Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

VERITAS - A VERSATILE MODELING ENVIRONMENT FOR TEST-DRIVEN AGILE SIMULATION

Anatoli Djanatliev
Winfried Dulz
Reinhard German
Vitali Schneider

University of Erlangen-Nuremberg
Department of Computer Science (Informatik 7)
Martensstr. 3
D-91058 Erlangen, GERMANY

ABSTRACT

An approach is presented in which both simulation and testing based on UML are combined in one framework to achieve an improved overall quality. System models are specified by UML diagrams, and are then mapped on C++-code and executed in the OMNeT++ network simulation framework. State-space oriented test models are defined independently from this to express requirements by selected system usages. From these test models it is possible to generate test cases and to execute them on the simulation code level. By adding Markov chain usage profiles to the test model it is possible to apply statistical test case generation as well. Altogether this allows to validate both kinds of models systematically and iteratively during the development cycle. The methodology is realized by combining appropriate tools and new software components based on the Eclipse RCP. The approach is also well suited for software engineering because standard modeling languages are used.

1 INTRODUCTION

The implementation of complex software systems usually consists of a series of development and testing phases. Starting from requirements definition the development process is divided into a number of design, specification, programming and testing steps. Each software engineering step is guided by an appropriate method and generally supported by dedicated tools, e.g., Rational DOORS (IBM 2011), Papyrus UML (Papyrus 2008), Enterprise Architect (Sparx Systems Ltd. 2011), and .getmore (sepp.med gmbh. 2011).

Due to the increasing complexity of software systems model-driven approaches are gaining in popularity. In particular, graphical representations like UML (*Unified Modeling Language*) diagrams of special system features allow to deal with the complexity and also enable advanced analysis and validation capabilities. The UML allows to describe the structure and behavior of a system and eventually such a model can be executed or simulated in order to gain insight in its dynamics. The UML also allows to define models of tests of the later system, which is then considered as the SUT (*system under test*). Following these observations a simulation framework Syntony (Dietrich, Dressler, and German 2009) has been developed, in which system models can be described by UML models, compiled to C++ and executed in the OMNeT++ network simulation framework. This work has been extended by the possibility of defining and executing test cases for single components of the system model according to the UML testing profile (Dietrich et al. 2010). In this paper this framework is complemented by adapting advanced concepts from model-based testing such that truly integrated simulation and testing becomes possible.

Model-based testing techniques make use of models of either the behavior of the SUT or the expected usage of the users of the SUT. In the former case a system specification is derived from the requirement definitions and serves as a starting basis to automatically generate test cases in order to test the SUT (Rosaria and Robinson 2000). In the latter case usage models are deduced from the requirements and may be considered as independent from the specification. Because exhaustive testing of real systems is infeasible in practice an appropriate set of test cases must be provided for accomplishing a given test goal. In contrast to other testing techniques, the statistical testing approach benefits from usage driven testing based on a set of statistically sampled test cases (Dulz and Zhen 2003, Prowell 2005). Statistical testing relies on a Markov chain usage model (Walton, Poore, and Trammell 1995; Whittaker and Thomason 1994) that characterizes the possible uses of given software in its intended environment and consists of usage states and transitions between these states. Test cases are automatically derived by randomly traversing the Markov chain between two dedicated start and end states.

Design and implementation errors are the more expensive the later they are detected and fixed. Therefore, we call for the earliest possible validation of the specification models in the course of the software engineering process. Otherwise, errors are detected too late while testing the SUT in the system or even the acceptance test.

In the VeriTAS environment we apply, among others, a UML modeling tool, a simulation framework and a test case generator to perform a so-called *test-driven agile simulation* (TAS); statistical usage testing can also be applied. Altogether a new methodology is defined with the following three contributions: first, a systematic way is presented with which simulation models can systematically be validated iteratively during the development phase. This is considered important since correctness of complex simulation models is hard to control. Second, as model-based testing gets more popular, test models are getting more complex and their correctness is also hard to control; they can be validated early against the simulation model. Third, an overall process is defined which if combined with code generation can provide high quality software with respect to quantitative and functional aspect.

In the following sections we explain the idea of test-driven agile simulation and the main parts of the VeriTAS environment in order to support model-driven software development, testing and simulation. To illustrate various concepts throughout this paper a model of the Stop-And-Wait protocol (Dietrich et al. 2010) will be used as an example.

2 TEST-DRIVEN AGILE SIMULATION

As mentioned before, UML-based models are growing in popularity for system modeling as well as for test specifications. Usually, the testing model is derived from a system model and aims to validate the functional as well as the non-functional behavior specified for that system. Thereby, the validation is performed by executing tests on the system after it has been implemented. To better assure the validation of the required system behavior we propose to start from the requirements and to derive the system specification as well as the test specification independently from each other (Figure 1).

