A tertiary study on model-based testing areas, tools and challenges: Preliminary results

Leonardo Villalobos-Arias, Christian Quesada-López, Alexandra Martinez, Marcelo Jenkins

University of Costa Rica, San Pedro, Costa Rica
{leonardo.villalobos, cristian.quesadalopez, alexandra.martinez, marcelo.jenkins}@ucr.ac.cr

Abstract. Context: Model-based testing (MBT) is one of the most studied approaches by secondary studies in the area of software testing. A tertiary study that aggregates knowledge from secondary studies on MBT can be useful to both academia and industry. **Objective**: The goal of this study is to identify and characterize secondary studies in model-based testing, in terms of the areas, tools and challenges they have investigated. **Method**: We conducted a tertiary study in MBT. Our systematic mapping of secondary studies included 12 literature surveys and 10 systematic reviews over the period 1996–2016.**Results**: We found that the two most studied areas of MBT are UML models and Transition-based notations. We also found that only 5 studies compared and classified MBT tools. The main challenges and limitations found were related to the need for more empirical evidence that supports the selection of MBT approaches and tools. **Conclusions**: Not many systematic reviews on MBT were found, consequently some areas still lack secondary studies: test execution aspects, language types, model dynamics, and some model paradigms and generation methods.
We thus encourage the MBT community to perform further systematic reviews and mapping studies, following known protocols and reporting procedures, in order to increase the quality and quantity of empirical studies in MBT.

Keywords: Model-based testing, software testing, mapping study, software engineering, tertiary study

1 Introduction

Software testing is an essential part of software engineering [1], but it has several limitations. Traditional testing techniques are based on the design of a test suite, which is later translated into a series of test cases. These test cases are then manually executed by testers on the software under test (SUT) [1, 2]. Model-based testing (MBT) attempts to solve this by automating parts of the software testing process, mainly the test design step and the test case execution. MBT has been proven to find at least as many errors as manual testing [3]. Investing in MBT has been shown to reduce the overall cost, time and effort of the testing process [2]. Also, MBT may improve testing quality by ensuring that certain criteria (e.g., code coverage) is always met [2]. Finally, MBT tools can aid with requirements traceability and evolution, which can be hard in the traditional testing process [2].

Despite the benefits of MBT, it has some limitations. Particularly, it requires a different set of skills than manual testing [2] since the model designer must be capable of abstracting and designing the model from the requirements (i.e., knowing which aspects should be model and which should not). This also implies training costs and a learning curve before starting to use MBT [2]. Besides, a considerable amount of effort has to be invested in building the model, since the quality of generated tests depends on the quality of the model.

Some tertiary studies have been recently conducted in software engineering [4, 5, 6], but only one [4] focuses on software testing. Garousi et al. [4] identified 15 secondary studies related to MBT, positioning it as the testing method with more secondary studies. However, we did not find any tertiary study specific to MBT. A tertiary study that aggregates and synthesizes knowledge from secondary studies on MBT can be useful to both academia and industry.

To address the lack of tertiary studies in MBT, our research goal was to identify and characterize secondary studies in model-based testing, in terms of the areas, tools, challenges, problems and limitations they have investigated. We conducted a tertiary study based on the guidelines described in [7, 8] and recommendations stated in [9, 10, 11]. The studies were characterized in terms of the specific areas and research questions, model types, test selection criteria, test generation and test execution techniques, testing tools, as well as challenges and limitations. We established three specific research questions to guide our analysis:

RQ1: What model-based testing specific areas have been investigated in secondary studies? Answering this question will enable us to determine the specific MBT areas addressed by secondary studies, including model representations, criteria for test case selection, and techniques for both test case generation and test case execution.

RQ2: How have model-based testing tools been characterized in the literature? This question will allow us to identify the characteristics of model based testing tools as well existing tool classifications.

RQ3: What are the challenges, open problems and limitations reported in the literature of model-based testing? This question will allow us to identify common challenges and limitations of the current research on MBT, as well as potential areas of future research.

The remainder of our paper is structured as follows. We discuss the background in Section 2. Section 3 presents the methodology. Section 4 answers our research questions. And section 5 offers our conclusions and future work.

2 Background

Software testing consists of verifying if a program matches its expected behavior [1]. It can be performed at different levels: unit, integration, and system. Software testing approaches can also be categorized as white-box or black-box, depending on whether they use information about the internal structure of the SUT [1]. Model-based testing is the automation of the design of black-box tests [2]. It is based on a model of the SUT that represents aspects that will be tested. From this model, test cases can be generated in an automated way, usually with the help of a tool. Model-based testing is usually performed at the unit level and covers the functional behavior of the SUT [2].

Utting and Legeard's book on model-based testing [2] is essentially a body of knowledge in MBT, containing existing approaches, process stages, tools and examples. It offers valuable insight for those new to the area. The process of model-based testing has five stages [12]: (1) building the model of the SUT from the requirements, (2) choosing the test selection criteria, (3) transforming these criteria (and the model) into test case specifications, (4) generating test cases, and (5) executing the test cases. The model should be built for testing purposes (i.e., simpler than the model used for development) [2]. A MBT tool may provide support for activities in any of these stages.

