Introducing a Framework and Methodological Guidance for Model Based Testing

M.George[1,4], K.P.Fischer-Hellmann[1,2], M.Knahl[3], U.Bleimann[1] and S.Atkinson[4]

[1]Aida Institute of Applied Informatics Darmstadt, Darmstadt, Germany
[2]Digamma Communications Consulting GmbH, Mühltal, Germany
[3]Furtwangen University of Applied Sciences, Furtwangen, Germany
[4]Centre for Security, Communications, and Network Research,
University of Plymouth, Plymouth, United Kingdom
e-mail:george.mathew@hotmail.com

Abstract

The growing complexities of software and the demand for shorter time-to-market are two important challenges that face today's IT industry. These challenges demand the increase of both productivity and quality of software. Early integration of test development into the system life cycle becomes more and more important, which helps the developer to find the errors in an early stage of design and development. A promising technique for the early integration of test development in system life cycle is model based testing that automatically generates test artefacts from design models. A major challenge of model based testing is the lack of methodological guidance, approaches, and a framework. My research aims to define a methodological guidance and a framework that facilitates model based testing which ranges from requirement specification, modelling guide lines, test generation from models, test management, and traceability of requirements to models and generated tests. In this paper, we define and justify the research problem and present a research approach.

Keywords

Model based testing, UML, Model, Test oracles, Traceability

1. Introduction

Today's software testing methodologies and techniques have to meet the growing complexities of the software and shorter time-to-market. Testing is an essential activity in software engineering. In the simplest terms, it amounts to observing the execution of a software system to validate whether it behaves as intended and identify potential malfunctions (Bertolino, 2007). Today's software testing ranges from unit test to software acceptance tests, but all these tests are confined in the later stages of development. Major problem of this approach is that testing is not done at the early stages of development and defects in the design and specification detected in the later stages of software development process are expensive to fix.

Automation of software development and software testing on the basis of executable models and simulation promises significant reductions in fault-removal cost and development time. (Utting et al. 2006) defines Model based testing as a variant of testing that relies on explicit behaviour models that encode the intended behaviour of

a system and possibly the behaviour of its environment. According to Bertolino (Bertolino, 2007), the major goal of model based testing (MBT) is automatic generation of test artefacts from models. Model Based Testing offers considerable promise in reducing the cost of test generation, increasing the effectiveness of the tests, and shortening the testing effort. (Dalal et al. 1999)

While using the model based testing in practical software life cycle, the main problem arise is the lack of a methodological guidance and a process for defining the requirements, developing models, generating tests, tracing requirements to models and the test cases generated from models, test selection criteria (example, transition coverage) and test management(defining priorities and number of execution of each tests, defining priorities of requirements, "etc..") that suites for model based testing. Our research will focus to develop and define a framework for the above.

This paper is organized into an overview of Model Based Testing, its challenges, problem description of the research work, problem justification and an overview of the research approach.

2. Model Based Testing and Challenges

Fundamental approach of Model Based Testing is to effectively use models defined in software development to drive the testing process, in particular to automatically generate test cases and test suite and enable the tests at the early stages of development. There are different definitions for Model Based Testing. Mark Utting (Utting et al. 2006) defines it as the "generation of test cases with test oracles from behavioural model". Since Models describes the behaviour of the system, tests can be executed based on the model for quality software. Bertolino refers it as the test case derivation from a model representing software behaviour. (Bertolino et al 2005)

Model-based testing typically involves the following steps: (Utting et al. 2007, Legeard, 2010)

- Building an abstract model of the behaviour of the system under test. The model captures a subset of the system requirements.
- Definition of test selection criteria. The criteria define what test cases to be generated from the model.
- Generating abstract tests from the model, using the defined test selection criteria. At this stage, the generated test cases are expressed in terms of the abstractions used by the model.
- Transforming (concretize) the abstract test cases into executable test cases.
- Executing the test cases. At execution time, an adaptor component transforms the output of the system to the abstraction of the model.
- Assigning of a pass/fail verdict to executed test case.
- Analyzing the execution result.

Model based testing significantly reduces testing time (Bringmann et al. 2008). According to Legard (Legeard, 2010), Model Based Testing is an important and

useful technique that brings significant progress over the state of the practice for functional software testing effectiveness, increasing productivity, and improving functional coverage. Following section describes various challenges of model based testing.