The key idea of the TAS approach is to use the same test cases that are otherwise used in the latter system test for an early validation step of the involved models. This is done by simulating a given system specification and running test cases at the model level. The method also implies the automatic generation of executable simulations and of test cases from corresponding models, which can be executed within a simulation framework. This approach lets us detect modeling errors or inconsistencies in the simulated system model and the used test model as early as possible, even before any code or hardware has been implemented. Prior to expensive tests on the real system in customer's target environment, generated test cases as well as the system specification can easily be validated by executing the simulation. Hence, one of the main advantages of the TAS approach is to provide a cheap and agile technique to check the models in several iteration steps during early engineering phases.

In addition to the dynamic validation, when tests are performed by means of simulation, it is also reasonable to compare the static characteristic of system and test models, e.g., if the models cover all specified (usually functional) requirements or if the same name conventions have been used. In the next section we describe the VeriTAS environment, which supports the TAS approach.

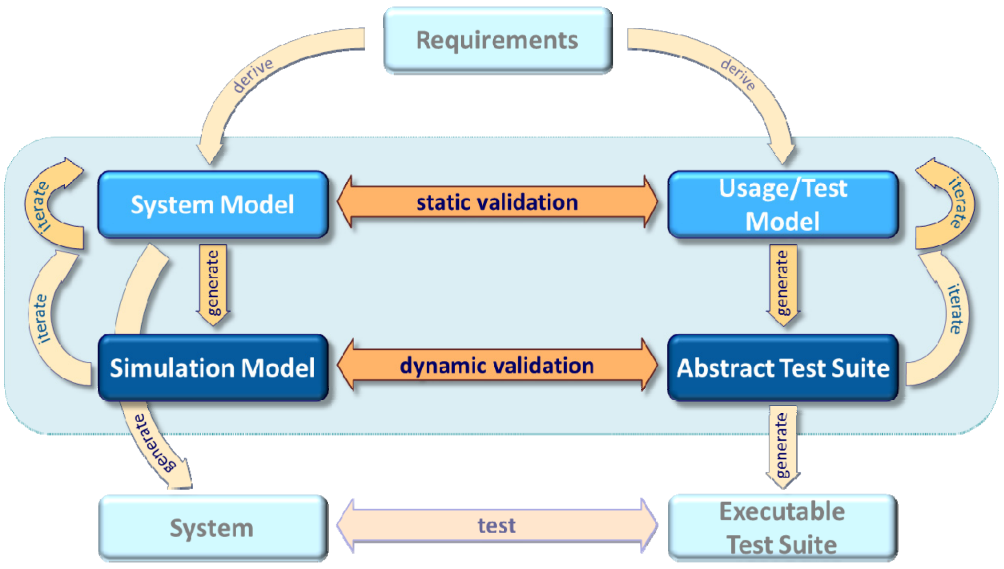


Figure 1: The concept of the Test-driven Agile Simulation

3 THE VERITAS ENVIRONMENT

VeriTAS is based on the Eclipse Rich Client Platform that allows combining specific components supporting the main steps of a quality-assurance process within a common environment (Figure 2) and which implies a typical tool-chain.

