations about the relative inconsistencies between specification, implementation, and properties are reported to the user under the form of test verdicts. The approach is illustrated on the BRP protocol.

Test Case Generation for an Electronic Payment System with CSP-CASL

Holger Schlingloff (Fraunhofer Institut - Berlin, D)

In this talk, we report about an on-going project on the formalization and test case generation for the ep2 (electronic funds transfer / point of service) banking system.

As modelling language, we use the new formalism CSP-CASL, which combines the process algebra CSP (for the reactive part) and the common algebraic specification language CASL (for the data part).

In this project we plan to investigate how test cases for ep2-terminals can be automatically derived from our formal model.

Keywords: Test case generation, CSP-CASL, algebraic specification, process algebras, EMV, EP2, EFT/POS2000

Joint work of: Schlingloff, Holger; Roggenbach, Markus

Model-based Test Data Generation for Testing Integrated Modular Avionics

Aliki Tsiolakis (Universität Bremen, D)

Next generation aircrafts use generic computing resources based on the concept of Integrated Modular Avionics (IMA). These so-called IMA Modules provide standardised hardware and interfaces as well as a common standardised operating system. IMA can achieve a high degree of functional and physical integration by ensuring spatial and temporal partitioning and deterministic scheduling of avionics applications which run concurrently but without disturbing each other on one IMA module. Consequently, using IMA technology has an important impact on verification, validation and testing activities. The presentation outlines a strategy for testing IMA starting with single IMA modules and continuing with networks of IMA modules. In particular, it has to be verified that inter-module and intra-module communication flow complies with the application specific configuration and the defined module performance. For testing all possible communication flows, a test data generation algorithm is suggested which considers the I/O and scheduling configuration of the IMA modules and the performance information about the IMA modules and the network. To reduce the number of resulting test cases or to sort them according to their importance for testing, the algorithm can take restriction functions or heuristic functions into consideration which help to focus on a specific test aim. For test execution, a test setting is suggested where test control specifications load the test cases into the test applications using specific communication links. Similarly, the test execution logs are transmitted to test checker specifications for evaluation of the overall test result.

First evaluations have shown promising results that suggest future investigations particularly in the area of restriction and heuristic functions and considerations of additional parameters.

Testing Software with Spec# and SpecExplorer

Margus Veanes (Microsoft Research - Seattle, USA)

We give an overview of the model-based testing tool SpecExplorer and the modeling language Spec# used by SpecExplorer.

Spec# is an extension of C# with contracts and high-level data structures like sets, maps and sequences.

SpecExplorer allows the user to explore the state space of the model, to generate a finite state machine from the, typically infinite state, model, and to generate and run tests against an implementation under test. The tool supports nondeterminism and provides the view of testing a nondeterministic system as a game between two players. Generated tests are considered as game strategies.

Keywords: Model-based testing, games

Symbolic Test Case Generation for Primitive Recursive Functions

Burkhart Wolff (ETH Zürich, CH)

We present a method for the automatic generation of test cases for HOL formulae containing primitive recursive predicates. These test cases can be used for the animation of specifications as well as for black-box testing of external programs.

Our method is two-staged: first, the original formula is partitioned into test cases by transformation into a Horn-clause normal form (HCNF). Second, the test cases are analyzed for ground instances satisfying the premises of the clauses. Particular emphasis is put on the control of test hypotheses and test hierarchies to avoid intractability.

We applied our method to several examples, including AVL-trees and the red-black tree implementation in the standard library from SML/NJ.

Keywords: Symbolic test case generations, black box testing, theorem proving, Isabelle/HOL

Joint work of: Wolff, Burkhart; Brucker, Achim