Dagstuhl Seminar 10421
Model-Based Testing in Practice
October 18–22, 2010

Wolfgang Grieskamp[1], Robert M. Hierons[2] and Alexander Pretschner[3]

[1] Microsoft, USA
wrwg@microsoft.com
[2] Department of Information Systems and Computing, Brunel University
Uxbridge, Middlesex, UB8 3PH United Kingdom
rob.hierons@brunel.ac.uk
[3] Karlsruhe Institute of Technology
alexander.pretschner@kit.edu

1 Model-Based Testing

Software testing is one of the most cost-intensive tasks in the modern software production process. Model-based testing is a light-weight formal method which enables the automatic derivation of tests from software models and their environment. Model-based testing (MBT) has matured as a rich research area in the last decade, with a significant body of research and applications. The academic community is well established with many conferences, workshops, and research projects. Tools for model-based testing have been developed both as research prototypes and as commercial or semi-commercial applications brought to users by midsize and enterprise-level companies, and applied in large scale projects.

In the family of model-driven approaches, model-based testing can be seen as a success story in particular with respect to the degree of mechanical processing and automation that has been achieved, and the adoption in practice. The successful deployment of model-based testing in industrial settings can be seen in the telecommunication domain, chip cards, specific Windows components, and embedded systems in general. An interesting issue is under which circumstances we can expect these successes to carry over to other domains and families of systems as well (e.g., distributed systems; testing the cloud).

2 The Seminar

This Dagstuhl Seminar brought together top researchers, young scientists, and practitioners to discuss the state of the art, compare it with practical experiences, and derive future directions for model-based testing research and industrialisation. Model-based testing has proved promising even in industrial terms and currently seems at the verge of large-scale deployment. While previous Dagstuhl seminars around model-based testing (04371 in 2004 and 98361 in 1998) were dedicated to bringing research results into practice, in this seminar we aimed

Dagstuhl Seminar Proceedings 10421
Model-Based Testing in Practice
http://drops.dagstuhl.de/opus/volltexte/2011/2925

to take advantage of the relative successes of model-based testing and have the discussion guided by available industrial experience. For this reason, this seminar was designed with a particularly high industry participation rate. We also started with a day of talks from industrialist with the focus being on the use of MBT in industry and the challenges faced.

3 Challenges from 2004

In Seminar 04371, in 2004, 10 main MBT challenges were identified and we discussed the progress that has been made on these challenges. We now briefly describe the outcome of this discussion.

- **Challenge 1**: Measures for coverage and test quality: Model-based test generation algorithms can produce many test cases, but there is no method yet to compare the quality of two test suites. Quantification of test quality is desirable to compare test suites and to select the best one.
  Conclusions: It was agreed that there has been little progress on this challenge. Mutation based methods can help us to compare the quality of test suites, and these approaches have been extended to models. However, the relationship between the ability of a test suite to detect mutants and to find real faults is still an open question, despite some recent promising empirical results.
- **Challenge 2**: Test purposes and test scenario control: Often, it is necessary to guide or control the model-based generation of test cases so that the interesting, error-prone, or tricky parts of a system under test (SUT) are tested. How to identify and specify suitable test purposes, and how to control and guide the generation of tests is still unclear.
  Conclusions: The view was that this is still a challenge. There has been some limited progress, for example showing how test purposes can be produced based on coverage, but it is unclear how this relates to detecting faults in real systems.
- **Challenge 3**: Merging different models: Many test generation methods work for particular aspects of behaviour, e.g., state-based models mainly test for control flow. For real systems many aspects must be tested at the same time: state, control flow, data flow, data transformation real-time, etc. This requires integration of the corresponding modeling formalisms, and of the test generation methods.
  Conclusions: Current practice still involves using separate models for different aspects of a system and typically these are not integrated. As a result, this was seen as still being a challenge. However, there are models that combine control and data and also control and time and MBT methods have been devised for such models.
- **Challenge 4**: The role of quiescence, timed and untimed: An SUT should do what it is required to do. Doing nothing is a particular form of such a requirement and is expressed as being quiescent. Quiescence is treated differently

in different test generation methods. Better understanding of quiescence is necessary, in particular, when real-time is involved.

