Introducing Model-Based Testing Techniques in Industrial Projects

Andreas Ulrich

Corporate Research & Technologies CT SE 1
Siemens AG
Otto-Hahn-Ring 6
81730 München / Germany
andreas.ulrich@siemens.com

Abstract: The paper is an experience report on the introduction of model-based testing (MBT) approaches in industrial projects. It reflects the author's subjective view on that matter that he gained in various software development projects at Siemens. Existing MBT approaches are classified and evaluated according to their importance in recent projects. In addition the necessary requirements and efforts to successfully introduce MBT into existing projects are discussed.

1 Introduction

In recent time, model-based testing (MBT) gained importance in connection with upcoming concepts of test-driven development and the model-driven architecture (MDA) initiative of the OMG. Other advances in software engineering, like the use of executable specification languages, the pattern-based detection of faults in source code, or the inference of program behavior from runtime observations, contribute to a renaissance of MBT approaches as well. This trend can be also observed by a recent increase in the number of research workshops in this area. Though, the roots of MBT go back to the early beginnings of computer science [Moo56].

In software testing practice, MBT approaches evolved as the latest innovation step as depicted in Figure 1. To apply these approaches successfully in today's industrial software development projects, one has to put efforts to automate the execution of tests in the first place. Today, a full range of test automation solutions of varying abstraction levels exists depending on the actual application domain and the project history. However, with an improved applicability of model-driven approaches due to the provision of better tools and supporting development processes, a high impact of MBT approaches on software testing can be expected now and in the near future.

The rest of the paper is organized as follows. The next section classifies existing MBT techniques in order to allow a better understanding of the underlying technologies. Section 3 discusses the items necessary to successfully introduce MBT in industrial projects. Section 4 continues with a short discussion on promising application domains for MBT and MBT tools in general. Eventually Section 5 concludes the paper.

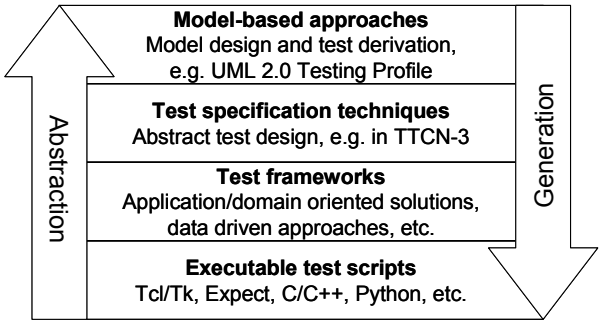

Figure 1: Evolution of testing approaches.

2 Classification of MBT Techniques

The notion of MBT is sometimes extended to cover approaches that merely use a graphical language to specify test cases, e.g. UML sequence diagrams. In this paper, MBT is defined as an approach to derive executable tests from a given formal model of the system under test (SUT) using a variety of test selection criteria. A model is an abstract view on the system and specifies typically (parts of) the behavior of the system in terms of its dynamics (control flow, data flow). Three different types of models typically exist (Figure 2):

1. *Test models* that reflect the requirements on the system by modeling the behavior of a potential user (tester),
2. *Design models* that specify the intended behavior of the system as it is implemented,
3. Models constructed from the *source code* directly.

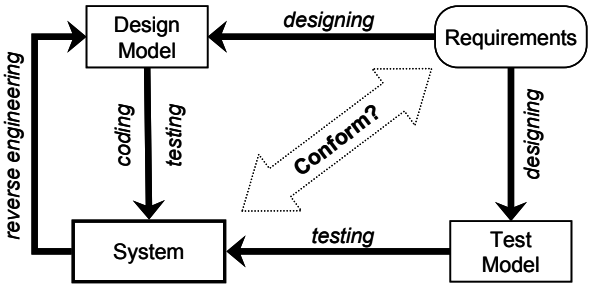

Figure 2: The design and test process.

The test derivation approaches applied on the different model types are typically classified into specification-based testing (or black-box testing) if they are based on requirement models and program-based testing (or white-box testing) if the source code is used as the underlying model. Here a different classification scheme is applied that refers to the used test selection criteria: (a) coverage-based, (b) fault-based, and (c) mutation-based approaches (Table 1).

	Coverage based	Fault based	Mutation based
Test models	X		
Design models	X	X	X
Source code models	X		X

Table 1: MBT approaches applied to different model types.

Coverage-based approaches interpret the underlying model as a (directed) graph structure with a set of vertices and a set of edges. A large number of syntactical coverage criteria are known, e.g. a test case shall cover all vertices in the graph or all paths through a digraph. The usefulness of them in terms of the error detection power has been underpinned by empirical studies (only). Besides the impractical all-path criterion, no "best" coverage criterion exists. Because of their abstract interpretation, coverage-based test derivation approaches are applicable for any types of models mentioned above.

