Guest Editorial Preface

Special Issue on Model-Based Testing

Ina Schieferdecker, TU Berlin/Fraunhofer FOKUS, Germany

Colin Willcock, Nokia Siemens Networks, Germany

Dragos Truscan, Åbo Akademi University, Finland

With the ever-increasing penetration of software-intensive systems into technical, business and social areas, not only the requirements on system functionality and features, but also the requirements on system quality and reliability are increasing. With the increasing requirements, the complexity of such software-intensive systems is growing – this combined with ever shortened development times has led to major problems with the classical (non-model-based) way of developing and testing such software. In order to remain competitive, an early and continuous consideration and assurance of system quality and reliability becomes an asset of ever-increasing importance in industrial software development.

For high-quality software-intensive systems, it is known that testing takes 40-60% of the total development costs and that poorly tested software can result in product failure or even in catastrophic cases and losses of human life. Thus, it makes sense to develop and deploy new testing methods and tools to cope with the growing requirements, complexity and testing costs.

Lately, an increased popularity of model-based approaches has been observed not only in the academic world but also in industrial environments. This increased interest can be traced back to the complexity and quality issues mentioned above, but it is also driven by the improved graphical support of modelling languages, the potential for testing automation, and improved tooling. This increase in popularity is also reflected in the software and system testing field, where the relatively old paradigm of *model-based testing* (MBT) has become better enabled and supported by the technological developments. The main idea behind model-based testing is the use of abstract specifications of the system under test (SUT) and/or of its environment, which are used for checking the implementation of the system against its specification. This is done by using these specifications for automatically designing, deriving, validating, and optimizing tests.

Model-based testing refers to software testing where test cases are derived in whole or in part from a model that describes selected, often structural, functional, sometimes non-functional aspects of the system under test and/or its environment.

In model-based testing, the efforts traditionally put in eliciting test requirements and in

designing tests are lifted to the modelling level where structural and/or conceptual elements of the system and/or environmental models can be used to guide the test development process, to provide information on the quality and coverage of the designed tests, or to prioritize and select test cases with respect to the relevance of tests (for examples in relation to risk models, resource models, etc.).

Furthermore, as the models abstract from technical and technological details, the reuse of test designs across different test phases or kinds of tests becomes possible. In addition, the test designs can be used across different versions of a product, with different technological target platforms, or across different products of product lines.

Above all however, the modelling paradigm causes testers themselves to define and document their design decisions right at the beginning of a test process, which allows to reflect and to synchronize them with the design decisions from the system developers. By its very nature, modelling used for the sole purpose of test development improves the system quality: missing, inconsistent, or ambiguous system requirements are questioned by testers and improved along with the design of corresponding test requirements. The design and refinement of models used for test generation brings likewise an improvement of the system design and its specification. Errors can be detected a lot earlier which results also in reduced error correction costs.

In addition, abstraction by itself makes testers widen their view from one dedicated, singular test task to a more general approach for testing that is potentially applicable for a wide spectrum of test phases, kinds of tests, product versions, or products. And last but not least, these test designs and test specifications are at best all in sync with the system designs and system specifications.

These advantages require higher design and modelling efforts in early system development phases. However in view of reduced residual error rates, much time and resources for error correction can be saved. In this way, the advantages of model-based software development naturally translate into advantages of model-based test development.

The gain of model-based testing is primarily due to the potential for automation to increase effectiveness and efficiency in test generation. The test generation is typically guided by test selection criteria that also serve as termination criteria for testing. Test criteria are used to define a set of testing goals (also called testing targets or test requirements) that define which parts of the specification have to be covered during testing and to what extent the specification is covered by a given set of test cases. If the system specification (i.e. model) is machine-readable and formal to the extent that it has a well-defined behavioural interpretation, test cases can in principle be derived mechanically. Often, the system or environment models are translated to or interpreted as a finite state automaton or a state-transition system. In order to identify test cases, the automaton or the state-transition system is searched for executable paths. A possible execution path can be the basis for a test case – extended e.g. by timers, verdicts and defaults to cover unexpected responses. Depending on the complexity of the SUT and its model(s), the number of paths can be very large as there is often a huge amount of possible behaviours of the system. For finding appropriate test cases, i.e. paths that refer to a certain requirement to be checked, the search for these paths has to be guided. For test case selection, multiple techniques such as constraint logic programming, model checking, deductive theorem proving, symbolic execution, etc. can be applied.

