A Systematic Literature Review on Environment Modeling Techniques in Model-Based Testing

Siavashi, Faezeh; Truscan, Dragos

Published: 01/01/2015

Document Version
Publisher's PDF, also known as Version of record

Faezeh Siavashi | Dragos Truscan

A Systematic Literature Review on Environment Modeling Techniques in Model-Based Testing

Turku Centre *for* Computer Science

TUCS Technical Report
No 1129, February 2015

A Systematic Literature Review on Environment Modeling Techniques in Model-Based Testing

Faezeh Siavashi
Åbo Akademi University, Department of Computer Science
Joukahaisenkatu 3-5A, 20520 Turku, Finland
faezeh.siavashi@abo.fi
Dragos Truscan
Turku Centre for Computer Science
Joukahaisenkatu 3-5 A, 20520 Turku, Finland
dragos.truscan@abo.fi

TUCS Technical Report
No 1129, February 2015

Abstract

In this paper, we present a systematic literature review (SLR) on the use of environment models in model-based testing (MBT). The objectives of this paper are to identify what are the main characteristics of environment models, to which type of systems and in which application domains they have been applied, as well as what are the benefits and research challenges when using them for testing and test generation. The SLR is performed for publications retrieved from well-known academic publication databases in order to answer a set of research questions. Sixty one publications have been identified and selected as primary studies on which we performed our analysis. The results show that the environment models are especially useful in testing systems with high complexity and nondeterministic behavior, and that they facilitate automatic test generation. However, building environment models is not always easy due to the lack of automation and tool support.

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

1 Introduction

Testing is the process of executing a system with the goal of detecting possible failures. From given specifications of a system, a set of test cases can be defined and executed as test inputs to the system. The test outputs will be compared with the expected outputs. The goal of testing is to obtain the confidence that the system meets its requirements, but also that it does not preform any unwanted behavior [80]. The testing process is an essential part of developing a system. However, it is time consuming and costly. In order to speed up and make testing process more efficient, different techniques and approaches have been developed.

Model-Based Testing (MBT) is a black-box testing technique that generates tests from abstract behavioral models [79]. 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 the *abstraction* is beneficial in hiding unnecessary details of the implementation and reduces complexity in testing. Nevertheless, it is also essential that a test model be *detailed* enough in order to generate effective test cases. Finding the right level of abstraction for the test model is one of the challenges of MBT [80].

In recent years, the complexity of computer systems has increased significantly and as a result testing has become more difficult. Complex systems operate in the environments with large numbers of events and with different timings. Performing a testing procedure in these systems 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 advance methodology in MBT is required in order to optimize the number of test cases and reduce the complexity of testing [44].

One technique to reduce the testing effort in MBT is to use an *environment model*. 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 [54]. Environment modeling can help in dealing with testing complex systems, since one can use environment models to generate automatic test cases for particular behavior of the SUT. In fact, in formal verification and testing, one of the most challenging tasks is the construction of an adequate model of the environment [25].

We searched and studied a number of papers in MBT and found that many of them employed environment models in testing in a variety of approaches. However, we did not find any concrete study or survey which discusses about all the approaches available for environment modeling in the context of MBT. Therefore, we identified the need for a SLR to explain these issues.

The main objective of this SLR is to understand how an environment model can enable MBT, to which types of systems and at which testing levels it has been applied and what are the current problems. To our best of knowledge, this is

the first paper focusing on different approaches in environment modeling using a SLR.

For this work, we have followed the guidelines of conducting a SLR as presented by Kitchenham et al. [58]. We searched the papers reporting experiences of using environment models in their work or studying the applications and tools which automate the environment modeling process. We extract the information as presented by the authors of the available literature.

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 analyses from our findings. In Section 4, we discuss validity threats and in Section 5 we provide a discussion and conclusions.

2 Research method

Many authors proposed methods for performing SLRs [19, 26, 52, 53, 58]. In this work, we follow the approach suggested by Kitchenham and Brereton [58], based on following steps:

- Identify the need for a systematic literature review (discussed in Introduction)
- Define and formulate research questions (Section 2.1)
- Conduct a comprehensive search (Section 2.2)
- Quality assessment (Section 2.3)
- Classify data needed for the research questions
- Extract data from each selected paper
- Summarize the study results (Section 3)
- Interpret applicability of the results (Section 4)
- Write up the results as a report

In the introduction, we clarified the importance of providing a systematic literature review in environment modeling, which is in line with the first step. In the following sections, we show in detail how we applied the other steps and then report the results.

2.1 Research questions

Our SLR addresses the following research questions:

- *RQ1:* What are the characteristics of the environment models used for model-based testing?

- *RQ2:* What are the advantages of using environment models in testing?
- *RQ3:* To which types of systems are the environment models applied?
- *RQ4:* At which testing levels are the environment models applied?
- *RQ5:* What formalism and tools have been used for testing with environment models?
- *RQ6:* What problems and challenges have been observed by researchers using environment models in testing?

2.2 Search and selection process

This process includes defining search terms, selecting databases to be searched, and defining criteria for selection. Each of them is described below.

2.2.1 Search terms

We collected a set of keywords from the research questions for searching the academic publication databases. The keywords from research questions were: ”environment model”, ”model-based testing” and ”testing”. In order to cover all syntactic combinations, we made search strings with syntactic variations of the keyword terms. For instance, one of the search strings is shown below:

(”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”)

For each database, we changed the search string to adjust with the database input formats. For instance, some of the databases do not allow conditional searches (i.e., using Boolean *AND* and *OR* to operators) or nesting searches, therefor we have split the sting and manually input it.

