RIGA TECHNICAL UNIVERSITY
Faculty of Computer Science and Information Technology Institute of Applied Computer Science

Jurijs GRIGORJEVS
Computer Systems doctoral programme student

MODEL-DRIVEN TESTING APPROACH FOR EMBEDDED SYSTEMS NON-FUNCTIONAL FEATURES

Doctoral thesis summary

Scientific adviser
Dr. sc. ing., professor
O. NIKIFOROVA

Riga 2012

UDK 004.031.6+004.2](043.2)
Gr 543 m

Grigorjevs J. Model-Driven Testing Approach for Embedded Systems Non-Functional Features Doctoral thesis summary. -R.:RTU, 2012.-21 pp.

Printed in accordance with the decision of the Institute of Applied Computer Systems on November 10, 2011, the minutes No 73.

This work has been partly supported by the European Social Fund within the National Programme "Support for the carrying out doctoral study programmes and post-doctoral researches" project "Support for the development of doctoral studies at Riga Technical University".

ISBN 978-9934-10-281-3

DOCTORAL THESIS SUBMITTED FOR THE DOCTORAL DEGREE OF COMPUTER SYSTEMS AT RIGA TECHNICAL UNIVERSITY

The assertion of the thesis submitted for the doctoral degree of computer systems will take place at an open session on June 11, 2012 in Room 202, Meza 1/3, at Riga Technical University, Faculty of Computer Science and Information Technology, Riga, Latvia.

OFFICIAL OPPONENTS:
Professor, Dr.habil.sc.ing. Leonids Novickis
Riga Technical University

Senior IS auditor, Dr.sc.comp. Martins Gills
Norvik Banka

Associate professor, doc. Ing., CSc. Jiří Kunovský
Brno University of Technology

CONFIRMATION

I confirm that I have developed this thesis submitted for the doctoral degree at Riga Technical University. This thesis has not been submitted for the doctoral degree in any other university.

Jurijs Grigorjevs ……………………………….(Signature)

Date: 04.01.2012

The doctoral thesis has been written in Latvian, and includes Introduction, 4 Parts, Conclusion, 5 Appendices, Bibliography, 37 figures and illustrations, 5 tables, in total 147 pages. Bibliography includes 100 information sources.

GENERAL CHARACTERISTICS OF THE RESEARCH

We regularly interact with a multitude of computer controlled devices of all shapes and sizes, which are being used to perform more and more functions in the household and elsewhere. The number of such devices increases every day. Some of the simplest and most popular ones are the electronic oven, microwave oven, refrigerator, washing machine, various kinds of alert and alarm systems, elevator systems, railway crossings, ATM, ticket processing systems, and many others. These systems can mostly be called small and medium-scale embedded systems. Examples of larger-scale embedded systems include trucks and light motor vehicles, helicopters, ships, certain kinds of military systems, as well as spaceships, nuclear power stations, multitudes of automated robots in the manufacturing and service industries. There also exist embedded systems that basically consist of minimal software and a few hardware tools. This category generally covers simple lighting system solutions, automatic door systems etc. Their primary difference from previous categories of systems is the complexity of software and integrated hardware tools, as well as the conditions of their functionality and the importance of their proper functioning. In this review of embedded systems, the author of the doctoral thesis focused on medium and large-scale embedded systems.

Medium and large-scale embedded systems have similar specific features and functioning peculiarities. This is caused by the fact that they must manage a number of external devices simultaneously, collect and process information from the external environment, operate independently without requiring human intervention, and continue functioning where possible in spite of minor faults and errors. To ensure proper system functioning, the specific features of embedded systems must also be tested for all potential cases. This study investigates the non-functional properties of embedded systems.

Relevance of the thesis

Embedded systems have more non-functional properties, more complex software structure and development process compared to simple systems. In just the past few years, there have been a number of cases where errors in the software and hardware of embedded systems led to human casualties and massive losses. Even though only about 20% of all aircraft accidents are caused by technical issues [ACRO 2011], the resulting figure is very high, and it depends on the software and hardware installed in aircraft. Thus, for instance, the largest recent aircraft accidents caused by technical issues are [LEY 2010] [BEA 2011] [BBC 2009]. Besides aircraft, satellites are regularly affected by similar issues as well [SVO 2011], although unlike aircraft, technical faults account for nearly 100% of the accident statistics, seeing as the satellites are launched into orbits with pre-programmed parameters, without human intervention. Errors in embedded systems are also discovered in smaller systems, such as motorcars [VWREC], coffee machines, cell phones and elsewhere. The task of system testing becomes particularly crucial for development of embedded systems.

To make development of embedded systems reliable, new methodologies, programming and specification languages are introduced, and support tools are created. Automation becomes ever more popular with contemporary software development. Generation of programming code is a well-known process that has been used for many years, but full development-cycle automation is still being researched, and the trend is to replace manual work. Standardized principles of model-driven architecture (MDA) [MDA], the available development environments and tools stimulate automation of the entire software development cycle, including execution of testing tasks, which generally take up around 50% of total development time. One of the tools used by MDA is a Unified Modeling Language (UML) [UML], which provides a testing profile [UTP] to support the testing process; this, in turn, ensures standardized specification of artifacts in the testing process. However, even though the testing profile was standardized in 2005, there are still no generally accepted methods for automating the testing process and generating test cases based on the system model.

Related works

Existing methods for testing embedded systems are incomplete and do not ensure automation of the testing process or correspondence to current trends in software development. The methods are based mostly on general testing standards and techniques. Certain methods are known, for instance, for covering as many nodes and routes as possible, obtaining maximum coverage during automatic generation of test cases [STH 2001] [WU 2007]. This method has a number of advantages, including the authors' emphasis on the necessity of generating test cases automatically. Nonetheless, such coverage cannot ensure verification of the non-functional properties of embedded systems in successful and unsuccessful cases.

Attempts to formalize existing testing methods [AGR 2001] [MAT 1995] [KRS 2002] [ZHI 1999] [BRO 2003] [STI 2008], do not support automatic generation of test cases from the system model, which is one of the tasks addressed as part of the doctoral thesis.

Goal of the thesis

The goal of the doctoral thesis is to offer a testing method based on model transformation principles and available standards that allows automatic generation of test cases for testing the non-functional properties of embedded systems. The method proposed must ensure verification of the non-functional properties of embedded systems based on generally accepted system modeling standards.

Research tasks

To achieve the goal of the thesis, the author has formulated the following tasks:

1. analyze and classify existing testing methods;
2. study the non-functional properties of embedded systems, methods and techniques of testing them;
3. perform the analysis of model transformation principles and evaluate the option of applying them to test case generation;
4. define a testing method to ensure generation of test cases based on UML models;
5. apply the proposed method to testing of non-functional properties of an embedded system and conclude about the possibilities of applying the method.

Research methods

Based on the results of the tasks performed, an understanding is obtained regarding the specific features of testing, automation of the necessary testing methods and contemporary automation methods, as well as analyzing and defining the non-functional requirements of embedded systems and existing methods of their modeling and testing. The thesis specifies the following non-functional properties of embedded systems: timing, asynchronism, synchronization, scheduling, and reliability. Based on model-driven principles and the properties described, a hypothesis is proposed regarding the testing method for generating automatic test cases. The hypothesis determines a concept of the testing method and its application. At the same time, the hypothesis ensures the application of existing MDA standards in presenting, processing and transforming models within the testing method. Application of the XML Metadata Interchange (XMI) and general MDA principles improves the efficiency of the hypothetical method and its universal integration with various development processes. The proposed method is not linked to any specific development tool and supports generally accepted standards. The primary universal characteristics of the method are support of the system model in XMI 2.1 format and storage of resulting test cases in a UML-defined testing profile. The first of these characteristics ensures independence from system model development tools, since existing modeling and MDA support tools, such as [EA] or [EMF], ensure exporting of models in XMI format. In turn, the use of a UML testing profile allows structuring and storage of the result of the transformation in a standardized way. This principle supports the subsequent application of the generated results by various testing management tools that allow importing UML-standardized test data from external sources.