Fault-based approaches attempt to derive tests according to a given fault model that is imposed on the design model of the system. Known approaches are the W method and UIO method and their descendants and variants [Pet00]. It is possible to derive a finite set of exhaustive tests of a given fault model and additional assumptions on the design model and the SUT. That means these tests are able to detect any error in the SUT if its error class has been specified in the fault model. Though fault-based test methods are very powerful and have a long history in research, they are restricted to only few application domains, e.g. communication protocol testing. Consequently these techniques are rarely applied in industrial projects.

Last but not least, mutation-based testing approaches deploy the fact that a test with a high error detection capability should discriminate a large number of mutated models (i.e. erroneous models) from the correct model [OfUn00]. Here a mutant is a syntactic modification of the correct model, e.g. changed or added vertices in the digraph. Because the number of mutants for a given model is typically infinite, the selection of tests with a high error detection capability depends mainly on the extent of generated mutants.

From the perspective of their current practical usage, MBT approaches based on test models and appropriate coverage criteria are dominant. Besides the fact that they can be easily implemented using graphical specification techniques like UML and the UML 2.0 Testing Profile in particular, these approaches benefit from methodologies for test-driven development that are currently *en vogue*. They also circumvent the test oracle problem associated to the test derivation from design models. If production code and test cases are generated automatically from the same model, it might be difficult to argue about the purpose of the derived tests. In the long run though, advances in fault-based testing approaches are expected because they produce tests of defined high quality, but are hampered currently by the non-availability of appropriate design models for most application domains. Mutation-based testing is expected to play a role in some niche applications.

3 Preparing for MBT in Industrial Projects

In depth knowledge about the best practices in MBT is a key prerequisite to reach excellence in MBT. Basically, technologies supporting MBT have at least two characteristics:

they are based on (or assume the presence of) some type of model and they implement (or support) a process for test case derivation and test result evaluation.

There are various modeling technologies that can be used for MBT. Examples are finite state machines (FSMs), tabular condition/action models, Petri nets, Markov chains, UML models and hybrids. The orientation towards an industrial context imposes some extra requirements and constraints.

First of all, it is not wise to assume that a formal (complete and consistent) model of the software system always exists. What is reasonable to assume instead is an informal model that describes the system (or parts of it) mostly in natural language. Therefore, domain understanding and modeling play the mayor role for a successful application of MBT in a typical industrial context. Semi-formal models, such as UML, are becoming increasingly common in industrial projects contributing to the reduction in MBT modeling efforts. However, the development of a domain-specific language (DSL) that is adequate for modeling the requirements of a given application domain remains a crucial factor that decides about the success of MBT.

Second, MBT can only be effective in industrial projects when the total efforts for applying a MBT approach are affordable. This means that the testing approach should produce readily executable tests, even if the software system is modeled only partially. The model should also have some level of resilience to modifications (changing requirements, product evolution). Another important aspect is that the level of abstraction in the model must be adequate for describing the intended test purposes concisely. This implies that the different abstraction levels must be bridged in order to obtain executable tests (cf. Figure 1) [PrPh05].

It is seldom possible in practice to perform comparative studies between conventional test approaches as used in previous projects and new, advanced MBT approaches because of a permanent pressure on the project efforts (cost, time, and people). As a result, the decision for or against the introduction of an MBT approach into a software development project must be done based on known experiences in similar fields and also often depends on the openness and willingness of the people involved in the decision-making. Good knowledge of best practices in MBT is therefore indispensable to promote MBT in practice successfully.

As explained in Section 2, test derivation approaches can be classified into different categories (cf. Table 1). An investigation of best practices involves a more detailed assessment of test derivation approaches and their applicability to the diverse models in MBT. Issues to review include:

- Which modeling standards and formalisms are appropriate for applying MBT approaches in different application domains? What are the difficulties on introducing and using these modeling standards? Are they applicable without or with modifications? Are even new modeling approaches required?
- Are the currently known test derivation approaches appropriate for a given model? Is there a systematic, logical basis for determining which one is more appropriated for a certain application domain? Is it possible to produce meaningful metrics for test cov-

erage and reliability growth? Would the investigation into a new test derivation approach for a particular application domain be justified?
- MBT approaches are typically applied to functional testing. How could the approaches be extended to cope with non-functional tests, in particular load and stress tests over a prolonged time?

