# Environment Modeling in Model-Based Testing: Concepts, Prospects and Research Challenges

## A Systematic Literature Review

Faezeh Siavashi
Åbo Akademi University
Joukahainengatan 3-5
20520 Turku, Finland
faezeh.siavashi@abo.fi

Dragos Truscan
Åbo Akademi University
Joukahainengatan 3-5
20520 Turku, Finland
dragos.truscan@abo.fi

## ABSTRACT

In this paper, we describe a systematic literature review (SLR) on the use of environment models in model-based testing (MBT). By applying selection criteria, we narrowed down the identified studies from two hundred ninety seven papers to sixty one papers which are used in this analysis. The results show that environment models are especially useful in testing systems with high complexity and non-deterministic behaviors in terms of facilitating automatic test generation. However, building environment models is not a trivial task due to the lack of a systematic methodology and of supporting tools for automation.

## Categories and Subject Descriptors

I.6.7 [**Simulation and Modeling**]: Simulation Support Systems—*Environments*; D.2.4 [**Software Engineering** ]: Software/Program Verification—*Model checking*

## Keywords

environment model, software testing, model-based testing, systematic literature review

## 1. INTRODUCTION

Model-Based Testing (MBT) is a black-box testing technique that generates tests from abstract behavioral models [23]. The models can represent either the expected behavior of the system under test (SUT) or of its environment, or in some cases of both. In this context, *abstraction* is beneficial in hiding unnecessary details of the implementation and reducing the complexity of testing. Nevertheless, it is also essential that a test model is *detailed* enough in order to generate effective test cases. Finding the right level of abstraction for the test model is one of the challenges in MBT [23].


*EASE* '15, April 27 - 29, 2015, Nanjing, China

http://dx.doi.org/10.1145/2745802.2745830.

MBT can be used for both online and offline testing. In online testing, the test inputs are generated and executed on-the-fly, whereas in offline testing, test inputs are first generated and later on executed as a batch [22].

In complex computer systems, which operate in environments with large numbers of events and different timings, testing leads to a large number of test cases to cover all possible states of the system. Executing all possible test cases becomes time consuming and unfeasible. Therefore, more advanced methods are required in MBT in order to optimize the number of test cases and reduce the complexity of testing [9].

Environment modeling is an activity that specifies a part of the real world, in which the system is integrated. The process of environment modeling results into an *environment model*, which captures all relevant assumptions and contains all interactions with the SUT [11]. Environment modeling can help addressing the problem of testing complex systems, since one can use environment models to generate automatic test cases for a particular behavior of the SUT.

The main objective of this SLR is to understand how an environment model can enable MBT and what are the current problems and research challenges. In this paper, we do not attempt to compare different approaches, instead, we extract the information as presented by the authors of the available literature in order to present a complete picture of the research done on this topic. To our best of knowledge, this is the first systematic literature review on different approaches in environment modeling.

The remainder of this paper proceeds as follows: in Section 2, we define the research method, provide research questions, and describe the material selection process based on the defined selection criteria. In Section 3, we answer the research questions and present the data analysis from our findings. In Section 4, we discuss validity threats, while in Section 5, we provide a discussion and conclusions.

## 2. RESEARCH METHOD

In this work, we follow the research method suggested by Kitchenham and Brereton [13] for conducting a systematic literature review. However, we describe a summary of the process here, while deferring more details to [20].

**Research questions**: The following research questions are addressed in this paper:

- *RQ1:* What are the characteristics of the environment models used for MBT?
- *RQ2:* What are the advantages of using environment models in MBT?
- *RQ3:* What formalism and tools have been used for creating environment models in MBT?
- *RQ4:* What problems and challenges have been observed by researchers using environment models in MBT?