2.2.2 Sources of studies

The following electronic databases/libraries are searched using keywords defined in the section above:

- ACM digital library
- IEEE Explore
- Science Direct
- Springer
- Google Scholar

2.2.3 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 as follows:

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

The exclusion criteria are:

1. The studies for which only extended abstracts were available.
2. The papers about environmental engineering or biological studies or other studies outside the scope of software engineering.
3. Master's theses and Doctoral monographs. We assumed that these works have been 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.

The studies that are selected after the selection procedure are known as *primary studies* and used in the analyses. Different papers that describe the same environment modeling approaches and have same authors are marked as *linked papers* [58]. We include linked papers in primary studies because they investigate their approaches in different testing purposes and the results can be useful for the analyses.

The studies with the same authors and same contribution in different publication databases are marked as *repeated papers*, and we only included the most recent study or the most comprehensive study.

2.2.4 Procedure of selecting primary studies

Figure 1 gives an overview of the search and selection process. In the first round of the search, we found 297 studies from publication database, which we selected 120 studies based on their *titles* and *abstracts*. After reading the *content* of these studies and applying the selection criteria (mentioned in previous section), we collected 63 studies.

In parallel with reading the content, we made a data extraction form in Excel spread sheet to record details of each study. From each primary study we recorded answers of the research questions along with other details such as year of publication, name of authors, etc. We synthesized the data for each research question

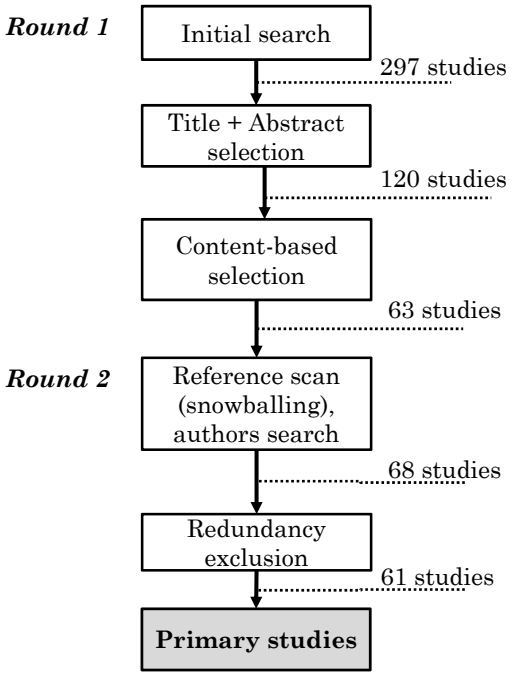

Figure 1: Paper selection process

and categorized the answers. In Section 3 the frequencies of the primary studies for each category are reported.

During data extraction, we checked all related work for each paper and marked the relevant references. The list of the references were used in second round of the selection.

In the second round, we read all the related references and applied selection criteria. This round is also known as *snowballing* [52]. In this round we selected five more primary studies and recorded their details in the data extraction form.

After filling the information of all primary papers, we identified linked papers. These papers help in finding the applicability areas of environment modeling in MBT. Hence, we included them in the primary studies. Ten groups of linked papers are identified and marked in the data extraction form. Moreover, we found that some of the selected studies were redundant (i.e., identical papers that we found from different databases). Seven studies were detected as redundant papers and removed in the last stage of the selection procedure. Finally, we ended up with a list of 61 papers.

Table 1 presents the databases that we used for searching the studies. Each row belongs to one database and each column represents the number of selected papers in each step (similar to Figure 1). The redundancy exclusion column shows negative numbers indicates the number of papers that excluded in the last step.

The last column shows the final numbers of the primary studies from databases.

It should be noted that some of studies were found by Google Scholar whereas they are originally from other databases. Therefore, we removed them from the list of studies retrieved from Google Scholar. As a result, the numbers which is shown in Table 1 is less than the actual number of papers that were found.

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

Database name	**Initial search**	**Title+Abs Selection**	**Content-based selection**	**Reference scan**	**Redundancy exclusion**	**Primary studies**
ACM	33	12	7	8	-1	7
IEEE Explore	125	48	25	25	-1	24
ScienceDirect	26	7	4	4	0	4
Springer	72	36	20	22	-4	18
Google Scholar*	41	17	7	9	-1	8
Total	**297**	**120**	**63**	**68**	**-7**	**61**

* After removing the studies that were published in other databases

Figure 2 shows the number of primary studies from 2000-2014 by five years interval. It indicates that in recent years there has been more attention towards environment modeling in testing. It may be because of the growing rate of the complexity of computer systems and applications and subsequently the testing process is becoming more complex and using environment models as a technique for reducing the complexity is becoming more popular.

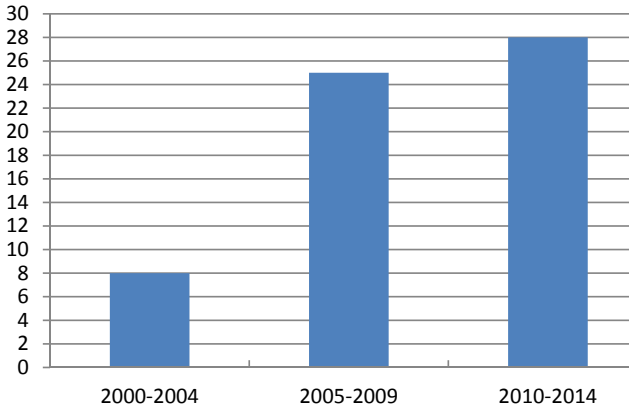

Figure 2: Number of primary papers by five years intervals

2.3 Quality Assessment

We independently assessed quality of each of the 61 primary studies, by a set of questions. We used the quality assessment questions that are developed by Dybå and Dingøsyr [28], with some changes to fit in our study. For instance, we include lessons learnt papers to cover more range of studies.