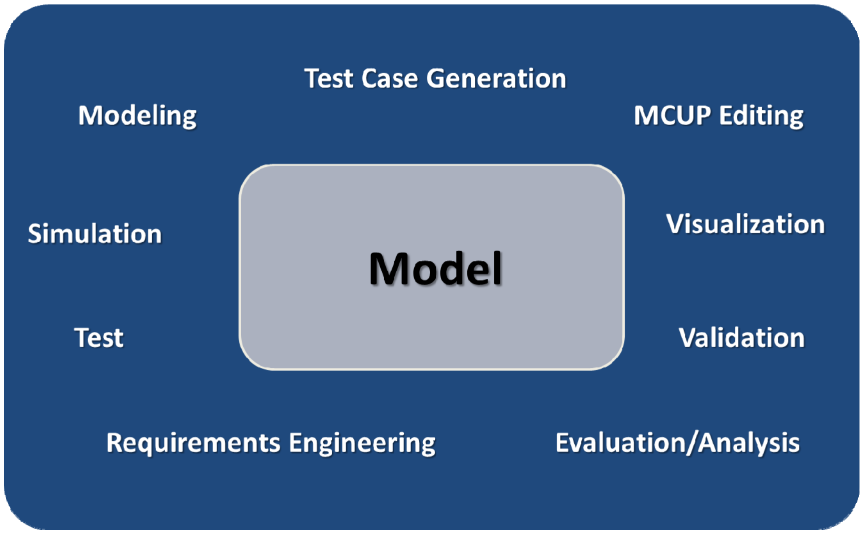


Figure 2: Overview of the VeriTAS environment

The current realization of the VeriTAS tool chain applies Papyrus UML and Enterprise Architect to produce UML conform specifications for system simulation and test. In the next step the usage model is imported by the test case generation tool .getmore (sepp.med gmbh. 2011) that enables the generation of abstract test cases dependent on the desired strategy. Additionally, a visualization framework offers customized layout algorithms to visualize models that are managed by .getmore and allows highlighting diverse aspects within a model, e.g., generated test cases. It also provides an editor for Markov chain usage profiles *MCUPEditor* in order to generate test cases statistically. To simplify the usage profile definition nodes and transitions may be highlighted by different colors that indicate different weights. This new feature allows focusing on special regions of interest within a usage model that are considered by the statistical test case generation strategy. In our TAS approach test cases are used to validate system specifications by means of simulations. In order to generate an executable simulation model from a given system specification, i.e., a UML state chart diagram, the simulation framework Syntony (Dietrich et al. 2010) is applied within the VeriTAS tool chain. Furthermore, a dedicated exporter has been developed for automatically transforming generated test cases into the UML format. Finally, simulation results and test metrics can be calculated and visualized within Syntony by using the public domain tool R.

3.1 Model Definition

Different types of UML diagrams can be used to describe certain aspects of the simulated or tested system, like architecture, data and behavior. In our example model of the Stop-And-Wait protocol we follow the modeling paradigm of communicating automata. Among others, we take the use of class, composite structure and state chart diagrams to describe the system (Figure 3). In this example the system represents protocol units that are responsible for reliable data transfer (RDT) over an unreliable channel. To specify non-functional performance attributes and measurements, that are relevant in scope of simulation, we use the UML profile for MARTE (Modeling and Analysis of Real-Time and Embedded Systems).

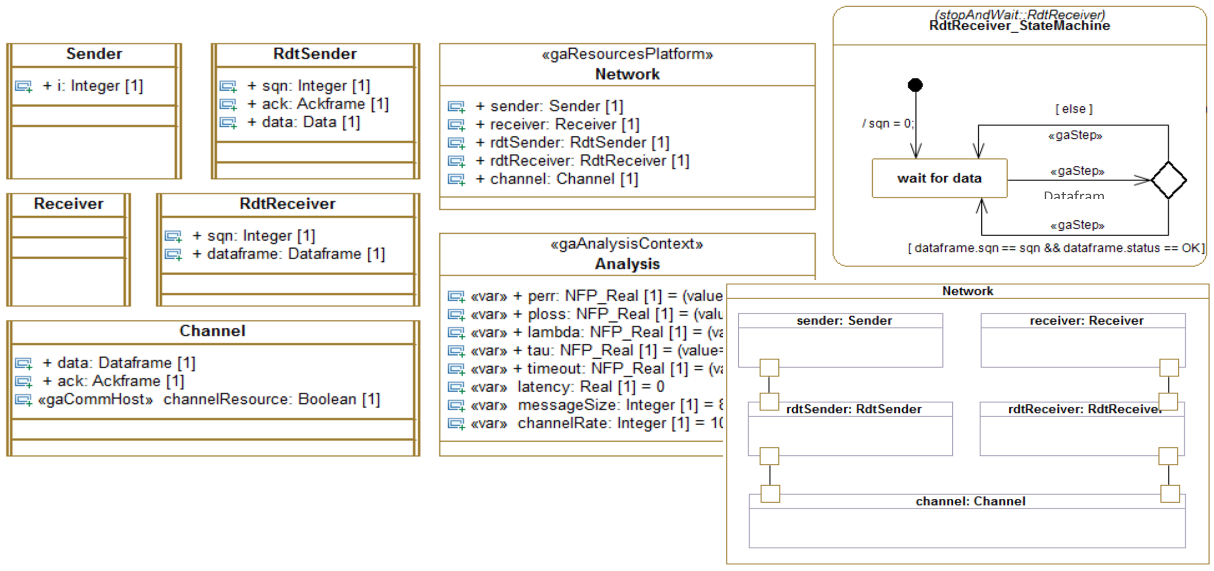


Figure 3: Extract of the Stop-And-Wait model in PapyrusUML

To validate the behavior of the specified units we can define model-based unit tests (Dietrich et al. 2010). According to the UML Testing Profile (UTP) a test context describes the SUT as well as involving test components and contains test cases. Usually, test cases are modeled as sequence of simple test steps, which express the interaction with the SUT. In our approach we make use of automated test case generation from activity diagrams or Markov chain usage models, which specify the usage of a system or a single component from user's point of view. Thereby, single test steps are modeled as parameterized interactions and are combined to a sequence during the generation.

