Automatic Execution of Test Cases on UML Models of Embedded Systems

Marco A. Wehrmeister, Gian R. Berkenbrock

Automatic Execution of Test Cases on UML Models of Embedded Systems*

Marco A. Wehrmeister[1] and Gian R. Berkenbrock[2]

[1] Federal University of Technology – Paraná (UTFPR)
Av. Sete de Setembro, 3165, 80230-901 Curitiba, Brazil
wehrmeister@utfpr.edu.br
[2] Santa Catarina State University (UDESC)
Rua Paulo Malschitzki, s/n, 89219-710 Joinville, Brazil
gian@joinville.udesc.br

Abstract. During the design of an embedded system, fixing errors discovered only in later stages is a very expensive activity. In order to decrease such costs, the engineers have to identify and fix the introduced errors as soon as possible. Therefore, it makes sense to facilitate the errors detection during the whole the design cycle, including the initial specification stages. This work proposed a test-based approach to aid the early verification of embedded and real-time systems. The proposed approach applies test cases on the system behavior described in the high-level specifications. A tool to automate the execution of the test cases upon UML models has been created. Its initial goal is to improve the errors detection on the system behavior before the implementation phase, since test cases are based on the system requirements. Test cases are platform independent and describe: runtime scenarios; the behaviors to be tested along with their input; and the expected results. The tool executes automatically each test case, in which the specified behavior is simulated Thereafter, the obtained results are compared with the expected ones, indicating the success or failure of the test case. A case study was performed to validate the proposed approach. The achieved results demonstrate that it is feasible to test the system behavior even though when the implementation is still not available.

Keywords: Model-Driven Engineering, UML, testing, test cases execution, simulation

1 Introduction

The design of embedded systems is a very complex task. The engineering team must cope with many distinct requirements and constraints (including timing constraints), whereas the project schedule shrinks due to the time-to-market pressure. Moreover, modern embedded and real-time systems are demanded to

* This work is being supported by National Council for Scientific and Technological Development (CNPq - Brazil) through the grant 480321/2011-6.

deliver an increasing amount of services, affecting directly the complexity of their design. As the system size increases in terms of the number of functions, the number of potential errors or bugs also increases. As a consequence, additional resources (e.g. extra time, money and people) are needed to fix such problems before delivering the final system.

A common approach to deal with the system complexity is to decompose hierarchically a complex problem into smaller sub-problems, increasing the abstraction level [1]. *Model-Driven Engineering* (MDE) [2] has been seen as a suitable approach to cope with design complexity of embedded systems. It advocates that specifications with higher abstraction levels (i.e. models) are the main artifacts of the design. These models are successively refined up to achieve the system implementation using components (hardware and software) available in a target execution platform. Specifically, the engineers specify a *Platform Independent Model* (PIM) that is refined and mapped into a *Platform Specific Model* (PSM), from which source code can be generated automatically. CASE tools helps with such a transformation process.

As one can infer, the quality and correctness of the generated code is directly related with the information provided by the created models and their transformations. Thus, as the system implementation relies strongly on the created models, an undetected error in any model is easily propagated to latter design phases. An error introduced in early design stages and detected only in advanced stages leads to a higher repair cost. The total cost would be lower (from 10 to 100 times lower [3]) if the error had been detected and fixed in the phase in which it was introduced. Taking into account that MDE approaches strongly rely on the PIM and its transformation to a PSM, it is very important to provide techniques to test and verify the produced models. Consequently, the automation of these tasks (e.g. automatic execution of test cases on the system models) is requited.

Aspect-oriented Model-Driven Engineering for Real-Time systems (AMoDE-RT) [4][5] has been successfully applied to design embedded and real-time systems. The engineers specify the system's structure and behavior using UML[3] models annotated with stereotype of the MARTE profile[4]. In [6], AMoDE-RT was extended to include a verification activity in the specification phase, in order to enable the engineers to simulate the execution of the behavior specified in the UML model. However, such an approach was not adequate, since there is a considerable effort to create and run new tests.

