ISSN 2186-7437

NII Shonan Meeting Report

No. 2014-16

Integration of Formal Methods and Testing for Model-based Systems Engineering

Tetsuya Tohdo (DENSO CORPORATION, Japan)
Werner Damm (Carl von Ossietzky Universität Oldenburg, Germany)
Alexander Pretschner (Technische Universität München, Germany)

Jun Sun (Singapore University of Technology and Design, Singapore)

December 1-4, 2014

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Integration of Formal Methods and Testing for Model-based Systems Engineering

Organizers:
Tetsuya Tohdo (DENSO CORPORATION)
Werner Damm (Carl von Ossietzky Universität Oldenburg)
Alexander Pretschner (Technische Universität München)
Jun Sun (Singapore University of Technology and Design)

December 1-4, 2014

Overview

This meeting aims to provide a forum for researchers and practitioners working on Formal Methods (FM) and Testing to identify research issues and bring related ideas, techniques, theories and tools in both research areas to real industrial solutions.

Critical systems manufactured in automobile, railways and avionics need to be designed, developed, operated and maintained with high-level of confidence. These systems need to keep evolving so as to cope with the ever-changing environments, and respond to new demands in a timely manner with high productivity. Observing the industry, the so-called "Semi-Formal" approaches are widely adopted, e.g., the use of executable models (one notable example is Simulink in the automotive industry) in conjunction with the use of traditional testing methods instead of formal techniques. While the adoption of results from Formal Methods research is gaining momentum, there continue to be gaps between practical industry needs and academic achievements. Among others, these gaps reflect concerns with return-on-investment, scalability, practicality, educational challenges, and the seamless embedding of methods and tools into an ever changing landscape of development processes. This current situation motivates us to organize this meeting. Model-based engineering approach based on the use of modeling descriptions with formal mathematical semantics (as used in FM) in combination with Testing is considered to be one of the best practical solutions over the system-lifecycle to ensure high level of confidence without decreasing productivity. However, the best way of combining FM and Testing techniques is not obvious, and we believe that there are numerous research issues on this approach.

The main purpose of this meeting is to recognize and identify research issues which need to be solved in order to develop engineering methods which effectively combine FM and Testing theories, techniques and tools. This forum has a strong focus on collaboration between academia and industry in order to

foster further development of this cutting-edge technology and aims to establish a strong tie between academia and industry especially to Japanese industry. In order to make this happen, we carefully selected invitees from academia who are working on systems engineering, foundational theories between Testing and Formal Methods, and principal researchers conducting research projects on embedded systems, systems engineering and verification, and practitioners who are practicing verification in model-based systems engineering.

We had twenty seven participants including organizers at this seminar. As the theme of this seminar is reflected, we had a good number of participants from several industries, e.g., automotive, aeronautic and IT-related, which add a unique feature to this seminar.

The following research topics are considered particularly relevant to the meeting.

- Test case generation using formal proofs and model checking
- Unified theory of Testing and Formal Methods which underpins both technique in a single framework,
- Contractual framework for specification, design and verification
- Combination of static analysis and dynamic analysis/testing
- Testing and symbolic verification
- Approximation and abstract interpretation
- Refinement calculus and Testing
- Requirement-based verification
- Model-based testing
- Runtime Verification

In order to identify the gaps between between research (on formal methods and testing) and practice, and then propose potential ways of closing the gap, the meeting is organized as follows.

Day 1 09:00 - 15:30: Introduction presentation and gap analysis introduction which open discussion to share understandings of major gaps and key obstacles

Day 1 16:00 - Day 2 15:30: Individual presentations and discussion on their research and its relation to the gaps

Day 2 15:30 - 18:00: Special industry session where we invite Japanese industrial persons who are interested in the seminar topic. This session features three keynote talks.

