Int J Softw Tools Technol Transfer (2012) 14:383–386
DOI 10.1007/s10009-012-0240-3

INTRODUCTION

Model-based testing of software and systems: recent advances and challenges

Alexandre Petrenko · Adenilso Simao · José Carlos Maldonado

Published online: 22 June 2012

Abstract Model-based testing is focused on testing techniques which rely on the use of models. The diversity of systems and software to be tested implies the need for research on a variety of models and methods for test automation. We briefly review this research area and introduce several papers selected from the 22nd International Conference on Testing Software and Systems (ICTSS).

Keywords Software testing · Model-based testing

1 Introduction

The importance of testing in the development of various software and systems has been growing steadily with the increasingly crucial role they play in the modern society. Even though testing is a familiar and unavoidable practical activity, the ever growing variety of software-based systems poses big challenges for the development teams. It is not surprising that testing has evolved during the last decades from an ad hoc and under-exposed area of systems development to an important and active research area.

The quest for high quality systems and for a rapid development cycle has demanded testing techniques which are both rigorous and automatic. Model-based testing (MBT) offers techniques which meet these requirements.

A. Petrenko
Centre de recherche informatique de Montreal (CRIM), Montreal, Quebec, Canada
e-mail: petrenko@crim.ca

A. Simao (✉) · J. C. Maldonado
São Paulo University, São Carlos, São Paulo, Brazil
e-mail: adenilso@icmc.usp.br

J. C. Maldonado
e-mail: jcmaldon@icmc.usp.br

In this Special Section, we first provide a brief overview of MBT main techniques and outline several research challenges. We highlight some possible future work on MBT, as an invitation for further contributions in this area. Second, we introduce the papers of this Special Section which were selected from the 22nd International Conference on Testing Software and Systems (ICTSS), held in Natal, Brazil, in 2010.

2 Model-based testing

Model-based testing is an approach that relies on a formal model built to support the testing activity. This approach can offer a number of benefits, such as high fault detection ratio, reduced cost and time, traceability, and ease of handling requirements evolution [11,30].

As models play a crucial role in MBT, a major testing challenge moves to the modeling the SUT itself. The use of models for various purposes, from capturing the requirements, model checking of properties of designed systems to testing existing systems, has a long history in computer science and engineering. However, it seems that it is only recently that test models have been combined with test purposes and properties for test generation.

As Lund et al. [21] justly observe, there are numerous languages for modeling computer systems; the semantics for the languages ranges from English explanations of modeling language constructs to highly formal mathematical or logical definitions. A number of issues need to be taken into consideration when choosing a language and accompanying semantics. For instance, it is important to know who will use the models and what level of training they would have. As stated by Lund et al. [21], the language needs to have a notation that is understandable by the users of the models, at least at the intuitive level. In this context, formal and semi-formal models with intuitive notation and semantics may be useful, since mathematical and logical formulas may be well understood by computer scientists and some developers, but will be incomprehensible for most practitioners.

The system modeling for testing has been facing a number of challenges. First, test models should indeed match the level of mastering formal approaches by the existing testing personnel. Then, models should be related to the test requirements or at least provide some means to trace features of test models to the requirements. Models should have formal semantics to allow for test automation and to ensure soundness of the produced verdicts. On one hand, a test model has to capture the relevant characteristics of the system, so that the tests derived from it are meaningful. On the other hand, the model should not be too detailed, since otherwise the testing activity would be unfeasible due to the complexity of handling all the details. Finding a compromise between these two conflicting requirements is an engineering task. To ease this task, different modeling techniques have been investigated.

Two main types of models are currently considered in the testing context, holistic models and scenario models. The former are models which attempt to describe the expected behavior of a system under test (SUT) as completely as possible in a single formal specification, while the latter focus on a number of test scenarios describing various aspects of the expected behavior.

Holistic models describe the behavior of the SUT with some level of detail, trying to capture the features that are relevant to testing. They can be divided into sequential and true-concurrency models.

Sequential models include finite state machines (FSMs), input/output transition systems (IOTSs), and various, e.g., UML, state machines. Testing with classical deterministic Mealy machines is a topic which was initiated more than 50 years ago [22]. The family of FSMs currently used for testing includes machines that are not necessarily completely specified and deterministic and extended (with data) FSMs. The main research challenges are related to nondeterminism, fault models, and executability of EFSM tests. IOTSs differ from FSMs in dropping coupling of inputs with outputs resulting in "unstable" states reached by inputs from which outputs could be produced. If no input is enabled in unstable states, then the two models, FSM and IOTS, have the same input–output traces, though the former needs fewer states than the latter. Otherwise, if inputs are also enabled in unstable states, the IOTS has traces no FSM can have. This feature along with the use of nonobservable transitions makes the IOTS model more suitable for specifying certain types of nondeterminism and concurrency than the FSM model.