In [12], Utting et al. attempt to sort out the MBT body of knowledge by defining a taxonomy of MBT approaches. This taxonomy has six dimensions: (1) model scope, which encompasses aspects of the SUT that are being represented by the model, (2) model characteristics, like non-determinism, (3) model paradigm, which groups modeling notations into families, (4) test selection criteria, which covers the criteria that is used to generate the test cases, (5) test generation technology, which comprises techniques that generate test cases from the model, and (6) test execution, which includes the different ways of generating and/or executing the test cases. Authors suggest that other taxonomies may be defined to express different aspects of MBT's approaches or tools.

The International Software Testing Qualifications Board's Foundation Level Certified Model-Based Tester Syllabus [13] is a guide for certification on MBT. It explains multiple aspects of model-based testing in a concise way, and it focuses on the practical part of MBT. This guide also covers an important topic on how to evaluate and deploy an MBT approach.

Anand et al. [14] performed an orchestrated survey on existing test case generation techniques. In their paper they include five generation techniques. In their model-based testing section, they cover some of the previous work in MBT. They explain the finite-state machine and labeled transition system approaches. They also showcase some commercial tools.

We have found few tertiary studies related to model-based testing. However, we haven't stumbled upon a tertiary study in which MBT is the main emphasis, rather it has been only one area of knowledge from multiple that each study covers. Our study will perform a deeper, more accurate mapping of model-based testing than the existing tertiary studies. We believe it will useful for those new to research in model-based testing as a guide for other studies.

3 Methodology

A tertiary study using the same methodology stated in [7, 8] was conducted. In our design, recommendations stated in [9, 10, 11, 15] were also incorporated.

3.1 Search process and study selection

We initially conducted an exploratory search to identify the terms (or keywords) relevant for our mapping. Then we selected this search string:

```
(``model driven * test*'' OR ``model based * test*'' OR ``model-based
    test*'') AND (``review'' OR ``survey'' OR ``systematic mapping'' OR
    ``mapping study'' ) AND (``Software'' OR ``system under test'' OR ``
    SUT'') AND (``tool'' OR ``test* automat*'' OR ``automat* test*'')
```

This search string was used to conduct a search on the Scopus (subareas: "COMP" and "ENG") and Web of Science online academic engine, up to April 2017. The search terms could appear anywhere in the title, abstract or keywords.

Inclusion and exclusion criteria. Articles that met the following criteria were included: (I1) Language: English, (I2) Type of study: Secondary study (survey, map or review) and (I3) Approach: Model based testing. Articles that met the following criteria were excluded: (E1) Papers not available in full text, (E2) Publications not related to the software engineering domain and (E3) Non peer reviewed publications. When a secondary study was published in more than one journal/conference, both versions of the study were compared for the purpose of data extraction.

Selection process. Our search string produced an initial set of 47 potentially relevant studies. The inclusion and exclusion criteria were applied to the title, keywords and abstract of extracted articles. Any paper that was clearly irrelevant became excluded. To complete the inclusion and exclusion process, the following steps were performed: (1) each paper was evaluated by 5 members of the research group, (2) each member independently screened the paper for inclusion and exclusion criteria, and decided whether or not to include the paper, offering a justification, (3) the team met and compared their results. Any disagreement about the inclusion of a paper was discussed until a final consensus was reached, and the decision was documented. We tried not to reject potentially relevant papers. After applying the exclusion criteria, 13 studies were selected. To expand the scope of our search, backward snowballing was applied to the selected secondary studies as well as to three relevant tertiary studies [4, 5, 6]. As a result, 9 more papers were included, for a total of 22 papers. Our automated search did not find the snowball papers because they did not math our search string: some targeted specific models such as UML and FSMs, some used a different name for MBT (e.g., formal testing approach) and others focused on a specific aspect of MBT (e.g., test case generation). From these, 12 were literature surveys and 10 were systematic reviews. Figure 1 shows the number and type of secondary studies over time. The complete list of secondary papers along with their origin (automated search or snowballing), type (literature surveys or systematic reviews), quality assessment and extraction form, is available at `https://goo.gl/gX2WKe`.

3.2 Quality assessment

The quality of all secondary studies was evaluated according to the DARE assessment based upon four questions [6, 15]: (Q1) Are the study's inclusion and exclusion

Table 1: Quality assessment of secondary studies.