Further issues that need to be addressed when considering MBT are:

- Automation on test set-up, test execution, test result verification, and traceability
  - How could the set-up and initialization of the test infrastructure be automated most efficiently? How can test cases be managed to provide support for regression tests?
  - What are efficient ways to bridge the abstraction contended in models to executable test cases? How to create test drivers cost-effectively?
  - What are the efficient ways to support automated test result verification? Can test oracles be generated from models or should they be specified separately?
  - Is it possible to have meaningful traceability between the model and test execution results? How to trace the requirements from the models to the test cases?
- Model modification: How to handle the rework resulting from changes in the requirements and/or interfaces of the SUT? How to manage versions of models to support model-based regression testing?
- Training: How can a test team be supported best to achieve modeling skills for MBT? What can be done to leverage the introduction of MBT approaches?

4 Application Domains and Tools for Model-Based Testing

In addition to investigating the current practices in industrial projects, it is important to identify application domains where MBT approaches can be applied easily or applied. Of course, application domains of systems that can be described naturally by (state-based) models are the first candidates for MBT. An application domain lends itself to a specific model type and consequently to a particular MBT approach. Some intuitive observations on the usefulness of models in different application domains can be made. For example, many systems are reactive and exhibit continuous dynamics. These systems are best modeled in terms of communicating FSMs. Examples of FSM based systems can be found in nearly all industries, e.g. call processing units in telephone switches, automotive power train control systems, Web-based hospital information systems, and others.

Besides the identification of the right application domain, it must be possible to integrate a chosen MBT approach into an existing software development process that is deployed by the organization, in which the project is performed. In particular, *software product line development* deserves special attention. This topic is of increasing interest.

A recent report [UtLe07] lists the available main tools in MBT in both the academic and industry. It also distinguishes between test generators and model-based input generators, which do not output the expected response of the SUT. Furthermore, it discusses a distinction between test generators and test automation frameworks. A test automation framework accepts manually created or pre-recorded test sequences and runs them with-

out human supervision. Typical examples of such tools are HP Mercury's WinRunner or a TTCN-3 based test system. On the other hand, a test generator actually derives tests automatically from a given abstract model, e.g. as in Conformiq's Qtronic test generator.

An evaluation of existing MBT tools shall be always performed before deciding on the development of an own tool. Many times the available support for commercial tools outweighs the risk associated to the proprietary development of similar tools. It can be also observed that more and more commercial MBT tools enter the market. Nevertheless, a proprietary MBT tool development within Siemens happened so far with the TDE/UML tool [DHV06].

The overall goal of the home-grown TDE/UML tool is to generate functional test cases based on UML use cases, corresponding activity diagrams, and additional test annotations. TDE/UML implements a test generation process that combines model coverage with data coverage. Model coverage is achieved through different coverage criterions for the activity diagrams detailing a given use case. Data coverage is achieved by means of special annotations in the model that specify activities of the underlying category/partition method [OsBa88].

5 Conclusions

There is a promising future for MBT as software becomes even more ubiquitous and quality becomes the only distinguishing factor between brands. When all vendors provide software with a similar set of features and have the same shipping schedules, the only reason to buy one product over another is its quality. MBT, of course, cannot and will not guarantee or even assure quality. However, by its very nature, it leads to thinking thoroughly of use cases and test scenarios well in advance of product development while still allowing for the addition of new insights, which makes it an ideal choice for testers concerned about completeness and effectiveness of their tests.

References

[DHV06] K. Didrich, St. Herbst, M. Vieira: *Applying Model-based Testing to a Train Control System*; in: Proc. of the Workshop Modellbasiertes Testen, 36. GI-Jahrestagung Informatik 2006, Dresden, Germany, October 2006.

[Moo56] E. F. Moore: *Gedanken-Experiments on Sequential Machines*, in: Automata Studies, Princeton University Press, Princeton, New Jersey (1956), pp. 129–153.

[OfUn00] J. Offutt, R. H. Untch: *Mutation 2000: Uniting the Orthogonal*; in: Mutation Testing in the Twentieth and the Twenty First Centuries, pp. 45–55, San Jose, CA, October 2000.

[OsBa88] T. J. Ostrand, M. J. Balcer: *The Category Partition Method for Specifying and Generating Functional Tests*; ACM Comm. 31 (6), pp. 676–686, June 1988.

[Pet00] A. Petrenko: *Fault Model Driven Test Derivation from Finite State Models: Annotated Bibliography*; in: Proc. of the 4th Summer School on Modeling and Verification of Parallel Processes (MOVEP2000), Springer LNCS 2067, 2000.

[PrPh05] A. Pretschner, J. Philipps: *Methodological Issues in Model-Based Testing*, in: LNCS 3472, Model-Based Testing of Reactive Systems, Springer Verlag, pp. 281–291, 2005.

[UtLe07] M. Utting, B. Legeard: *Practical Model-Based Testing. A Tools Approach*, Elsevier Books, Oxford, 2007.