- "Motivation of NII shonan meeting 048" by Prof. Werner Damm (One of the organizers of this seminar)
- "An Academia-Industry Collaboration toward Open System Dependability" by Dr. Mario Tokoro (leader of DEOS association, Sony Computer Science Labs, Inc)

- "Expectation to Researchers" by Mr. Masahiro Goto (Director of ePF R&D division DENSO coroporation)

Tool vendors which exhibit their tools are the followings:

- ChangeVision Inc.,
- CATS Co. Ltd.,
- Gaio Technology Co. Ltd.,
- BTC Embedded Systems AG

Participants of this seminar, attendees from industry and tool vendors exchanged opinions and Q&A actively, and we believe that this interactions between academia and industry were very fruitful for the both sides.

Day 3 and 4: Roadmap session which consolidates the discussion on technology pushes (to close the identified gaps) into a single roadmap

Summary of Identified Issues

The participants of the meeting had a heated discussion on the gap between research (on Formal Methods and testing) and practice. The following is a comprehensive list of identified research questions and issues.

- Formal methods do not stick, e.g., one manager leaves, back to square 1.
- There are missing necessary skills/work force.
- We need convincing effort/reward evidence to justify modeling.
- How do we understand/measure ROI/effectiveness?
- New techniques are required for security, e.g., in dealing with malicious attacks.
- How do we make models generative (e.g., generating code, test)?
- How do we support supply chain (e.g., legacy systems, modularization, characterization)?
- We need debuggability of formal specifications (e.g., in the form of IDEs for formal specifications).
- How do we formalize requirements (specifically for adaptive systems) to avoid the problem of invalid specification?
- Formal methods and model-based testing must be connected with upstream/downstream activities.
- Formal methods solves only one half of the problem.
- Formal methods are not change-aware.

- How do we analyze evolving systems?
- We need interoperable analysis tools and techniques.
- We need methods/tools/theories for logically safe combination of formalisms, models, and techniques.
- There are missing open source software as well as missing industry standards.
- How do we address the inability to tell when to prove vs. when to test (e.g., how long will your model checker take?).
- How do we keep the tester in the loop (e.g., interactions with tester, have intermediate artifacts)?
- Formal methods tools need to produce reviewable evidence (e.g., test cases).
- Formal methods are not particularly scalable which make them unusable (e.g., users often spend more effort in tool tuning than modeling).
- We need complete and reduced and generic strategies for very large concurrent systems.
- How do we produce heterogeneous evidence (e.g., combine incomplete testing and proofs) and chain of arguments?
- Can we trust the formal method tools?
- How do we obtain "good" test cases/properties?
- We need regulatory compliance (e.g., using de facto tool chain, tool qualification, recommended techniques).
- How do we deal with the risk of change or provide liability?
- How do we cope with disruptive technologies?
- how do we deal with complex environments and uncertainty?
- How do we test open systems (e.g., interface to cloud)?
- We need techniques/methods/tools for handling concurrency.
- For hybrid systems, we need to model not just the software, but also the (physical) environment.
- How do we define and achieve consistency/conformance between different models or approximation?
- How do we make formal methods transparent to the user?
- Developing reliable tools often requires an effort too high for academia.
- How do we deal with highly parametrized/parameterizable software
- How do we do generic testing for product lines?

- Tool vendors need to provide transparency, e.g., evidence in terms of completeness, soundness and coverage.
- How much confidence does a proof give for a system to be working correctly in its real environment? How much confidence does testing yield? What do we do/need at which stage of the development process?
- How do we allow/handle feature interactions?
- We need taxonomy for system design artifacts allowing placement/role of tools.
- We need crowdsourcing for testing.
- How do we conduct performance testing?
- How do we handle domain specificity?
- How do we provide predictability?

Summary of Identified Technology Pushes

All of the above-mentioned questions and issues have been discussed during the meeting. Some technology pushes which ought to be helpful in solving the issues are identified. In the following, we document some of them, in the form of "what" (i.e., what is the issue), "why" (i.e., why such an issue exists) and "how" (i.e., how do we overcome the issue). Naturally, we do not have all the remedies.

Technology Push 1

What: When using implementation-based formal methods or model-based testing techniques, executives want to know how "good" this methodology, tool chain, or technology is. Currently, we do not have generally agreed-upon characteristics of the "quality" of formal methods or model-based testing techniques, and we do not have a lot of evidence that these characteristics are met.