Ref.	Type	Year	Guid.	Q1	Q2	Q3	Q4	Total	Ref.	Type	Year	Guid.	Q1	Q2	Q3	Q4	Total
S01	LS	1996		0	0	0	0.5	0.5	S12	SR	2010	[8, 9]	1	1	0.5	1	3.5
S02	LS	2005		0	0	0	1	1	S13	LS	2012		0	0	0	1	1
S03	LS	2006		0	0	0	1	1	S14	SR	2012		1	0.5	0	1	1.5
S04	SR	2007	[8, 9]	1	1	0	1	3	S15	LS	2013		0	0	0	1	1
S05	SR	2008	[8, 9]	1	1	0	1	3	S16	SR	2014	[8]	1	1	1	1	4
S06	SR	2008	[8, 9]	1	1	0	1	3	S17	LS	2015		1	1	0	1	3
S07	LS	2009		0	0	0	0	0	S18	SR	2015	[7, 8, 9]	1	0.5	1	1	3.5
S08	LS	2009		0	0	0	0.5	0.5	S19	SR	2015	[8]	1	1	0	1	3
S09	LS	2010		0	0	0	0	0	S20	SR	2015	[5]	1	1	1	1	4
S10	LS	2010		0	0	0	1	1	S21	LS	2016		0	0	0	0	0
S11	LS	2010		0	0	0	0	0	S22	SR	2016	[8, 5]	0.5	1	0.5	1	3

criteria explicitly and clearly defined? Y (yes): the inclusion criteria are explicitly and clearly defined; P (partially): the inclusion criteria are implicit; N (no): the inclusion criteria are not defined and cannot be easily inferred. (Q2) Is the literature search likely to have covered relevant studies? Y: the authors have either searched a digital library and included additional search strategies or identified and referenced all journals addressing the topic of interest; P: the authors have searched searched a digital library with no extra search strategies; N: the authors have searched an extremely restricted set of journals or conferences. (Q3) Did the reviewers assess the quality/validity of the included studies? Y: the authors have explicitly defined quality criteria and extracted them from each primary study; P: the research question involves quality issues that are addressed by the study; N: no explicit quality assessment of individual papers has been attempted, or quality data has been extracted but not used. (Q4) Were the basic data/studies adequately described? Y: information is presented about each paper so that data summaries can clearly be traced to relevant papers; P: only summary information is presented about individual papers, it is not possible to link individual studies to each category; N: the results of the individual studies are not specified, the individual primary studies are not cited. The score assignment was Y = 1, P = 0.5, N = 0, and the overall score was computed by summing the scores of individual questions. A high quality score corresponds to a detailed report of the secondary study. Our quality assessment is presented in Table 1. We observe from this table that all but one of the systematic reviews (SR) exhibited high quality scores (3 or more, out 4). On the other hand, only one literature surveys (LS) had a high quality score (3), the rest had low scores (1 or less, out of 4), as expected. Studies with low quality scores were not excluded from the analysis.

3.3 Data extraction and analysis

The extraction process was performed in four steps, described next. First, each paper was randomly assigned to a pair of researchers (from the research team of six in-

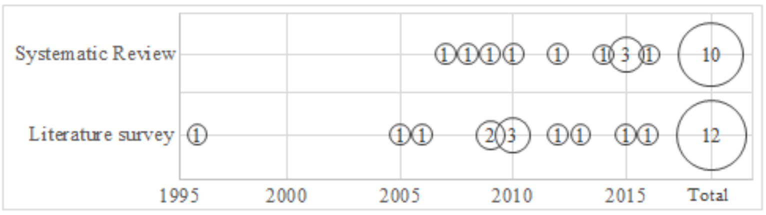

Fig. 1: Number and type of secondary studies in time.

Table 2: Extracted data items.

Category	Data Items
Id	Reference, forum (conference or journal), year, title, authors, google scholar hits
Study	Type of study and guidelines used
Areas	Model types, test selection criteria, test generation and execution techniques, taxonomies and classifications used
Tools	Tools classification (model representation, test case generation, test case execution and analysis techniques), system under test (SUT) domain(s)
Challenges	Challenges, open problems and limitations

cluding the authors of this paper). Then, each member of a pair independently screened the paper for extraction. After that, the pair met and compared their results, discussing any disagreement until a consensus was reached and the decision was documented. Finally, for each research question, one author of this paper validated the data extraction and complemented any information when necessary. We analyzed each study and extracted any relevant data to answer our research questions. In Table 2, we list the data items extracted from each study. A spreadsheet was created to aid in the organization and categorization of data. For RQ1, our initial hierarchy was the union of the categories in our reference taxonomies [12, 16] ([16] is also S17 in Table 1). Then, we kept adding categories (areas) that emerged from the studies. Since we wanted to create a tree rather than a flat structure (for better understanding and categorization of data), we had to group subareas sharing one or more characteristics into an upper-level node (area). We tried to respect the category names originally given by the authors, but sometimes we had to choose (whenever two different names were given to the same area), and in one case, we had to create the label. When in doubt about the classification of an approach, we followed our reference taxonomies. For RQ2, MBT tools mentioned in secondary studies were identified. We extracted information reported about each tool and attempted to group them in categories. For RQ3, reported challenges and limitations were identified. We perform grounded analysis to allow the categories to emerge from the data.