Verification of the hypothesis is performed on a real payment card system using real system models with time constraints. The Card Suite payment card system [TIE] belongs to real-time systems with soft time constraints, allowing the time constraints to be exceeded while at the same time strictly limiting the number of such violations with requirements specified for the system. Exceeding of time constraints in system operation may lead to financial consequences, meaning that observing and verifying them is very important. Considering the specific features of Card Suite, it may be used as the platform for verifying the hypothesis. Validation of the proposed testing method takes place by comparing the automatically generated results to test cases obtained manually, based on the analysis of the model using classical testing techniques.

Novelty of the doctoral thesis

The novelty of this study lies in the automatic method for generating a set of test cases based on MDA principles and existing standards. MDA principles are mostly applied to generating software and rarely to devising test cases, mostly for testing the functional system properties. The author of the doctoral thesis offers a method that allows verification of non-functional requirements based on model transformation principles in an MDA context.

In the thesis, this method is applied and approbated for timing properties using the automated generation of test cases. The example used in method approbation provides generating of test cases in order to verify timing on a unit testing level.

The method proposed as part of the study is based on the application of designed tools that support model transformation and generation of test cases, for which the author has defined a specific tool chain. A component-based design was deliberately used in developing the tools allowing to separate import of models in XMI format into the database, simplified representation of models, and transformation itself. The model simplification stage is necessary to define high-quality, content-based transformation rules. Separation of components allows mutual independence of all components, focusing on specific operations. This principle allows existing tools to be adapted and improved, applying the proposed method for verifying other non-functional properties and ensuring compatibility with development of the UML and XMI standards in the future.

Practical significance of the thesis

Having worked for more than 10 years at SIA Tieto Latvia as a real-time payment card system tester, head of implementation, testing and integration testing units, the author has observed that insufficient testing of non-functional properties may bring an entire system to a standstill and lead to other negative consequences. The results of this study were produced based on the author’s experience with testing of real-time systems and current trends in software development; the proposed method was also successfully applied in verifying a financial message processing scenario from a time constraint viewpoint. The practical application allows making conclusions about the functioning of the method and generation of necessary test cases based on the standard system models.

Author of this thesis participated as performer in the following research and methodological study projects:

1. Research project of Latvian Ministry of Education and Science Nr. 09.1269 “Methods and Models Based on Distributed Artificial Intelligence and Web Technologies for Development of Intelligent Applied Software and Computer System Architecture” (2009-currently).
2. Participation in ESF funded project „Development of Study Module on Model Driven Software Development Technology within the Study Programme “Computer Systems””(Contract Nr. 2007/0080/VPD1/ESF/PIAA/06/APK/3.2.3.2./0008/0007) (2006-2008).
3. Participation in research project of Latvian Ministry of Education and Science and Riga Technical University R7389 „Development of prototype for class diagram generation tool based on two-hemisphere model” (2008).

Research publications and presentations at the scientific conferences

The results of the research are obtained and published in 10 publications in the internationally edited and in 6 locally edited conference proceedings. The results are presented at 12 international and 4 local conferences and one student scientific conference of Riga Technical University. A list of publications is attached in the end of this summary.

The main results of the research were presented at next international conferences:

1. Grigorjevs J. „Model-Driven Testing Approach for Embedded Systems Specifics Verification Based on UML Model Transformation", 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development June 8-11, 2011, Beijing, China
2. Grigorjevs J. „Card Suite Testing Toolbox – product launch case study", 12th Annual Software Testing Conference TAPOST2011, May 26, 2011, Riga, Latvia
3. Grigorjevs J. „Model-Driven Testing Approach Based on UML Sequence Diagram", RTU 51st International Scientific Conference, October, 2010, Riga, Latvia
4. Grigorjevs J. „Several practical approaches in testing automation", 11th Annual Software Testing Conference TAPOST2010, July 6, 2010, Riga, Latvia
5. Grigorjevs J., Nikiforova O. „Several Outlines on Model-Driven Approach for Testing of Embedded Systems", RTU 50th International Scientific Conference, October, 2009, Riga, Latvia
6. Nikiforova O., Pavlova N., Grigorjevs J. „Several Facilities of Class Diagram Generation from Two-Hemisphere Model in the Framework of MDA", 23rd International Symposium on Computer and Information Sciences, ISCIS 2008, 27-29 October, 2008, Istanbul, Turkey
7. Grigorjevs J., Nikiforova O. "Compliance of Popular Modelling Notations to Non-functional Requirements of Embedded Systems", International Scientific Conference Informatics in the Scientific Knowledge, June, 2008, Varna, Bulgaria
8. Grigorjevs J. "Testing of Embedded System's Non-functional Requirements", International Baltic Conference Baltic DB&IS 2008, June 2-5, 2008, Tallinn, Estonia
9. Grigorjevs J., Nikiforova O. "Modelling of Non-Functional Requirements of Embedded Systems", 42nd Spring International Conference MOSIS2008, April 22-24, 2008, Hradec nad Moravici, Czech Republic
10. Grigorjevs J., Nikiforova O. „Features of embedded systems that require specific testing approaches", RTU 48th International Scientific Conference, October, 2007, Riga, Latvia
11. Grigorjevs J., Nikiforova O. „Testing Process Adjustment for Real Time Systems", RTU 47th International Scientific Conference, October, 2006, Riga, Latvia
12. Grigorjevs J., Nikiforova O. „Unit Testing for Real Time Systems", RTU 46th International Scientific Conference, October, 2005, Riga, Latvia

Structure of the thesis

The thesis is structurally divided into 4 chapters. The first two chapters are devoted to an overview of the subject of the study, analysis and formulation of the author's hypothesis. The other chapters describe in detail the method and its approbation on a real-time system, followed by an evaluation of the method.

The first chapter considers the specific features of embedded systems, reviews various types of embedded systems and their common properties. General testing principles are also reviewed with particular attention to specific features of testing of embedded systems. In continuation of the chapter, the author turns to the characteristics of embedded systems and their non-functional properties. Each of these is defined in detail, observing the essence of the property, techniques used for specification, modeling, and testing.

Current trends in software development and model-driven development are described in Chapter 2 of the thesis. This chapter describes the testing process and the process of creating software using MDA tools, as well as puts forward a hypothesis about a new testing method based on MDA transformation principles that might be applicable to testing the non-functional properties of embedded systems. Chapter 3 is devoted to detailed specification and description of the method. The proposed inspection of the method is focused on verifying timing properties, and from a structural standpoint is responsible for modeling its application for testing non-functional properties. An example demonstration of the method provides an insight into the principles on which it operates and allows an initial evaluation.

Chapter 4 is devoted to approbating the method on a real-time payment card system and provides a description of the functionality of the system. Based on the approbation results, the author evaluates the method, describing its drawbacks, advantages and application opportunities.

1. TESTING PRINCIPLES OF EMBEDDED SYSTEMS

Embedded systems include hardware and software components and are designed to perform dedicated functions without human interference [JEN 2011].

Embedded systems are distinguished with their independent functioning and control of connected hardware. This specific nature imposes additional requirements to the testing process and requires special verification of characteristics of embedded systems. Mostly, classical testing methods are applied in verification and validation of embedded systems, and there are only few approaches that consider additional verification of characteristics of such systems. In Chapter 1, the author analyzes and classifies the existing testing methods, defines the non-functional properties of embedded systems, and discusses their modeling and testing techniques.

1.1. Properties of embedded systems

There is a wide variety of embedded systems; however, a common feature for all of them is a built-in interface to external environment and control of connected devices. This also defines additional requirements as to the structure and functional principles of embedded systems. Embedded systems are to ensure the following options [ZIM 2009]:

- Physical coupling – a capability of converting physical phenomena into analog data (ensured by all kinds of sensors).
- Interfaces – a capability of obtaining software-friendly data from analog data.
- Timing – specification of time constraints and their attainment [GRI 2005] [GRI 2006].
- Resource control – a capability of controlling resources.
  - Asynchronism – normally, providing an interrupt processing mechanism that would allow concurrent operation of several interfaces and software parts.
  - Data synchronization – along with interception of interrupts, some other synchronization mechanisms, for example, semaphores must be provided as well.
  - Scheduling – competing processes always appear in multiprocessing systems; therefore, systems must be capable of processing several tasks at the same time, according to the created strategy.
- Reliability – software must be fault tolerant, and systems are to intercept errors of a certain type during program compilation (for example, typing and syntax errors).