3.2 Visualizing Hierarchical Usage Models

As mentioned before we operate with usage models expressed as graphs consisting of nodes and arcs. To support the visualization of hierarchical diagrams the visualization framework has been developed based on the Eclipse Zest Visualization Toolkit. As test-cases represent single paths of a model, the visualization component of VeriTAS offers a parallel view of the test-cases as a tree and the corresponding diagram with highlighted paths (Figure 4). Furthermore it is easily possible to navigate through diagram hierarchies and to configure the appearance of single model elements.

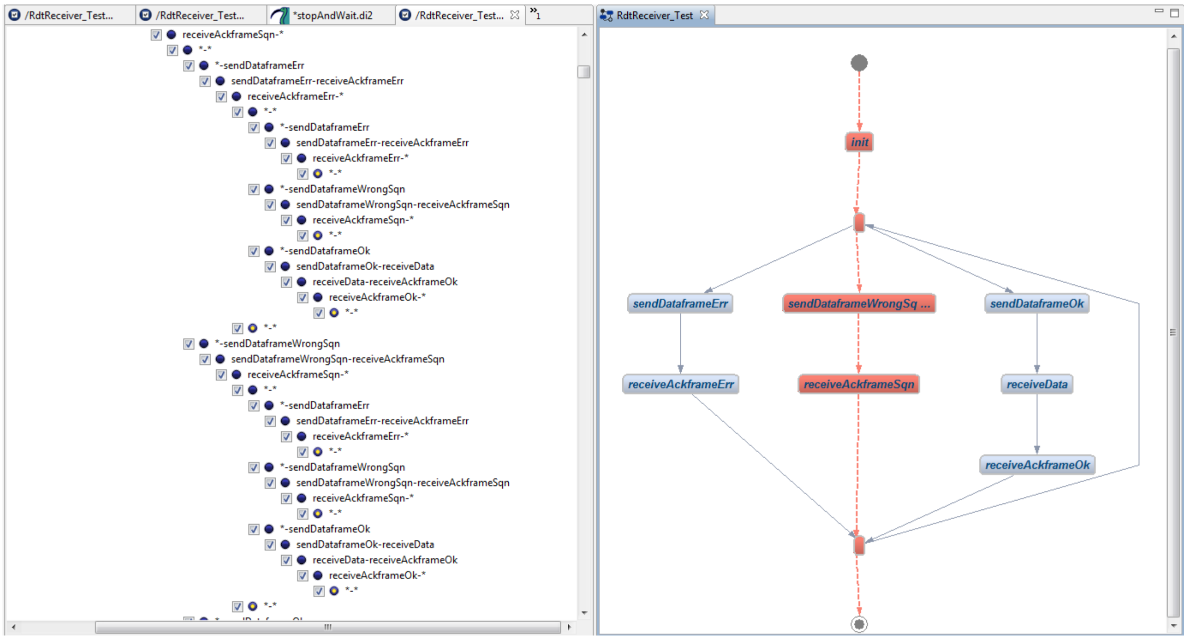


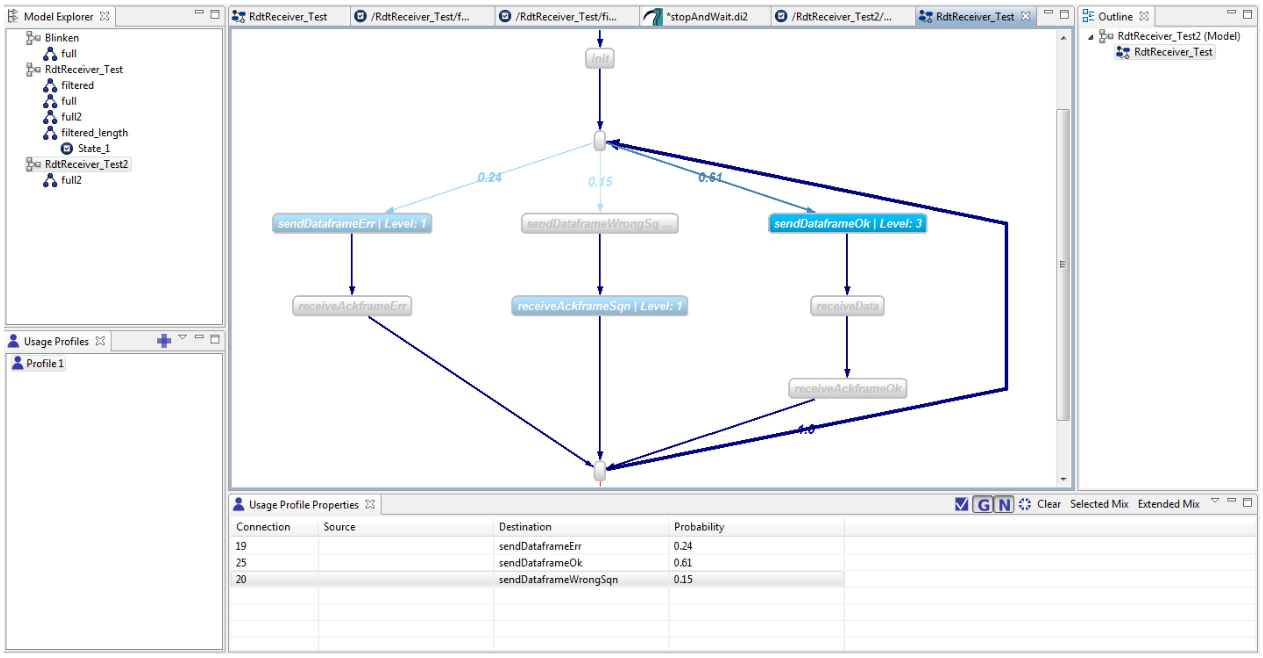
Figure 4: Test case visualization as highlighted paths