2.1. Modelling

A great deal of research focuses on automatic test generation from models Since Model Driven Development is emerging as the de-facto standard in software development (Frankel, 2003), research has to focus on building ideal models for quality software. (Bertolino, 2007).

Bertolino outlines "Test Based Modelling" as a dream in the software testing process. Instead of taking a model and see how well it is exploited for testing, it is better to consider how one should ideally build a model so that the software can be effectively tested (Bertolino, 2007). Today's software industry demands time to market and a high quality for the delivered software. This demands a growing need of research in Model Based testing as well with an extra focus in building models. Since models serve as the back bone of Model Based Testing, we have to develop concepts and methodologies that guide the model development.

2.2. Test Oracles in Model Based Testing

The functionality of the system is tested by comparing the output of the system for a predefined input, using a test oracle. The purpose of the test oracle is to give a test verdict ("Pass", "Fail", "Inconclusive", "etc."). A test oracle is a mechanism that determines the expected output for a given input and compares it with the actual output. The precision and efficiency of oracles greatly affects testing cost/effectiveness. Efficient test oracles are also a major factor for efficient test automation. An important component of testing is the oracle. Indeed, a test is meaningful only if it is possible to decide about its outcome (Bertolino, 2003). So an approach is needed for realizing and automating test oracles from models, an approach to generate test oracles automatically from models and requirements. Generated test oracles will be later used in the test execution phase to determine whether the test has passed or failed.

2.3. Traceability of Requirements in Model Based Testing

The automation of bidirectional traceability between requirements and test cases is a key aspect of MBT (Legeard, 2010). Bidirectional traceability is the ability to trace links between two parts of the software development process with respect to each other (Legeard, 2010). Dalal et al. suggests that defects of a model can be minimized by ensuring the traceability from requirements to the part of models (Dalal et al. 1999). Traceability of requirements is an integral part of Model Based Testing and provides effective verification to determine the coverage of requirements with respect to a model or models. Study has to be focused on how traceability of requirements can be propagated from system model and the tests generated from the model. That means the study has to be focussed on how system specification can be

formulated and how system models can be developed for automatic test generation that helps to trace the requirements.

2.4. Automatic Test Generation and Evaluation

Fundamental approach and the effectiveness of model based testing is the automation that it offers (Bertolino, 2007) .Test suites and test cases are generated algorithmically from models and an execution framework executes the generated tests. The results are evaluated with respect to the expected outputs. Test cases can be generated using different techniques and can be executed in different target languages or execution frameworks such as TTCN3 or JUnit Tests. Here the TTCN3 & JUnit are only considered as a framework for executing the tests generated from models. Tests generated from models could be tests defined using TTCN3, or using some other high level languages or scripting languages. Bertolino views 100% automation (test generation and evaluation of test results) as a challenge in model based testing (Bertolino, 2007). These all demands an additional focus in 100% automation in model based testing, which can only be achieved by efficient test generation techniques.

3. Problem Description

Using models in testing demands a high quality modelling; otherwise the tests generated will not meet the intended goals. Correctness of a model is fundamental necessity to start the test case generation process. If the model is wrong, the generated tests will be invalid.(Dias Neto et al. 2007). Requirements specification also plays a major role in effective and efficient Model Based Testing. Zave classifies the goals of requirements engineering in her article. 'Understanding the priorities of the system', 'obtaining specifications that are well-suited for design and implementation activities' are two among them (Zave, 1997). Requirements specified by the business analyst can be annotated by the Test analyst. Requirement specification in the MBT process should define the test priority of each requirement, test selection criteria, model dependency, "etc.". In model based testing literature several test selection criteria and test generation algorithm can be found as shown in several surveys like [9] or [10]. Most of these algorithms generate thousands of tests (Test case explosion) and several tests generated are difficult to interpret and unambiguous. Also the tests generated need not be executed each and every time. There should be a mechanism for defining a test management process and requirements specification that provides the effective test generation and test execution. Traceability is another issue in model based testing, tests generated from models without tracing them to requirements cannot minimise the defects in models and it will be difficult to analyse the test results. (Dalal et al. 1999)

The problems mentioned above demand a framework /methodological guidance for defining a process that defines requirement specification, modelling, test management, test generation that is well suited for Model Based Testing. Table 1 summarises the research problem and criteria.