All the above properties are not unique for embedded systems specifically, and some of them are characteristic of other systems as well. Thus, for instance, timing is an integral part of real-time systems, data synchronization and scheduling, in turn, are mandatory for many processing systems, whereas means of physical coupling can be encountered in the daily use of a mouse or a keyboard. At the same time, all the above mentioned requirements are mandatory for all embedded systems.

1.2 Testing principles and techniques of embedded systems

Testing is the process of running and analyzing the system with the purpose of making sure that it will be capable of long-term functioning in certain circumstances and performing pre-defined actions. According to the author's opinion, the most comprehensive definitions of testing are as follows:

Testing is the process of executing and investigating a program with the intent of finding defects [MYE 2004] – a general definition of testing.

The process of operating a system or component under specified conditions, making an evaluation of some aspect of the system or component, the results being observed and recorded [COP 2004] – a more formal definition provided by IEEE Standard 610.12-1990.

Conceptually, testing of embedded systems corresponds to a generally accepted testing process. It consists of the following stages [SPI 2007]: planning, specification, execution, recording (for example, documenting of results), verification of whether the scheduled work has been performed, and closure activities (for example, final evaluation of the results).

Chapter 1.2 of the doctoral thesis presents and categorizes a number of testing methods that can be applied in testing of embedded systems. A special emphasis is put on the methods based on model transformation. However, even though the methods described can be used for testing of embedded systems, they mostly focus on the process of testing practical functional requirements and its implementation. Characteristics of embedded systems, their modeling and representation options, as well as generation of a group of test cases of non-functional requirements still remain an unresolved problem in the researchers' field of interest.

The research in the field of testing of embedded systems provided in the subchapter mostly focuses on verification of the functional properties of systems, ensuring faster and wider generation and execution of test cases. In turn, methods for verifying non-functional properties are based on old modeling notations, thus focusing on verification of a particular property, without employing the modern universal modeling options and tools. This statement is also supported by other researchers, who analyzed various methods and approaches to verification of both functional and non-functional properties, and their complexity in embedded systems, such as researchers of embedded system designing process [ACK 2008], authors of the method of implementation of embedded system compilers [KUG 2008], as well as the authors of MARTE framework [PER 2010], and others. The study of the author of this doctoral thesis focuses directly on testing of non-functional properties of a system.

1.3 Properties of embedded systems and their modeling and testing techniques

Non-functional properties of embedded systems include asynchronism, synchronization, timing, scheduling and reliability [GRI 2007]. The properties of asynchronism, synchronization, and timing are implemented in software in both cases – when the operating system is and is not used. While scheduling mostly refers to the functionality of an operating system as one of the mandatory properties. Therefore, nowadays, this property is increasingly not implemented in the embedded system software, being left within the competence of operating systems. Provision of the reliability property is a complicated issue that is still under study. Although there exist certain methods of reliability computation and modeling, this property is still a separate subject of research from the implementation point of view. In order to present verification of non-functional properties of embedded systems in sufficient detail while focusing on the current non-functional properties, the author – within the scope of this thesis – investigates asynchronism, synchronization, and timing as described in detail in Chapter 1.3.

2. REPRESENTATION OF BASIC ELEMENTS OF MODEL-DRIVEN ARCHITECTURE IN TESTING ARTIFCATS

The specific features of embedded systems set increased demands as to testing of non-functional properties. In order to execute a testing task, UML allows creating system models that manifest the essence of each non-functional property in detail. In Chapter 2, the author hypothesizes that principles of model-driven software development can be used for automating the verification of embedded systems.

2.1 Basic principles of model-driven architecture

There are a number of MDA definitions, but, in the author's opinion, the concise and constructive way of revealing the essence of MDA is as follows:

„MDA is an approach of specifying information systems, which separates specification of the functionality from specification of its implementation on a specific technological platform" [MDA].

„The essence of MDA is to create various models on different levels of abstraction and to bind these models later with the purpose of system implementation. Some models exist independently from a software platform, while others are platform-specific. Each model is created using the text and multiple complementary and interconnected diagrams" [MEL 2004].

The principle of model-driven architecture is based on the essence of a model that reflects and specifies functioning of a system and transformation of its models [KLE 2003], [NIK 2007]. Fundamental concepts and principles of MDA are described in Chapter 2.1 of the thesis.

2.2 Application of model-driven architecture principles in testing

In model-driven architecture, the main artifacts are models because they describe both the system business functionality and the internal technical properties, and are sources in the code generating process. In terms of classical testing, test cases are generated using the functional descriptions of a system, which can be realized by way of models in model-driven architecture. This fact indicates that test cases can be generated from models. Such process is called model-based testing or model-driven testing – software testing, where test cases are fully or partially derived from a model that describes some (or all) aspects of the system under test [ENG 2006]. The diversity of the level of abstraction of models ensures and supports a relevant testing variant. Considering the MDA-based and classical software development and testing chain, it becomes obvious that artifacts of the levels of abstraction remain unchanged. A conclusion is made in the thesis about the similarity of the levels of the V testing model to the model-driven development chain. This analogy is one of the aspects taken into consideration upon developing the method.

In order to ensure a unified approach to managing and structuring of testing artifacts, OMG [OMG] have initiated the development of a testing profile that would support model-driven testing. In 2007, the UML testing profile (UTP) version 1.0 was published. It is based on the general UML 2.0 and is currently the latest version. UTP defines a language for designing, visualizing, specifying, analyzing, constructing, and documenting the artifacts of test systems [UTP]. UTP can be used stand alone or in an integrated manner with UML system description in order to handle system and test artifacts together. UTP expands the general UML with such testing-specific concepts like test context, verdict, test case, test behavior, and others. UTP is another aspect on which the author's proposed testing method relies.

2.3 Hypothesis for a possibility of relying the testing of non-functional properties of embedded systems on MDA principles

The UTP standard tools offer using system models to generate test cases for functional properties [ZAN 2009]. In turn, testing of non-functional properties with MDA principles is still an evolving field. In Chapter 2.3, the author of the thesis **hypothesizes** the following:

Testing of a system's compliance with non-functional requirements can be based on general MDA principles of model transformation which, in turn, can be used for automatic generation of test cases for the purpose of verifying the non-functional properties of embedded systems.

Model-driven software design principles are based on model transformation, where new models are generated from the existing models. Transformation among models is ensured by the definition of transformation – a set of transformation rules documented in the form of an unambiguous specification, where the whole model or its part is used for generation of another model in full or in part [KLE 2003]. Transformation rules are documented in a specific transformation definition language. In order to ensure generating of a testing model, a definition of a transformation language is introduced in the thesis, which is focused on testing artifacts generation and would be easy to use while formulating transformation rules.

The automated model transformation can be realized by means of a special tool capable of recognizing and handling objects of the source and destination models, along with application of defined transformation rules to a source model. In the result of transformation, a destination model with appropriate generated objects is derived in line with the notation. The general principle of model-driven software design defines a general scheme for model transformation according to the appropriate conceptual scheme of transformation [KLE 2003]. Figure 1 illustrates the concept of the formulated hypothesis as projected to the general model transformation scheme.