Model-driven testing may use models that contain information from different sources and views included by requirement engineers, software architects, test engineers and so on. For that reason it is often cumbersome to find and represent the relevant data. In VeriTAS the visualization component provides an extension point to add user-defined perspectives and to display only those parts that are needed for a given engineering aspect. The user can define specific figures, colors, fonts for diagram elements to highlight relevant parts in the model. Also new views and buttons may be added to the user-defined perspective.

For models that are imported without any graph positioning information, the visualization component provides some default graph layout algorithms. For instance the tree layout, the grid layout and the spring layout of the visualization framework Zest can be used. In addition, the visualization component offers an extension point for user-defined layout algorithms realized as Eclipse plug-in. Because Zest layout algorithms fail in some important visualization criteria like bend points, length of the connections and overlapping areas VeriTAS provides its own configurable layer-based layout algorithm (Purchase 1997).

3.3 Statistical Test Case Generation

Usage models act as a source for many different generation algorithms to derive a set of test cases. Well known techniques, such as coverage algorithms, are already supported by various test case generation tools like .getmore. Because exhaustive testing is not practicable and too expensive for complex systems, it is reasonable to make use of advanced generation methods, in particular in early phases of the software development. In our approach we focus on the statistical test case generation based on Markov chain usage models. VeriTAS contributes to this approach by providing a Markov perspective that enables the provision of usage profiles with an integrated consistency check. Sometimes, the exact profile definition is not possible or requires an additional effort (Dulz, Holpp, and German 2010). To take the profit of the statistical generation methods, the so-called *focus method* has been developed for VeriTAS. Therewith, one is capable to define special regions of interest, which are called *focus levels* without the exact specification of probabilities for transitions in the focus level (Figure 5). The exact values are calculated automatically afterwards for each branch.



Connection	Source	Destination	Probability
19		sendDataframeErr	0.24
25		sendDataframeOk	0.61
20		sendDataframeWrongSqn	0.15

Figure 5: Usage profile definition using the focus method

We assume that the set of all outgoing edges is E. If the uniform distribution at a branch cannot be applied, the probability p_e of an edge e with the *level* l_e and the *weight* of the destination node $w(e)$ is calculated by

$$p_e = \frac{l_e + w(e)}{\sum_{\forall i \in E} (l_i + w(i))}$$

Using N as the set of all reachable nodes from node n and LLF as the so-called *level loss factor*. $D_{n,i}$ is the *minimal distance* from node n to node i. Hence, the weight $w(e)$ can be calculated using the level of the node l_n by