Why: Deploying formal methods or model-based testing techniques can contain significant risk in terms of cost and schedule. Making the judgement about when to apply them currently requires understanding of both the application domain and analysis strengths and limitations. In addition, management tends to be risk-averse. The scalability of formal methods or model-based testing tools is problem specific. Many work well only for specific kinds of systems or properties, e.g., embedded systems: complex mode logic (systems) or buffer overflow (properties). The tools do not work well on systems with complex data types or mathematics. Certification authorities have previously not allowed credit for formal methods activities.

How: The possible solutions are as follows.

- ROI: Start from pilot projects in "sweet spot" for formal methods or model-based testing tools in known problem areas to gain credibility.
- ROI: Designate technical champion and provide training.
- Risk reduction: If possible, choose standard notation that is already in use. Focus on formalizing requirements in standard notation. Improves V&V process throughout design cycle: proof, autotest generation target, oracles for standard test process, and runtime monitoring. Relatively low cost. Integrate tools into existing IDEs: Simulink / Eclipse. Educate industry on benefits and risks of formal methods.

Technology Push 2

What: There is a shortage of engineers skilled in advanced software engineering topics such as model-based testing and formal methods, in particular, people who have also domain knowledge in their product area (automotive, etc.). This results in a high managerial risk when using new technologies, because it is not clear whether they are sustainable with the newly hired workforce. New graduates in Engineering and Computer Science are too few in number and and have no or little cross-disciplinary skills.

Why: Many industries have been slow to recognize the importance of Computer Science as one of the technological cornerstones of their products. As a consequence, there are relatively few Computer Scientists in industries that have traditionally been the realm of mechanical and electrical engineers. University education of engineers and (computer) scientists has traditionally been putting emphasis on the depth in their main subject rather than on interdisciplinarity.

How: Universities and industry need to engage stronger in continuing education in advanced software engineering techniques. Industry must give employees the time to do so. Universities must cater better for the needs of industry by providing tailored courses where attendance is made flexible by e-teaching offers (e.g. MOOCs). It is important that courses for industry address scalable technologies that can actually be applied in an industrial context. Politics must give universities additional resources to fulfill this mission that goes beyond their traditional role. We need to develop and push joint curricula in CS/EE. Curricula in CS should include engineering tracks, not only selected random courses in application subjects. Curricula in EE should include some fundamental CS concepts (and taught by CS faculty), not only programming. One possibility to make topics such as MBT and FM more accessible to EE students is to teach them in Ninja-style, e.g., embedded into hands-on projects that demonstrate the usefulness of of advanced software engineering concepts. CS courses for engineers should make explicit the differences between the disciplines, for example: thinking in scenarios vs. thinking in terms of invariants. Basic insights into the underlying theoretical concepts are considered to be essential, even though they should be kept lightweight.

Technology Push 3

What: It is a business decision to weigh the potential benefits of introducing a technology offering clear market benefits against the risks stemming from confronting designers with a new disruptive technology. Technically such risks arise from the implicit dependence of all previous development on characteristics of the current technology base. The effort of re-engineering required to find all such technology dependence may be commercially inviable. At least as critical is the impact on designers, who can no longer rely on their acquired skills and experience but must master the learning curve of adopting the new technology. Examples of disruptive technologies in the automotive domain: Autosar, multi-core processors, and new IC fabrication processes resulting in complete new qualities of failures (e.g.. aging) and power-aware algorithm design.

Why: Causes for introducing disruptive technologies forced from external uncontrollable market conditions (e.g., new fabrication processes, multi-core, mismatch of innovation cycles of supported devices) estimated benefit of introducing disruptive technology is so significant that it outweighs the drawbacks of introducing a disruptive technology.

How: For external dependencies, we would suggest enforcing standards for integrating devices with shorter innovation cycles; enforcing early awareness of disruptive technology changes along supply chain; enforcing layered design for encapsulating effect of disruptive technologies as far as possible; and enforcing architectural style guide. For company controlled decision, we would suggest incremental role out strategy.