This work extends that previous work by proposing the repeatable and automatic verification[5] of embedded and real-time systems based on their high-level specifications. The *Automated Testing for UML* (AT4U) approach proposes the automatic execution of test cases on the behavioral diagrams of an UML model. The set of test cases exercise parts of the system behavior (specified in an UML model) via simulation. The test results can be analyzed to check if the system has

[3] `http://www.omg.org/spec/UML/2.4`

[4] `http://www.omg.org/spec/MARTE/1.1`

[5] In this paper, "verification" means checking the specification using a finite set of test cases to exercise system behavior, instead of the exhaustive and formal verification.

behaved as expected. To support the proposed approach, the AT4U tool executes automatically the set of test cases on the UML model. Each test case describes a runtime scenario (on which the system is exercised) and the behavior selected to be tested. *Framework for UML Model Behavior Simulation* (FUMBeS) [6] simulates the indicated behavior, using the input arguments specified in the test case. The obtained result is compared with the expected result to indicate whether the test case succeeded or not.

It is important to highlight that this automated testing is done already in the specification phase, so that the UML model being created (or its parts) are verified as soon as possible, even if the model is still incomplete. The proposed approach has been validated with a real-world case study. The experiments show encouraging results on the use of AT4U and FUMBeS to simulate and test the system behavior in early design phase, without any concrete implementation.

This paper is organized as follows: section 2 discusses the related work; section 3 provides an overview the AT4U approach; section 4 presents the experiments performed to assess AT4U and their results; and finally, section 5 draws some the conclusions and discusses the future work.

2 Related Work

This section discusses some recent related work regarding Model-Based Test (MBT) and test automation. In [7], the authors discuss MBT and its automation. Several characteristics are evaluated, e.g. quality of MBT versus hand-crafted test in terms of coverage and number of detected failures. Among other conclusions, the authors state that MBT is worth using as it detects from two to six times more requirements errors.

In both [8] and [9], UML models annotated with stereotype of the *UML Testing Profile* (UTP) are used to generate tests using the language TTCN-3(*Testing and Test Control Notation*, version 3) in order to perform black-box testing on software components. The work presented in [8] uses sequence diagrams to generate the TTCN-3 code for the test cases behavior. Additionally, test case configuration code (written in TTCN-3) are generated from composite structure diagrams. Similarly, in [9], TTCN-3 code is generated from sequence and/or activity diagrams decorated with UTP stereotypes. However, that approach uses MDE concepts and created a mapping between the UML/UTP and TTCN-3 meta-model. A TTCN-3 model is obtained from UML model by using a model-to-model transformation.

In [10], MBT is applied in Resource-Constrained Real-Time Embedded Systems (RC-RTES). UML models are annotated with stereotype of the UTP to generate a test framework, comprising: a proxy test model, UTP artifacts (i.e. test drivers and test cases), and a communication interface. The tests cases are executed directly on the RC-RTES, however the produced results are mirrored to the design level. In that approach, the test cases are manually specified as sequence diagrams.

The automated generation of test cases is addressed in [11]. MDE techniques are use to generate test cases from sequence diagrams. The Eclipse Modeling Framework (EMF) was used to create two PIM: the sequence of method calls (SMC) model and the xUnit models. By using the Tefkat engine, a model-to-model transformation translates a SMC model into a xUnit model. Thereafter, a model-to-text transformation combines the xUnit model, test data and code headers to generate test cases for a given target unit testing framework. The experiment conducted in [11] generated test cases for SUnit and JUnit.

Comparing the proposed verification approach with the cited works, the main difference is the possibility to execute the test cases directly on the high-level specification. This work proposes a PIM to represent test cases information likewise [11] and [9]. However, the proposed PIM seems to be more complete since it represents both the test case definitions and the results produced during testing. The test cases are manually specified (as in [10]) in XML files rather than using the UML diagrams as in [10], [8] and [9]. Finally, to the best of our knowledge, AT4U is the first approach that allows the automated execution of test cases on UML model.