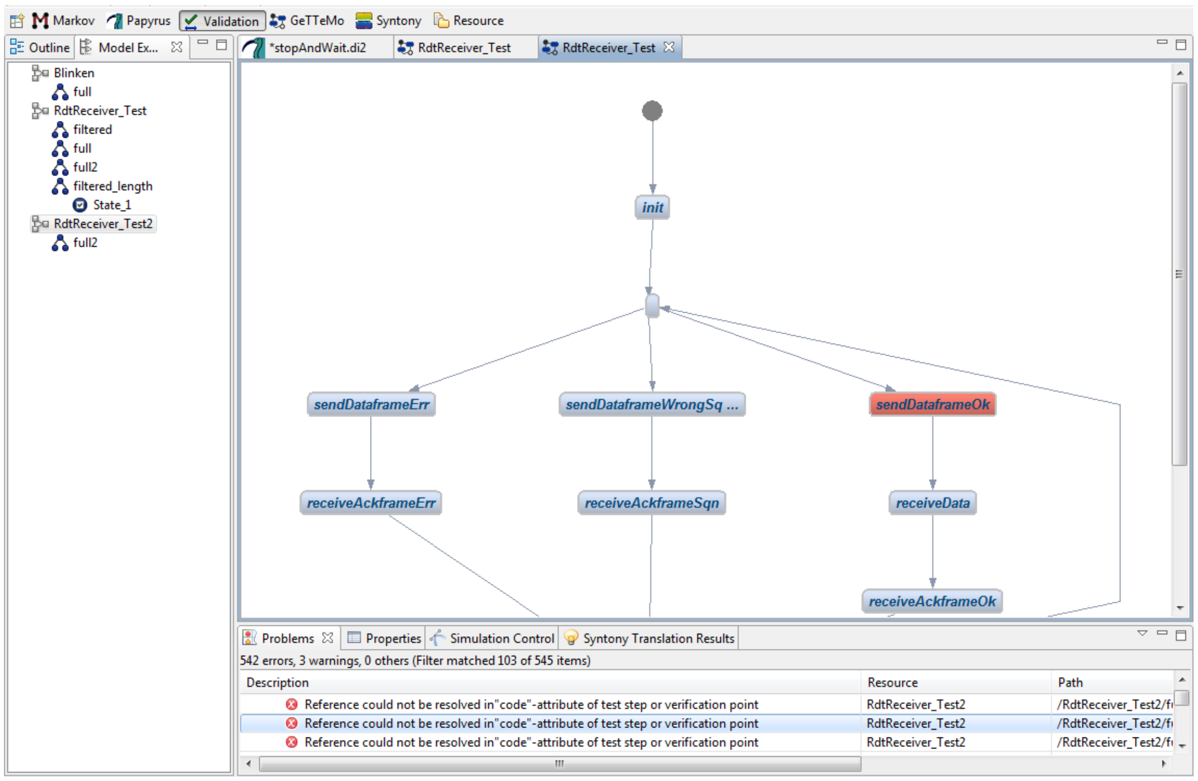
$$w(e) = l_n + \sum_{\forall i \in N} (l_i * (1 - LLF)^{D_{n,i}})$$

The global level loss factor will influence the probability calculation of the edges. A low LLF means a high propagation of the node's focus level to all previous paths. Thus, it results in a higher probability for test cases that contain nodes in focus levels and therefore nodes at focus levels are visited more often in the TAS approach.

4 MODEL VALIDATION USING SYNTONY AND AGILE SIMULATION

In the previous sections we have introduced our TAS approach and described modeling, visualizing and test case generation using the VeriTAS environment. In this section we focus on the validation of involved models by using the simulation framework Syntony (Dietrich et al. 2010), which is also integrated in VeriTAS.

Utilizing the extensibility of the visualization component, VeriTAS offers a perspective to perform static validation checks on models and to highlight the model elements that contain errors (Figure 6). All in all, it is very easy to add new validation constraints by using the Eclipse extension mechanism. To transform abstract test cases previously generated by .getmore into UML-conform interaction diagrams that are used by Syntony, VeriTAS provides an exporter component.



542 errors, 3 warnings, 0 others (Filter matched 103 of 545 items)

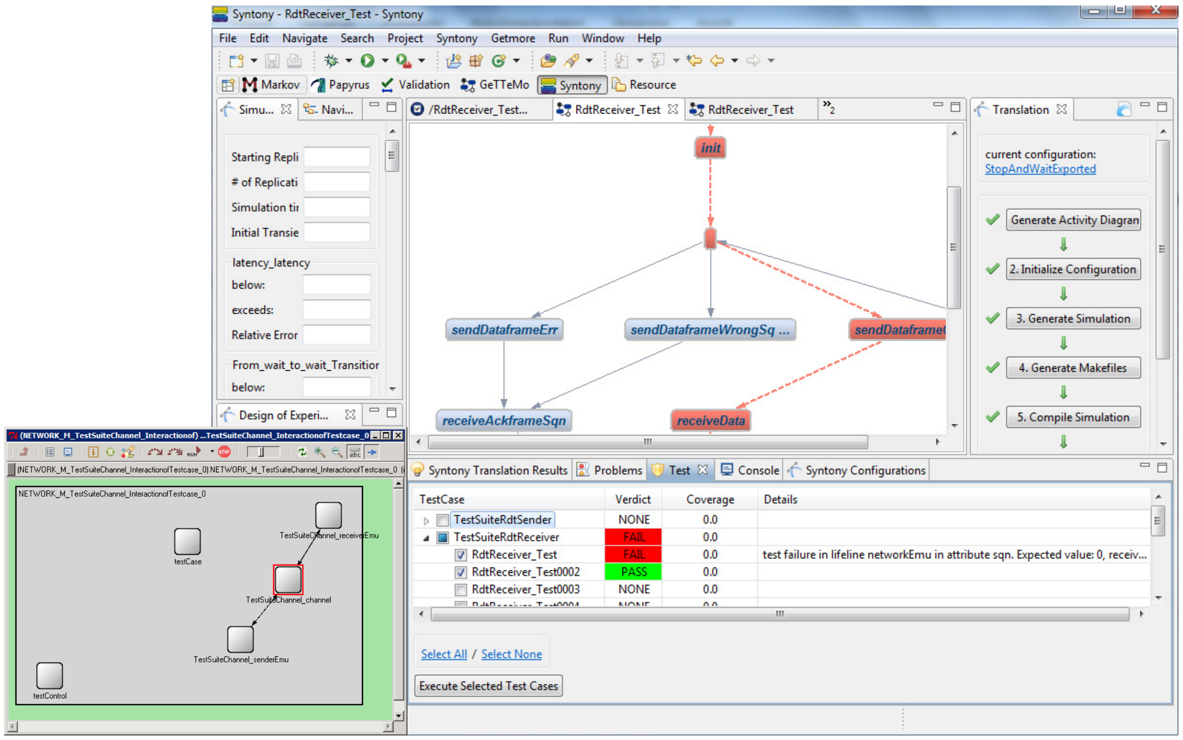
Description	Resource	Path
Reference could not be resolved in"code"-attribute of test step or verification point	RdtReceiver_Test2	/RdtReceiver_Test2/fı
Reference could not be resolved in"code"-attribute of test step or verification point	RdtReceiver_Test2	/RdtReceiver_Test2/fı
Reference could not be resolved in"code"-attribute of test step or verification point	RdtReceiver_Test2	/RdtReceiver_Test2/fı