1. Is the paper based on research?
2. Are the aims of the research understandable by the reader?
3. Is there a clear description of the context?
4. Was the research conducted properly to address the aims of the research?
5. Was there a evaluation process to compare related work or similar methods?
6. Was the data collected in a way that addressed the research issue?
7. Was the data analysis accurate?
8. Is there a clear statement of results?
9. Is the paper applicable in industry?
10. Do the conclusions address the aims of the research?

After measuring the quality of each paper, we calculate the quality scores of the 61 primary studies. The result shows that more than 84% of the studies are evaluated to high and very high quality.

3 Results

In this section, we present how the literature answers the research questions. We describe the results using the criteria we discussed earlier in Section 2.2.3. The information about the characteristics of environment models (RQ1), gives a picture of understanding the role of environment model in testing (RQ2). We then focus more on the types of systems (RQ3) and level of testing (RQ4). Next, we make a list of modeling formalism, tools and methodologies that are based on environment models (RQ5). Finally, we look at the current limitations and challenges that are reported by the literature (RQ6). Citations of the selected papers are given in this section for further reading.

Figure 3 shows that 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.

A large percentage of the publications, 39%, comes from IEEE explore (24 papers), followed by 30% papers from Springer (18 papers). Google Scholar and ACM have 14% and 11% respectively (8 and 7 papers). The reason that Google Scholar has a small percentage is that we removed the studies that are originally

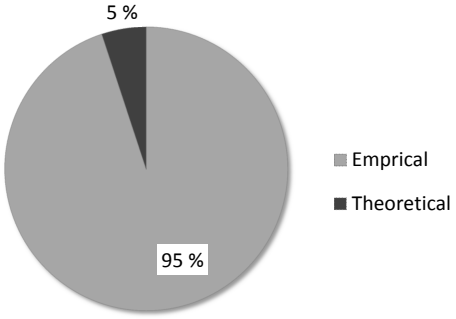

Figure 3: Types of primary studies

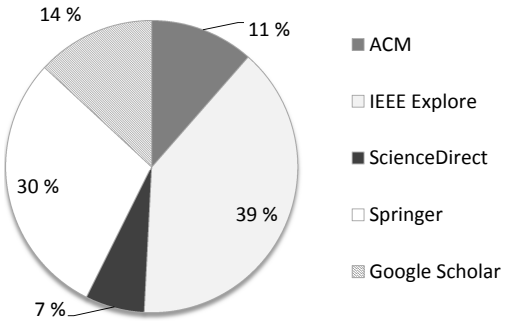

Figure 4: Distribution of the primary studies in databases

from other databases. Only 7% of the papers are selected from Science Direct (4 papers). Figure 4 shows distribution of the primary studies in different electronic resources.

3.1 RQ1 - What are the characteristics of the environment models used for model-based testing?

All primary studies, either implicitly or explicitly, presented general characteristics of using an environment model in their testing approaches. Table 2 lists the main characteristics of an environment model and their frequencies in the referenced primary studies. Below, we present them in more detail:

- *Specific aspects of the SUT:* An environment model can be specified in such a way that only specific part(s) of the SUT will be tested. It can contain a test scenario (test suit) to violate certain behaviors or functionality of the SUT. We found 12 papers that identify environment models for testing certain part(s) of the SUT.

- *Nondeterminism:* Three primary studies present environment models with nondeterministic behavior. Nondeterministic models are especially useful in modeling the environment of systems which have continuous and unpredictable interactions with their environment [29].

- *Include multiple entities*: Ten papers presented their work using different environment entities, such as users, other systems, or a part of the actual environment (i.e. 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.

- *Dynamic and static behavior:* Six papers clarified that environment models can have support of dynamic and static behaviors. 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, whereas dynamic behaviors model the interactions of the environment model with the SUT, the timing properties and the order of test inputs base on the current outputs at the time of testing [86].

- *Abstraction:* As we mentioned earlier, a model is an abstract specification of the real world. Abstraction in environment models can be achieved by ignoring internal interactions of the SUT. An abstract model can be defined by restricting the range of input values, omitting some functions or reducing the time span. Creating an environment model is one of the most challenging tasks in model-checking, because the model should be precise and at the same time abstract [81]. Six papers in our review emphasize on the importance of the abstraction level of the test model when using environment models. Utting et al. point out the different abstraction issues related to MBT [80].

- *Control of time:* Nine papers showed that environment models can be used for testing the systems with timing properties, such as real-time systems. Environment models generate timed input traces, which can occur in the real environment, to ensure that the system can satisfy the specified timing properties.

- *Explicit behaviors:* Having separate models for the SUT and its environment has advantages in 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. But, when the system specifications are not available, the assumptions of the environment can be observed from the actual environment and then formally can be 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. Sixteen papers explicitly claimed that they define their environment via assumptions.

Table 2: Main characteristics of environment models

Characteristics	**References**	#
Specific aspects of the SUT	[2, 14, 24, 27, 33, 36, 43, 50, 60, 63, 73, 77]	12
Non-determinism	[48, 61, 76]	3
Include multiple entities	[2, 15, 32, 39, 43, 46, 55, 57, 69, 76]	10
Dynamic and static behavior	[9, 21, 25, 50, 68, 86]	6
Abstraction	[6, 9, 12, 32, 41, 45]	6
Control of time	[6, 25, 27, 36, 46, 55, 68, 66, 76]	9
Explicit behaviors	[9, 33, 34, 38, 43, 64]	6
Source of knowledge for modeling	Requirements [13, 14, 16, 21, 24, 36, 39, 43, 48, 45, 49, 55, 61, 63, 64, 76, 78]	
	Assumptions [2, 3, 4, 5, 17, 32, 34, 57, 60, 69, 71, 77, 81, 83, 84, 85]	33

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

Based on the MBT taxonomy illustrated by Utting et al. [80], testing of a system using MBT consists of three main dimensions: modeling, test generation and test execution. Based of our findings, 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 [60, 80]. Thus, an environment model is useful in automatically creating test oracles. Three of the primary studies discussed explicitly about test oracle generation using environment models.