**Search terms**: First, we selected a set of keywords from the research questions and then defined the search term:

*("environment model" OR "environment behavior model" OR "environmental model" OR "environmental modelling" OR "environment modelling" OR "environment modeling" OR "environmental modeling") AND ("model-based testing" OR "model based testing"OR testing OR test OR "software testing")*

**Sources of studies**: The electronic libraries that we used for searching are: ACM digital library, IEEE Explore, Science Direct, Springer, and Google Scholar. The reason for using Google Scholar is to ensure that we covered all available and relevant papers that are published by miscellaneous publishers or shared in other databases.

**Selection criteria:** A set of inclusion and exclusion criteria has been defined in order to collect relevant studies and filter out irrelevant ones. The inclusion criteria were:

- The objective of the study should be to discuss, apply or investigate the environment model methodologies for the purpose of testing.
- The studies must be written in English.
- The study should be published in a journal or conference proceedings.
- The study should answer at least one of the research questions.
- The study should be published between the years 2000 and 2014 (September).

The exclusion criteria are:

- The studies for which only extended abstracts were available.
- The papers that are about environmental engineering or biological studies or other studies outside the scope of software engineering/testing.
- Master's theses and Doctoral monographs. We assumed that these works have been previously reported and presented as conference or journal publications.

The inclusion and exclusion criteria were applied during the selection process in parallel with reading the full papers.

**Procedure of selecting primary studies**:
*Step 1.* 297 studies were identified by using the search terms in the electronic libraries.
*Step2.* We reviewed *title* and *abstract* of the identified papers and selected 120 studies.
*Step 3.* We read the *content* of the selected studies and applied the selection criteria. In parallel, we made a data extraction form in our Excel spread sheet and recorded details of each study, such as authors, year of publication, etc. In this step, we reduced the number of studies to 63.
*Step 4.* We added all relevant references that we found in 63 papers and applied Steps 1-3 on them (*snowballing* [10]). From the references, we selected 5 more studies, so the total number from this step reached to 68.
*Step 5:* We found that 7 papers were redundant, so we removed them and 61 studies remained.

In this paper, we report the findings that we retrieved by studying 61 studies, which we refer to them as *primary studies.*

For each repository the number of selected papers in each step is shown in Table 1.

**Table 1: The number of selected papers in each repository and in each step of SLR**

| **Database** | **Step1** | **Step2** | **Step3** | **Step4** | **Step5** | **%** |
|---|---|---|---|---|---|---|
| ACM | 33 | 12 | 7 | 8 | 7 | 11% |
| IEEE Explore | 125 | 48 | 25 | 25 | 24 | 39% |
| ScienceDirect | 26 | 7 | 4 | 4 | 4 | 7% |
| Springer | 72 | 36 | 20 | 22 | 18 | 30% |
| Google Scholar* | 41 | 17 | 7 | 9 | 8 | 13% |
| **Total** | 297 | 120 | 63 | 68 | **61** | 100% |

* Only papers that are published in miscellaneous repositories

The last column in Table 1 shows that a large percentage of the publications, 39%, comes from IEEE Explore (24 studies), followed by 30% papers from Springer (18 studies). Google Scholar and ACM have 13% and 11% respectively (8 and 7 studies). Only 7% of the papers are selected from Science Direct (4 studies). Here, Google Scholar has a smaller percentage, since we removed the studies that were originally found in the other databases.

Figure 1 shows the number of primary studies from 2000-2014 by five years interval. It can be noticed that in recent years there has been increased attention towards environment modeling in MBT. This may be due to the growing rate of the complexity of computer systems and applications and subsequently the testing process is becoming more complex. Thus, using environment models as a technique for reducing the complexity is becoming more popular.