Figure 6: Validation perspective of VeriTAS

The Syntony framework provides mechanisms to transform UML system models into executable simulation code for the OMNeT++ simulation engine that we are using in our approach. On the right hand of the usage model in Figure 7 the different steps of a Syntony simulation process are depicted:

- In the first step Syntony generates UML activity diagrams from textual *Casual* statements (Dietrich et al. 2010), which are used to express simplified and more user-friendly activity definitions.
- After that, configuration parameters are initialized that are necessary for transforming UML models represented in the XMI model interchange format.
- During the transformation step all elements of the UML system model are translated to C++ code needed for generating an executable OMNeT++ simulation model.
- Next, make files are created in order to compile the generated C++ code.

- After transforming and compiling the UML input model, Syntony displays the contained test cases in its test view. In this way, Syntony lets the user select and execute test cases in combination with a simulation model.
- In the test view test verdicts are displayed after the simulation (Figure 7). Green or red colors are used to visualize, whether a test case will result in the verdict PASS or FAIL during the simulation. In addition, the coverage column indicates the percentage of nodes that are covered by each of the test cases and the test suite altogether.


Syntony - RdtReceiver_Test - Syntony
File Edit Navigate Search Project Syntony Getmore Run Window Help
Markov Papyrus Validation GeTTeMo Syntony Resource
Starting Repli
# of Replicati
Simulation tir
Initial Transie
latency_latency
below:
exceeds:
Relative Error
From_wait_to_wait_Transitior
Design of Experi...
init
sendDataframeErr
sendDataframeWrongSq ...
sendDataframe
receiveAckframeSqn
receiveData
Translation
current configuration:
StopAndWaitExported
Generate Activity Diagram
2. Initialize Configuration
3. Generate Simulation
4. Generate Makefiles
5. Compile Simulation
Syntony Translation Results Problems Test Console Syntony Configurations
NETWORK_M_TestSuiteChannel_InteractionofTestcase_0
testCase
TestSuiteChannel_receiverEmu
TestSuiteChannel_channel
TestSuiteChannel_senderEmu
testControl
Select All / Select None
Execute Selected Test Cases

TestCase	Verdict	Coverage	Details
TestSuiteRdtSender	NONE	0.0	
TestSuiteRdtReceiver	FAIL	0.0	
RdtReceiver_Test	FAIL	0.0	test failure in lifeline networkEmu in attribute sqn. Expected value: 0, receiv...
RdtReceiver_Test0002	PASS	0.0	
RdtReceiver_Test0003	NONE	0.0	

Figure 7: Test-driven Agile Simulation provided by the VeriTAS tool chain

Syntony also provides a dedicated evaluation view for importing results and for selecting metrics that have to be analyzed and plotted in graphs. This specific task is done in the background by the open source tool R (R Project 2011).

5 CONCLUSION

In this paper we introduced the versatile modeling environment VeriTAS. This approach intends to lower the cost and time barriers between simulating a system specification and testing the implemented SUT. The combination of different, dedicated tools within a common Eclipse-RCP environment provides an integrated support for quality-assurance processes and opens new perspectives for the versatile usage independent of specific application domains. Due to Eclipse it is very easy to substitute specific tools in the VeriTAS tool chain and to provide extensions and additional features with respect to more concrete issues and questions (Dulz 2011).