Technology Push 4

What: There are several technical issues to verify open systems, but we need to clarify what are closed and open systems first.

Why: As an example, analysis of protocols that are exposed to attacks rely on understanding what adversaries are capable of, while adversary models are not main focus for cyber-physical systems, or protocols and cloud APIs that build on top of a trust model. Lack of a security strategy for open systems has the obvious negative business impacts.

How: (Clopen systems approach - as in topology) Two areas of business opportunities exist (in some cloud vendor scenarios): model based testing for internal and external consumption. Internal use of model-based testing benefits from buy in and availability of skill-sets from the engineering organization. Commonly adapted strategies in cloud vendors is to rely on a small fraction of live customers for testing: you can fail some of the customers some of the time without them noticing (they just refresh their browser or their client software has built-in retry mechanisms). So the value proposition of model based testing really depends on whether it is intended to be used in such scenarios as opposed to, say, protocol documentation (which is a brilliant application of model based testing, but mainly driven by political requirements). External use has to rely on customer scenarios that can be well scoped and sufficiently common.

Technology Push 5

What: Systems are offered highly configurable and parameterizable for specific customers. How can we develop generic software platforms that can be customized easily and fast with low customer-specific effort?

Why: Frameworks of software (or system) are the most crucial for getting benefit of highly configurable software. Architecture of such frameworks requires coordination of business strategy comprehension and software (or systems) engineering, which is hard.

How: To develop methodology which

- models business requirements;
- bridges business requirements and software (or system) framework design;
- and models parameter structure by assistance of V&V technologies.

Technology Push 6

What: In a large company with divisions across many domains, modeling techniques, cultures, etc., how can methodologies, tools and knowledge be shared in order to increase efficiency?

Why: Managing interdisciplinary is essential. Because humans are flexible, methodologies and technologies need to handle artifacts from different culture in a consistent but lack of humans' flexible power..

How: To develop methodologies which assist organizational management by

- bridging different domains online;
- keeping consistency as much as possible;
- and allowing evolutional aspects.

Technology Push 7

What: Most existing tool providers for model-based testing tools are relative small, and offer proprietary solutions. It is also too expensive to build the technology in house. The big platform providers, on the other hand, do not offer officially supported tools. It is a big risk to lock-in a technology for which the provider can go out of business. Yet, vendors are liable for the complete product that is shipped to customers. Therefore, 3rd party software, integrated OSS, etc., needs to fulfill the same quality/safety/security standards as the proprietary development. As the development of third party components as well as legacy systems is not governed by the own software development process (security development process), the use of such components increases the risk and, thus, the decision from which vendors to buy/consume components is a business decision that needs to be taken on the CEO level. Academia and research, on the other hand is well-served with problem sets related to industry. So it is

useful to make industry partners aware of these formats and encourage making benchmarks non-proprietary (e.g., dynamic execution traces from SAGE, Static Driver Verification regression suite is available as anonymized Horn clauses in SMT-LIB, while the IC industry has so far been very reluctant to share benchmarks because one can directly reverse engineer designs, their main IP).

Why: Model-based testing is a relatively specialized field. It requires a substantial involvement from academia (as many problems are really hard and require research). But academia is not well posed for engaging in long running processes and working from point of view for productization. The market is also fragmented among many different specialized areas that each need their technologically sophisticated solution.

How: In automotive domains (and perhaps in some other engineering domains as well), a tool chain is generally built to standardize the development process. Some safety standards require to certify/qualify tools to be used in the tool chain. Particularly verification tools need to be certified/qualified at a higher level, which is an obstacle for any newcomers to go into the mainstream tool chain. A possible solution in this setting is to embrace OSS and/or interoperability/exchange format initiatives around model-based testing. One approach to enable academic impact is to promote formats that enable decomposing problems, such that subproblems can be solved using specialized tools (e.g., SAT solvers, SMT, Petri-nets, hybrid system flow-pipe analysis); industry solutions otherwise tend to favor holistic end-to-end experiences which can easily overflow into exchange formats.