- *Automated test generation:* Twelve papers present that environment models are used to generate test inputs for the SUT during testing. It prevents the human errors that might occur with manual testing, and reduces the time of generating test cases. Furthermore, two papers claim that the advantage of automating test generation can lead to a *test harness*, which is a collection of test data used to automatically run the tests and monitor the outputs [24, 56].

- *Optimal test generation:* In five papers, 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 of abstraction in environment modeling.

- *Reducing the size of the state space:* One of the main issues in executing and simulation of complex models (or models with wide range of inputs) is that the number of symbolic states that should be explored increases during the test execution, and thus the memory of the test machine will be fulled. This problem is known as state space explosion. Five studies report that well-defined environment models significantly reduce the search space by constraining the ranges of certain test inputs. Reducing the size of state space can be done by defining a limited range of numbers instead of defining a general data types, resetting the clock variables after passing the corresponding state, or defining model invariants which limit the enabled states at a given time.

- *Early validation of requirements:* Using explicit environment models can be helpful for validating the requirements in the early stage of the system design. They can be used to guide the simulation of early prototypes of the SUT. Two papers explain this as an advantage of using environment models.

- *Re-usability:* Five papers report that with the same environment model, different SUTs or different versions (regression) of the same SUT can be tested. Generally, environment models will be changed relatively rarely unless some errors are discovered from the requirements during testing. Therefore, the modeling efforts can be reduced by using same models in different testing contexts.

Table 3 presents a list of studies, which show how environment models can be useful in testing.

3.3 RQ3 - To which types of systems are environment models applied?

In this section, we aimed to synthesis our findings in which types of systems environment models have been used in the literature.

Table 3: Advantages of using environment models

Advantage	References	# of studies
Test oracle creation	[50, 60, 80]	3
Automated test generation	[9, 11, 14, 15, 24, 27, 41, 56, 60, 71, 73, 78]	12
Optimal test generation	[33, 43, 49, 70, 88]	5
Smaller size of state space	[63, 73, 81, 77, 85]	5
Early validation of requirements	[15, 39]	2
Re-usability	[11, 12, 22, 38, 39]	5

The types of systems can be divided into two categories, the systems containing hardware and software, and systems which are pure software. The first category includes different systems as introduced below:

- **Real-time Systems:** Systems with timing properties and activities, that must be performed within specific timing constraints [67].

- **Reactive systems:** Systems with continuous behavior interacting with their environment. A reactive system receives inputs from the environment and based on its configurations, changes its internal states and sends the outputs as results to the environment [59].

- **Hybrid systems:** Systems which combine two or more different systems. Hybrid systems may preform both continuous and discrete behaviors [40].

- **Embedded systems:** Systems with dedicated design or function for a part of hardware, dealing with the control of physical environment through sensors and actuators [31].

Embedded systems can act as both reactive systems and real-time systems. Thus, we present them in either the domain of real-time systems (for the real-time embedded systems) or reactive systems (for the reactive embedded systems).

The second category is defined as follow:

- **Software systems:** Software applications or programs which are based on interactions between their components, describing a system of a part of a system [37].

The majority of the studies are in the field of testing the first category. In total, 49 papers belong to this category, whereas 10 papers targeted to software systems.

A large number of studies are on applying environment models in testing real-time and embedded systems. Thirty-two studies present using environment models in testing time constrained systems. It is because the environment models can

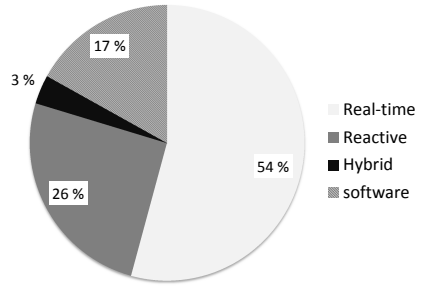

Figure 5: Distribution of application domains using environment models

control the timing of the SUT and also provides test cases to verify timing properties (as discussed in RQ1).

Fifteen papers used environment models for testing reactive systems, that contains continuous communications with its environment. Only two studies were found experimenting environment models on hybrid systems.

Ten papers used environment models in testing software components. When the entire software is not complete and a component needs to be tested, then an environment model of the whole software represents the behavior of the complete software in order to test the component. Table 4 lists system domains which are studied using environment models as identified in 59 primary studies.

Table 4: System domains using environment models

System domains	**References**	**# of studies**
Real-time & Embedded systems	[2, 3, 4, 5, 9, 13, 12, 16, 22, 23, 24, 25, 34, 36, 39, 41, 42, 43, 45, 46, 55, 56, 57, 60, 63, 64, 66, 69, 76, 82, 84, 86]	32
Reactive systems	[11, 13, 14, 15, 27, 38, 47, 48, 49, 50, 68, 73, 74, 78, 83]	15
Hybrid systems	[6, 87]	2
Software systems	[18, 17, 21, 32, 61, 71, 77, 33, 85, 88]	10

As the result of synthesizing the data, we identified that 54% of the primary studies investigate on real-time and embedded systems, 26% stud are about reactive systems and 17% experiment on software developments. Only 3% of total studies present testing in hybrid and embedded systems. Figure 5 shows the portion of each application domain in form of a pie chart.

3.4 RQ4 - At which testing levels are the environment models applied?

Based on our search results we could classify the papers as falling into two categories: functional and non-functional testing.

3.4.1 Functional Testing:

We categorize the primary studies based on different levels of testing by further analysis we find in which testing purposes environment modeling process has been applied.