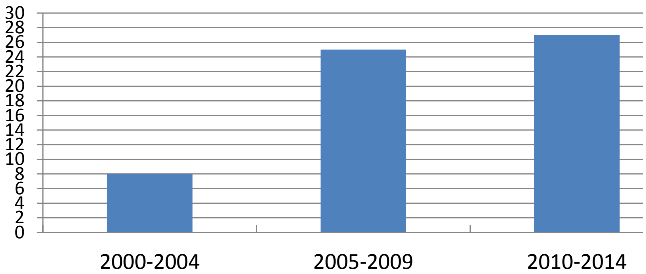


**Figure 1: Number of primary papers by five years intervals**

# 3. RESULTS

In this section, we present how the literature answers the research questions. We start with the information about the characteristics of environment models (RQ1), followed by an overview of the role of environment models in MBT (RQ2). Next, we present a list of modeling formalism, tools and methodologies that are based on environment models (RQ3). Finally, we look at the current limitations and challenges that are reported by the literature (RQ4). Citations of the selected papers are given in this section for further reading.

Out of 61 studies, 95% are empirical studies (i.e., experiments on case studies). 5% were theoretical studies which are based on providing concepts, formal definitions, methodology or references to other work.

Due to the lack of space, we only present a short overview of our findings. More details of the analysis and references to the primary studies are presented in [20].

## 3.1 RQ1 - What are the characteristics of environment models used for MBT?

All primary studies, either implicitly or explicitly, presented general characteristics of using an environment model in their testing approaches. Below, we present them in more detail:

- **Specific aspects of the SUT:** An environment model can be specified in a way that it covers only certain part(s) of the SUT in order to test those parts. Therefore, different parts of a system can be tested separately. Besides, environment models can be defined in a way that they contain different test scenarios to violate specific functionality of the SUT. Twelve out of 61 primary studies fall in this category.

- **Non-determinism:** Non-determinism is an important feature in modeling complex systems with unpredictable environments. It is not a trivial task to model a system which can accept and react to unpredictable conditions. Therefore, using environment models can help in defining the continuous and unpredictable interactions [7]. Non-deterministic environment models give more options to choose among enabled test inputs. We found that 3 primary studies present environment models with non-deterministic behavior.

- **Include multiple entities**: A SUT can have communications with different environment entities such as users, other systems, or a part of the actual environment (i.g. temperature or the sunlight). There are two ways in specifying multiple environments in a test model: it can be defined as a single model containing all environment interactions, or it can be defined as multiple environment components. In both cases, the environment model should capture all assumptions of the environment. Also it should control the interactions among the entities as well. Ten papers presented their work using different environment entities.

- **Dynamic and static behavior:** An environment model is able to support both static and dynamic behaviors of a SUT. Static behaviors mostly indicate what are the inputs of the environment model into the SUT and what type of data and properties are supported by the environment model. In change, dynamic behaviors specify the interactions of the environment model with the SUT, the timing properties and the order of test inputs based on the current outputs during test execution. Six papers used environment models to specify dynamic and static behavior.

- **Abstraction:** A model is an abstract specification of the real world. An abstract model can be defined by restricting the range of input values, omitting some functions or reducing the time span. Environment models can be modeled to only focus on the more abstract interactions. Six papers in our review emphasize on the importance of the abstraction level of the test model when using environment models.

- **Control of time:** Environment models generate timed input traces, which can occur in the real environment, to ensure that the system can satisfy specific timing properties. This characteristics can be beneficial in modeling and testing real-time systems. Nine studies show that environment models can be used for testing systems with timing properties, such as real-time systems.

- **Explicit behaviors:** Having separate models for the SUT and its environment has advantages modifying each of them separately. For instance, when environment models are used in test generation, they typically encode test goals. Whenever test requirements change, only the environment models should be changed. Six papers argue that environment models used in MBT have explicit nature depicting the expected behavior of the system.

- **Source of knowledge for modeling:** The source of knowledge about an environment can come from the requirements, or from the assumptions of the test designers. The requirements are a list of the specifications that a system must follow and need to be tested. When the system specifications are not available, assumptions of the environment can be observed from the actual environment and then formally defined. From the literature, we found that seventeen papers define their environment models from the requirements that are provided in the documentation of the SUT. Also, sixteen papers explicitly claimed that they define their environment via assumptions.

