Peer-to-Peer Comparison of Model-Based Test Tools

Christof J. Budnik[1], Rajesh Subramanyan[1], Marlon Vieira[1]

[1] Software Engineering, Siemens Corporate Research, Inc.,
755 College Road East, Princeton, NJ 08540
christof.budnik, rajesh.subramanyan, marlon.vieira@siemens.com

Abstract: The aim is to understand the differences between TDE/UML and peer tools available on the market to assess strength and weakness of TDE/UML. Evaluation comparison criteria are modeling, test case generation and extensibility of the model-based testing tools under consideration. The methodology can be adopted using tools other than TDE as a baseline for similar comparisons.

1 Introduction and Related Work

System testing is critical in the development and testing lifecycle. It entails functional testing of the software and hardware components together. Model-based testing (MBT) is a system test methodology to design test cases with adequate coverage using UML or other models artifacts of a system under test (SUT) [PP05]. Some MBT tools (MBTT) support single and not multiple test levels. TDE/UML [VLH+06] is a Siemens internal model-based testing tool, based on an internal test case generation methodology, and used within Siemens on projects of different domains over the last 10 years. This paper evaluates MBT competitiveness of TDE/UML to commercial peer tools and assesses the strengths and weaknesses.

MBT has been extensively studied last decade despite little industry adoption. Many support and test case generation tools have emerged leading to interest on tool surveys. [Ha02] is a comprehensive survey of academic prototypes to commercial, [UPL06] is a classification taxonomy and [Sc07] is on derived models. While other surveys cover pros and cons of large varied approached, this paper differs as it compares few peer tools at a higher granularity, more in-depth and the entire lifecycle and tool integration.

2 MBTT Evaluation and Selection Criteria

This section defines what peer MBT tools are and what makes a test tool an MBT tool. MBT is defined as an approach to derive executable tests from a given model of the system under test (SUT) using several test selection criteria. The model is built either from requirements and/or from specifications of test model. The model should contain both input and expected output in order to be able to generate test oracles [PP05]. Thus, no model based input generator and no test automation framework where test cases can be manually created or pre-recorded [Ha02] are considered to be MBTTs.

An MBT tool supports the test life cycle and interacts with other development phase elements. Selected evaluation criteria are (i) Test Modeling - design of the test model derived from the SUT, (ii) Test Generation – strategy for deriving test cases, and (iii) Extensibility – integration possibilities with other tools via import/export interfaces and/or extending tool to different domains. The usability criterion has been intentionally omitted. Test generation algorithm are better covered elsewhere and omitted here.

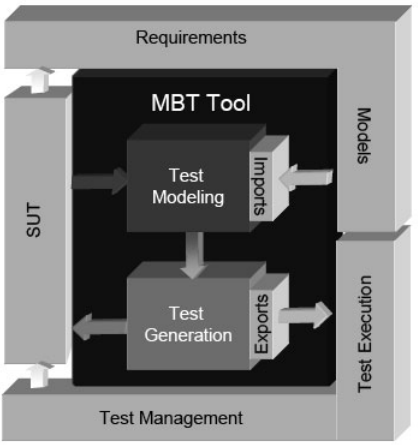

Figure 1: MBTT Categories of Interest

MBTT peers are selected if they satisfy the following criteria: (i) Usage of a test model (not a system model) from where tests are derived. (ii) Automate test generation covering test input data and system behavior. (iii) UML and system testing as used in TDE. (iv) It has to be a commercial product used to assess what is missing in TDE; i.e. gap analysis.

3 Analysis and Results

The following MBTTs met the above selection Leirios TestDesigner™ [UL06], Conformiq QTronic™ [UL06], and Innovator™ from MID [BDK+08].

Test Modeling compares support for modeling, SUT behavior and their test data. Behavior modeling is separated to those using statecharts (Qtronic, TestDesigner) vs. use-case and activity diagrams (TDE/UML, Innovator). The behavior model is extended by appropriate annotations of the test data derived from either formal languages or diagrams. Qtronic uses a proprietary language QML, a combination of OCL and XML. Test Generation compares approaches for generating test cases, how the test case set can be reduced or optimized, verifying test results against the expected output, and what kind of test reports can be created. All tools support the most common transition coverage criteria to test different sequences of the SUT. Differences arise in the generation of test cases depending on the coverage of different test data. The strength of TDE/UML is in the optimization criteria. Other tools do not support any approaches or just reduce the number of test cases for regression testing. External tools supported for each MBT tool (numbers below are references in Table 1).