Forty-three papers investigate testing with environment models at system level. The whole SUT is modeled along with its environment, and the test cases validate the functionality of whole system.

We found three studies that used environment models in integration testing, which verifies that different components work properly together.

Six papers focus on unit testing with environment models. Unit testing is mostly applied in testing software components and the environment model is specified as a model of a complete software, producing the input data for the component.

Table 5 presents different testing levels using environment models.

Table 5: Environment models in testing levels

Testing Level	**References**	**# of studies**
System	[2, 3, 4, 5, 6, 14, 16, 21, 22, 23, 24, 25, 27, 32, 34, 36, 39, 38, 42, 41, 43, 46, 45, 47, 48, 49, 50, 55, 60, 61, 63, 64, 66, 74, 76, 78, 81, 82, 83, 84, 85, 86, 87]	43
Integration	[57, 69, 73]	3
Unit	[10, 17, 56, 71, 77, 88]	6

3.4.2 Non-Functional Testing:

With further analyses, we identified that environment models are mostly employed in some certain non-functional testing approaches.

As it can be seen in Table 6, five papers report using environment models for testing safety and reliability of the SUT. An environment model can include hazardous states and by generating wrong inputs, which put the SUT into hazard. The responses of the SUT show how the system is safe against different hazardous states.

Few studies applied environment modeling on robustness testing, which is a crucial part of the verification process using invalid behaviors of the SUT to detect errors. For robustness testing, the environment model should provide stressful

conditions and invalid events to check the correctness of the SUT. This is important for testing critical systems where the behavior of the environment can not be predicted easily.

Three studies discuss applying environment models for regression testing, which indicates that by modifying the SUT, same environment model can be used to check the validity of the new version of the SUT.

Table 6: Non-functional testing approaches using environment models

Testing types	**References**	**# of studies**
Safety testing	[14, 16, 21, 57, 62]	5
Robustness testing	[45, 84]	2
Regression testing	[8, 15, 18]	3

3.5 RQ5 - What formalism and tools have been used for testing with environment models?

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 languages and tools and their references and briefly discuss some of the most referenced tools and explain how the environment models are defined.

The majority of the studies use the Unified Modeling Language (UML) [75] as the modeling language. In our review, 20 papers build on UML either by using its standard behavioral diagrams such as sequence and state diagrams, or use UML profiles such as MARTE [1], SysML [30] and MbRTE (Executable model-based robustness testing environment) [84]. The structure of the environment is a domain model (i.e., a model that describes all various entities and their relationships) and one or more environment components. The domain model provides the information of all relationships and properties between the components. In case of robustness testing, it can have some hazardous states for modeling hazardous test inputs.

Six papers present MBT approaches using Timed Automata (TA) [7]. The tools for modeling with TA are UPPAAL, its online testing tool (UPPAAL TRON) and Maude. UPPAAL is a model-checker which allows simulation and verification of TA-based specifications. The environment models can be specified UPPAAL as deterministic or non-deterministic. UPPAAL-TRON is online testing tool that generates test cases from the test models and execute them against the SUT [65].

Maude is a language and system supporting both equational and rewriting logic computation. It represents model generation rules by rewriting, instead of describing a model directly [66]. It can be applied for modifying the timed automata.

Table 7: Formalism and tools for environment modeling

Formalism	Tools and references	#
UML	UML diagrams [16, 39, 38, 45, 48, 57, 60, 73, 85, 87]	10
	UML/MARTE [9, 41, 42, 46, 47]	5
	UML Fondue [23, 68]	2
	UML/SysML [30]	1
	ESML[54]	1
	MbRTE [84]	1
Timed Automata	UPPAAL [36, 43, 63, 64]	4
	UPPAAL TRON[76]	1
	Maude [66]	1
Event Grammar	AEG [11, 12, 13, 14, 15, 78]	6
Petri nets	TINA [2, 3, 4, 5]	4
Lutin	Lurette [49, 50, 62]	3
Java	BEG[71, 77]	2
QR	QR models [6, 83]	2
TSML	AUTOSAR[69]	1
Esterel	Esterel[24]	1
SPIN	Promela [25]	1
TML	JUMBL [74]	1
Markov model	Markov model[21]	1
TTCN-3	TTCN-3 [27]	1
SLAM	SLAM [17]	1
DoB	Degree-of-Belief(DoB)[32]	1
BLAST	BLAST[56]	1

Six papers presented their experiments on testing with Attributed Event Grammar (AEG) [14] , which is built for testing real-time and embedded systems. Event grammars are text-based and are appropriate in specifying the dynamic environment with arbitrary number of actors and events. Models based on event grammars can be designed only for the environment or, for the environment and the SUT. It also can contain hazardous states to assess the safety of the SUT. The environment models can be used to automatically generate test cases.

Four studies are based on Petri Nets [35], using the TINA tool (TIme Petri Net Analyser) [20]. TINA is a software environment for the editing and analysis of Petri nets and Timed Petri nets. The environment models in TINA have same properties as the models defined in UPPAAL. Similar, to UPPAAL, the environment model support non-determinism and also deterministic assumptions.

Three papers presented testing with Lutin language [72], with environment models based on Lurette [51]. Lutin is a test-based language for specifying random reactive behaviors, and specially developed for modeling and testing reactive systems. Lurette is a test generator which provides random or guided test cases.

Two studies show how environment model can be designed in the Bandera Environment Generator (BEG) [77], a tool that automates the generation of environments for model-checking the Java programs. The tool is able to decompose a given java program into small modules and create the environment program out of it. Due to lack of space, we refer to the reader the overview of the remaining languages and tools in Table 7.

3.6 RQ6 - What problems and challenges have been observed by researchers using environment models in testing?

From all studies, we identified several studies that describe limitations and problems that occur using environment models in testing software applications or computer systems.