## 3.2 RQ2 - What are the advantages of using environment models in MBT?

Based on the MBT taxonomy illustrated by Utting et al. [23], testing of a system using MBT consists of three main dimensions: modeling, test generation, and test execution. Our findings from the primary studies show that applying environment modeling can be beneficial in all these dimensions. Environment modeling brings the following benefits:

- **Test oracle creation:** In MBT, a test oracle is usually encoded in the test model, and during test generation it is assigned to the generated test cases. In complex systems, in order to reduce the complexity of testing and focus on certain functionality, environment model can be used to model certain test oracles. Three of the primary studies discussed explicitly about test oracle generation using environment models.

- **Automated test generation:** In online testing, it is essential to automatically generate test cases. Automation prevents human errors, which might occur with manual testing, and reduces the time of generating test cases. A *Test harness* (automated testing framework) can be built by a set of test data to automatically run tests and monitor the outputs. Environment model can be used in automation of testing. Twelve papers present that environment models are used to generate test inputs for the SUT during testing.

• **Optimal test generation:** Optimized test case generation is discussed as a benefit of using environment models in testing, making the testing process more efficient. It is caused by having support for abstraction in environment modeling. This advantage is presented in five papers.

• **Reducing the size of the state space:** One of the main issues in executing and simulating complex models (or models with a wide range of inputs) is that the number of symbolic states that should be explored increases during test execution, which causes the system to run out of memory. This problem is known as state space explosion. Reducing the size of the state space can be done by using bounded data types, resetting clock variables, or defining model invariants which limit the enabled states at a given time. Five studies report that well-defined environment models significantly reduce the search space by constraining the ranges of certain test inputs.

• **Early validation of requirements:** Using explicit environment models can be helpful for validating the requirements at the early stage of the system design. Inconsistencies in specifications can be detected when building the models. In addition, they can be used to guide the simulation of early prototypes of the SUT. Two primary studies discuss this issue as an advantage of using environment models.

• **Re-usability:** Different SUTs or different versions (regression) of the same SUT can be tested using a single environment model (see for instance [4]). Generally, environment models will be changed relatively rarely unless some errors originating from requirements are discovered during testing. Therefore, the modeling efforts can be reduced by using the same models in different testing contexts. Five primary studies report this advantage.

• **Different testing types:** Our findings show that environment models can be applied in different testing approaches, such as *safety testing* (5 studies), *robustness testing* (2 studies) and *regression testing* (3 studies). Safety and robustness can be verified by creating erroneous test inputs to the SUT. Moreover, since environment models are able to test certain parts of the SUT, applying them in regression testing can improve the testing effort. Also, environment models have been studied in Aspect Oriented Modeling (AOM) [6] where are known as context models. In AOM, an environment consists of some smaller models, which communicate with each other and with the SUT.

• **Applicable into all testing levels:** The primary studies show that environment models can be applied at all testing levels: *system*, *integration*, and *unit*. The majority of the studies describe applying environment modeling at the system level (43 studies) and few number of them report using environment models at the integration level (3 studies) and the unit level (6 studies).

### 3.3 RQ3 - What formalisms and tools have been used for creating environment models in MBT?

We detected a large range of modeling languages and variety of modeling and testing tools from the primary studies. In this section, we provide the list of the most frequent languages and tools, their references, and briefly discuss some of the most referenced tools.

• **UML:** The majority of the studies use the Unified Modeling Language (UML) [19] as the modeling language. In our review, 20 primary studies are built on UML either by using its standard behavioral diagrams such as sequence and state diagrams, or UML profiles such as Fondue [16], MARTE [1], SysML [8] and MbRTE (Executable model-based robustness testing environment) [24]. The structure of the environment is a model that describes all various entities and their relationships (also known as a domain model in the literature) and consists of one or more environment components. The domain model provides the information on all relationships and properties between the components.