Technology Push 8

What: Tools need to be usable.

Why: Adoption depends on normal workforce using those tools.

How: Making models debuggable via animation, model analysis, etc. This could be achieved through

- IDE integration and quality;
- avoiding excessive tool complexity;
- ninja formal methods (i.g, don't bother the user with unnecessary detail);
- making it as simple as possible but not simpler;
- and exposung relevant information in "user language" (domain-specific).

Concluding Remarks

As was shown above, we have come up with a very extensive list of issues as to gaps between research and practice and their relevant technology pushes, and finally identified roadmaps (how section of each technology push) for make FM

and Testing more applicable for industry. This is the major achievement of this seminar. Our future plan is to publish the results obtained in this seminar.

Overview of Talks

Challenge to Requirement-based Verification

Tetsuya Tohdo, DENSO CORPORATION

Recently DENSO CORPORATION initiates a program to enhance research activities on Software-intensive Systems. Requirement-based Verification is ongoing research according to this program, which focuses better use of formal methods targeting test case generation. This talk presents an experience of re-constructing abstract models from existing detailed specifications in order to generate test cases as a feasibility study.

Defect-Based Testing

Alexander Pretschner, Technische Universität München

Whats a good test case? Text book, practice and intuition suggest: one that reveals a fault. But the thought experiment of a perfect program shows the deficiency of this definition in this case, there would not be any good test cases. A more adequate definition is that a good test case reveals likely potential faults with good cost effectiveness. The model-based testing community tends to answer this question in one of two ways: good test cases are defined by coverage (because we can) and by explicit test purposes (because we sense that there must be more, but others should do the work). We argue why coverage-based testing is inherently problematic, if not useless, and propose to complement explicit test purposes by defect models. These encode what typically goes wrong in a specific domain, technology, company, or application family, and describe what can potentially go wrong, thus catering to the above definition of good test cases. We discuss the nature of potential defects, formalize them, provide examples, and discuss their operationalization for test case generation also outside the domain of model-based testing.

Bridging Formal Methods and Real-World with Testing and Learning

Jun Sun, Singapore University of Technology and Design

One of the difficulties in adopting formal methods is that users must develop models first. One way of obtaining (perhaps imprecise or incomplete) models is through testing and learning. We argue that a model learned from the actual implementation, with a good test suite and a capable machine learning algorithm, could potentially have better and more controllable quality. Such models would be useful to bridging the gap between the current practice and formal methods. This is especially so if the model can be later improved systematically through different means, for instance, through a formal verification and refinement loop. The questions are then: can we design effective testing techniques which would facilitate better learning? How do we decide what to learn (for certain analysis task) and how do we learn?

Creating Confidence in the Correctness with formal methods and testing

Toshiaki Aoki, JAIST

(Position Presentation)

Cyrille Artho, Nat. Inst. of Advanced Industrial Science and Technology (AIST)

We present a brief overview on ongoing research in the fields of model-based testing and the verification of networked systems at AIST. In model-based testing, we are pursuing work on combinatorial testing with tool Calot, and the testing of state-based systems with tool Modbat. To verify networked software exhaustively, we use our own extension to Java PathFinder. In addition to technologies that we develop, our experience has also shown that writing a model at the right level of abstraction and expressiveness is a challenge for a human, requiring insight and experience.

SMT for Networks and Quantities

Nikolaj Bjorner, Microsoft Research, Redmond

Many modeling and static analysis questions rely on formulating queries in a logical form. Satisfiability Modulo Theories solvers have been developed in the context of significant technological advances in theorem proving coupled with innovative uses of logic solvers for analysis and software design. My work currently involves extending the use of SMT along two vectors: Network Verification and Optimization. Network Verification is an exciting opportunity for formal methods. The networking domain is undergoing a revolution thanks to a rapid build-out of large scale cloud data-centers, the availability of commodity devices for network control. These changes are driven by market forces, but can really only be enabled and enhanced by better software engineering methods. We have so far used the SMT solver Z3 to check configurations in Azure, perform model checking of routing configurations, and verify controllers. My work on optimization in Z3 aims at broadly supporting applications of SMT solvers from different domains to allow them to solve satisfiability with objective functions. Many applications require not only establishing that there is a feasible solution, but require a best solution given some objective functions.