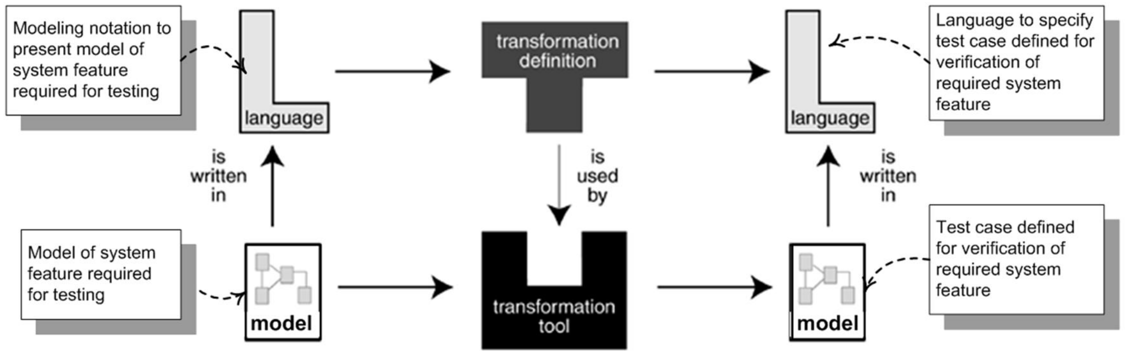

Modeling notation to present model of system feature required for testing
language
transformation definition
language
Language to specify test case defined for verification of required system feature
is written in
is used by
is written in
Model of system feature required for testing
model
transformation tool
model
Test case defined for verification of required system feature

Figure 1. An illustration of the proposed hypothetical solution in the conceptual transformation scheme [GRI 2009]

The model of a respective non-functional property is suggested by the author as the source model, which is generated in the UML sequences and state diagram notation. Thus, the UML language is regarded as modeling notation that specifies the source model. The author suggests considering testing examples as the destination model, which are expressed in the form appropriate for the UML testing profile. Thus, the UML testing profile notation serves as the language for modeling the destination model. Transformation of the source model to the destination model is realized using the transformation definition based on the syntax and semantics of both modeling languages. The principles shown in the figure above reflect the hypothesis stated in the thesis and represent the basic artifacts of the testing method and their interconnections.

3. TESTING METHOD FOR VERIFICATION OF EMBEDDED SYSTEMS' PROPERTIES

In Chapter 3 the author provides a detailed description of the suggested testing method for embedded systems, based on hypothetical statements presented in Chapter 2, and illustrates the principles of applying the method as exemplified by testing of an abstract fragment of an embedded system. The author also analyzes the advantages and drawbacks of the proposed method, and illustrates the effect produced by applying the method in performing a testing task. It is based on the analysis of the set of identified test cases obtained from method application.

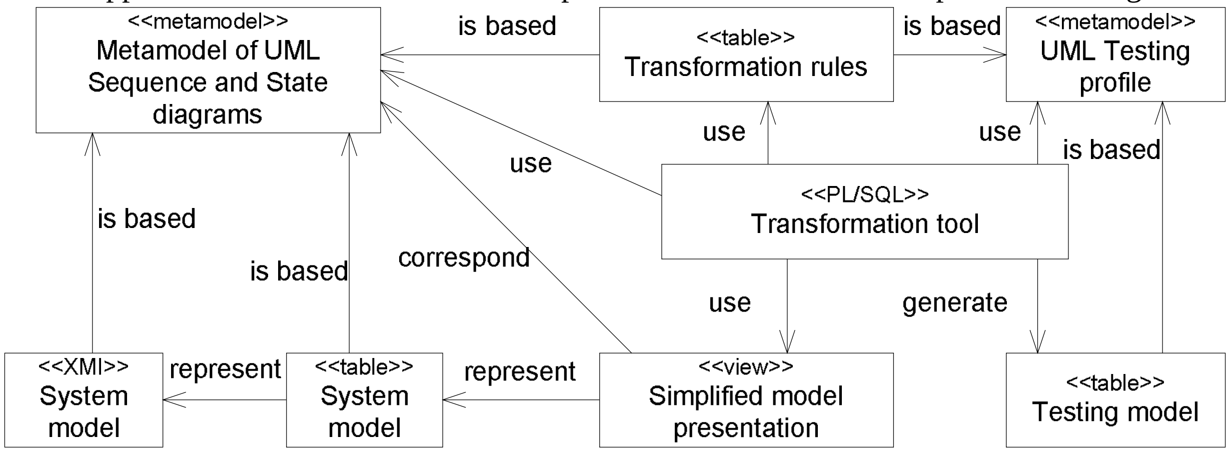
3.1 Analysis of source and destination models

The properties of synchronization, asynchronism, and timing of embedded systems can be modeled with UML standardized diagrams [GRI 2008b] [GRI 2008c]. The UML sequences and state diagrams are chosen as source models. The UML sequences and state diagrams are the adequate modeling notation in order to display the properties of synchronization, asynchronism, and timing [GRI 2008a] [PIL 2005] [UML]. Chapter 3.1 presents the definition of a metamodel of the source model describing the elements of the sequence diagram and their interrelations.

The UML testing profile was selected as the destination notation. UTP provides a black box testing principle [COP 2004] [UTP], when there is no information about the test software design and testing is performed externally by calling various conditions of test objects. Despite the fact that the current research is based on verification of system properties, which, essentially, is the analysis of inner processes and generation of corresponding test cases, UTP is chosen as the suitable metamodel for testing data management, as described in Chapter 3.1.

3.2 Detailed description of the method

The method of test case generation is based on the stated hypothesis for the existence and potential application of such method. A conceptual model of the method is presented in Figure 2.



<<metamodel>> Metamodel of UML Sequence and State diagrams
is based
<<table>> Transformation rules
is based
<<metamodel>> UML Testing profile
use
use
is based
use
<<PL/SQL>> Transformation tool
is based
is based
correspond
use
generate
<<XMI>> System model
represent
<<table>> System model
represent
<<view>> Simplified model presentation
<<table>> Testing model

Figure 2. A conceptual model of the suggested test method [GRI 2011b]

The model depicted in Figure 2 shows the most important concepts of the suggested method and dependencies among them. The implemented method relies on sequences and state diagrams of the UML modeling notation and UML testing profile. Both notations are represented by the

<<metamodel>> stereotype in the diagram in order to illustrate the essence of these concepts and their difference from other concepts. Both metamodels rely on MOF principles and are described in detail in Chapter 3.1.

System models, which act as the initial models of the method, correspond to the UML sequences and state diagram modeling principles. The method provides presentation of system models in XMI format [XMI] (the detailed description of the format is found in Chapter 3.3) shown with a corresponding *<<XMI>>* stereotype in the diagram. The XMI format has wide application as the common-use standard for exchanging of UML models among various supporting tools of the designing process.

The implementation of sequence diagram mapping from XMI format to tables provides a new technical presentation of system models. It means that system models in tables are the same models, in essence and in content, that are available in XMI format. At the same time, it also means that they correspond to the abovementioned metamodel of UML sequences and state diagrams. Similarly to the *<<XMI>>* stereotype, the *<<table>>* stereotype depicts the model presentation format in the diagram.

Testing-oriented representation of system models is the concept of the method that ensures preparation of the handled models for the transformation process. A specific feature of UML model transformation to test cases is in the necessity of analyzing data of multiple related objects in order to generate one test case. The implementation of the method provides for showing UML models transferred to database tables as views, being rendered with a corresponding *<<view>>* stereotype in the diagram. According to this approach, views are a simplified summary of a complex system model, presenting the aspects required for testing. Views are constructed on the basis of system models, encapsulating the required elements and attributes in the demand format. Views introduce no new data and contain solely and exclusively data of original models. Relying on these statements, an evaluation can be made as to the conformity of views to the metamodel of the initial system models.

The transformation tool allows generating a destination model in line with the UML testing profile, namely reading and processing of transformation rules in relation to a simplified system model and generation of a testing model. By implementing the tool, a certain notation of transformation rules is processed, which is described in greater detail in Chapter 3.2.1. The tool as such is realized in PL/SQL programming language with dynamic SQL [DSQL].

A set of transformation rules is a concept, which describes conditions for transforming system models to a testing model. Rules are stored in a separate table with rights to modify and read them during the process of transformation. Transformation rules correspond to the pre-defined notation mentioned above, which, in its turn, is based on the UML sequences and state diagram and the UML testing profile metamodel.

The destination model describes aspects that are required for testing and corresponds to the UML testing profile. Similarly to the presentation of the tables of system models, a testing model consists of a set of interrelated tables describing classifiers specified in the testing profile. A table is prepared for each classifier, where the relevant generated data is recorded during transformation according to transformation rules and the system model.

In order to provide application of transformation rules to the system model and generation of test cases, it has to contain the selection criteria of verified elements and test behavior criteria. The author suggests the notation for defining transformation rules as described in Chapter 3.2.1 of the thesis.