• **Timed-Automata:** Six primary studies present the MBT approaches using Timed Automata (TA) [3]. The tools used with TA are UPPAAL, its an online testing tool (UPPAAL TRON) and Maude. UPPAAL is a model-checker which allows simulation and verification of TA-based specifications. Environment models can be specified in UPPAAL as deterministic or non-deterministic. UPPAAL-TRON is an online testing tool that generates test cases from TA and executes them against the SUT [14]. Maude is a tool based on supporting equational logic and rewriting logic [15]. It represents model generation rules by applying rewriting theories, instead of describing a model directly. It can be applied for modifying the TA model.

• **AEG:** Six primary studies present their experiments on testing with Attributed Event Grammar (AEG) [4], which is used for testing real-time and embedded systems. Event grammars are text-based and are appropriate for specifying the dynamic environment with an arbitrary number of actors and events. Models based on event grammars can be designed either for the environment or for the environment and the SUT. They can also contain hazardous states to assess the safety of the SUT. The environment models can be used to automatically generate test cases.

• **Petri Nets:** Four primary studies are based on Petri Nets, using the TINA tool (TIme Petri Net Analyser) [5]. TINA is a software environment for the editing and analysis of Petri nets and Timed Petri nets. The environment models in TINA have the same properties as the models defined in UPPAAL. Similar, to UPPAAL, the environment model supports both non-deterministic and deterministic assumptions.

• **Lutin:** Three primary studies discuss testing with the Lutin language [18]. Lutin is a test-based language for specifying random reactive behaviors, specially developed for modeling and testing reactive systems. The Lurette test generator is used for random or guided test case generation.

• **BEG:** Two primary studies show how environment models can be designed in the Bandera Environment Generator (BEG) [21], a tool that automates the generation of environments for model-checking Java programs. The tool is able to decompose a given Java program into small modules and create the environment models out of it.

Table 2 shows all formalism and modeling tools that have been used for environment modeling.

Table 2: Formalism and tools for environment modeling

| Formalism/ Languages | Tools | # Studies |
|---|---|---|
| **UML** | UML tools | 10 |
| **UML/MARTE** | | 5 |
| **UML Fondue** | | 2 |
| **UML/SysML** | | 1 |
| **ESML** | | 1 |
| **MbRTE** | | 1 |
| **Timed Automata** | UPPAAL | 4 |
| | UPPAAL TRON | 1 |
| | Maude | 1 |
| **Event Grammar** | AEG | 6 |
| **Petri nets** | TINA | 4 |
| **Lutin** | Lurette | 3 |
| **Java** | BEG | 2 |
| **QR** | QR models | 2 |
| **TSML** | AUTOSAR | 1 |
| **Esterel** | Esterel | 1 |
| **SPIN** | Promela | 1 |
| **TML** | JUMBL | 1 |
| **Markov model** | Markov model | 1 |
| **TTCN-3** | TTCN-3 | 1 |
| **SLAM** | SLAM | 1 |
| **DoB** | Degree-of-Belief(DoB) | 1 |
| **BLAST** | BLAST | 1 |

### 3.4 RQ4 - What problems and challenges have been observed by researchers using environment models in MBT?

We identified several studies that describe problems in MBT using environment models. Also, they identify research areas in MBT for further investigations.

• **Lack of methodology for environment modeling:** Many of the identified studies use environment models for testing, but without discussing explicitly how they are created. Methodological aspects of creating environment models are only discussed in a limited number of papers (e.g., in [9] for UML models). Kishi and Noda emphasize the importance of having a strategy for defining environment model in aspect oriented approaches [12]. Dividing an environment model into several sub-models requires a well-defined methodology as well.

• **Test adaptation is manually implemented:** The studies show that although once an environment model is specified, then the test generation will be automatic. Yet creating the test adapter which can transform the model-level test inputs into executable test cases is manual and error prone process (e.g. in [17]).

• **Multiple test adaptations:** In systems with multiple environment entities, multiple test adaptations are required [2]. The reason is that the interactions among the environment entities as well as interactions between the environments and the SUT are usually complex.