Combining MBD, MBT and FM

Udo Brockmeyer, BTC Embedded Systems

Integration of Formal Methods and Testing for Model-Based Security Engineering

Achim D Brucker, SAP AG

We present a brief overview of various security testing works that range from applying off-the-shell tools (both dynamic tools as well as static program

analysis) to theorem-prover based testing for ensuring the compliance of systems to high-level security policies. Moreover, we report on the process of selecting the most appropriate (security) testing tools during product development derive open research questions based on our experience in developing, introducing, and applying (security) testing tools at SAP SE.

Prove If You Can, Test If You Cannot

Rance Cleaveland, Department of Computer Science Exec. And Sci. Dir., Fraunhofer USA CESE,
University of Maryland

Current formal methods focus on mathematical proof as a means for establishing that a system is correct with respect to a formal specification. This perspective can limit the applicability of formal methods, since the development of such proofs remains a very difficult task requiring specialized expertise, even with computer assistance. This presentation argues that formal-specification approaches that support both proof and testing as V&V technologies can enhance the practical usefulness of formal methods.

Formal Verification for Flight Critical Systems

Darren Cofer, Trusted Systems Advanced Technology Center Rockwell Collins

The complexity of software onboard aircraft is increasing rapidly. Current methods for verification and certification are at (or beyond) their limits. We need better engineering tools for designing and assuring safety and correctness. This must include analyzable design models and formally specified requirements. Test generation and formal verification should be driven from the same set of formal requirements. Important issues to deal with include use of composition to improve scalability and bridging the gap between theory and practice.

Integration of Formal Methods and Testing

Wolfgang Grieskamp, Google Inc.

(Position Presentation)

Reiner Haehnle, Department of Computer Science Software Engineering Group
Technische Universität Darmstadt

Toward System-level Testing

Tomoyuki Kaga, Toyota Motor Corporation

Model-based Combinatorial Testing

Takashi Kitamura, Nat. Inst. of Advanced Industrial Science and Technology (AIST)

A model-based approach for combinatorial interaction testing is proposed. In this technique, system models are described using extended logic trees, and combinatorial test cases such as T-way tests can be constructed automatically. The testing technique is developed using formal techniques, such as formal semantics, model transformation, and correctness proof.

(Position Presentation)

Takuro Kutsuna, Toyota Central R&D Labs. Inc.

Evolution-Aware Monitoring-Oriented Programming

Darko Marinov, University of Illinois at Urbana-Champaign

Monitoring-Oriented Programming (MOP) helps develop more reliable software by means of monitoring against formal specifications. While MOP showed promising results, all prior research has focused on checking a single version of a target software application. We propose to extend MOP to support multiple software versions and thus be more relevant in the context of rapid software evolution. Our approach, called eMOP, is inspired by regression test selection, a well studied, evolution-centered technique. The key idea in eMOP is to monitor only the parts of code that changed between versions. We illustrate eMOP by means of a running example, and show the results of preliminary experiments. eMOP opens up a new line of research on MOP. It can significantly improve usability and performance when applied across multiple versions of an application, and is complementary to algorithmic MOP advances on single versions.

(Position Presentation)

Kenji Taguchi, CAV Technologies, Nat. Inst. of Advanced Industrial Science and Technology (AIST)

(Position Presentation)

Willem Visser Stellenbosch University

(Position Presentation)

Michael W. Whalen, University of Minnesota

Despite considerable research in software testing over 30 years, the factors that influence the effectiveness of testing are still poorly understood. Much testing research is concerned with meeting a particular testing objective, whether

or not the objective is good at finding faults given the system under test. Our research has been concerned with examining the range of factors influencing software testing and creating test metrics and test generation strategies that are robust and effective.