Problem	Criteria
Modelling	Model structure, Test Oracles
Requirement specification	Traceability, Requirements priority, Model dependency, Test Oracles
Test Generation	Test generation algorithm
Test Management/Test Explosion	Test priority, Test Selection, Result Analysis

Table 1: List of research problems and appropriate criteria

4. Related Work

A great deal of research has been already done in the field of Model Based Testing especially in field of test generation from Models. Offutt and Abdurazik (Offut et al 2000 and Offut et al. 1999) introduce methodologies for test generation from state-chart diagrams as well as from collaboration diagrams. The test generation is accomplished both using the static and dynamic models. This is considered to be the first approach in Model Based Testing (Dias Neto et al. 2007).

Mingsong use Unified Modelling Language (UML) activity diagrams as design specifications and present an automatic test case generation approach. The approach first randomly generates test cases for a JAVA program under test (Mingsong, 2006).Vieira et al. use UML use cases and activity diagrams to describe which functionalities should be tested and how to test them, respectively. This combination has the potential to create a very large number of test cases. The approach is demonstrated and evaluated based on use cases developed for testing a graphical user interface (GUI) (Vieira et al. 2006)

Soldal defines an algorithm for deriving tests from sequence diagram specifications that takes into consideration the partial nature of sequence diagrams as well as the notion of invalid and universal behaviour introduced in UML version 2.0 with the operators neg and assert (Soldal, 2006). Naslavsky proposes an approach that leverages model transformation traceability techniques to create fine-grained relationships among model-based testing artefacts (Naslavsky, 2006). Cavarra and Chrichton present a architecture for model-based testing using a profile of the UML. Class, object, and state diagrams are used to define essential models: descriptions that characterize the entire range of possible behaviours, in terms of the actions and events of the model. Object and state diagrams are used to introduce test directives (Cavarra et al. 2000).

Emanuela et al. suggest a procedure based on model-based testing techniques with test cases generated from UML sequence diagrams. The proposal is focused on mobile applications (Emanuela et al. 2007). Samuel et al. [19] have proposed automatic test case generation for UML state diagrams. It covers all the events associated with state diagrams (Samuel et al. 2008).

Prasanna and Chandran propose an approach for test generation from object diagrams using a “genetic tree crossover” algorithm. They present a banking system as a case study to justify this approach (Prasanna et al. 2009). Sarma et al. suggest

test case generation from use case and sequence diagrams. These diagrams are converted to a “use case diagram graph” and a “sequence diagram graph” that are then used for test generation (Sarma et al. 2007).

There are also academic proposals available in the field of test generation using the UML & UML Testing Profile (U2TP). With the increasing complexity of Software Systems, effective tests are needed. UML does not provide the means of describing test. OMG (Object Management Group) has defined the profile of UML tests, called U2TP. U2TP fills the gap between designers and testers by providing the means to use UML in system modelling and tests specification [22]. Dai introduces a methodology on how to use the U2TP to transform a model of UML system design. He presents the definition of transformation rules from UML models to U2TP models (Dai, 2004). Zander et al. present an approach to automatically derive executable tests from UML diagrams using the UML Testing Profile (U2TP). The proposal presents a transformation from UML to executable tests (model to executable tests) using U2TP (Zander et al. 2005).

A close examination of the research work reveals the lack of a methodological guidance and an approach for the MBT. All the proposals described above provide one or several approaches for generating tests from models but do not attempt to provide a framework and guidelines for requirement specification, modelling, tests generation, test management, and traceability of requirements to models and tests that suites model based testing.

5. Research Approach

This section presents a research approach, early ideas for solving the above mentioned problems. The proposed solution is intended in the area of UML and business information systems and an evaluation of this approach will be accomplished by a case study. The solution proposed in this section is still at an early stage and only provides an overview and direction of the research approach. Figure 1, shows an overview of the proposed solution.Our research aims to propose a model-based testing framework, process, and methodology to cope with the above mentioned problem. At the current stage of the research, this framework is expected to incorporate the following steps: Figure 1 shows an overview of the proposed model based testing process and Figure 2 shows detailed approach of the proposed framework.