While sequential models do not distinguish nondeterminism and concurrency, as they model the latter by interleavings, true-concurrency models, such as Petri Nets, see, e.g., [2,18] and partial order input/output automata [12], avoid interleaved representation of concurrent events achieving better compactness.

In spite of this disadvantage, sequential models currently offer more mature results to the practice of MBT.

Scenarios are usually described using message sequence charts (MSC) and UML sequence diagrams, including a variety of timed scenario notations, see, e.g., [13]. One of the main challenges in using scenarios for testing is to ensure that the chosen test scenarios are consistent and sufficiently formal to allow executable tester's generation.

Another type of scenario is test purposes and properties, which are used to force the test generation process to consider, instead of a complete test model, only a part of it which is related to a given test purpose. This allows one to reduce the number of tests generated from the model.

Once a test model of the system's behavior is available, it is then used to obtain a set of tests. Simplistic, but nonetheless sometimes useful, approaches just "walk" through the model according to some algorithm, collecting the events along the way which are used to build tests. For instance, tests can be obtained from a state-based model by navigating from state to state, according to some predefined schema, and registering the traversed path. One of the main challenges here is the chosen path executability, especially when the model uses variables and predicates, as mentioned above. More elaborated approaches use coverage criteria to guide the test case generation. A coverage criterion says when certain elements of the model are "covered" by the test cases; the goal is then to cover all such elements, see, e.g., [25].

The existing approaches for model-based test generation could be grouped according to a criterion used to terminate the test generation process. Among the mostly investigated criteria are the model coverage criteria, which include the use of test purposes. Another type of criteria is related to fault models. The main motivation for using fault models is to have tests which can detect, i.e., cover, certain types of implementation faults. Such tests offer a "guarantee" for the test quality in terms of fault coverage as opposed to model coverage. Fault coverage criteria are in fact also used in mutation-based approaches, which construct a test suite by choosing tests killing a selected number of mutants.

Mutants can be derived from test models with some fault models in mind. In case of holistic models, fault models include output, predicate, and transition faults. In case of test scenarios, if both negative and expected behaviors are described, then the former model, in fact, represents certain implementation faults which should be detected by tests.

Fault modeling approaches rely on conformance relations to discriminate between the expected and erroneous behaviors. These relations are also needed in model coverage approaches to design the verdict mechanism of the tester. One of the main challenges in choosing an appropriate conformance relation is to ensure that wrapping an SUT by a test harness or test adapter within a given test architecture does not violate the soundness of tests generated for the chosen relation.

There are ongoing discussions on distributed and asynchronous testing using not only FSM models, but also the IOTS model along with the ioco-like conformance relations [16,17,26,27,29]. These discussions are important, since, coincidentally or not, the existing tools, including Spec Explorer [31], TGV [19], UPPAAL/Tron [20] and TorX [28], rely on different assumptions about inputs and outputs, making some events synchronous (which could thus be blocked) and others asynchronous. Apparently, more research is needed to better address the problems with various test architectures.

A long-term challenge in test generation is related to the data aspect of test models. This is a continuous line of research where various approaches for treating symbolic models and tests are tried [3,4,8,9,23].

Other stringent problems should be dealt with when testing real-time systems, i.e., systems whose correctness can only be assessed if time instances when events occur are also taken into account. The test generation for real-time systems becomes more challenging [1,6,14]. The generation algorithm should determine not only which input to send to (and which output to receive from) the SUT, but also when an input should be sent and when an output should be received. One of the biggest issues is the inherent nondeterminism of real-time systems; see, e.g., [5]. Testing is somewhat simplified if nondeterminism in outputs is forbidden in both, the specification and the SUT [15], e.g., assuming the so-called isolated outputs in UPPAAL/Tron [20].

Another important factor that constrains the algorithm used for test generation is the test harness available for executing tests. Test execution can be online or off-line. In off-line execution, the test cases are generated from the model *a priori*. Afterward, the SUT is executed with the concretized test cases. In online execution, the test cases are generated *on-the-fly* while executing the SUT. While the off-line execution schema is usually simpler to implement, online execution is more suitable to deal with nondeterminism.