3.4 Threats to validity

We briefly discuss here the threats to validity of our tertiary study. *Search terms and digital libraries:* In order to minimize the risk of missing relevant studies, we first did an exploratory search in order to identify relevant papers and keywords for our search string. Several trials and validations were performed on the search string by multiple researchers, in order to have a stable and final search string. We only used the Scopus and Web of Science search engine, but to counter we applied snowballing, as recommended by [11]. The snowballing process was a backward process based only in titles. Some studies could still be missing from our analysis. *Selection of studies:* In order to minimize the researcher bias in applying the exclusion and inclusion criteria, multiple researchers performed various validations of the excluded and included studies. Since we included only secondary studies in model-base testing, our findings apply only in this context. *Data extraction and categorization:* In order to minimize possible inaccuracies in the data extraction as well as researcher bias in the classification, various researchers validated the extraction process. Although the categorization was mainly performed by a single researcher, the other researchers gave feedback on the categories, as a way to validate them. Moreover, most of our categories for RQ1 and RQ2 were based on previous MBT studies.

4 Results and discussion

In this section we present the results of our mapping study, based on 22 secondary studies, and address our specific research questions.

4.1 RQ1: Model-based testing research areas

This research question helped us identify MBT areas that have been investigated by secondary studies. Particularly, we aimed to discover which model representations, test case selection criteria, test case generation methods and test case execution techniques have been most (and least) studied. For this, we built a hierarchy of MBT areas and subareas based upon two existing taxonomies: one proposed by Utting et al. [12] (aimed for MBT approaches) and other proposed by Marinescu et al. [16] (aimed for MBT tools). Areas (or categories) were added, merged or renamed as papers were read and analyzed. When the hierarchy was finalized, we reviewed again all the studies in order to complete the list of supporting studies per area. The final hierarchy of MBT areas is shown in Figure 2. References inside internal nodes are studies that actually use or propose the category represented by the node. The reference list per leaf node corresponds to studies that support that specific category. A study was considered to 'support' a category or area if it mentions the category (not necessarily if it was a mainstream topic).

We found that the two most studied areas in MBT are UML (16 secondary studies) and Transition-based notations (12 secondary studies), both falling into the Model paradigm area. Within UML, behavioral models (including state machine, sequence and activity diagrams) are the most common, with 15 secondary studies in total. Among Transition-based notations, Finite state machines is the most common, with 10 secondary studies. The third and fourth most studied areas in MBT are Test levels, particularly Unit testing (10 studies) and Integration testing (9 studies).

With respect to test case selection criteria, the most studied is Structural model coverage, with 6 studies. Regarding test case generation methods, Random and Search-based are the most studied algorithms, with 4 studies each. Among the most studied test artifacts found were Functional behavior (or properties), with 6 studies, and extra-functional behavior (or non-functional aspects), with 5 studies. On the other hand, among the most studied test types were Functional testing and Regression testing, with 5 studies each. Also, we found that the most studied test execution subarea is Technology (online/offline), with 4 studies. In regard to the type of SUT that MBT targets, the most studied SUT domains are Real-time systems, Reactive systems, Embedded systems, and Web applications (each with 4 studies), whereas the most studied software paradigm is Object-oriented (with 4 studies also).

During the construction of our hierarchy, we also found some inconsistencies among the classifications given by different authors. For example, some studies [S12] classified FSM and LTS as state-based models, whereas Utting et al. [12] and Marinescu et al. [S17] classified them as transition-based models in their taxonomies. Another example is regression testing, which Dias et al. [S04] classified as a test level but we consider it a test type (based on [17]). On the other hand, [S21] placed search-based algorithms for test case generation (also known as meta-heuristic search) in a category outside of MBT, while other authors [S13, S15, S21][12] categorized it as a test generation method in MBT.

4.2 RQ2: Model-based testing tools

This research question led us to identify classifications that have been used in secondary studies for MBT tools. In what follows, we present the results with respect