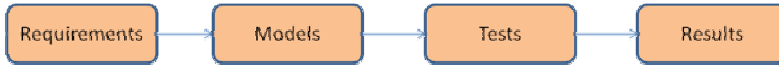

Figure 1: Model-based testing process, overview diagram

5.1. Requirement specification

In the first step, the business analyst specifies the requirements; framework/process defines all needed information such as requirement priority, dependency to other

requirements and related model artefact that are important in a business analyst perspective. Afterwards, the test analyst will annotate the requirements with all needed information for test generation, such as test priority and test selection criteria. See "Requirement specification" in Figure 2.

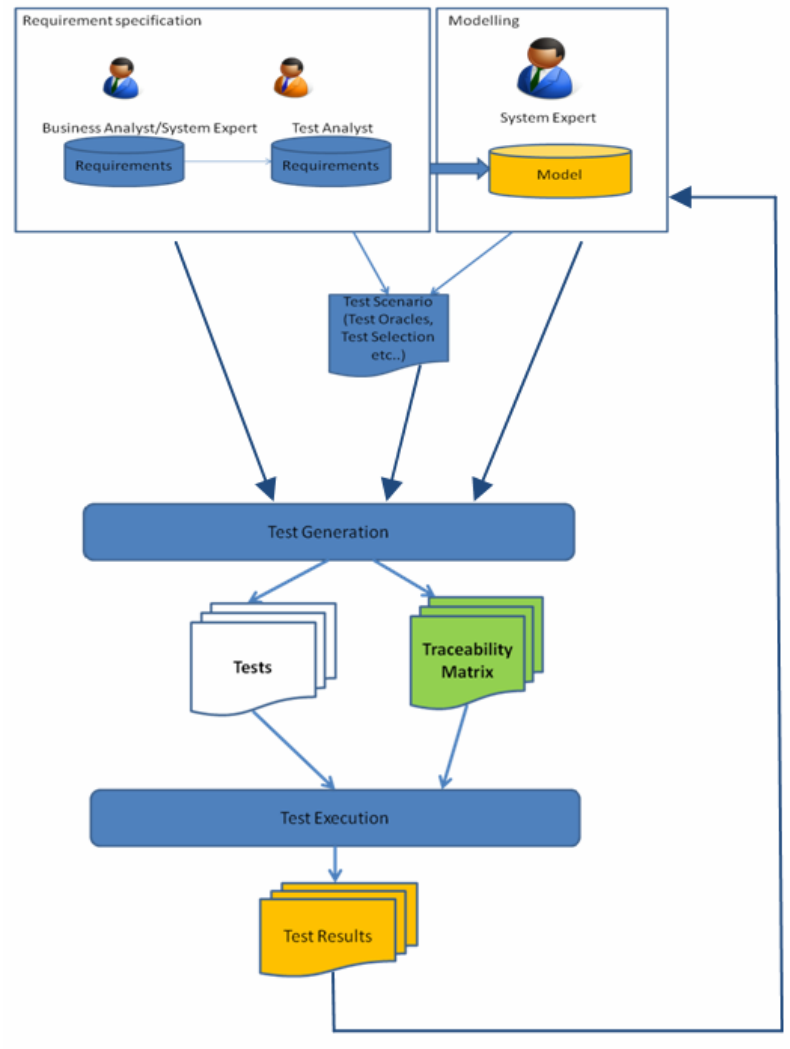

Figure 2: Model- based testing process, detailed diagram

5.2. Modelling

A design using UML will be described by the system expert. Additional information required for test generation has to be provided as per the guidelines defined in the proposed framework. Framework will customize the design model by taking the necessary information from the test analyst.

5.3. Test Generation

A test management criterion called test scenario that consists of test selection, requirement priority, test priority and test oracles will be generated from the model and requirements annotated by the test analyst. Semantics of the test cases will be considered and defined in the next stages of the research. Test scenario along with the requirements and the models will be used for generating the tests. That means the priorities defined in the test scenario will be later used for test generation and test execution. Suppose a user wants to execute all the tests with higher priority, then it is possible by defining the adequate test scenario. A traceability matrix will be generated that traces requirements to models and the generated tests. See test generation in Figure 2.

5.4. Test Execution & Analysis

Generated tests from the models will be executed and the test results will be generated for further analysis. See test execution and analysis in Figure 2.

6. Conclusion and Further Work

Proposed framework offers different advantages that include early detection of finding architectural, specification and design failures before implementation. A major contribution of the our research is to define and develop a framework, methodological guidance to facilitate model based testing as major part of the software development process by providing solutions for the problems defined above.