3.3 Principles of method implementation

The method allows implementing transformation in two steps. In the first step, the initial UML models are transformed from the standard XMI file to database tables by means of a dedicated mapping tool. Using the pre-defined views on UML models, a simplified presentation of system models is obtained, which is later applied in transformations. In the second step, transformation takes place by means of reading and processing the simplified UML models with defined transformation rules. As the result of transformation, a testing model with generated test cases is obtained.

3.4 Method application steps

Method application includes several steps, some of which may be omitted depending on the specific nature of a project. A common series of method application steps is shown in Figure 3, which defines the necessary activities of the previously described testing method. The presented scheme shows the required steps, including both manually and automatically executed steps. Automated steps are used to refer to activities, the results of which are prepared by a relevant supporting tool. Manual steps may refer to activities 1, 4, and 7, whereas all the rest activities are provided by standard or custom tools, as well as by tools that were specifically designed within the scope of the current research.

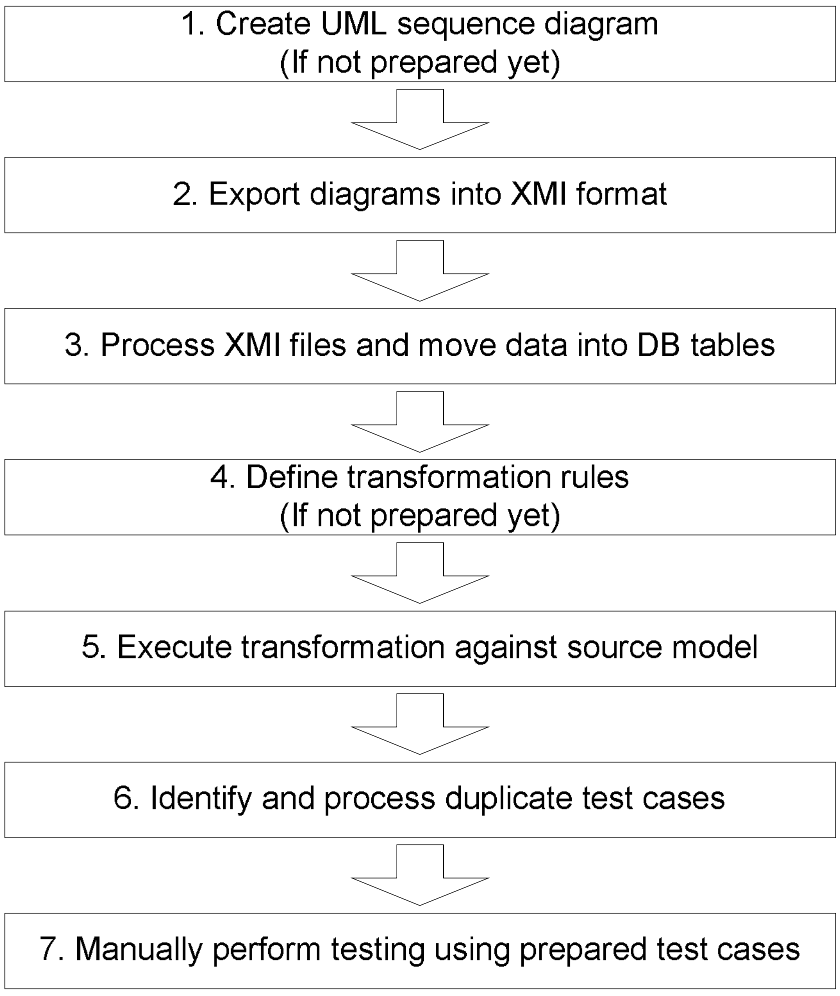

1. Create UML sequence diagram
(If not prepared yet)
2. Export diagrams into XMI format
3. Process XMI files and move data into DB tables
4. Define transformation rules
(If not prepared yet)
5. Execute transformation against source model
6. Identify and process duplicate test cases
7. Manually perform testing using prepared test cases

Figure 3. Execution steps of proposed method

The scheme shown in Figure 3 describes a common sequence of method application steps. A detailed description of method application with technical essentials is provided in the next chapter.

3.5 Method application example

In order to illustrate the presented method, an application example is provided with regard to generation of test cases for abstract software with synchronous calls with time constraints. The solution proposed in the doctoral thesis allows implementing a number of steps that require using a supporting tool of the appropriate activity at each step. Thus, to apply the developed algorithm, a tool chain is defined with its general structure shown in Figure 4.

In order to provide a system model for transformation, method demonstration involves software specification in the form of UML sequences diagram with the Enterprise Architect tool. This commercial tool enables system modeling, model transformations to the code, generating testing scripts, and exporting a developed model in XMI format. With this function applied, the UML sequences diagram created in the tool is exported in XMI format, providing initial data for the described testing method. According to the XMI file processing tool specified in Chapter 3.3.1, a file with INSERT queries is obtained for uploading data to the database tables. After uploading the data, the tables will contain a complete system model in consistence with the processed XMI file as to the content. Afterwards, the uploaded model is represented as a view, where data necessary for transformation is displayed in a simplified and easy-to-handle format.

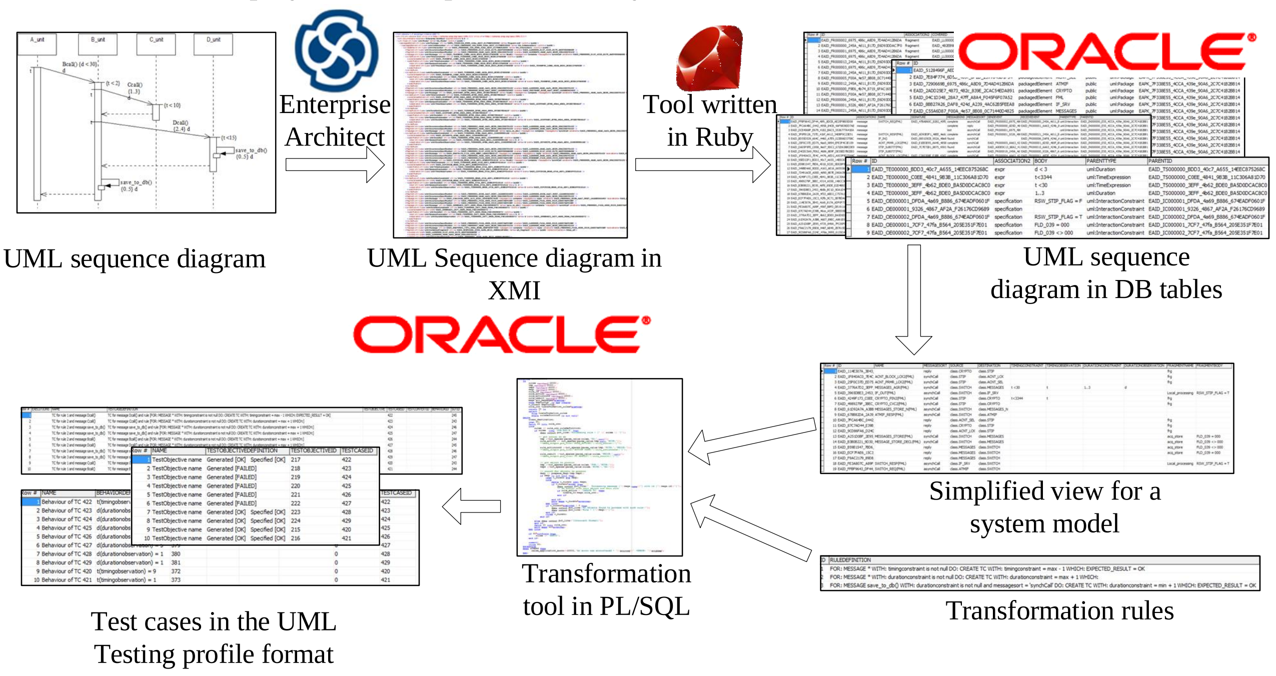

UML sequence diagram
Enterprise Architect
UML Sequence diagram in XMI
Tool written in Ruby
ORACLE
UML sequence diagram in DB tables
ORACLE
Simplified view for a system model
Transformation rules
Transformation tool in PL/SQL
Test cases in the UML Testing profile format

Figure 4. Tool chain of the proposed method [GRI 2011a] [NIK 2008]

By activating the transformation tool with defined and uploaded data, a testing model is generated in compliance with the UML testing profile. Handling of a system model and generation of test cases is performed in line with the principles specified in Chapter 3.