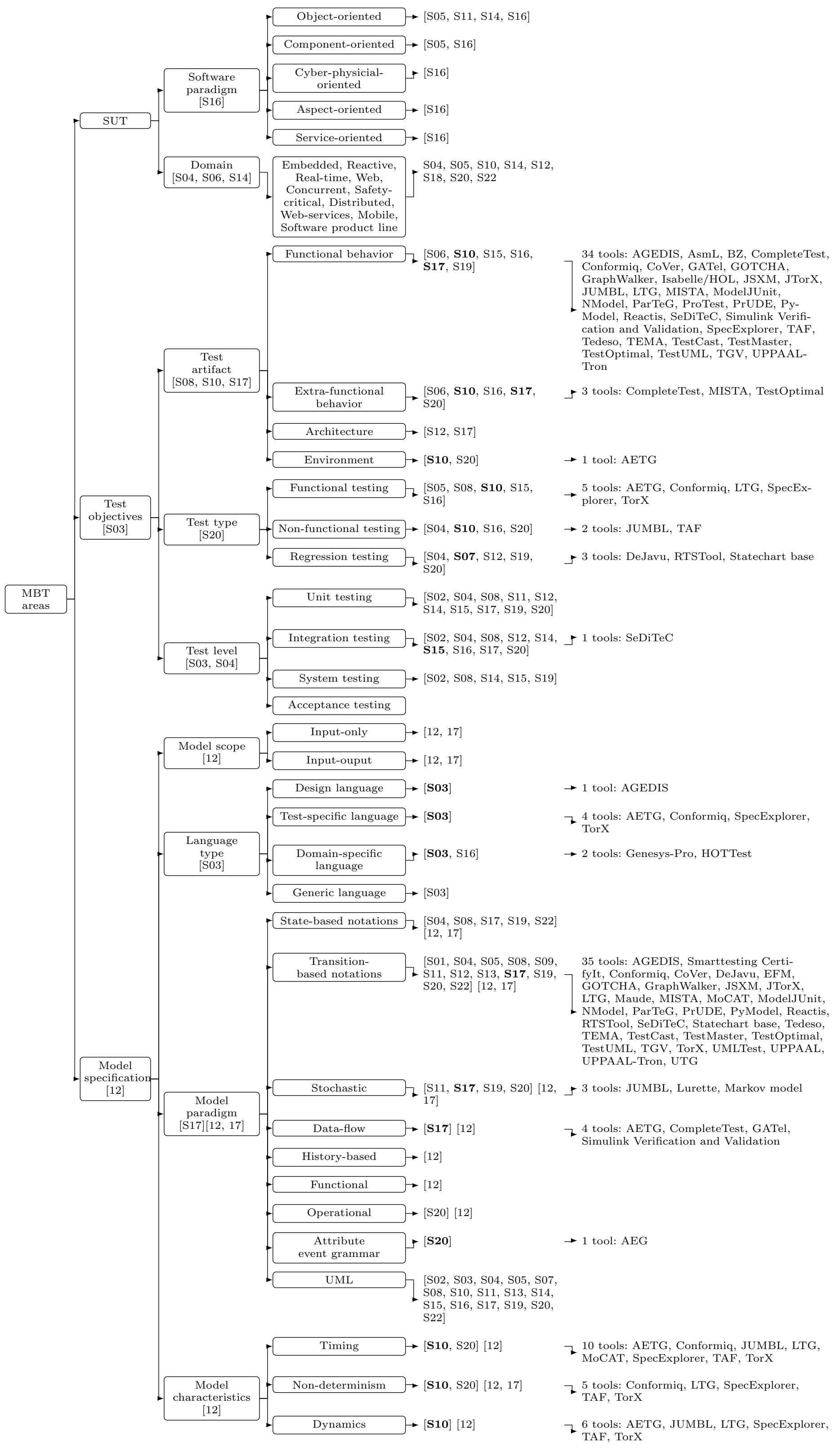
MBT areas
SUT
Software paradigm [S16]
Object-oriented
[S05, S11, S14, S16]
Component-oriented
[S05, S16]
Cyber-physicial-oriented
[S16]
Aspect-oriented
[S16]
Service-oriented
[S16]
Domain [S04, S06, S14]
Embedded, Reactive, Real-time, Web, Concurrent, Safety-critical, Distributed, Web-services, Mobile, Software product line
S04, S05, S10, S14, S12, S18, S20, S22
Test objectives [S03]
Test artifact [S08, S10, S17]
Functional behavior
[S06, S10, S15, S16, S17, S19]
34 tools: AGEDIS, AsmL, BZ, CompleteTest, Conformiq, CoVer, GATel, GOTCHA, GraphWalker, Isabelle/HOL, JSXM, JTorX, JUMBL, LTG, MISTA, ModelJUnit, NModel, ParTeG, ProTest, PrUDE, PyModel, Reactis, SeDiTeC, Simulink Verification and Validation, SpecExplorer, TAF, Tedeso, TEMA, TestCast, TestMaster, TestOptimal, TestUML, TGV, UPPAAL-Tron
Extra-functional behavior
[S06, S10, S16, S17, S20]
3 tools: CompleteTest, MISTA, TestOptimal
Architecture
[S12, S17]
Environment
[S10, S20]
1 tool: AETG
Test type [S20]
Functional testing
[S05, S08, S10, S15, S16]
5 tools: AETG, Conformiq, LTG, SpecExplorer, TorX
Non-functional testing
[S04, S10, S16, S20]
2 tools: JUMBL, TAF
Regression testing
[S04, S07, S12, S19, S20]
3 tools: DeJavu, RTSTool, Statechart base
Test level [S03, S04]
Unit testing
[S02, S04, S08, S11, S12, S14, S15, S17, S19, S20]
Integration testing
[S02, S04, S08, S12, S14, S15, S16, S17, S20]
1 tool: SeDiTeC
System testing
[S02, S08, S14, S15, S19]
Acceptance testing
Model specification [12]
Model scope [12]
Input-only
[12, 17]
Input-ouput
[12, 17]
Language type [S03]
Design language
[S03]
1 tool: AGEDIS
Test-specific language
[S03]
4 tools: AETG, Conformiq, SpecExplorer, TorX
Domain-specific language
[S03, S16]
2 tools: Genesys-Pro, HOTTest
Generic language
[S03]
Model paradigm [S17][12, 17]
State-based notations
[S04, S08, S17, S19, S22] [12, 17]
Transition-based notations
[S01, S04, S05, S08, S09, S11, S12, S13, S17, S19, S20, S22] [12, 17]
35 tools: AGEDIS, Smarttesting CertifyIt, Conformiq, CoVer, DeJavu, EFM, GOTCHA, GraphWalker, JSXM, JTorX, LTG, Maude, MISTA, MoCAT, ModelJUnit, NModel, ParTeG, PrUDE, PyModel, Reactis, RTSTool, SeDiTeC, Statechart base, Tedeso, TEMA, TestCast, TestMaster, TestOptimal, TestUML, TGV, TorX, UMLTest, UPPAAL, UPPAAL-Tron, UTG
Stochastic
[S11, S17, S19, S20] [12, 17]
3 tools: JUMBL, Lurette, Markov model
Data-flow
[S17] [12]
4 tools: AETG, CompleteTest, GATel, Simulink Verification and Validation
History-based
[12]
Functional
[12]
Operational
[S20] [12]
Attribute event grammar
[S20]
1 tool: AEG
UML
[S02, S03, S04, S05, S07, S08, S10, S11, S13, S14, S15, S16, S17, S19, S20, S22]
Model characteristics [12]
Timing
[S10, S20] [12]
10 tools: AETG, Conformiq, JUMBL, LTG, MoCAT, SpecExplorer, TAF, TorX
Non-determinism
[S10, S20] [12, 17]
5 tools: Conformiq, LTG, SpecExplorer, TAF, TorX
Dynamics
[S10] [12]
6 tools: AETG, JUMBL, LTG, SpecExplorer, TAF, TorX