As such, model-based testing is a known concept and has been investigated in research for over 20 years. However, there was a large gap to industrial reality: MBT approaches are not being used systematically and pervasively in industry yet. While the research solutions work fine for thousands of lines of code and hundreds of states, a mobile phone for example needs software with millions of lines of code

and with approximately 10,000 parameters. Practical and scalable solutions for MBT were yet to be developed.

Hence, in recent years, model-based testing has received increased attention as a solution for addressing the challenging tasks of test design and test execution automation for complex software-intensive systems. This attention is partly driven by the growing popularity of model-based specifications techniques as well as by its associated tool support. However, there are still a number of challenges to solve in order to be able to exploit the full potential of model-based testing. Such challenges include, for example, addressing the complexity of specifications by introducing abstraction levels, modelling the systems under test from different perspectives (behavioural, architectural, etc.), reusing design and development specifications for creating test models, validation of models before being used for test design, addressing changes in requirements and consequently in the models, evaluating test quality, and traceability of requirements to and from test cases.

Addressing the aforementioned challenges has been the focus of several European projects where different key players from industry and research have joined together to improve and apply model-based testing in practice, in the context of complex software-based systems. One example of such a European-level project is the ITEA2 project on "Deployment of Model-Based Technologies to Industrial Testing (D-MINT)" - http://www.d-mint.org. The goal of D-MINT was to provide partners and European industry with a new leading edge model-based testing technology that enables the production of high quality software-intensive systems at reduced expenditure in terms of time and money.

The D-MINT consortium consisted of 26 partners based in six European countries. D-MINT aimed at developing a new approach to model-based testing which combines various system aspects and system models during test generation. This integration of different system aspects and different system models becomes increasingly important as system complexity increases. It is also important for systems that are developed partially in software and partially in hardware or that are developed by different vendors with off-the-shelf components.

D-MINT set out to resolve the problems and to turn an academic discipline into an industrial reality: The typical academic model-based approach applies the same language and tools throughout and therefore makes the tool integration very straightforward. However, in industry we do not have just one specification language, not just one tool and not just one level of abstraction. The basic requirements specification may be written in DOORS, the electrical system specified in MATLAB and software in UML – completely different specification languages with different semantics and different tool chains. Hence, an MBT framework has been developed that allowed to integrate and combine the various modelling techniques and different MBT methods and tools.

As a key concept, every development within D-MINT has been driven based on industrial case studies. These case studies where used to validate the developed tools, methods and technologies and to provide empirical analysis for the results. The case studies covered several application domains, seven in total; telecommunication, automotive, control automation, video conferencing systems, industrial engineering, public lighting systems and milling machines. This extremely wide range of industrial domains in one single project is one of the unique features of D-MINT and makes the overall results obtained of special relevance. Despite operating in many different industrial domains, the results of the project were remarkable homogenous.

For each industrial case study, the D-MINT project developed an associated industrial demonstrator and a domain-specific tool chain for test generation, test refinement, distributed test process control and test quality. The demonstrators involved parallel developments using classical and model-based testing approaches for direct comparison. Analysis of the real costs involved in time and investments across all the consortium members showed that not only direct test costs could be reduced by 15%

using model-based testing, but at the same time test coverage could be improved by 10%. This translates into an overall improvement of some 20 to 25% in test costs.

In addition, the case studies indicated that adaptation efforts to model-based testing – such as initial training and integration into existing test processes – are high but are a one-off activity. They also showed that an MBT approach is particularly beneficial for testing activities with several iterations. Especially when modelling and specification techniques are part of the mainstream product development processes, the benefits in terms of reuse and shorter integration time become obvious.