4. APPLICATION OF THE PROPOSED TESTING METHOD FOR REAL-TIME SYSTEM VERIFICATION

In Chapter 4 the author approbates the suggested method and developed tools by testing the timing property of a real-time payment card system Card Suite [TIE]. A basic data flow processing scenario of the specified system, including conditions of several kinds of time constraints, acts as the object of testing. Based on the approbation results received, the author analyzes the advantages and limitations of the method.

4.1 Project overview

The suggested method is used for testing the Card Suite Real-Time Processing System (RTPS) version 8.0 at SIA Tieto Latvia (a reference on method application is provided in Attachment 1). Development and testing of this product version was completed in Q4 2010 and it is currently operated by several customers. The major enhancements of this version are designed to improve the system reliability properties and ensure compliance with the requirements of the Payment Application Data Security Standard (PA DSS) [PADSS].

The author of this thesis was actively involved in the above mentioned project and was responsible for a number of testing activities. The author designed a testing plan for all the RTPS 8.0 testing activities and performed the entire testing, including supervision and monitoring of functionality, system, integration, platform, and performance tests. The author was also responsible for verification of specific cases, except for usability tests. The project covered testing of timing properties of the real-time system on the basis of the method described in Chapter 3 of the current thesis. In view of the fact that the model-driven software development process is only partially implemented at the enterprise and models are used for specifying a small fraction of system functioning, the author of the thesis during the timing properties testing stage designed the UML sequence diagrams for system components within the Sparx Enterprise Architect application software, and applied those in testing henceforth. The designed UML sequence diagrams specify time constraints. If such diagrams were available to the system, additional development of diagrams would not be necessary and the existing UML sequence diagrams with time constraints could be used. When models were ready, the method suggested in the thesis was applied and test cases were generated as a result. Using the generated test cases, the author performed testing of RTPS time constraints. Running the system required simulating erroneous situations that would shut system components down, block Oracle tables and interrupt communication. The author of the thesis executed all the tests in full according to the automatically generated test cases. The results provided detailed verification of system time constraints and revealed incorrect configuration parameters that are set upon the initial installation and could have been delivered together with the payment card system.

4.2 Description of the system under test

The method of testing time constraints was approbated on the real-time payment card system Card Suite. Card Suite is a set of products that provides payment card issuing, acquiring by means of all popular methods (for example, via the Web, ATM (automated teller machine) and POS (point of sale) terminals, and through other devices), real-time authorization processing, and many other supporting operations.

4.3 Real-time authorization processing scenario

Normally, depending on the system configuration and message data, authorization processing involves from 10 to 20 and more separate services. The inner structure of RTPS complies with the component-based architecture; moreover, the Oracle Tuxedo transaction monitor [TUX] is used for managing the whole infrastructure and ensuring the entirety. The system operation utilizes time constraints of four kinds:

- Global Tuxedo transaction time constraint – the maximum allowed time from initiation of a transaction at any of the services till the completion of all operations, which occurs prior to sending a response message to some other service. A transaction may be initiated by one service and completed by the other. Services of global transaction initiation are defined in system configuration. Several global transactions may be initiated within processing of one message.
- Oracle session time constraint – the maximum allowed time for all operations within a session, including saving of all changes. An Oracle session starts with a global transaction initiation and ends with the commit operation.
- Duration constraint for Tuxedo service calls – a time period from initiating a call till transferring control to the called service.
- Time constraint for external system calls – the maximum allowed period of time between sending a request to an external system and receiving a response from it.

The diagram shown in Chapter 4.3 of the thesis illustrates processing of a financial authorization in RTPS services starting with system components that are responsible for providing communication and data transmission to external sources – international networks and various kinds of devices. The diagram presents all four kinds of time constraints mentioned above. UML sequence diagrams presented in the thesis describe authorization message processing scenarios within the most significant RTPS services. Due to the complexity of the system, it is impossible within the scope of the thesis to illustrate the entire processing scenario with all possible services and processing alternatives. The suggested model conforms to the system operation scenario and presents time constraints implemented in the system, which is a key moment in the given research.

4.4 Definition of transformation rules

Transformation rules describe principles of test cases generation. In manual testing such principles are based on extensively applied methods that have become standard testing methods. The following test conditions for verification of time constraints are defined for financial authorization processing described herein:

1. verify the *commit* operation of all services in cases of exceeding and not exceeding time constraints;

2. verify cases of exceeding and not exceeding time constraints in a particular *local* fragment;
3. verify cases when time constraints for calls of *local_processing* fragment are exceeded and not exceeded if *RSW_STIP_FLAG = F*;
4. verify time constraints for all calls from a specific *STIP* server;
5. verify time constraints for all synchronous calls;
6. verify time constraints for all asynchronous calls.

The above test conditions describe possible verification aspects that may be applied together or in any other combination. If these conditions are applied together, duplicate test cases will obviously be obtained as well, because conditions 5 and 6 overlap with previous conditions. Such choice of conditions was made in order to show possible and practically applied verification cases.

Attachment 2 of the thesis contains transformation rules that satisfy the aforesaid test conditions (60 transformation rules fulfill six conditions). The defined transformation rules allow analyzing time and duration constraints, and are focused on generating test cases in accordance with the presented testing methods – equivalence class partitioning and boundary value analysis.

4.5 Transformation process and the obtained model

The transformation tool is operated with defined transformation rules and a system model. As a result of transformation, 180 test cases are obtained as described in Attachment 3. Test conditions specified in the preceding chapter provide for generation of duplicate test cases. The uniqueness of test cases is provided by system behavior of a particular verified case – a set made of *behavior* and *sut* fields. Attachment 4 presents duplicate test cases generated and the number of their recurrence. At the same time, 21 unique test cases are not shown in the table. Exclusively of duplicate test cases, 72 unique test cases were generated during transformation.

4.6 Evaluation of the method

The method presented in the thesis allows defining rules for generating test cases and, by means of the transformation tool, creating test data necessary to verify time constraints. Removal or non-generation of duplicate test cases is deliberately not provided for in implementation of the current tool. The purpose is to generate all cases so that it would be possible to use them as the basis for estimating performance of the tool and analyzing the result of its processing.

To evaluate applied method, the author of current thesis performed manual test case generation using the same verification criteria that were used in transformation rules definition. In a result of manual test case preparation the same 72 unique test case have been prepared and took 5 times more hours. After manual and automated test case implementation author can state that efficiency of proposed method can grow for complex models. Although mentioned number could not be directly used to specify efficiency for all possible systems verifications, the tendency of method application presents one of the most import advantages of it.

The automated generation of test cases is one of the most significant positive aspects of the method. It means that errors in the chain of test case generation due to human factors are reduced. Owing to the pre-defined transformation rules, the system model is analyzed and transformed directly to the testing model without human intervention. The automated generation of test cases provides faster test data acquisition without wasting time on manual test generation. The use of the UML testing profile as the metamodel for testing data allows employing the data also in other systems. Compliance with the OMG standard helps avoiding potential stages that could be necessary for transferring testing data to some test management tool that supports this standard.

The suggested transformation language has two significant advantages. On the one hand, the language is simple and allows focusing on system functionality and testing artifacts, and on the other hand, it enables writing trivial and complicated transformation rules. Thus, for instance, it is possible to write a transformation rule that analyzes not only one particular message, but also searches other messages where the source server is the destination of the current message.

Another advantage of the suggested method is application of UML and XMI standards for defining a source model, which allows for easy integration of the method into various development environments. The UML modeling language is internationally recognized and extensively used. In turn, XMI standard is provided by the majority of modeling tools; it allows preparing an XMI file with a system model for further transformation to a testing model.