Fig. 2: MBT areas investigated by secondary studies.

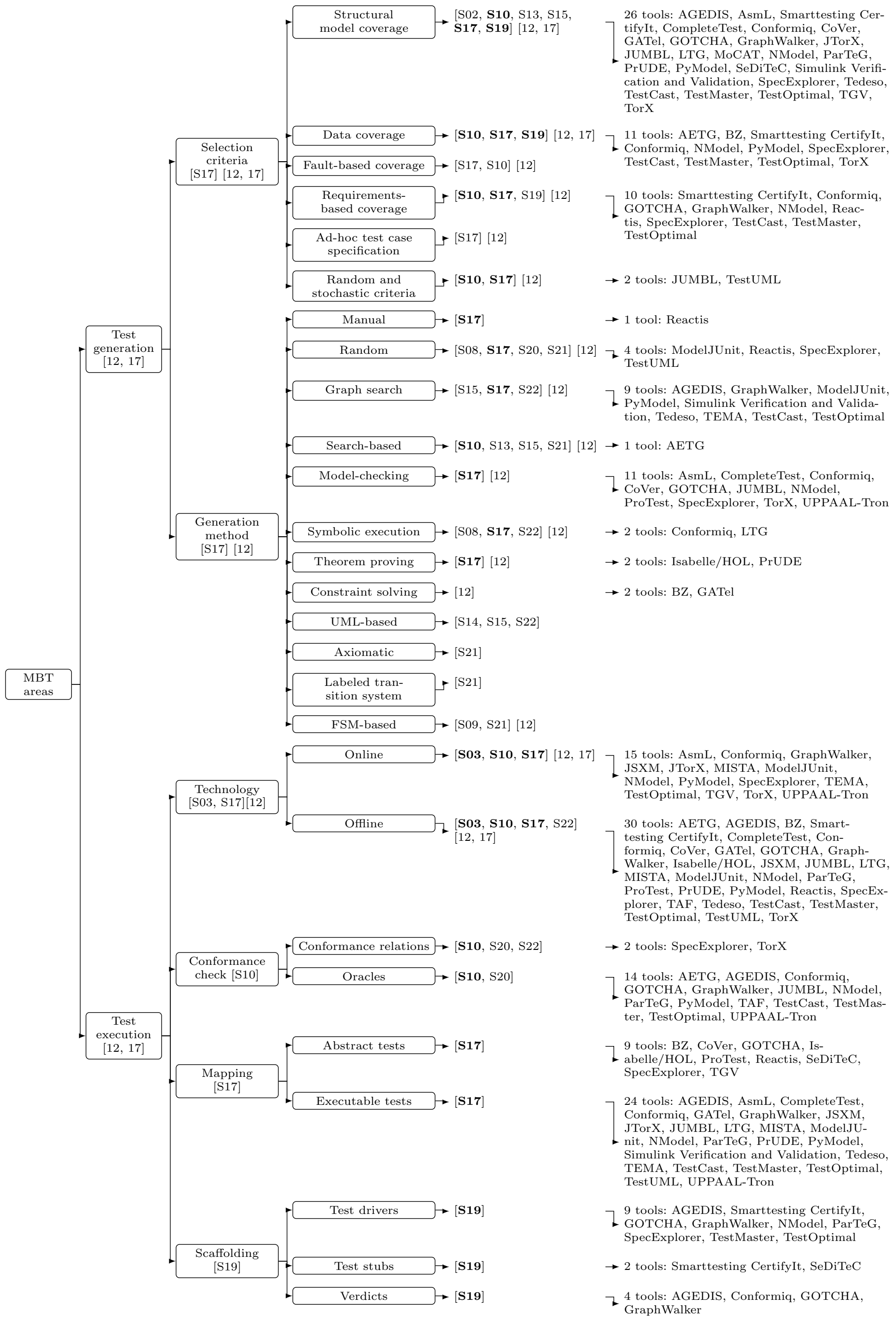
MBT areas
Test generation [12, 17]
Selection criteria [S17] [12, 17]
Structural model coverage
[S02, **S10**, S13, S15, **S17**, **S19**] [12, 17]
26 tools: AGEDIS, AsmL, Smarttesting CertifyIt, CompleteTest, Conformiq, CoVer, GATel, GOTCHA, GraphWalker, JTorX, JUMBL, LTG, MoCAT, NModel, ParTeG, PrUDE, PyModel, SeDiTeC, Simulink Verification and Validation, SpecExplorer, Tedeso, TestCast, TestMaster, TestOptimal, TGV, TorX
Data coverage
[**S10**, **S17**, **S19**] [12, 17]
11 tools: AETG, BZ, Smarttesting CertifyIt, Conformiq, NModel, PyModel, SpecExplorer, TestCast, TestMaster, TestOptimal, TorX
Fault-based coverage
[S17, S10] [12]
Requirements-based coverage
[**S10**, **S17**, S19] [12]
10 tools: Smarttesting CertifyIt, Conformiq, GOTCHA, GraphWalker, NModel, Reactis, SpecExplorer, TestCast, TestMaster, TestOptimal
Ad-hoc test case specification
[S17] [12]
Random and stochastic criteria
[**S10**, **S17**] [12]
2 tools: JUMBL, TestUML
Generation method [S17] [12]
Manual
[**S17**]
1 tool: Reactis
Random
[S08, **S17**, S20, S21] [12]
4 tools: ModelJUnit, Reactis, SpecExplorer, TestUML
Graph search
[S15, **S17**, S22] [12]
9 tools: AGEDIS, GraphWalker, ModelJUnit, PyModel, Simulink Verification and Validation, Tedeso, TEMA, TestCast, TestOptimal
Search-based
[**S10**, S13, S15, S21] [12]
1 tool: AETG
Model-checking
[**S17**] [12]
11 tools: AsmL, CompleteTest, Conformiq, CoVer, GOTCHA, JUMBL, NModel, ProTest, SpecExplorer, TorX, UPPAAL-Tron
Symbolic execution
[S08, **S17**, S22] [12]
2 tools: Conformiq, LTG
Theorem proving
[**S17**] [12]
2 tools: Isabelle/HOL, PrUDE