This special issue is another outcome of the D-MINT project. This issue is dedicated to the publication of recent results in applying model-based testing in industrial settings. It includes three articles that have applied model-based testing to industrial case studies from different application domains.

The first article, by Grosch, presents a formal approach for tracing requirements in model-based testing in which requirements-based test cases are generated using model-checking techniques. In this approach, requirements are specified using path fragments of timed automata or timed state charts. The graphical representation of these paths is transformed automatically into temporal logic formulae, which in turn are used by a model-checking algorithm for deriving test cases. The suggested approach was applied in practice and evaluated in the automotive domain.

Streitferdt et al. present, in the second article, a practical instantiation of the model-based testing paradigm based on an industrial case study in the automotive domain. The article addresses the challenge of applying model-based testing to configurable embedded software systems in order to reduce the development efforts, while measuring the benefits (development quality, test case quality, and test effort) against traditional testing. The model-based testing support described in the article integrates existing model-based testing methods and tools such as combinatorial design and constraint processing. The main conclusion of the study was that by using model-based testing techniques the testing effort could be reduced by more than a third compared to the traditional testing process.

In the third article, Ferrari et al. present the adoption of model-based testing in conjunction with abstract interpretation techniques for the verification of railway signalling systems. The focus of the article is on evaluating, against two industrial case studies, the benefits of moving the testing process from a code-centric to a model-centric one. The conclusions of the study show that the model-based testing process became more efficient after several iterations. In addition, it resulted not only in a significant reduction in verification costs, but also in more efficient bug detection and correction capabilities.

Only one of the above articles was done directly in the context of the D-MINT project. However, the conclusions of each article confirm the results of the D-MINT project, namely that model-based testing is an effective technique which, with the necessary adaptation, can provide industry with real improvements over the traditional testing processes.

In the end, we would like to warmly thank all the contributors to this special issue and to gratefully acknowledge the reviewers for their effort in providing valuable comments and suggestions for improving the quality of the included articles. We would also like to acknowledge IJERTCS's Editor-in-Chief Seppo Virtanen for assisting us in producing and publishing this special issue.

Ina Schieferdecker
Colin Willcock
Dragos Truscan
Guest Editors
IJERTCS

Ina Schieferdecker is Professor on Engineering and Testing of Communication-Based Systems at Technical University Berlin and is heading the Competence Center Modelling and Testing for System and Service Solutions (MOTION) at the Fraunhofer Institute on Open Communication Systems (FOKUS), Berlin. She studied Mathematical Computer Science at Humboldt-University Berlin and did her PhD at Technical University Berlin on performance-extended specifications and analysis of quality-of-service characteristics. Prof. Schieferdecker works in the area of design, modelling, and testing of software-intensive, distributed and/or embedded systems using specification-based techniques like UTP (UML Testing Profile) and TTCN-3 (Testing and Test Control Notation). She is active member of ETSI MTS, of the German Testing Board and was elected to be a member of the German Academy of Technical Sciences (acatech).

Colin Willcock is currently manager for 3GPP Radio Access Network Standardisation at Nokia Siemens Networks. He received a BSc from Sheffield University in 1986, an MSc from Edinburgh University in 1987, and a PhD in parallel computation from the University of Kent in 1992. Colin was part of the core ETSI team that developed the TTCN-3 language and spent many years leading and participating in the TTCN-3 language maintenance. In the past, he has worked on numerous standardization efforts at ETSI, ITU-T, and 3GPP, focusing on various aspects of formal specification languages. He was the project leader for the European TT-medal project, which strove to improve test methodology and languages for software-intensive systems and also lead the D-MINT project, which aimed to improve test methodology and languages for software-intensive systems and explored the use of model-based testing in an industrial context.

Dragos Truscan is currently a postdoctoral researcher with the Software Engineering Laboratory, at Åbo Akademi University, Finland. He obtained his doctoral degree from the same university in 2007, on the topic of applying model-based development principles to the design of applications targeted to custom processor architectures. Since then he has been interested in research topics related to model-based testing, testing processes, model-based development of software-intensive systems, and cloud computing.