The method is realized with a number of tools observing the component-based design. This approach allows improving separate components where appropriate. Implementation of the current tool supports XMI version 2.1, and in case of the requirement of migrating to the latest XMI version due to the introduced changes, then changes will be necessary in Ruby software only, which provides translating of data to database tables. In this case, no changes in the transformation tool are necessary.

The suggested method has also some limitations. Firstly, the method does not allow for automatic operation of the system. In order to provide a universal method and its application for various development environments and systems, the automatic operation of the system is deliberately not realized within the scope of the doctoral thesis as part of the study.

Another limitation of the method is the implementation of the transformation tool within the Oracle DBMS environment, which is a commercial product, and method application may therefore require certain expenses for obtaining Oracle licenses. Oracle was chosen as the most popular DBMS [GAR 2011] that can provide functions required for the method. Development of the transformation tool involved PL/SQL blocks that are supported in the Oracle environment only. In turn, the aforesaid shortcoming cannot be referred to critical disadvantages, because the use of DBMS at software development companies and educational institutions requires no additional licenses, thus not limiting the options for applying the method. At the same time, the component-based design, if necessary, enables rewriting a transformation tool in other technology.

THE MAIN RESULTS OF THE RESEARCH AND CONCLUSIONS

This study is devoted to applying the model-driven software principles to testing of the non-functional properties of embedded systems. As a result of the study, a method was developed for generating testing cases from a system model prepared previously using the UML modeling language. To implement the proposed method based on the existing OMG standards, the author defined a tool chain and implementation of required components that ensures processing of system models and generation of test cases. To test the application of the method, it was approbated by verification of time constraints for a real-time payment card system, for which these non-functional properties are critical for performing standard activities. The following **tasks** were performed in order to achieve the specified goal:

- the existing testing methods were reviewed and analyzed, and their application to testing of embedded systems was considered;
- the non-functional properties of embedded systems and the existing testing methods and modeling techniques were researched;
- the analysis of MDA principles was performed and options of applying them to generation of test cases were evaluated;
- a hypothesis was proposed about the possibility of basing the testing method on MDA principles, ensuring generation of test cases from UML models;
- the hypothesis was tested on the non-functional properties of a real-time payment card system, and conclusions were drawn about the possibilities of applying the method.

*The **main result** of the doctoral thesis is the proposed method for testing the non-functional properties of embedded systems, as well as the set of tools developed to ensure transformation of UML models and generation of test cases. The suggested method is based on the fundamentals of model-driven software development and general principles of model transformation. The tool was applied to testing the timing property of the real-time payment card system.*

Other **important results** of the doctoral thesis are:

1. Information about the non-functional properties of embedded systems was systematized; a description of techniques for modeling and testing the analyzed properties was provided.
2. Based on the fact that classical testing methods are also used to test embedded systems, the general testing process and its methods were reviewed as well.

3. The principles of the model-driven software development process were defined, and its artifacts were described in detail.
4. A model for compatibility between the principles of the model-driven architecture and the general V model of testing was proposed.
5. A model-based testing method was defined which can be applied to testing of non-functional properties.
6. Based on the specific features of testing, a transformation notation was developed which has been applied to defining rules for transforming system models to the testing model.
7. Based on the proposed testing method, tools were developed, including a transformation tool ensuring transformation of UML models to test cases recorded in a format compatible with the UML testing profile.
8. The proposed method and developed tools were approbated by verifying the timing properties of a real-time payment card system.

Based on the research performed and results obtained within the framework of the doctoral thesis, **the following conclusions** were made:

1. Classical testing methods are applied to functional testing of embedded systems, verifying as many cases as possible and achieving the maximum coverage of the software code.
2. Existing methods for testing the non-functional properties of embedded systems are based on modeling notations from earlier generations and on manual generation of test cases.
3. No generally accepted model-driven testing approach exists for verifying functional as well as non-functional properties.
4. Test cases may be presented in a specific notation and obtained from system models automatically by applying the principles of model-driven software development.
5. In order to ensure that transformation rules focus on logical operations and while avoiding descriptions of a complicated structure, the testing method envisages simplified presentation of the initial system model, which is processed using transformation rules.
6. Based on the fact that the proposed method was successfully used for verification of the timing property of a real system, the method may also be applied to other non-functional properties.

The directions of further research may be related to the improvement of the method to ensure fully automated testing of the system, which would include generating test cases, running them and reviewing the results.

The implemented method and tools should be applied to testing of properties like timing, synchronization, and asynchronism. The method may be applied not only to embedded systems, but also to other types of systems which implement the aforementioned properties. The method may be easily integrated into the development process, which already entails specification of system operation using UML models, since it involves no additional steps for preparing initial data, and processes UML models in the standard XMI format.

The author suggests applying the implemented method and tools to testing of other non-functional properties as well, because the implementation of tools relies on a component-based design that ensures their mutual independence, allowing flexibility for improvement if necessary. In addition, the author of the thesis suggests applying the proposed testing method, which is based on simplified presentation of the system model, to testing of functional properties.

BIBLIOGRAPHY

Publications of the author in internationally reviewed scientific papers:

[GRI 2005] Grigorjevs J., Nikiforova O. Testing Process Adjustment for Real Time Systems // In Proceedings of the 46th Scientific Conference of Riga Technical University, 5th Series, Vol. 22, Computer Science, Applied Computer Systems, Riga, Latvia: RTU Publishing. 2005. - pp. 229-241

[GRI 2006] Grigorjevs J., Nikiforova O. Unit Testing for Real Time Systems // Scientific Journal of Riga Technical University, Computer Science, Applied Computer Systems, 5th series, Vol. 26, Riga, Latvia: RTU Publishing. – 2006. – pp. 67-77

[GRI 2007] Grigorjevs J., Nikiforova O. Features of embedded systems that require specific testing approaches // In Proceedings of the 48th Scientific Conference of Riga Technical University, 5th Series, Vol. 30, Applied Computer Systems, Vol. 26, Riga, Latvia. - 2007.- pp. 47-56

[GRI 2008a] Grigorjevs J., Nikiforova O. Modeling of Non-Functional Requirements of Embedded Systems // In Scientific Proceedings of 42nd Spring International Conference MOSIS2008, Ostrava, Czech Republic. – 2008. – pp. 13-20

[GRI 2008b] Grigorjevs J., Nikiforova O. Compliance of Popular Modeling Notations to Non-functional Requirements of Embedded Systems // In Proceedings of the International Scientific Conference Informatics in the Scientific Knowledge 2008, Varna, Bulgaria. – 2008. – pp. 139-149

[GRI 2008c] Grigorjevs J. Testing of Embedded System's Non-functional Requirements // In Scientific Proceedings of the 8th International Baltic Conference Baltic DB&IS 2008, Tallinn, Estonia. – 2008.

[GRI 2009] Grigorjevs J., Nikiforova O. Several Outlines on Model-Driven Approach for Testing of Emb. Systems // Scientific Journal of RTU, 5th series, Computer Science, 38.vol.2009.pp.96-107

[GRI 2011a] Grigorjevs J. Model-Driven Testing Approach for Embedded Systems Specifics Verification Based on UML Model Transformation // In Proceedings of 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development, China. 2011. pp.26-35

[GRI 2011b] Grigorjevs J. Model-Driven Testing Approach Based on UML Sequence Diagram // In Scientific Journal of Riga Technical University, 5th series, Computer Science, Applied Computer Systems, Vol. 47, Riga, Latvia – 2011 – pp. 85-90

[NIK 2008] Nikiforova O., Pavlova N., Grigorjevs J. Several Facilities of Class Diagram Generation from Two-Hemisphere Model in the Framework of MDA //In proceedings of 23rd International Symposium on Computer and Information Sciences, ISCIS 2008.Piscataway,2008. 6p.

