The UML 2.0 Test Profile as a Basis for Integrated System and Test Development

Ina Schieferdecker

Technical University Berlin/Fraunhofer FOKUS
Kaiserin-Augusta-Allee 31
D-10589 Berlin
ina.schieferdecker@fokus.fraunhofer.de

Abstract: Model centric development and engineering according to Model-Driven Architectures (MDA) has recently gained much attention. This paper presents an approach how a model centric approach cannot only be used for system development but also at the same time to support the provision of system tests which are to be an integral part of the overall system development. The paper discusses the various artefacts for model-based testing along MDA and their relation to the artefacts in system development. The testing artefacts can be designed and modelled with the UML 2.0 Testing Profile (U2TP) which extends UML 2.0 with test specific concepts.

1 Introduction

Test software is a special kind of software dedicated to the analysis and evaluation of software systems. It can be modelled and developed the same way as system software, so that development processes established for the design and development of systems can theoretically be used also for test systems. Practically however, system specification techniques do not take into account the specifics of test systems, like being able to define test components, test data or verdicts. Many efforts have to be spent to enable system developers and testers to understand each other, to transform system artefacts into test artefacts, to keep them consistent, to relate and trace system requirements to test cases and test results, etc. The situation can be improved if both the software system and its test system(s) can be designed, specified and visualized within one technology. Then, the people involved as well as the reviews, transformations, consistency checks, coverage analysis, etc. do not have to cross technology borders, but can focus on the content of the artefacts such as the system and test logic. As UML has been established as the industrial standard in system design, its extension towards test systems was a compelling idea, which resulted in the initiation and definition of the UML 2.0 Testing Profile (U2TP [U2TP04]) By doing so, UML based development processes can also be used for test design and test generation.

Model centric development of software system has recently become an important software engineering strategy for handling the complexity and the increasing requirements to larger and highly distributed software systems. The OMG initiative on Model Driven Architectures (MDA) prescribes certain model artefacts to be used along system development, how those models may be designed and their relationship [MDA03]. It is an approach to system development that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform [QVT04]. Main MDA artefacts are computationally independent platform models (CIMs), platform independent system models (PIMs), platform specific system models (PSMs) and system code [KWB03]. There is a clear distinction between CIM, PIM, PSM and system code although it depends on the context, the development process and the details of the system and target platform, where the borders between CIM, PIM, PSM and system code are to be placed. Within these abstraction levels, transformation techniques are used to translate model parts of one abstraction level into model parts on another abstraction level. These transformations can also be used to specify the relations and invariants between the models on different abstraction levels, which are the base to check the consistency between models and to validate models against each other. These MDA abstraction levels can also be applied to test modelling [G03] as according to the philosophy of MDA, the same modelling mechanism can be re-used for multiple targets [S01]. Similarly, test models can be specified independent from the computations, independent from the platform details and specifically for a target platform before generating executable test codes [D04].

2 The UML 2.0 Testing Profiles

The UML 2.0 Testing Profile (U2TP) defines a language for designing, visualizing, specifying, analyzing, constructing and documenting the artefacts of test systems. It is a test modelling language that can be used with all major object and component technologies and be applied to test systems in various application domains. Being a profile, the U2TP seamlessly integrates into UML. It is based on the UML 2.0 meta-model [2] and reuses UML 2.0 syntax.

The UML 2.0 Testing Profile provides concepts that target both the pragmatic and systematic development of concise test specifications and test models for black-box and grey-box testing [U2TP04,U2TP04a]. In particular, the profile introduces concepts covering: test architecture, test behaviour, test data, and time. The profile defines testing concepts, including test context, test case, test component, and verdicts that are commonly used during testing. Behavioural elements from UML 2.0 can be used to specify the dynamic nature of test cases. These include interactions, state diagrams, and activity diagrams. Additional concepts for behaviour include several types of actions (validation actions, log actions, final actions, etc.), defaults for managing unexpected behaviours, and arbiters for determining the final verdict for a test case. The definition and handling of test data is supported by wildcards, data pools, data partitions, data selectors and coding rules. Timers and time zones are also provided to enable specifications of test cases with appropriate timing capabilities.

The philosophy being adopted for the development of U2TP has been to make use of existing UML 2.0 concepts wherever possible, thereby minimizing the introduction of new concepts. The U2TP concepts are structured into

- Test architecture concepts defining concepts related to test structure and test configuration, i.e. the test components and system components involved in a test and their relationships

- Test behaviours concepts defining dynamic aspects of test procedures covering stimuli, observations and evaluation activities during a test,

- Test data concepts defining concepts for test data used in test procedures, i.e. the structures and meaning of values to be processed in a test, and

- Time concepts defining concepts for a time quantified definition of test procedures, i.e. the time constraints and time observation for a test execution.

The use of U2TP for the development of test suites from system models has been for example described in [U2TP04b]. While these tests have been manually derived and checked for consistency, a more systematic approach for test generation and validation is possible with model-driven test techniques as outlined in the following section.

3 MDA++ and the Test Artefacts

MDA tries to overcome common problems in existing development processes: requirements, specifications and development information are often documented and communicated via paper – consistency is therefore hard to achieve. Typically, experts responsible for different development tasks in different development phases do not interact on a common terminology base and face therefore communication and efficiency problems. Transitions between different phases in development process are not or not sufficiently automated. And last but not least, developers and testers do not share artefacts – not only but also as technologies and methods for early and continuous tests throughout the development process are missing. U2TP bridges the gap between project leaders, developers and testers by providing means for using UML also for test development and modelling. This allows the reuse of UML design documents for testing and enables test development in an early system development phase. MDA++ - the extension of MDA with test support – adds to this the various test artefacts and their relations to the system artefacts. The overall approach of MDA++ is depicted in Fig. 1.