3 The 22nd International Conference on Testing Software and Systems

The 22nd ICTSS, held in Natal, Brazil, in November 2010, is the merging of two traditional and important events in the model-based testing area, which has served the testing community as an important venue for discussing advancements in the area.

The objective of ICTSS 2010 was to be a forum for researchers from academia as well as from industry, developers, and testers to present, discuss, and learn about new approaches, theories, methods, and tools in the field of testing of software and systems.

The authors of four papers published in the ICTSS proceedings were invited to submit extended versions of their papers for this special issue. The first two papers discuss how formal models and properties of tests can be enhanced; the other two papers address the problem of test generation; one proposes an approach for test case generation using genetic algorithms, whereas the other proposes an approach based on solving a first-order logic. In summary, the chosen four papers represent a wide range of active research topics in testing of software and systems.

The first paper [32] by Margus Veanes and Nikolaj Bjørner, "Alternating Simulation and IOCO", proposes a symbolic framework called *Guarded Labeled Assignment Systems* (GLAS) and discusses how it can be used for analysis of formal specification languages. The analysis of symbolic models is becoming a realistic approach, mainly due to advances in satisfiability modulo theories technology.

The second paper [7] by Yliès Falcone, Jean-Claude Fernandez, Thierry Jéron, Hervé Marchand and Laurent Mounier, "More testable properties", discusses the testability of various properties w.r.t. to several relations between the SUT and the specification. A property is testable if it is possible to establish that it holds for a given SUT.

The third paper [10] by Christoph D. Gladisch, "Model Generation for Quantified Formulas with Application to Test Data Generation", proposes a model generation algorithm, which solves first-order logic formulae with quantifiers. The model can be converted into a test preamble for state initialization.

Finally, the fourth paper [24] by Aurora Pozo, João Carlos Garcia Árias, Rafael da Veiga Cabral, Silvia Regina Vergilio, and Tiago Nobre, "Multi-objective Optimization Algorithms Applied to the Class Integration and Test Order Problem", describes an algorithm for integration test, in which the best order for testing the classes of an object-oriented software is sought. The main criteria used to determine which test order is better are the effort required to develop test drivers and test stubs. The algorithm is based on multi-objective optimization techniques, such as Pareto ant colony, multi-objective Tabu search and nondominated sorting.

In conclusion, the presented techniques have a solid formal background and robust tool supporting. Together, they indicate that research on model-based testing has been reaching the level of maturity required for transferring theoretical results to industry.