Constraint solving
[12]
2 tools: BZ, GATel
UML-based
[S14, S15, S22]
Axiomatic
[S21]
Labeled transition system
[S21]
FSM-based
[S09, S21] [12]
Test execution [12, 17]
Technology [S03, S17][12]
Online
[**S03**, **S10**, **S17**] [12, 17]
15 tools: AsmL, Conformiq, GraphWalker, JSXM, JTorX, MISTA, ModelJUnit, NModel, PyModel, SpecExplorer, TEMA, TestOptimal, TGV, TorX, UPPAAL-Tron
Offline
[**S03**, **S10**, **S17**, S22] [12, 17]
30 tools: AETG, AGEDIS, BZ, Smarttesting CertifyIt, CompleteTest, Conformiq, CoVer, GATel, GOTCHA, GraphWalker, Isabelle/HOL, JSXM, JUMBL, LTG, MISTA, ModelJUnit, NModel, ParTeG, ProTest, PrUDE, PyModel, Reactis, SpecExplorer, TAF, Tedeso, TestCast, TestMaster, TestOptimal, TestUML, TorX
Conformance check [S10]
Conformance relations
[**S10**, S20, S22]
2 tools: SpecExplorer, TorX
Oracles
[**S10**, S20]
14 tools: AETG, AGEDIS, Conformiq, GOTCHA, GraphWalker, JUMBL, NModel, ParTeG, PyModel, TAF, TestCast, TestMaster, TestOptimal, UPPAAL-Tron
Mapping [S17]
Abstract tests
[**S17**]
9 tools: BZ, CoVer, GOTCHA, Isabelle/HOL, ProTest, Reactis, SeDiTeC, SpecExplorer, TGV
Executable tests
[**S17**]
24 tools: AGEDIS, AsmL, CompleteTest, Conformiq, GATel, GraphWalker, JSXM, JTorX, JUMBL, LTG, MISTA, ModelJUnit, NModel, ParTeG, PrUDE, PyModel, Simulink Verification and Validation, Tedeso, TEMA, TestCast, TestMaster, TestOptimal, TestUML, UPPAAL-Tron
Scaffolding [S19]
Test drivers
[**S19**]
9 tools: AGEDIS, Smarttesting CertifyIt, GOTCHA, GraphWalker, NModel, ParTeG, SpecExplorer, TestMaster, TestOptimal
Test stubs
[**S19**]
2 tools: Smarttesting CertifyIt, SeDiTeC
Verdicts
[**S19**]
4 tools: AGEDIS, Conformiq, GOTCHA, GraphWalker