Conclusions: The view was that significant progress has been made on this topic and in real-time testing in general and this is no longer a major challenge.

- **Challenge 5**: Modelling method invocations: Many test generation methods have their origins in a message-oriented paradigm. Current component-based systems work differently: they are based on the object-oriented paradigm of method invocations. The modeling of method invocations for testing is not yet well-understood, in particular if parallelism and multi-threading is involved, so that several method invocations can exist concurrently.

  Conclusions: The view was that there has been significant progress in the use of languages for such systems, an example being languages in the UML. There are MBT tools for such languages and this is no longer a challenge.
- **Challenge 6**: Integration of techniques: The boundaries between such techniques as model-based testing, model checking, static analysis, abstract interpretation, theorem proving, constraint solving, run-time verification, etc. diminuish. Integration of these techniques is necessary to be able to choose for every task the best combination of techniques.

  Conclusions: It was agreed that there has been significant progress on this challenge. Among others, (bounded) model checkers and constraint solvers are standard tools in the model-based test case generation toolbox today.
- **Challenge 7**: Modelling test interfaces: For the execution of tests, the tester is connected to the SUT via some kind of test interface. The behaviour of this interface, e.g., an operating system pipe which behaves as a FIFO queue, must be taken into account when tests are generated. Research on different kinds of test interfaces and their influence on test generation and observation is desirable.

  Conclusions: It was agreed that there has been relatively little progress in this area. There has been some work on specific cases, such as when there are buffers and (to a lesser extent) when the testers are distributed. However, many issues remain.
- **Challenge 8**: Tool architecture frameworks: Integration and interoperability of different (test) tools is desirable, e.g., interoperability of test generation tools and (on-the-fly) test execution tools.

  Conclusions: The integration of different tools is still seen as being desirable but there appears to have been little progress on this challenge.
- **Challenge 9**: Model based testing of non-functional properties: Most theory, methods, and tools for model-based testing have been devoted to testing of functionality. Model-based testing of other quality characteristics, such as security, reliability, performance, usability, etc., often referred to as non-functional properties, is an interesting field of research.

  Conclusions: There has still been some work in this area, in particular on reliability, safety, performance and security. However, this is still a major challenge.

– **Challenge 10**: Promoting MBT in industry: To advance the industrial usage of MBT, it is necessary that the methods scale well to industrially sized problems, that they are sold with the right level of expectation, and that feedback from case studies is used in the next generation of MBT methods and tools.
Conclusions: MBT is now used in many more companies and significant case studies are appearing. While it is still crucial to promote MBT in industry, there has been significant progress on this challenge.

We can see that there has been progress on meeting several of the challenges, but some also remain. Given that these challenges were identified only five years ago, it is natural that not all challenges have been overcome. However, we are encouraged by the fact that MBT has clearly become industrially applicable in this short period of time; this is a sign of very lively and productive communities of researchers and test engineers. Naturally, new challenges have arisen.

4 Challenges Identified

During the Seminar we discussed the key challenges for the community. Within this discussion we identified three groups of challenges and specific challenges within these groups. We now discuss the challenges identified by group.

4.1 Challenges for conformance and test selection

– **Concurrency and Distribution.**
Concurrency and distribution in the SUT affect the ability of the users or testers to observe behaviours. In particular, if there are distributed testers then we obtain a set of local observations rather than a global observation. There is a need to develop new conformance relations that reflect this reduced observational power. Naturally, this also affects the oracle problem: if we have different conformance relations then we need new ways of checking observations against a model.
Traditional deterministic models are unsuitable for distributed and concurrent systems. There is therefore the question of what type of models are suitable. In addition, there is potential to reflect some of the issues met, such as communications being asynchronous, in either the conformance relation or in the model (by e.g. modelling the queues). It is unclear which approach leads to least complexity and greatest potential to reuse current methods and tools. Naturally, there are also consequences for test selection and test execution is radically changed if we have separate independent testers at the different system interfaces.
– **Model analysis and test selection.**
There are clear benefits to using model analysis techniques to detect anomalies as early as possible. For specifications and design models, this can lead to the early identification of potential problems. The analysis of test models can also identify requirements errors but can also avoid testing from an

incorrect model. However, such analysis techniques can have an additional benefit: they can be used to drive test selection. There has already been some work along these lines but there is scope to extend this. In particular, most work in this area has used model checkers and there has been relatively little work using other analysis techniques. There is also the question of whether models can be modified to simplify the process of test generation and possibly also to lead to simpler test cases.