The artefacts used along the development process are based on metamodels which define the process, technologies, methods and system specifics in form of concept spaces reflecting the concept structures and their relations. There exist metamodels for system artefacts to capture the computationally independent system model (CIM), the platform independent system model (PIM), the platform model (PM), the platform specific model (PSM) and the system code (SC).

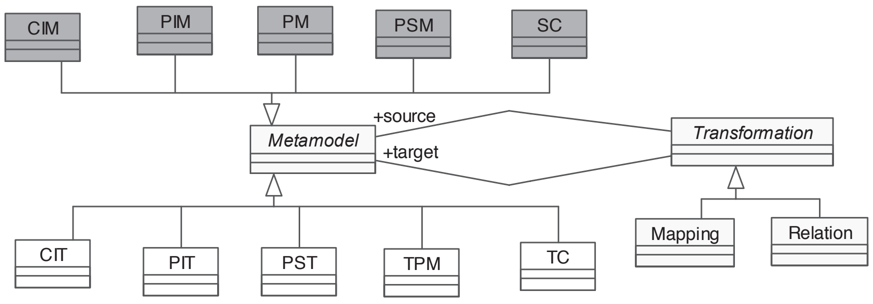

Figure 1: MDA++: MDA with testing support

The models can be associated via various transformations being either mappings or relations. For example, a PSM is typically derived from a PIM by taking into account the characteristics of a PM.

Likewise, for the test development there exist a computational independent test (CIT), a platform independent test (PIT), a platform specific test (PST), a test platform model and a test code (TC). Please note that the platform in PST relate to the system platform, i.e. the PST contains tests being specific to the characteristics of the system platform. The test platform is taking into account when generating the TC from a PST.

In addition, there are mappings from system models to test models such as from CIM to CIT or from PIM to PIT. There are also relations between system and test models which define e.g. consistency or coverage criteria. Another relation is the relation between test code and system code which reflects the application of the tests onto the real system and which will result in test reports. Examples for the mappings from system to test models include:

- The CIM to CIT mapping which derives test objectives and test suite structures from the business objectives combined with overall test strategies used in a specific development process,

- The PIM to PIT mapping which can be based on a mapping from use cases (and their formalization in form of activity diagrams or sequence diagrams) onto sets of abstract test cases, which define the behavioural test procedures but leaving out details of test data,

- The PSM to PST mapping which derives test dedicated to the platform specific interfaces and to the special treatment of system components on the target platform. It will also resolve data dependencies and complete abstract test cases into concrete test cases such that specific test data is sent to and expected from the system under test.

The approach of generating test models from system models follow established techniques for e.g. structural aspects of tests such as types, test components and their test configurations and for e.g. behavioural aspects such as the derivation of tests from (extended) finite state machines (represented by UML state machines) or message sequence charts (represented by UML interaction diagrams) can be used. The first uses mainly state or transition coverage methods, while the second uses branch or path coverage methods to derive the various test sequences.

Open questions include for example if transformation from PIM to PSM to PST will result in the same tests when transforming PIM to PIT to PST. We are currently at the beginning of this research, where we use our results in test specification with U2TP and test generation from UML 2.0 models to U2TP. We expect interesting results when analysing MDA++ in application domains such as the financial domain or in the automotive domain.

References

[U2TP04] OMG: UML 2.0 Testing Profile. Final Adopted Specification, ptc/04-04-02, 2004.

[U2TP04a] P. Baker, Z. R. Dai, J. Grabowski, Ø. Haugen, S. Lucio, E. Samuelsson, I. Schieferdecker, and C. Williams: The UML 2.0 Testing Profile, Conquest 2004, ASQF Press, September 2004, Nuremberg, Germany.

[U2TP04b] I. Schieferdecker: The UML 2.0 Testing Profile U2TP, TIMNA 2004, Telecommunications Information Multi-Management Networking Architecture, October 2004 Beijing, China.

[MDA03] OMG: Model-Driven Architecture (MDA) www.omg.org/docs/omg/03-06-01.pdf, www.omg.org/docs/formal/02-04-03.pdf.

[QVT04] OMG: MOF Query/Views/Transformations, 2nd Revised Submission, ad/04-01-06, 2004.

[UML03] OMG: UML 2.0 Superstructure Final Adopted Specification, www.omg.org/cgi-bin/doc?ptc/2003-08-02.

[KWB03] A. Kleppe, J. Warmer, W. Bast: MDA Explained: The Model Driven, Architecture–Practice and Promise. Addison-Wesley Pub Co, 2003.

[G03] Gross, H.: Testing and the UML – a perfect fit. Fraunhofer IESE, Technical Report 110.03E, 2003.

[S01] J. Siegel, OMG Staff Strategy Group: Developing in omg's model-driven architecture., 2001.

[D04] Z. R. Dai: Model-Driven Testing with UML 2.0, Second European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations (EWMDA'04), Canterbury, England, September 2004.

[AWL04] D. Amyot, M. Weiss., L. Logrippo: UCM-Based Generation of Test Goals. ISSRE'04 Workshop on Integrated-reliability with Telecommunications and UML Languages (ISSRE04:WITUL), Rennes, France, November 2004.

[ZDSD05] J. Zander, Z.R. Dai, I. Schieferdecker, G. Din: From U2TP Models to Executable Tests with TTCN-3 - An Approach to Model Driven Testing, IFIP 17th Intern. Conf. on Testing Communicating Systems - TestCom 2005, Montreal, Canada, March 2005.