On Theorem-proving based Testing

Burkhart Wolff, University Paris-Sud, LRI

While Formal Testing and Theorem-Proving are still perceived as antagonisms by many, there is a growing research field using the combination of both to increase the applicability of Formal Methods in industry, in particular in the area of Safety-and Security critical systems requiring formal certifications. My research on model-based testing is centered around the HOL-TestGen System (a plugin of Isabelle/HOL), which places formal theory development, theorem proving, symbolic computation, and constraint solving in the center of formal test development: we call this paradigm "theorem prover based testing".

(Position Presentation)

Mark Utting, University of the Sunshine Coast

A brief overview of several recent model-based testing and verification projects that I've been involved in. These include our book on "Practical Model-Based Testing", the Jumble mutation analysis tool for JUnit tests, unit testing of Z specifications with positive and negative tests, correctness checking of the JStar parallel programming language using SMT solvers, and work with the Whiley verified programming language in collaboration with Victoria University of Wellington. The common theme is getting computers to automate more of the checking for errors in our programs.

Model-based Testing Industrial Application and Challenges for Basic Research

Jan Peleska, Universität Bremen

Jan Peleska's research is specialised on MBT based on formal methods, with application domains avionics, railways, and automotive. His contributions to the technology vectors are mainly to (1) requirements-based modelling and verification, and (2) fault-based modelling and testing, but he is also interested in learning approaches for incremental test model creation. Together with Verified Systems International GmbH, Jan Peleska and his research group have developed the industrial-strength MBT tool RT-Tester which is continuously updated with the latest MBT strategies and supporting technologies.

Symbolic Automata Theory

Margus Veanes, MSR Redmond

Symbolic automata theory lifts classical automata theory to rich alphabet theories. It does so by replacing an explicit alphabet with an alphabet de-

scribed implicitly by a Boolean algebra. How does this lifting affect the basic algorithms that lay the foundation for modern automata theory and what is the incentive for doing this? We investigate these questions here. In our approach we use state-of-the-art constraint solving techniques for automata analysis that are both expressive and efficient, even for very large and infinite alphabets. Symbolic automata enable a separation of concerns that can be beneficial for many different purposes. One potential usage of symbolic automata is within compositional approaches to testing, since symbolic automata are closed under different forms of composition that are typically needed in testing.

(Position Presentation)

Alexandre Petrenko, Computer Research Institute of Montreal

Hybrid Systems Modeling and Verification at Bosch

Matthias Woehrle, Bosch Corporate Research

In automotive systems, software controls physical processes with complex dynamics using sensors and actuators. Examples include engine control, active safety systems, and braking systems. System functionality is evaluated on the composed (closed-loop) system of software and its effect on the physical plant. Hence, in design, development and verification, we deal with so-called hybrid systems that include discrete parts, such as operating mode switches and software computations, as well as continuous parts, such as the physics of the plant. Within Bosch Corporate Research, we investigate and evaluate novel methods for the efficient design, development and verification of hybrid systems. In particular, we are interested in model-based techniques on different levels of abstractions. One specific strand of research tries to enhance the state-of-the-art in testing using model simulations by automating the testing process leveraging formal specifications and falsification-based methods.

Model-based Testing a Modular Message Oriented Middleware in Intra-logistics

Benjamin Kraemer, Jungheinrich Logistiksysteme GmbH

Depending on the focus of a warehouse, its intralogistic processes highly vary and no warehouse is like another. The used software has therefore to be very flexible to be useful in all occurring scenarios. Jungheinrich is one of the leading international companies in the material handling equipment, warehousing and material flow engineering sectors. As general contractor, it does not only produce the fork lift trucks to transport goods but also a middleware software to integrate the hardware into the IT infrastructure of its customers to maximize the use of the trucks. This software is build using a component based, message oriented framework. It is hard to manually define a good set of test cases due to the high configurability of each component. To improve the software quality, model-based testing caught our interest since we are already using models to generate parts of the software. It still has to be analyzed if the costs justify

the benefits. Also other questions like how to keep the specification and code synchronized have to be cleared.