- **Non-determinism in models and SUTs.**
  Many models and systems are non-deterministic. Non-determinism leads to additional challenges in testing. One practical problem is that testing often has to be on-the-fly: methods that devise preset test cases suffer from a combinatorial explosion. However, in order to apply on-the-fly methods one requires a test tool/infrastructure that is sufficiently quick and this can be difficult for systems that rapidly interact with their environment.
  There are alternative sources of non-determinism, such as concurrency and abstraction. It seems likely that these different sources of non-determinism will require different test methods.
- **Coverage and fault detection.**
  Many test selection methods are based on a notion of coverage. However, there is only limited evidence regarding the relationship between coverage and fault detection; the community would benefit from the development of an elegant theory that explains this relationship. In addition, there are often many alternative models at different levels of abstraction. It is unclear how the level of abstraction influences the effectiveness of a coverage criterion and how we can evaluate models with regards to potential fault detection.
- **Fault models.**
  Fault models can be used to represent possible or likely faults. When using a model M for testing, a fault model can be represented as a set of mutations of M: versions of M with faults/differences introduced. This allows one to reason about test effectiveness. However, there is a need to better understand what types of fault models are suitable and how these relate to faults in real systems.

4.2 MBT and Cloud Computing

The Cloud is special because is brings together the following aspects: it is large scale; there is significant amount of concurrency; it is highly distributed; and it is dynamically changing. These properties introduce significant challenges for any engineering method and here we describe some of the challenges for using MBT for cloud computing.

- **Controllability, observability and test oracle**
  It is known that controllability and observability are affected by distribution: if the system interacts with its environment at physically distributed interfaces then a tester at one interface only observes observes events at its interface and no tester has a global view. As a result of testers not having a

global view, a tester might not know when to supply an input. These issues make testing more difficult. Sometimes we can overcome these problems by allowing the testers to communicate during testing but this can make testing more expensive since there is a need for an external communications network. It can also make testing take longer since testers have to wait to receive messages. There is a need to understand the trade-off between the cost of including communications between testers and the benefits it brings. There is also the question of whether there should be a global test oracle of whether each tester should have its own local oracle. Finally, there is the potential to have a Cloud infrastructure that is designed for testability.

- **Testing against simulator vs. real cloud**
  In testing software designed to run on a Cloud infrastructure we might simulate the infrastructure. This can make testing much simpler but has some clear disadvantages: the simulation might be incorrect and even if it is correct it will be an abstraction of the Cloud infrastructure and this abstraction could lead to misleading results. If we do not use a simulator then we may need to deploy the test cases into the Cloud. This makes testing more complex and relies on properties of the Cloud. There are also likely to be additional costs where the provider of the Cloud charges for its use. There are challenges relating to the development of appropriate simulations and also the understanding of the trade-off between cost and effectiveness when using a simulation.
- **Non-functional testing becomes more important**
  Cloud systems will typically have important non-functional requirements contained in Service Level Agreements. There is the challenge of producing MBT methods for testing such requirements, a problem that is complicated by the nature of Cloud systems. There is also a potential opportunity for MBT since MBT methods might be used to certify cloud service quality?
- **How to get repeatability?**
  Cloud systems can be dynamic and non-deterministic and so we can lose repeatability in testing. For example, we might run a test, observe a failure, but not be able to reproduce this failure. This can make it more difficult to locate and fix a fault. There is the challenge of devising methods to help make testing repeatable.
- **Applying existing methods of concurrency testing**
  Cloud systems are likely to be highly concurrent. Several testing methods have been produced for testing concurrent systems. However, these are typically for testing multi-threaded systems. There is a need to devise such methods that are specialised for Cloud systems: they need to scale to large systems and also to work with systems that are highly distributed. The properties of Cloud systems are likely to introduce significant theoretical and practical challenges.
- **MBT vs. Runtime Monitoring**
  In runtime monitoring we observe the behaviour of the system and check that the observations are consistent with certain requirements. There is the potential to use the same model in MBT and runtime monitoring. Runtime

monitoring could be a good complement for testing. It is potentially less costly and easier to conduct when compared to MBT since there is not the need to generate and apply test cases. However, runtime monitoring is a form of passive testing and so the tester does not decide which parts of the system are to be tested: this depends on how the system is used. There is a need to understand how MBT and runtime monitoring can complement and where each is suitable.