In the future, a framework will be developed (See Figure 2) which facilitates requirement specification, modelling, test management, test generation, test execution and test analysis. A case study will be carried out to prove the efficiency and effectiveness of the framework. The degree of completeness of the solution will be analysed and measured based on the case study.

7. References

Andrews, A., France, R. B., Ghosh, S. and Craig, G. (2003), “Test adequacy criteria for UML design models”, Software Testing, Verification Reliability, vol. 13

Bertolino, A. (2007), “Software testing research: Achievements, challenges, dreams” FOSE ’07: 2007 Future of Software Engineering. IEEE Computer Society.

Bertolino, A., Marchetti, E., and Muccini, E. (2005), “Introducing a Reasonably Complete and Coherent Approach for Model- Based Testing”, Electronic notes in theoretical Computer Science.

Bringmann, E., and Kraemer, A. (2008), “Model-based Testing of Automotive Systems”,2008 International Conference on Software Testing, Verification, and Validation.

Cavarra, A, and Chrichton, C. (2000), "Using UML for Automatic Test Generation", Oxford University Computing Laboratory, Tools and Algorithms for the Construction and Analysis of Systems.

Dai, Z.R., (2004), "Model-driven testing with UML 2.0.", Fraunhofer FOKUS, Berlin, Germany.

Dalal, S.R., Jain, A., Karunanithi, N. Leaton, J.M., Lott, C.M., Patton, G.C and Horowitz, B.M. (1999), "Model-Based Testing in Practice" ICSE 1999.

Dias Neto, A., Subramanyan, A. R. , Vieira, M., and Tracassos, G. (2007) "A survey on model-based testing approaches: A systematicreview," Siemens Corporate Research, Tech. Rep., 2007.

Emanuela, G.C, Franciso, G.O., and Patricia, D.L.M. (2007), " Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems", Systems, Man and Cybernetics, 2007. ISIC.

Legeard, B. (2010), "Model-based Testing: a new paradigm for manual and automated functional testing", Testing Experience magazine, March 2010.

Mc Quillan, J.A. and Power, J.F. (2005), "A Survey of UML Based Coverage Criteria for Software Testing," Department of Computer Science, Tech. Rep.

Mingsong, C. (2006), " Automatic test case generation for UML activity diagrams", AST '06: Proceedings of the 2006 international workshop on Automation of software test.

Naslavsky, L. (2007), " Towards traceability of model-based testing artifacts", Proceedings of the 3rd international workshop on Advances in model-based testing, 2007.

Offutt, J. and Abdurazik, A. (1999), " Generating Tests from UML Specifications", George Mason University, Fairfax VA 22030, USA.

Offutt, J. and Abdurazik, A. (2000), "Using UML Collaboration diagrams for static checking and test generation", Third International Conference on UML in York, UK.

Prasanna.M and Chandran,K.R. (2009) Automatic Test Case Generation for UML Object diagrams using Genetic Algorithm, ICSRS Publication

Samuel, P., Mall, R. And Bothra A.K. (2008), "Automatic Test Case Generation Using UML State Diagrams".

Sarma, M. and Mall,R. (2007), "Automatic Test Case Generation from UML Models", 10th International Conference on Information Technology

Soldal, M. (2006), "Deriving tests from UML 2.0 sequence diagrams with neg and assert" AST '06: Proceedings of the 2006 international workshop on Automation of software test.

UML Testing Profile, (2009). http://utp.omg.org

Utting, M. and Legeard, B. (2007), "Practical Model-Based Testing: A Tools Approach. Morgan Kaufmann"

Utting, M., Pretchner, A. and Legeard, B. (2006), "A Taxonomy of Model Based Testing , a white paper."

Vieira, M., Leduc, J., Hasling, B., Subramanyan, R. and Kazmeier, J. (2006)"Automation of GUI testing using a modeldriven approach," in AST '06: Proceedings of the 2006 international workshop on Automation of software test.

Zander, J., Dai, Z.R., Schieferdecker, I. and Din, G. (2005). "From U2TP models to executable tests with TTCN-3 -an approach to model driven testing", Fraunhofer FOKUS, Berlin, Germany.

Zave, P. (1997), "Classification of research efforts in requirements engineering, " Computing Surveys (CSUR)" , Volume 29 Issue 4.