One of the main problems that is reported when applying environment models in case studies is that for the tests generated from multiple environment models, multiple test adaption is needed since the interactions among the environments as well as the environments and the SUT are usually integrated. The test adapter are written manually and thus it is not an efficient process (i.e., in [69]).

Modeling an environment with multiple entities (modular or aspects) should be carefully done, because it is complicated to define inter-relations among small modules. Kishi and Noda study modular environment (context) modeling for analyzing safety of their case study [57]. They emphasize on the importance of having a strategy for defining environment modules.

Once an environment model is specified, it will be changed seldom unless there are significant modifications in the requirements or there are some errors in test generation during testing. Therefore, as mentioned in [14], the tests which are derived from the environment model can be reused for regression testing. Reusability of the environment models is an open research question and should be investigated.

Robustness testing by environment models is studied in a small number of papers. However, it is still a challenge to expand the environment modeling in complex systems and for more complicated environments [84].

4 Validity threats

There are three main threats subjected 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 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 where reading titles and abstracts. However, in the second round of the search (snowballing), we did the effort of finding all the studies that were we did not find (or excluded) in the initial round.

Another threat is that the measurements may not be reliable. This can be caused from lack of reliability in the searching the resources, 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 classifications that are defined by the literature. Besides, for each classification, we provide the referenced definition, to prevent ambiguity.

5 Discussion and Conclusions

In this SLR, we defined research questions about environment modeling in MBT. We searched for 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.

One of the initial conclusions is that many of the identified studies use environment models for testing, but without discussing explicitly how the environment models are created and the process being it. Methodological aspect of creating environment models are only discussed in a limited number of papers.

In the further analysis, we classified the studies based on different concepts: system types and testing levels. The results show that majority of the studies employ the environment models in real-time and embedded systems. The reason might be that as a characteristic of the environment models, they are able to model and focus test generation on the timing constraints of the system under test.

It was also concluded that the environment models are mostly applied for system level testing and there are very few studies related to the other testing levels (integration and unit testing). We argue that the reason behind this might be stem the other main characteristic of the environment model (abstraction), which focuses on a system in abstract way and does not go into the details of the system components or their internal communications. More work is necessary in using environment models for testing at lower levels of the testing process.

While scanning the primary studies, we also found a few papers describing environment models in non-functional testing approaches such as safety testing,

robustness testing and regression testing. The approaches presented in these papers rely on the the explicit behavior exhibited by environment models which facilitates the modeling and testing of non functional properties without modifying any part of the SUT. For instance, in the context of testing safety properties, environment models were used for modeling hazardous states which violate the properties of the SUT.

The formalism and modeling tools used in the context of environment models have also been presented along with their references. The results show that most of the studies use UML or its different profiles as the base of the modeling the environments. Timed Automata, in particular UPPAAL Timed Automata is also a popular formalism in modeling and simulating test models with their environment. One of the advantages of using UPPAAL is that it allows the simulation and verification of timing properties of real-time systems. Another major work in investigating environment modeling is done in AEG, where the environment is based on a set of event grammars and applied for safety assessment in different case studies.

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. As noted by Auguston, more empirical evaluations is required in the case of large and complex SUTs with large number of test cases automated by the environment [14].

As mentioned earlier, one of the advantages of using environment models is reusing same models for different testing purposes. There is not enough work practicing environment modeling in regression testing.

Re-usability of environment models in testing new versions of a software/hardware should be experimented [39]. There are very few reports, though, about using environment models in hybrid systems. The communications among diverse applications with environments as well as communication among the environments themselves should be studied.

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.

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] N. Adjir, P. de Saqui-Sannes, and K. M. Rahmouni. Test of preemptive real-time systems. In *Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on*, pages 734–742. IEEE, 2008.

[4] N. Adjir, P. de Saqui-Sannes, and M. K. Rahmouni. Time-optimal real-time test case generation using prioritized time petri nets. In *Advances in System Testing and Validation Lifecycle, 2009. VALID'09. First International Conference on*, pages 110–116. IEEE, 2009.

[5] N. Adjir, P. d. S. Sannes, M. K. Rahmouni, and A. Adla. Timed test case using labeled prioritized time petri nets. *Computing Research Repository (CoRR)*, 2012.

[6] B. K. Aichernig, H. Brandl, E. Jöbstl, and W. Krenn. Model-based mutation testing of hybrid systems. In *Formal Methods for Components and Objects*, pages 228–249. Springer, 2010.

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

[8] K. Androutsopoulos, D. Binkley, D. Clark, N. Gold, M. Harman, K. Lano, and Z. Li. Model projection: simplifying models in response to restricting the environment. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 291–300. ACM, 2011.

[9] A. Arcuri, M. Iqbal, and L. Briand. Black-box system testing of real-time embedded systems using random and search-based testing. In A. Petrenko, A. Simao, and J. Maldonado, editors, *Testing Software and Systems*, volume 6435 of *Lecture Notes in Computer Science*, pages 95–110. Springer Berlin Heidelberg, 2010.

[10] C. Artho, K. Hayamizu, R. Ramler, and Y. Yamagata. With an open mind: How to write good models. In *Formal Techniques for Safety-Critical Systems*, pages 3–18. Springer, 2014.

[11] M. Auguston. New directions in software quality assurance automation. Technical report, DTIC Document, 2009.

[12] M. Auguston. Software architecture built from behavior models. *ACM SIGSOFT Software Engineering Notes*, 34(5):1–15, 2009.

[13] M. Auguston, J. B. Michael, and M.-T. Shing. Test automation and safety assessment in rapid systems prototyping. In *Rapid System Prototyping, 2005.(RSP 2005). The 16th IEEE International Workshop on*, pages 188–194. IEEE, 2005.