3 Automated Testing for UML models

The *Automated Testing for UML* (AT4U) approach have been created to aid the engineers to verify the behavior of embedded and real-time systems in earlier design stages. AMoDE-RT was extended to include a verification activity: the execution of a set of test cases upon the behavior described in the UML model. Therefore, the proposed approach relies on two techniques: (i) the automatic execution of many test cases, including test case scenario setup, execution of selected system behaviors, and the evaluation of the produced results; and (ii) the execution of the specified behavior via simulation.

The proposed approach is based on the ideas presented in the family of code-driven testing frameworks known as xUnit [12]. However, instead of executing the test cases on the system implementation, the test cases are executed on the UML model during the specification phase in an iterative fashion (i.e. a closed loop comprising specification and simulation/testing) until the specification is considered correct by the engineering team. For that, AT4U executes a set of test cases upon both individual elements (i.e. unit test) and groups of dependent elements (i.e. component test) that have been specified in the UML model. Further, as the execution of test cases is performed automatically by a software tool named *AT4U tool*, the testing process can be repeated at every round of changes on the UML model, allowing regression test. If inconsistencies are detected, the UML model can be fixed, and hence, the problem is not propagated to the next stages of the design. An overview of the AT4U approach is depicted in figure 1.

Usually, high-level specifications such as UML models are independent of any implementation technology or execution platform. The use of any platform specific testing technology is not desirable since engineers need to translate the high-level specification into code for the target execution platform before per-

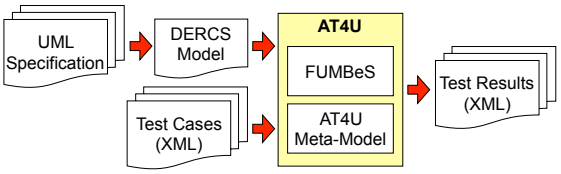

Fig. 1. Overview of AT4U verification approach

forming any kind of automated testing. In this situation, the testing results may be affected by both the errors introduced in the specification and the errors of this translation process.

In order to to allow testing automation for platform independent specifications, AT4U provides: (i) a platform independent description of test cases, and (ii) a mechanism to execute these platform independent test cases. AT4U proposes a *test suite model*, whose meta-model is based on the concepts and ideas of the xUnit framework [12] and the UML Testing profile[6]. AT4U meta-model represents the following information: (i) the test cases used to exercise the system behavior; and (ii) test case results, which are produced during the execution of each test case. For details on AT4U meta-model, see [13]. It is important to note that this *test suite model* is platform independent, and thus, it could be used later in the design cycle to generate test cases in the chosen target platform.

The AT4U tool automates the execution of test cases on high-level specifications. It takes as input a DERCS model[7] (created from the UML model that represents the embedded real-time system under test) and an XML file containing the description of all test cases that must be executed. Once the system model and the test suite model are loaded, the test cases are executed on the model as follows. For each test case, the AT4U performs: (i) the setup of the initial scenario; (ii) the simulation of the selected methods; and (iii) the evaluation of the results obtained from the simulated execution of the methods set.

In the *scenario initialization phase*, the information provided by the AT4U meta-model is used to initialize the runtime state of DERCS' objects. Each object described in the input scenario provides the values to initialize the DERCS's objects, i.e. these values are directly assigned to runtime information of the corresponding attributes.

Thereafter, in the *method testing phase*, AT4U executes the methods specified within the test case, checking if the associated assertions are valid or not. This phase is divided in two parts: (i) method setup and execution; and (ii) the evaluation of the assertions associated with the method under test. Once all input arguments are set, FUMBeS simulates the execution of the behavior asso-

[6] `http://www.omg.org/spec/UTP/1.1/`

[7] *Distributed Embedded Real-time Compact Specification* (DERCS) is a PIM suitable for code generation and simulation. Unfortunately, due to space constraints, a discussion on the reasons that led to the creation and use of DERCS is out-of-scope for this text. Interested readers should refer to [5] and [6].

ciated with the method under test. Each individual actions associated with the simulated behavior is executed and its outcomes eventually modify the runtime state of the DERCS model (for details see [6]).