• **Lack of extensive experiments:** The results of our findings show that environment modeling is still immature in some aspects of MBT. For instance, reports have shown that environment models are good choices in robustness testing [24] and regression testing. However, there are very few studies which applied the environment models in practice. Moreover, reusability of environment model can be investigated more and other advantages of using environment models can be studied in more details.

• **Complex specifications:** It is still a challenge to expand the environment modeling in complex systems and for more complicated environments. As it is noted by Auguston et al., more methods are required in order to evaluate environment modeling in large and complex SUTs with large number of test cases automated by the environment [4].

## 4. VALIDITY THREATS

There are four main threats related to our SLR. One is related to studies that we might have missed in our search. Despite the fact that we followed all the steps mentioned in the systematic review process, we cannot be certain that all of the approaches that use environment models in MBT have been identified. Some exclusions were made during reading titles and abstracts, which could have removed studies with relevant content. However, in the second round of the search (snowballing), we made the effort of finding all the studies that were we did not find (or excluded) in the initial round.

Next threat is that there might be some studies that can not be found in any of the selected repositories. We are aware that there are some repositories (e.g. Scopus) that may have more collections of studies. Nevertheless, we converged our search into those repositories to which we could have access and in addition we included Google Scholar to find additional works.

Another threat is that the measurements may not be reliable. This can be caused from lack of reliability in the searching databases, or from the lack of metrics of comparing and selecting the papers. We made all efforts to obtain all published studies that are available in the databases. For each resource, we recorded the details and the information about how and where we searched, in order to make the search repeatable in the future. Moreover, as mentioned in the search and selection process, we searched several different repositories as well as books, conference proceedings and journals, where the most updated works and tools are presented.

Moreover, judgmental errors may have happened during the classification of the papers. We followed the terminologies and classifications that are defined by the literature. Besides, for each classification, we provide the referenced definition, to prevent ambiguity. Based on the quality assessment that we presented in [20], more than 84% of the studies are evaluated as high or very high quality. Thus, the reliability of our measurements can be acknowledged.

## 5. DISCUSSION AND CONCLUSIONS

In this SLR, we defined research questions about environment modeling in MBT. We searched the keywords in different resources based on the defined inclusion and exclusion criteria. Sixty-one primary studies are found answering the research questions and the data are extracted and analyzed.

We identified the main characteristics of an environment model and provided a list of its advantages that are reported in the literature. From the characteristics and advantages, we clarified that using environment models can be helpful in robustness testing, safety testing and regression testing. Also, we showed that in what modeling languages environment models have been studied.

The limitations and current challenges in testing with environment models were summarized as well. The studies report that although the environment modeling helps in the automation of test case generation, yet some case test cases are written manually. Also, the transformation from the symbolic test cases to test scripts is still a manual process.

More research is needed to develop some statistical methods to evaluate and analyze the applicability of environment models in MBT.

From the literature, we clearly conclude that there is still plenty of potential for investigating environment modeling and automating test generation specially w.r.t. non-functional testing approaches. Extensions of the current methodologies are needed to overcome these limitations.

# 6. REFERENCES


[1] The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems. `http://www.omgmarte.org/`, 2013. [Online; accessed 22-December-2014].

[2] N. Adjir, P. De Saqui-Sannes, and K. Rahmouni. Testing Real-Time Systems Using TINA. In M. Núñez, P. Baker, and M. Merayo, editors, *Testing of Software and Communication Systems*, volume 5826 of *LNCS*, pages 1–15. Springer Berlin Heidelberg, 2009.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[4] M. Auguston, J. B. Michael, and M.-T. Shing. Environment behavior models for automation of testing and assessment of system safety . *Information and Software Technology*, 48(10):971 – 980, 2006. Advances in Model-based Testing.

[5] B. Berthomieu and F. Vernadat. Time petri nets analysis with TINA. In *Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on*, pages 123–124. IEEE, 2006.