[14] 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.

[15] M. Auguston, J. B. Michael, M.-T. Shing, and D. L. Floodeen. Using attributed event grammar environment models for automated test generation and software risk assessment of system-of-systems. 2005.

[16] J. Axelsson. Unified modeling of real-time control systems and their physical environments using uml. In *Proceedings of the Eighth Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, pages 18–25, 2001.

[17] T. Ball, V. Levin, and F. Xie. Automatic creation of environment models via training. In K. Jensen and A. Podelski, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 2988 of *LNCS*, pages 93–107. Springer Berlin Heidelberg, 2004.

[18] T. Bao and M. Jones. Test case generation using model checking for software components deployed into new environments. In *Software Testing, Verification and Validation Workshops, 2009. ICSTW '09. International Conference on*, pages 57–66, April 2009.

[19] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp. Protocol for a systematic literature review of motivation in software engineering. 2006.

[20] 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.

[21] G. v. Bochmann, G.-V. Jourdan, and B. Wan. Improved usage model for web application reliability testing. In *Testing Software and Systems*, pages 15–31. Springer, 2011.

[22] M. Broy, S. Kirstan, H. Krcmar, B. Schätz, and J. Zimmermann. What is the benefit of a model-based design of embedded software systems in the car industry. *Emerging Technologies for the Evolution and Maintenance of Software Models. IGI Global*, pages 410–443, 2011.

[23] D. Buchs, L. Pedro, and L. Lúcio. Formal test generation from uml models. In *Dependable Systems: Software, Computing, Networks*, pages 145–171. Springer, 2006.

[24] S. M. Cho and J. W. Lee. Lightweight specification-based testing of memory cards: A case study. *Electronic Notes in Theoretical Computer Science*, 111:73–91, 2005.

[25] D. Cofer and M. Rangarajan. Event-triggered environments for verification of real-time systems. In *Simulation Conference, 2003. Proceedings of the 2003 Winter*, volume 1, pages 915–922 Vol.1, Dec 2003.

[26] H. R. Collard, S. Saint, and M. A. Matthay. Prevention of ventilator-associated pneumonia: an evidence-based systematic review. *Annals of Internal Medicine*, 138(6):494–501, 2003.

[27] W. Denissen. Amultidisciplinary model-based test and integration infrastructure. In *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, pages 1916–1921, Oct 2006.

[28] T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50:833 – 859, 2008.

[29] 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.

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

[31] D. D. Gajski, F. Vahid, S. Narayan, and J. Gong. Specification and design of embedded systems. 1994.

[32] I. Gheta, M. Baum, A. Belkin, J. Beyerer, and U. D. Hanebeck. Three pillar information management system for modeling the environment of autonomous systems. In *IEEE International Conference on Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS)*, pages 12–17. IEEE, 2010.

[33] W. Grieskamp. Multi-paradigmatic Model-Based Testing. In K. Havelund, M. Nunez, G. Rosu, and B. Wolff, editors, *Formal Approaches to Software Testing and Runtime Verification*, volume 4262 of *LNCS*, pages 1–19. Springer Berlin Heidelberg, 2006.

[34] J. D. Griffith and S. J. Lee. Environment modeling for sense and avoid sensor safety assessment. In *IEEE/AIAA 30th Digital Avionics Systems Conference (DASC)*, pages 5B5–1, 2011.

[35] M. Hack. Petri net language. 1976.

[36] J. Hänsel, D. Rose, P. Herber, and S. Glesner. An evolutionary algorithm for the generation of timed test traces for embedded real-time systems. In *International Conference on Software Testing, Verification and Validation*, pages 170–179. IEEE, 2011.

[37] G. T. Heineman and W. T. Councill. Component-based software engineering. *Putting the Pieces Together, Addison-Westley*, 2001.

[38] M. Heisel, D. Hatebur, T. Santen, and D. Seifen. Using uml environment models for test case generation. In *Software Engineering (Workshops)*, pages 399–406, 2008.

[39] M. Heisel, D. Hatebur, T. Santen, and D. Seifert. Testing Against Requirements Using UML Environment Models. *Softwaretechnik-Trends*, 28(3), 2008.

[40] T. A. Henzinger. *The theory of hybrid automata*. Springer, 2000.

[41] F. Herrera, P. Penil, H. Posadas, and E. Villar. A model-driven methodology for the development of systemc executable environments. In *Specification and Design Languages (FDL), 2012 Forum on*, pages 177–184. IEEE, 2012.

[42] F. Herrera, H. Posadas, P. Peñil, E. Villar, F. Ferrero, and R. Valencia. A mdd methodology for specification of embedded systems and automatic generation of fast configurable and executable performance models. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 529–538. ACM, 2012.

[43] A. Hessel, K. Larsen, M. Mikucionis, B. Nielsen, P. Pettersson, and A. Skou. Testing Real-Time Systems Using UPPAAL. In R. Hierons, J. Bowen, and M. Harman, editors, *Formal Methods and Testing*, volume 4949 of *LNCS*, pages 77–117. Springer Berlin Heidelberg, 2008.

[44] 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.

[45] M. Iqbal, A. Arcuri, and L. Briand. Environment modeling and simulation for automated testing of soft real-time embedded software. *Software and Systems Modeling*, pages 1–42, 2013.

[46] M. Z. Iqbal, S. Ali, T. Yue, and L. Briand. *Experiences of applying UML/MARTE on three industrial projects*. Springer, 2012.

[47] M. Z. Iqbal, S. Ali, T. Yue, and L. Briand. Applying UML/MARTE on industrial projects: challenges, experiences, and guidelines. *Software & Systems Modeling*, pages 1–19, 2014.