After the simulation, FUMBeS returns the value produced during the execution of the method under test. Then, *evaluation phase* takes place. The assertions associated with the method under test are evaluated. AT4U tool compares the expected result with the one obtained after the method execution, using the specified comparison operation. In addition, AT4U tool evaluates the assertions related the expected scenario of the whole test case. For that, it compares the expected scenario described in the test case with the scenario obtained after executing the set of methods. Each object of the expected scenario is compared with its counterpart in the DERCS model. If the state of both objects is similar, the next object is evaluated until all objects described in the expected scenario are checked. The test case is marked as successful if all assertions are valid, i.e. those related to each individual method must be valid, along with those related to the whole test case. If any of these assertions is not valid, the test case failed and is marked accordingly.

It is worth noting that an important issue of any automated testing approach is to provide the feedback on the executed test cases. AT4U approach reports the outcomes of the test cases by means of an XML file. This file is generated from the output elements provided by the AT4U model.

The root of the XML document (*TestSuite*) is divided in various nodes, each one reporting the results of one test case. Each *TestCase* node has two subtrees. *Scenario* subtree reports the input, expected and result scenarios. Each of these scenarios reports the snapshot of the objects states in the following moments: *Input* describes the objects before the execution of any method; *Expected* indicates the objects specified in the expected scenario of the test case; and *Result* reveals the objects after the execution of all methods within the test case. *Methods* subtree presents information on the methods exercised during the test case. For each *Method*, the following information is reported: *InputParameters* describes the values used as input to simulate the execution of the method's behavior; *ReturnedValue* indicates the expected and the obtained value returned after the execution of the method; *Scenario* reports the input, expected and result scenarios (the snapshots are taken after the method execution).

However, although a comprehensive amount of data is provided by this report, the XML format is not appropriate for reading by human beings. This XML file could be processed (e.g. using XSLT - eXtensible Stylesheet Language Transformations[8]) to produce another document in a human readable format, such as HTML or RTF, so that the test cases results are better visualized. Such a feature is subject for our future work. However, it is important to highlight that, by generating an XML file containing the results of the execution of each test case, other software tools could process such information analyze the obtained results. For instance, test coverage could be evaluated, or software documentation could be generated based on such an information.

[8] `http://www.w3.org/TR/xslt`

4 AT4U Validation: UAV Case Study

This section presents a case study conducted to validate the AT4U approach. This case study has two main goals. The first one is to check if the proposed approach is practicable, i.e. if the engineers can specify a set of test cases and execute them automatically on the UML model of an embedded and real-time system. The second one is to evaluate the performance of the AT4U tool, in order to assess how much time AT4U tool takes to execute the set of test cases.

The AT4U approach was applied in the movement control system of an Unmanned Aerial Vehicle (UAV) presented in [5]. More precisely, 20 test cases have been created to test parts of the movement control system in 4 different situations: (i) the system operating under normal environmental conditions (4 test cases); (ii) the sensing subsystem is running, but the helicopter is powered down and it lies on the ground (4 test cases); (iii) the helicopter is powered on, but it lies on the ground (7 test cases); (iv) the system operating under hostile environmental conditions (5 test cases).

The set of test cases exercises a total of 31 distinct methods, including simple get/set methods and methods with more complex computations. Some of these methods have been simulated more than once (in different situations), and hence, 143 different simulations have been performed to execute the complete set of test cases. The normal and abnormal values have been chosen as input for the test cases. The complete set of test cases was executed and all assertions were evaluated as true, indicating that all test cases were successful.

Figure 2 shows a small fragment of the report generated by AT4U tool regarding the test case for the method *TemperatureSensorDriver.getValue()*. Lines 102-105 show that the expected and returned values are equal. Hence, the assertion on the behavior of this method was evaluated as true (*assertResult="true"*). The set of test case has been repeated 10 times and their results remained the same. Thereafter, the behavior of the *TemperatureSensorDriver.getValue()* was modified in the UML model, in order to check if its test case would fail. Now, this method returns the value of *sTemperature.Value* plus one instead of returning directly this value. The complete set of test cases was executed again. As expected, this test case failed, since its assertion is not valid anymore.