Looking ahead, different extensions and improvements are planned for VeriTAS, such as calculating performance metrics or determining the real-time capability for a given system specification by means of an agile simulation approach.

REFERENCES

Dietrich, I., F. Dressler, and R. German. 2009. "Syntony: A Framework For Model-Driven Simulation, Analysis, And Test." Presentation during PhD Colloquium at *2009 Winter Simulation Conference*, Austin, TX.

Dietrich I., F. Dressler, W. Dulz, and R. German. 2010. "Validating UML Simulation Models with Model-Level Unit Tests." In *3rd ACM/ICST Int. Conf. on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2010)*, Malaga, Spain.

Dulz, W. 2011. "A Comfortable TestPlayer for Analyzing Statistical Usage Testing Strategies." In *Proc. of the 6th IEEE/ACM International Workshop on Automation of Software Test (AST 2011)*. Waikiki.

Dulz, W., and F. Zhen. 2003. "MaTeLo - Statistical Usage Testing by Annotated Sequence Diagrams, Markov Chains and TTCN-3." In *IEEE Proc. Of Third International Conference on Quality Software (QSIC 2003)*, 336–342.

Dulz, W., S. Holpp, and R. German. 2010. "A Polyhedron Approach to Calculate Probability Distributions for Markov Chain Usage Models." *Electronic Notes in Theoretical Computer Science* 264(3):19–35.

IBM. 2011. "Rational DOORS: A Requirements Management Tool for Systems and Advanced IT Applications." Accessed March 31. http://www-01.ibm.com/software/awdtools/doors.

Papyrus. 2008. "Open Source Tool for Graphical UML2 Modelling." Accessed November 5, 2010. http://www.papyrusuml.org.

Prowell, S. J. 2005. "Using Markov Chain Usage Models to Test Complex Systems." In *Proceedings of the 38th Hawaii International Conference on System Sciences*, 318c–318c.

Purchase, H. C. 1997. "Which Aesthetic has the Greatest Effect on Human Understanding?" In *Proceedings of the 5th International Symposium on Graph Drawing*, 248-261. Springer.

R Project. 2011. "The R Project for Statistical Computing." Accessed March 30. http://www.r-project.org/.

Rosaria, S., and H. Robinson. 2000. "Applying Models in Your Testing Process." *Information and Software Technology* 42:815–824.

sepp.med gmbh. 2011. "The Test Case Generator: .getmore." Accessed March 30. http://www.seppmed.de/produkte/getmore.html.

Sparx Systems Ltd. 2011. "Enterprise Architect 9.1 Full Lifecycle UML Modeling Software." Accessed March 30. http://sparxsystems.eu/.

Walton, G. H., J. H. Poore, and C. J. Trammell. 1995. "Statistical Testing of Software Based on a Usage Model." *Software - Practice and Experience* 25(1):97–108.

Whittaker, J. A., and M. G. Thomason. 1994. "A Markov Chain Model for Statistical Software Testing." *IEEE Transactions on Software Engineering* 20(10):812-824.

AUTHOR BIOGRAPHIES

ANATOLI DJANATLIEV received a diploma in computer science in 2009. Currently he is a research assistant at the Department of Computer Science, University of Erlangen-Nuremberg. His research interests include prospective assessments of health technology by simulation. His email address is anatoli.djanatliev@cs.fau.de.

WINFRIED DULZ received the MSc and PhD degrees from the Department of Computer Sciences, University of Erlangen, in 1979 and 1989, respectively. Since 1981, he has been with the Computer Networks and Communication Systems Group at the University of Erlangen, where he is leading the Design and Testing Group. His research interest is on statistical usage testing based on Markov chain usage models. Application domains have been in the automation, communication, medical and automotive areas. He

was involved in several national and international scientific and industrial research projects and is a member of the ACM. His e-mail is Winfried.Dulz@informatik.uni-erlangen.de.

REINHARD GERMAN received a diploma in computer science in 1991, the PhD degree in 1994, and the habilitation degree in 2000 from the Computer Science Department, Technical University of Berlin. Thereafter, he joined the Department of Computer Science at the University Erlangen-Nuremberg. First, he was an associate professor (system simulation), then he became a full professor in 2004 (computer networks and communication systems), served as head of the department and serves currently as dean of the faculty of engineering. His research interests include performance and dependability analysis of networked systems based on numerical analysis (Markovian and non-Markovian models, approximate analysis of large models as well as network calculus), discrete event simulation, measurements, monitoring and testing. Vehicular communications and autonomous sensor/actor networks constitute major application domains. His e-mail is german@cs.fau.de.

VITALI SCHNEIDER received a diploma in computer science in 2009. Currently he is a research assistant at the Department of Computer Science, University of Erlangen-Nuremberg. His research interests include combination of simulation and testing methodologies.
His e-mail is vitali.schneider@cs.fau.de.