References

1. Andrade, W.L., Machado, P.D.L., Jeron, T., Marchand, H.: Abstracting time and data for conformance Testing of teal-time systems. In: A-MOST 2011, 9–17 (2011)
2. Bochmann, G.v., Jourdan, G.-V.: Testing k-safe petri nets. In: TestCom/FATES 09, LNCS, vol. 5826, pp. 33–48, Springer, Berlin (2009)
3. Clarke, D., Jéron, T., Rusu, V., Zinovieva, E., Katoen, J.-P., Stevens, P.: STG: A symbolic test generation tool. ETAPS 2002 and TACAS 2002, LNCS, vol. 2280, pp. 151–173. Springer, Heidelberg (2002)
4. Constant, C., Jéron, T., Marchand, H., Rusu, V.: Integrating formal verification and conformance testing for reactive systems. IEEE Trans. Softw. Eng. **33**(8), 558–574 (2007)
5. El-Fakih, K., Yevtushenko, N., Fouchal, H.: Testing timed finite state machines with guaranteed fault coverage. In: TestCom 2009, LNCS, vol. 5826, pp. 66–80. Springer, Eindhoven (2009)
6. En-Nouaary A.: A scalable method for testing real-time systems. Softw. Qual. J. **16**(1), 3–22 (2008)
7. Falcone, Y., Fernandez, J.-C., Jéron, T., Marchand, H., Mounier, L.: More testable properties, in this volume
8. Frantzen, L., Tretmans, J., Willemse, T.: A Symbolic Framework for Model-Based Testing. FATES 2006 and RV 2006. LNCS, vol. 4262, pp. 40–54. Springer, Heidelberg (2006)
9. Gaston, C., Le Gall, P., Rapin, N., Touil, A.: Symbolic execution techniques for Test Purpose Definition. In: TESTCOM 2006, LNCS, vol. 3964, pp. 1–18. Springer, Heidelberg (2006)
10. Gladisch, C.: Model Generation for Quantified Formulas with Application to Test Data Generation, in this volume
11. Grieskamp, W., Kicillof, N., Stobie, K., Braberman, V.: Model-based quality assurance of protocol documentation: tools and methodology. Softw. Test. Verification Reliab. **21**(1), 55–71 (2011)
12. Haar, S., Jard, C., Jourdan, G.-V.: Testing input/output partial order automata. In: TestCom 2007, LNCS 4581, pp. 171–185, Springer, Berlin (2007)
13. Hassine, J., Rilling, J., Dssouli, R.: An evaluation of timed scenario notations. J. Syst. Softw. **83**(2), 326–350 (2010)
14. Hessel, A., Larsen, K.G., Mikucionis, M., Nielsen, B., Pettersson, P., Skou, A.: Testing Real-Time Systems Using UPPAAL. In: FORTEST 2008. LNCS, vol. 4949, pp. 77–117. Springer, Heidelberg (2008)
15. Hessel, A., Larsen, K.G., Nielsen, B., Pettersson, P., Skou., A.: Time-optimal real-time test case generation using UPPAAL. In: FATES'03, LNCS, vol. 2931, pp. 136–151. Springer, Heidelberg (2003)
16. Hierons, R.M.: Controllable testing from Nondeterministic Finite State Machines with Multiple Ports. IEEE Trans. Comput. **60**(12), 1818–1822 (2011)
17. Huo J., Petrenko A. (2009) Transition covering tests for systems with queues. Softw. Test. Verification Reliability, 19(1):55–83
18. Jard, C.: Synthesis of distributed testers from true-concurrency models of reactive systems. Int. J. Inform. Softw. Technol. **45**(12), 805–814 (2003)
19. Jard, C., Jéron, T.: TGV: theory, principles and algorithms. Softw. Tools Technol. Transf. **7**(4), 297–315 (2005)
20. Larsen, K., Mikucionis, M., Nielsen, B., Skou, A.: Testing Teal-time embedded software using UPPAAL-TRON: an industrial case study. In: 5th ACM international conference on Embedded software, pp. 299–306. ACM Press, NY (2005)
21. Lund, M.S., Refsdal, A., Stølen, K.: Semantics of UML Models for Dynamic Behavior: A survey of different approaches. In: Model-Based Engineering of Embedded Real-Time Systems, LNCS, vol. 6100, pp. 77–103. Springer, Berlin (2011)
22. Moore, E.F.: Gedanken-experiments on sequential machines. Automata Studies. vol. 34, 129–153. Princeton University Press, Princeton (1956)
23. Petrenko, A., Boroday, S., Groz, R.: Confirming configurations in EFSM testing. IEEE Trans. Softw. Eng. **30**(1), 29–42 (2004)
24. Pozo, A., Árias, J.C.G., Cabral, R.V., Vergilio, S.R. and Nobre, T.: Multi-objective optimization algorithms applied to the class integration and test order problem, in this volume
25. Simao, A., Petrenko, A., Maldonado, J.C.: Comparing finite state machine test coverage criteria. IET Softw. **3**(2), 91–105 (2009)
26. Simao, A., Petrenko, A.: Generating asynchronous test cases from test purposes. Inform. Softw. Technol. **53**(11), 1252–1262 (2011)
27. Tretmans, J.: Test generation with inputs, outputs and repetitive quiescence. Softw. Concept Tools. **17**(3), 103–120 (1996)
28. Tretmans, J., Brinksma, E.: TorX: Automated model-based testing. In: First European Conference on Model-Driven Software Engineering, pp. 31–43 (2003)
29. Tretmans, J.: Model based testing with labelled transition systems. In: Formal Methods and Testing, LNCS, vol. 4949, pp. 1–38. Springer, Berlin, (2008)
30. Utting, M., Legeard, B.: Practical model-based testing: a tools approach. Morgan Kaufmann, San Francisco (2007)
31. Veanes, M., Campbell, C., Grieskamp, W., Schulte, W., Tillmann, N., Nachmanson L.: Model-based testing of object-oriented reactive systems with Spec Explore. In: Formal Methods and Testing, LNCS, vol. 4949, pp. 39–76. Springer, Berlin, (2008)
32. Veanes, M., Bjørner, N.: Alternating Simulation and IOCO, in this volume.