The first goal of this case study was achieved as various test cases have been specified and executed repeatedly on the high-level specification of the embedded and real-time system. By using AT4U approach, it is possible to perform regression tests. Modifications on the UML model can be automatically evaluated by the test cases previously developed. An error introduced in the system's parts that are covered by the test cases is quickly detected; at least one of these parts is going to eventually fail, as it was illustrated in the previous paragraph.

As mentioned, the second goal of this case study is to evaluate the performance of the AT4U tool. The technologies reused were implemented in Java (i.e. DERCS, FUMBeS, and EMF), and thus, AT4U tool was developed using the JDK 1.6. The experiments have been conducted with a MacBook equipped with an Intel Core 2 Due processor running at 2.16 MHz, 2 GB of RAM, Snow Leop-

```
001 <?xml version="1.0" encoding="UTF-8"?>
002 <TestSuite>
003   <TestCase id="TC-1.3">
004     <Scenario assertResult="true">
   ...
099     </Scenario>
098     <Methods>
099       <Method name="getValue" obj="sTemperature"
100               assertResult="true">
101         <InputParameters></InputParameters>
102         <ReturnedValue assertResult="true">
103           <Expected>20</Expected>
104           <Result>20</Result>
105         </ReturnedValue>
106         <Scenario assertResult="true">
   ...
201         </Scenario>
202       </Method>
203     </Methods>
204   </TestCase>
205 </TestSuite>
```

Fig. 2. XML file reporting test case results

ard as operating system, and Java SE Runtime Environment version 1.26.0_26 (build 1.6.0_26-b03-384-10M3425). To run the experiment, the system was rebooted, all background applications finished, and the experiments executed in a shell session.

The set of test cases was executed 100 times. The complete set was executed in 3712 ms on average, and the average execution time per test case was 185,6 ms (25,96 ms per method). It is important to note that this performance is highly dependent on the tested behaviors. In fact, depending on the complexity of the tested behaviors (e.g. the amount of loop iterations, branches, or executed actions), this execution time can be longer or shorter. Hence, the numbers presented in this case study show only that it is feasible to run test cases to simulate the behavior specified in an UML model.

Intuitively, one may claim that this execution time is longer than the ones required by the native implementations. However, AT4U has been created to allow automated testing for high-level specifications. Considering that there is no implementation available in the specification phase, the AT4U execution time would not be a problem, since engineers do not need to spend time implementing a functionality to verify a solution. Hence, the errors in the specification may be quickly detected without having any implementation or the complete UML model. This performance allows the execution of the whole set of test cases at every round of changes on the UML model to assess if errors were introduced in the specification.

5 Conclusions and Future Work

To decrease the overall cost, it is very important to provide methods and tools to check the created artifacts as soon as possible in the design cycle. The later an error is detected, the higher the cost and effort to fix it [3]. This works proposes

an additional activity in the AMoDE-RT approach. Specifically, the AT4U approach supports the automation of the verification activities of system specifications, more precisely, UML models. The specification phase became an iterative process comprising modeling, model-to-model transformations, and model testing and simulation. By using the AT4U verification approach, engineers specify and execute a set of test case during the creation of the UML model.

AT4U tool supports the proposed approach by automating the execution of the test cases on the UML model. It uses the FUMBeS framework to simulate (parts of) the system behavior. In other words, AT4U executes (the parts of) the embedded and real-time system under controlled and specific situations that have been defined within the test cases. Test cases are represented in a platform independent fashion by means of a test suite model. Engineers write test cases in an XML file which is used to instantiate the mentioned testing model. Similarly, AT4U tool reports the results of the test cases execution in an XML file. This report provides information such as the initial, expected and resulting scenarios used in each test case, as well as the data produced by the executed behaviors and the results of the assertions evaluation.

Although this work is the initial step towards automatic execution of test cases in both models and system implementation, it already presented encouraging results. The proposed approach has been validated with a real-world application of embedded and real-time systems domain. The experiments show indications that AT4U approach is suitable for the purpose of an early assessment of system behavior. Engineers may verify the system specification, and also evaluate different solutions, while the specification is being created in early stages of design. It is worth pointing out that there is no need to implement any part of the system under design to check the suitability of a solution.