Other bibliography used in this summary (all internet sources checked on March 19, 2012):

[ACK 2008] Ackermann C., Cleaveland R., Ray A., Shelton C., Martin C. Integrated Functional and Non-Functional Design Verification for Embedded Software Systems // In proceedings of SAE World Congress & Exhibition, Paper Number 09AE-0202, USA – 2008. / Internet: http://www.cs.umd.edu/Grad/scholarlypapers/papers/Ackermann.pdf

[ACRO 2011] Aircraft Crashes Record Office. Statistics – 2011. / Internet: http://www.baaa-acro.com/Statistiques%20diverses.htm

[AGR 2001] Agrawal S., Bhatt P. Real-time Embedded Software Systems – 2001. / Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.202.2819&rep=rep1&type=pdf

[BBC 2009] BBC. Yemen jet crashes in Indian Ocean – 2009. / Internet: http://news.bbc.co.uk/2/hi/8125664.stm

[BEA 2011] Bureau d'Enquêtes et d'Analyses. Interim report on the accident on 1st June 2009 – 2011. / Internet: http://www.bea.aero/docspa/2009/f-cp090601e1.en/pdf/f-cp090601e1.en.pdf

[BRO 2003] Broekman B., Notenboom E. Testing embedded software. Pearson Education – 2003. – 348 pages.

[COP 2004] Copeland L. A Practitioner's Guide to Software Test Design. Artech House Publishers, London, England. – 2004. – 294 p.

[DSQL] Oracle. Coding Dynamic SQL Statements. / Internet: http://download.oracle.com/docs/cd/B10500_01/appdev.920/a96590/adg09dyn.htm

[EA] Sparx Systems. Enterprise Architect./ Internet: http://www.sparxsystems.com/products/ea/index.html

[EMF] Eclipse Foundation. Eclipse Modeling Framework Project (EMF). / Internet: http://www.eclipse.org/modeling/emf/

[ENG 2006] Engels G., Guldali B., Lohmann M. Towards Model-Driven Unit Testing // In Proceedings of the 9th International Conference on Models in Software Engineering MoDELS'06, Genova, Italy. – 2006. – pp. 182-192. / Internet: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.62.200

[GAR 2011] Gartner. Magic Quadrant for Data Warehouse Database Management Systems. / Internet: http://www.sybase.com/files/White_Papers/Gartner_MagicQuad_forDataWarehouseDMS.pdf

[JEN 2011] Jensen E. D. Real-Time Overview – 2011. / Internet: http://www.real-time.org/realtimeoverview.htm

[JOU 2006] Jouault F., Kurtev I. Transforming Models with ATL // In Proceedings of the MoDELS’06 Conference, Montego Bay, Jamaica. – 2006. – pp. 128-138. / Internet: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.99.6117

[KLE 2003] Kleppe A., Warmer J., Bast W. MDA Explained: The Model Driven Architecture: Practice and Promise. Addison Wesley – 2003. – 167 p.

[KRS 2002] Krstic A., Lai W.-C., Chen L. u.c. Embedded Software-Based Self-Testing For SoC Design // 39th Design Automation Conference, DAC 2002, New Orleans, USA. – 2002. – pp. 355-360. / Internet: http://esdat.ucsd.edu/~lichen/esdat/paper/dac02_krstic.pdf

[KUG 2008] Kugele S., Haberl W., Tautschnig M., Wechs M. Optimizing Automatic Deployment Using Non-Functional Requirement Annotations // In Proceedings of the 3rd International Symphosium ISOLA 2008, Communications in Computer and Information Science, Leveraging Applications of Formal Methods, Verification and Validation, Margaria T., Steffen B. (Eds.), Springer, Greece – 2008. – pp. 400-414

[LEY 2010] Leyden J. Trojan-ridden warning system implicated in Spanair crash – 2010. / Internet: http://www.theregister.co.uk/2010/08/20/spanair_malware/

[MAR 2003] Marschall, F., Braun, P. Model Transformations for the MDA with BOTL // In Proceedings of the Workshop on Model Driven Architecture: Foundations and Applications, Enschede, The Netherlands. – 2003. – pp. 25-36. / Internet: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.7991

[MAT 1995] Mattai J. Real-Time Systems: Specification, Verification, and Analysis. Prentice Hall, PTR Upper Saddle River, New Jersey, USA. – 1995. – 278 p.

[MDA] Object Management Group (OMG). Model Driven Architecture. / Internet: omg.org/mda

[MEL 2004] Mellor S. J., Scott K., Uhl A. u.c. MDA Distilled: Principles of Model-Driven Architecture, Addison Wesley Professional – 2004. – 176 p.

[MYE 2004] Myers G. J. The art of software testing, Second Edition. New Jersey: John Wiley & Sons, Inc. – 2004. – 255 pages.

[NIK 2007] Ņikiforova O., Ņikuļšins V., Buzdins D. u.c. Modeļvadamas programmatūras izstrādes pamati (II daļa), izstrādāts mācību metodiskais materiāls mācību kursam „Programmatūras attīstības tehnoloģijas" datorzinātnes un informācijas tehnoloģijas akadēmiskās bakalaurantūras 3. kursa studentiem , RTU – 2007. – 24 lpp.

[OMG] Object Management Group (OMG). About OMG®. / Internet: http://www.omg.org/gettingstarted/gettingstartedindex.htm

[PADSS] PCI Security Standards Council. Documents Library. PCI Standards Documents. / Internet: www.pcisecuritystandards.org/security_standards/documents.php?association=PA-DSS

[PER 2010] Peraldi-Frati M., Mallet F., Deantoni J. MARTE for time modeling and verification of real-time embedded system // In proceedings of 1st Inter. Colloque. RUNSUD, Sophia Antipolis, France – 2010. – pp. 480-489, Internet: http://www.i3s.unice.fr/~map/map_fichiers_publi/RUNSUD2010.pdf

[PIL 2005] Pilone D., Pitman N. UML 2.0 in a Nutshell. First Edition. O'Reilly Media Inc., Sebastopol, USA. – 2005. – 240 p.

[SPI 2007] Spillner A., Linz T., Schaefer H. Software Testing Foundations: A Study Guide for the Certified Tester Exam, 2nd Edition, Rocky Nook. – 2007. – 288 p.

[STH 2001] Sthamer H., Baresel A., Wegener J. Evolutionary Testing of Embedded Systems // 14th International Internet & Software Quality Week, San Francisco, USA. – 2001. / Internet: http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.86.1798&rep=rep1&type=pdf

[STI 2008] Stiles G. S., Rice D. D., Doupnik J. R. Design, Verification, and Testing of Synchronization and Communication Protocols with Java – 2008. / Internet: http://www.neng.usu.edu/ece/research/rtpc/projects/JavaCSP/Synch_Comms.pdf

[SVO 2011] Свободная Пресса. Неудачные запуски спутников в 2010 году – 2011. / Internet: http://svpressa.ru/world/photo/36421/

[TIE] Tieto. Card Suite. / Internet: http://www.tieto.com/industries/financial-services/transaction-banking/card-suite

[TUX] Oracle. Tuxedo. / Internet: http://www.oracle.com/technetwork/middleware/tuxedo/overview/index.html

[UML] Object Management Group (OMG). Unified Modeling Language™ (UML®). / Internet: http://www.omg.org/spec/UML/

[UTP] Object Management Group (OMG). UML Testing Profile. / Internet: www.omg.org/utp

[VWREC] Volkswagen. Recall Compaigns./ Internet: http://www.volkswagen.co.nz/nz/en/service_and_parts/maintenance_care/mcare_recalls.html

[WU 2007] Wu X., Li J.J., Weiss D.M., Lee Y. Coverage-Based Testing on Embedded Systems // In Proceedings of AST, Minneapolis, USA. – 2007. – pp. 31-36. / Internet: http://arnetminer.org/viewpub.do?pid=2798644

[XMI] Object Management Group (OMG). XML Metadata Interchange (XMI®). / Internet: http://www.omg.org/spec/XMI/

[ZAN 2009] Zander-Nowicka J. Model-based Testing of Real-Time Embedded Systems in the Automotive Domain, doctoral thesis at Technical University Berlin – 2009. / Internet: http://opus.kobv.de/tuberlin/volltexte/2009/2186/pdf/zandernowicka_justyna.pdf

[ZHI 1999] Zhiming L., Mathai J. Specification and Verification of Fault-tolerance, Timing and Scheduling – 1999. / Internet: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.2264

[ZIM 2009] Zimmer U. R. Real-Time and Embedded Systems – 2009. / Internet: http://cs.anu.edu.au/student/comp4330/Level-0/Contents.html