Modeling Tool Integration: IBM Rational Software Modeler[1], Borland Together[2], Telelogic Rhapsody[3], Artisan Studio[4], Enterprise Architecture[5], Vitechcorp Core[6]

Requirement Tool Integration: IBM Rational Requisite Professional[1], Borland Caliber RM[2], Telelogic DOORS[3]

Test Execution Tool Integration: IBM Rational Functional Tester[1], Borland SilkTest[2], HP Quick Test Pro[3], Compuware TestPartner[4], Interpretation[5] (TTCN-3, Python, Ruby, etc.), Functional (JUnit, NUnit, etc.) [6], imbus Testbench[7]

Test Management Tool Integration: IBM Rational ClearQuest Test[1], Borland SilkTest[2], HP Quality Center[3], Compuware QA Center[4], imbus Testbench[5]

Table 1: Comparison of Test Modeling, Test Generation, and Extensibility

	SCR - TDE/UML	MID - Innovator	Conformiq - Qtronic	Leirios - TestDesigner
Test Modeling				
Data Coverage	Category Partition	Class-D., Object-D.	QML (Proprietary Language)	OCL conditions and actions
Behavior Coverage	Use Case/ Activity D.	Use Case/ Activity.-D., MS Excel	Statecharts	Statecharts
Model Creation	Integ. UML editor	Proprietary Integ. Editor	Conformiq Modeler	None
Model Verification	General/ Customized Checkers	None	Built-in analyzer and debugger	None
Test Generation				
Test Case Generation	Transition Coverage, Choice (Data) Coverage:	Coverage Criteria C4, Path-Coverage	Transition Coverage, Boundary value analysis	Transition-Coverage, Any one input value
Optimization/ Selection	Risk Analysis, Prioritization	None	None	Test Requirement Changes
Test Verification	Automatic Verification	Manual insert of expected output	Symbolic Execution, Simulation	Actual system state, manual data output
Test Reports	Word, HTML	XML	HTML, XML	Generic XML
Extensibility and Tool Integration				
Modeling Tool Integration	2,6	None	1, 3, 4, 5	1, 2
Requirement Tool Integration	None	None	Traceability and mapping to high-level Req.	1, 2, 3
Test Execution Tool Integration	Custom Test Frameworks, 3, 4, 5, 6	2, 7	5, 6	1, 2, 4, 6
Test Management Tool Integration	3	5	3	1, 2, 3, 4
Customization	Yes, external: API	API exist	Open API	None

Figure 2 highlights tool strengths for each comparison criteria. The results are classified into three levels of compliance (high – outer ring, medium to low – inner ring). Although Leirios TestDesigner lacks an integrated editor for model design and API for tool extension, it supports most interfaces to third-party tools. The Conformiq Qtronic fares positively against all criteria. It does not have a test set optimization, but has requirement traceability. The MID Innovator supports the model-based test lifecycle but has limited features. SCR TDE/UML is strong across the comparison criteria. The test model is created by an integrated UML editor, followed by model verification using static model checking rules, a test generation strategy, and finally the possibility of custom extensions. TDE/UML is competitive in tool integration and customization. Its strengths and weaknesses are understood and can be used towards the improvement of the tool as the next steps.

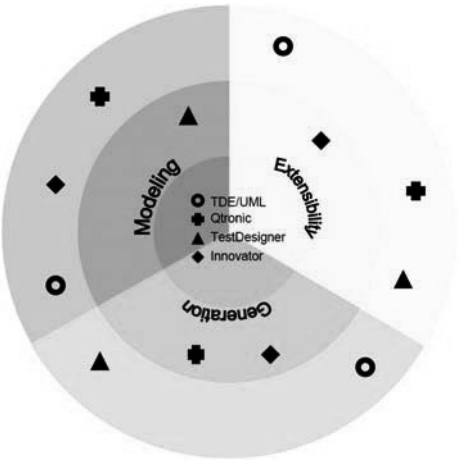

Figure 2: Peer-to-Peer Comparison Result

Concluding, a peer-to-peer comparison of four strongly similar tools using a different approach was presented. The approach is reusable where another tool replaces TDE/UML as the baseline. Future work includes extending with formal empirical studies and a comparison between a MBT paper study and its adaptation.

References

[BDK+08] Brandes, Ch., Ditze, A., Kollee, Ch., Kreische, D.: Modellbasiertes Testen praxisgerecht realisiert: Vom UML2-Profil zum Testmanagement-Werkzeug, Journal OBJEKTspektrum, SIGS-DATACOM, vol. 03, 2008, p. 80-97.

[Ha02] Hartmann, A.: AGEDIS Model based Test Generation Tools, AGEDIS Consortium, 2002.

[PP05] Pretschner, A., Philipps, J.: 10 Methodological Issues in Model-Based Testing, Model-Based Testing of Reactive Systems, Lecture Notes in Computer Science, vol. 3472, 2005, p.281-291.

[UL06] Utting, M. and Legeard, B.: Practical Model Based Testing: A Tools Approach, Morgan-Kaufmann Publishers Inc., 2006.

[UPL06] Utting, M., Pretschner, A., Legeard, B.: A Taxonomy of Model-Based Testing, Working Paper 4/2006, University of Waikato, 2006.

[Sc07] Schieferdecker, I.: Modellbasiertes Testen, Journal OBJEKTspektrum, SIGS-DATACOM, vol. 03, 2007, p. 39-45.

[VLH+06] Vieira, M., Leduc, J., Hasling, B., Subramanyan, R., Kazmeier, J.: Automation of GUI testing using a model-driven approach, In Proceeding of the 2006 International Workshop on Automation of Software Test, AST'06, ACM Press, 2006, p. 9-14.