[48] M. Z. Iqbal, A. Arcuri, and L. Briand. Empirical Investigation of Search Algorithms for Environment Model-based Testing of Real-time Embedded Software. In *Proceedings of the International Symposium on Software Testing and Analysis*, ISSTA, pages 199–209, New York, NY, USA, 2012. ACM.

[49] E. Jahier, S. Djoko-Djoko, C. Maiza, and E. Lafont. Environment-model based testing of control systems: Case studies. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 636–650. Springer, 2014.

[50] E. Jahier, N. Halbwachs, and P. Raymond. Engineering functional requirements of reactive systems using synchronous languages. In *Industrial Embedded Systems (SIES), 2013 8th IEEE International Symposium on*, pages 140–149. IEEE, 2013.

[51] E. Jahier, P. Raymond, and P. Baufreton. Case studies with lurette v2. *International Journal on Software Tools for Technology Transfer*, 8(6):517–530, 2006.

[52] 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.

[53] M. Jorgensen and M. Shepperd. A systematic review of software development cost estimation studies. *Software Engineering, IEEE Transactions on*, 33(1):33–53, 2007.

[54] 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).

[55] 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.

[56] M. Kim, Y. Kim, and H. Kim. A comparative study of software model checkers as unit testing tools: An industrial case study. *Software Engineering, IEEE Transactions on*, 37(2):146–160, 2011.

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

[58] 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.

[59] I. Kruger. Service-oriented software and systems engineering-a vision for the automotive domain. In *Formal Methods and Models for Co-Design, 2005. MEMOCODE'05. Proceedings. Third ACM and IEEE International Conference on*, page 150. IEEE, 2005.

[60] B. Kumar, B. Czybik, and J. Jasperneite. Model based TTCN-3 testing of industrial automation systems; First results. In *Conference on Emerging Technologies Factory Automation*, pages 1–4, 2011.

[61] A. Lakehal, F. Ouabdesselam, I. Parissis, and J. Vassy. Models for synchronous software testing. In *Model, Design and Validation, 2004. Proceedings. 2004 First International Workshop on*, pages 41–50, Nov 2004.

[62] A. Lakehal, F. Ouabdesselam, I. Parissis, and J. Vassy. Models for synchronous software testing. In *First International Workshop on Model, Design and Validation*, pages 41–50. IEEE, 2004.

[63] K. Larsen, M. Mikucionis, and B. Nielsen. Online Testing of Real-time Systems Using Uppaal. In J. Grabowski and B. Nielsen, editors, *Formal Approaches to Software Testing*, volume 3395 of *LNCS*, pages 79–94. Springer Berlin Heidelberg, 2005.

[64] 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*, pages 299–306. ACM, 2005.

[65] 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.

[66] 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.

[67] F. Liu, A. Narayanan, and Q. Bai. Real-time systems. 2000.

[68] 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.

[69] 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.

[70] M. Mikucionis, K. G. Larsen, and B. Nielsen. *Online on-the-fly testing of real-time systems*. Citeseer, 2003.

[71] P. Parizek and F. Plasil. Specification and Generation of Environment for Model Checking of Software Components . *Electronic Notes in Theoretical Computer Science*, 176(2):143 – 154, 2007. Proceedings of the Workshop on Formal Foundations of Embedded Software and Component-Based Software Architectures (FESCA 2006).

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

[73] A. Pretschner, O. Slotosch, E. Aiglstorfer, and S. Kriebel. Model-based testing for real. *International Journal on Software Tools for Technology Transfer*, 5(2-3):140–157, 2004.

[74] S. J. Prowell. Jumbl: A tool for model-based statistical testing. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 9–pp. IEEE, 2003.

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

[76] C. Rütz and J. Schmaltz. An Experience Report on an Industrial Case-Study about Timed Model-Based Testing with UPPAAL-TRON. In *International Conference on Software Testing, Verification and Validation Workshops*, pages 39–46, March 2011.

[77] 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.

[78] H. Tummala, M. Auguston, J. B. Michael, M.-T. Shing, D. Little, and Z. Pace. Implementation and analysis of environment behavior models as a tool for testing real-time, reactive systems. In *International Conference on System of Systems Engineering,*, pages 5–pp. IEEE/SMC, 2006.

[79] M. Utting and B. Legeard. *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

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

[81] W. Visser, M. B. Dwyer, and M. Whalen. The hidden models of model checking. *Software & Systems Modeling*, 11(4):541–555, 2012.

[82] Y. Wang, Y. Wang, and C. Jiang. Theory and implemetion of simulation testing framework for embedded software. In *Reliability, Maintainability and Safety (ICRMS), 2011 9th International Conference on*, pages 751–756. IEEE, 2011.

[83] F. Wotawa. Generating test-cases from qualitative knowledge preliminary report. In *Proceedings of the 21st Annual Workshop on Qualitative Reasoning, Aberystwyth, UK*, 2007.

[84] 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.

[85] K. Yatake and T. Aoki. Automatic generation of model checking scripts based on environment modeling. In *Model Checking Software*, pages 58–75. Springer, 2010.

[86] W. Yichen and W. Yikun. Model-based simulation testing for embedded software. In *2011 Third International Conference on Communications and Mobile Computing*, pages 103–109, 2011.

[87] L. Yu, W. T. Tsai, Y. Jiang, and J. Gao. Generating test cases for context-aware applications using bigraphs. In *Software Security and Reliability, 2014 Eighth International Conference on*, pages 137–146. IEEE, 2014.

[88] L. Zhang, T. Xie, N. Tillmann, P. De Halleux, X. Ma, and J. Lv. Environment modeling for automated testing of cloud applications. *IEEE Software, Special Issue on Software Engineering for Cloud Computing*, 2012.

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

ISBN XXX-XXX-XXX-X
ISSN 1239-1891