– **Strong collaboration between research and industry**
There is a need for more significant real collaboration between academia and industry. In order to make experimental results more relevant, academia needs real environments and subjects. These have to come from industry. All should gain from such collaboration: if companies can be persuaded to provide such environments and subjects then the results of experiments will be more relevant and will be particularly relevant to these companies.

4.3 Adoption, Model-Driven Development, Functional and Extra-Functional Requirements

– **Ensure Model Quality.**
The quality of the model used is important, but how can we measure this? Are there criteria that we can use to evaluate the quality of models? It appears that there are no standard criteria and it seems likely that criteria will vary with problem and/or domain. There is also likely to be a trade-off between factors, such as clarity/abstraction and containing all of the important aspects.
There are also issues related to the modelling language since different modelling languages can lead to different tests being derived. The goals of tests generated from models depend on the features of the modelling language. It is unclear how we can decide which modelling languages are most suitable and also how we measure the suitability of a language.

– **How to Teach / Learn Modelling?**
It is often said that many developers and testers find it difficult to produce appropriate models. Assuming this is the case, there is the challenge of finding effective ways to teach modelling. It might be helpful if researchers and practitioners published 'good' models and possibly also 'bad' models. Naturally, what is meant by a 'good' or 'bad' model depends on the intended use of the model and so there may be a need to produce several sets of examples. Naturally, tool support for model authoring and analysis is also important.

– **Is There a Methodology for Modeling?**
The development of good models would be facilitated by there being suitable methodologies. The methodology may well depend on the type of language used and, for example, whether it is compositional. It may also depend on the intended use of the model since the model can completely change for different concerns.

– **Do we need Multi-Faceted Models?**

Different kinds of models are needed for non-functional concerns and it seems that this is particularly the case for distributed systems. As it turned out, this same question was, in a slightly different context, discussed in the context of the cloud; a second listing of the insights is omitted here for brevity's sake.

- **Adoption of Model-Driven Engineering**
  Much of industry still does not use model-driven engineering (MDE) despite the significant amount of research in MDE. In part this may be because the MDE community has not been effective in promoting some of the advantages of using formal models. There should be particular emphasis on this in MBT since the use of a formal model can allow test generation and execution to be automated but adoption is still limited. We are starting to see empirical results that show significant savings resulting from the use of MBT in industry and this may help promote the use of MBT and, in turn, MDE. However, the MDE community would benefit from additional case studies, possibly in an area such as the nuclear power plans in which there are several different concerns, such as safety and security.
  There are several alternative types of modelling languages, For example, the UML has the advantages of being standardised and having significant tool support. However, some prefer domain specific languages (DSLs). In addition, different languages are appropriate for different stakeholders. For example, one might use UML for designers and a language such as C++ for testers.
- **Relate requirements and models.**
  Requirements are typically text documents that cover several kinds of aspects such as security, safety, and timing. There is the question of how models can be derived directly from requirements and there are some methods, such as sequence enumeration, for doing this. If we can relate parts of the model to requirements then we gain traceability. This should help the tester to communicate the purpose of a test case (what requirements it tests) and also for failures to be traced back to requirements. Such traceability is crucial if MBT is to be used more widely and there is therefore a need for the development of methods for deriving a model in a manner that facilitates traceability.

5 Conclusions

There has been significant progress in MBT since the previous Dagstuhl seminar in 2004. In particular, many more tools are available and there is significant industrial uptake. However, many challenges remain. Some of these challenges were identified in 2004, an example being understanding the relationship between coverage and test quality. In addition, new challenges have arisen. It appears that the move towards highly distributed systems, such as Cloud systems, introduces many interesting scientific and engineering challenges for the community. However, it also provides a great opportunity: these systems are extremely difficult to test in a systematic manner and if effective MBT approaches can be developed for such systems then this should further promote the industrial use of MBT.