Moreover, AT4U enables the regression test of UML model. This helps the engineers to identify whether an error has been included in the specification in the latter refinement steps. Although the benefits already mentioned in this paper, AT4U is not intended to be the unique technique used to verify the system. It shall be used along with other verification methods and tools including code-driven testing automation frameworks, in order to improve the confidence on the design and also to decrease the number of specification errors.

Future work directions include the improvement of AT4U in order to support the use of the UML testing profile to specify the set of test cases, instead of using an XML file Two new tools are needed to support AT4U: one to facilitate the specification of the test cases XML file; and another to facilitate the visualization of the testing report. A test cases generation tool is also important. Such a tool would use the information provided by AT4U PIM to generate the corresponding test cases for a selected target platform. In addition, this tool could create the test cases automatically based on the execution flow of the system behavior. An UML virtual machine using FUMBeS is also envisaged. It should simulate the whole embedded and real-time system, i.e. the execution of its active objects and their concurrent execution, respecting the time constraints.

References

1. Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Communications of the ACM **15**(12) (December 1972) 1053–1058 DOI: 10.1145/361598.361623.
2. Schmidt, D.C.: Guest editor's introduction: Model-driven engineering. Computer **39**(2) (feb. 2006) 25 – 31
3. Broekman, B.M., Notenboom, E.: Testing Embedded Software. Addison-Wesley, Boston, USA (2002)
4. Wehrmeister, M.A., Freitas, E.P., Pereira, C.E., Wagner, F.R.: An aspect-oriented approach for dealing with non-functional requirements in a model-driven development of distributed embedded real-time systems. In: Proc. of 10th International Symposium on Object Oriented Real-Time Distributed Computing, Washington, IEEE Computer Society (2007) 428–432
5. Wehrmeister, M.A., Pereira, C.E., Rammig, F.: Aspect-oriented model-driven engineering for embedded systems applied to automation systems. IEEE Trans. on Industrial Informatics. (2013) To appear in special issue on *Software Engineering in Factory and Energy Automation*. DOI: 10.1109/TII.2013.2240308.
6. Wehrmeister, M.A., Packer, J.G., Ceron, L.M.: Support for early verification of embedded real-time systems through UML models simulation. SIGOPS Operating Systems Review **46**(1) (January 2012) 73–81 DOI: 10.1145/2146382.2146396.
7. Pretschner, A., et al.: One evaluation of model-based testing and its automation. In: Proc. 27th International Conference on Software Engineering, New York, NY, USA, ACM (2005) 392–401
8. Baker, P., Jervis, C.: Testing UML2.0 models using TTCN-3 and the UML2.0 testing profile. In: Proc. 13th Intl. SDL Forum: Conf. on Design for Dependable Systems. SDL'07, Heidelberg, Springer-Verlag (2007) 86–100
9. Zander, J., Dai, Z.R., Schieferdecker, I., Din, G.: From u2tp models to executable tests with TTCN-3 - an approach to model driven testing. In: Proc. 17th Intl. Conf. on Testing of Communicating Systems, Berlin, Springer-Verlag (2005) 289–303
10. Iyenghar, P., Pulvermueller, E., Westerkamp, C.: Towards model-based test automation for embedded systems using UML and UTP. In: Proc. of IEEE 16th Conference on Emerging Technologies Factory Automation. (sept. 2011) 1 –9
11. Javed, A.Z., Strooper, P.A., Watson, G.N.: Automated generation of test cases using model-driven architecture. In: Proc. of the Intl. Workshop on Automation of Software Test, IEEE Computer Society (2007)
12. Beck, K.: Simple smalltalk testing. In Beck, K., ed.: Kent Beck's Guide to Better Smalltalk. Cambridge University Press, New York, NY, USA (1999) 277–288
13. Wehrmeister, M., Ceron, L., Silva, J.: Early verification of embedded systems: Testing automation for UML models. In: 2012 Brazilian Symposium on Computing System Engineering (SBESC), Porto Alegre, Brazilian Computer Society (nov. 2012) 1 –7