[6] T. Elrad, O. Aldawud, and A. Bader. Aspect-oriented modeling: Bridging the gap between implementation and design. In *Generative Programming and Component Engineering*, pages 189–201. Springer, 2002.

[7] G. Fraser and F. Wotawa. Test-Case Generation and Coverage Analysis for Nondeterministic Systems Using Model-Checkers. In *International Conference on Software Engineering Advances*, pages 45–45, Aug 2007.

[8] S. Friedenthal, A. Moore, and R. Steiner. *A practical guide to SysML: the systems modeling language*. Elsevier, 2011.

[9] M. Iqbal, A. Arcuri, and L. Briand. Environment Modeling with UML/MARTE to Support Black-Box System Testing for Real-Time Embedded Systems: Methodology and Industrial Case Studies. In D. Petriu, N. Rouquette, and Ø. Haugen, editors, *Model Driven Engineering Languages and Systems*, LNCS, pages 286–300. Springer Berlin Heidelberg, 2010.

[10] S. Jalali and C. Wohlin. Systematic Literature Studies: Database Searches vs. Backward Snowballing. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM, pages 29–38, New York, NY, USA, 2012. ACM.

[11] G. Karsai, S. Neema, and D. Sharp. Model-driven architecture for embedded software: A synopsis and an example. *Science of Computer Programming*, 73(1):26 – 38, 2008. Special Issue on Foundations and Applications of Model Driven Architecture (MDA).

[12] T. Kishi and N. Noda. Aspect-oriented context modeling for embedded systems. *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*, page 69, 2004.

[13] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic Literature Reviews in Software Engineering - A Systematic Literature Review. *Inf. Softw. Technol.*, 51(1):7–15, Jan. 2009.

[14] K. G. Larsen, M. Mikucionis, B. Nielsen, and A. Skou. Testing real-time embedded software using uppaal-tron: An industrial case study. In *Proceedings of the 5th ACM International Conference on Embedded Software*, EMSOFT '05, pages 299–306, New York, NY, USA, 2005. ACM.

[15] G. Li, S. Yuen, and M. Adachi. Environmental simulation of real-time systems with nested interrupts. In *Theoretical Aspects of Software Engineering, 2009. TASE 2009. Third IEEE International Symposium on*, pages 21–28, July 2009.

[16] L. Lucio, L. Pedro, and D. Buchs. A methodology and a framework for model-based testing. In *Rapid Integration of Software Engineering Techniques*, pages 57–70. Springer, 2005.

[17] M. Mews, J. Svacina, and S. Weissleder. From AUTOSAR Models to Co-simulation for MiL-Testing in the Automotive Domain. In *International Conference on Software Testing, Verification and Validation*, pages 519–528, April 2012.

[18] R. Pascal, R. Yvan, and J. Erwan. Lutin: A language for specifying and executing reactive scenarios. *EURASIP Journal on Embedded Systems*, 2008, 2008.

[19] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The*. Pearson Higher Education, 2004.

[20] F. Siavashi and D. Truscan. A systematic literature review on environment modeling techniques in model-based testing. Technical Report 1129, 2015, `http://tucs.fi/publications/view/?pub_id=tSiTr15a`.

[21] O. Tkachuk, M. Dwyer, and C. Pasareanu. Automated environment generation for software model checking. In *International Conference on Automated Software Engineering, 2003. Proceedings.* , pages 116–127, Oct 2003.

[22] M. Utting and B. Legeard. *Practical model-based testing: a tools approach*. Morgan Kaufmann, 2010.

[23] M. Utting, A. Pretschner, and B. Legeard. A Taxonomy of Model-based Testing Approaches. *Softw. Test. Verif. Reliab.*, 22(5):297–312, Aug. 2012.

[24] S. Yang, B. Liu, Shihai, and M. Lu. Model-based robustness testing for avionics-embedded software . *Chinese Journal of Aeronautics*, 26(3):730 – 740, 2013.