Fig. 2: (Continued) MBT areas investigated by secondary studies.

to the tool classifications found in secondary studies. We found 11 secondary studies that mentioned MBT tools. Of these, only 5 gathered multiple tools and classified them in some dimensions. The other 6 papers simply mentioned tools. Several studies hinted the existence of tools, but offered no name for them. For the purpose of our study, only tools with a name were considered.

Studies that classified tools used some common criteria that can be generalized into: model specification [S03, S10, S19, S20], test selection criteria [S10, S17, S19], test generation criteria [S10, S17], and test execution related [S10, S17, S19]. Model specification was the most common criterion used to classify tools, with 4 studies. Model specification refers in most cases to the model paradigm in Figure 2, but some also include model scope, language type and model characteristics. The next most common criteria used to classify tools were support for test selection criteria and test execution criteria, each with 3 studies. Test selection criteria can be found in the second level of Figure 2, while test execution criteria can be found in the first level. The least common criteria to classify tools was support for test generation criterion, with only 2 studies. Test generation refers to the generation method in the second level of Figure 2.

Marinescu et al. [S17] perform a state-of-the-art survey in tool-supported model-based testing by presenting and classifying some of the most mature tools available in the literature. The study approaches this by proposing a taxonomy based on the one provided by Utting et al. [12] and others. Many tools were characterized according to this new taxonomy. Shafique and Labiche [S19] study the support of MBT tools that rely on state-based models. They attempt this by finding these tools and comparing them according to many criteria, mainly test coverage. This is one of the studies that perform a more in-depth tool classification. The tools were characterized by their support of three general criteria: test, test scaffolding, and support for related activities criteria. These criteria are further sub-divided into more specific classes, like model-flow, script-flow, and data-flow coverage criteria. Saifan and Dingel [S10] perform a survey on how MBT has been used to test quality attributes of software. In their work, they extend the taxonomy of Utting et al. [12] by adding three additional criteria and characterize tools based on this categories. Siavashi and Truscan [S20] perform a study on the use of environment models in MBT. They gather a list of tools that use these models, and group them by model language. Mahdian et al. [S07] perform a survey on regression testing that employs UML models. In their study, they cite studies that research on regression approaches, types of UML diagrams, and the tool proposed.

The studies that did not perform a classification of MBT tools were also considered for this study. In general terms, they mention tools as examples of a particular approach or model, or comment on primary studies that propose and experiment with those tools. The information found in these type of studies was: tool description, the tool's model or language, tool's functioning, studies that proposed or employed the tool, and its usage in industry. The information found in these kind of studies was very heterogeneous, but it allowed to identify tools that were not cited in other papers.

We found a total of 99 MBT tools in the analyzed papers. Approximately 39 of these tools had little or no information, so they could not be associated to MBT areas. The table is sparse for most of the tools and some areas, indicating that no study covers all the dimensions found. The complete list of tools and their characterization can be found at `https://goo.gl/gX2WKe`. We classified these tools with respect to the hierarchy of areas depicted Figure 2, in order to find areas with tool support. For each low-level area ('leaves' of the hierarchy tree), we list the tools that support that area, and highlight (with boldface) the secondary studies from which the tools were extracted.

4.3 RQ3: Challenges, limitations and open problems

This research question directed us to identify the challenges, open problems and limitations that have been reported in the secondary studies. Next we describe the main challenges and limitations with respect to the factors of influence and empirical evidence for MBT approaches and tools.

We found that more empirical evidence is needed to support the selection of MBT approaches and tools. There is only a few evidence about experiences in the industry [S06], and case studies on practice using different approaches are needed [S03]. Moreover, evidence are usually published biased towards success stories [S03]. More research is needed in evaluating and analyzing the applicability of different models in MBT [S20] and the landscape of MBT approaches and tools should be compared [S07, S19]. It is necessary to evaluate not just small modules, but full systems to find the better suited approach for an specific context [S04] Evidence showing MTB approaches capabilities compared to conventional testing techniques is also required [S04]. Further, more evidence in checking a system against non-functional aspects should be investigated [S16, S20]. MBT approaches have to be evaluated on non-functional characteristics such as safety, security, usability, S10, S16, S17, S20]. In addition, research on the comparison and evaluation on how various criteria relate to each other in terms of the coverage and fault detection is required [S02]. There is no conclusive evidence about how one model outstands others when more than one model is appropriate for a specific context [S11]. Therefore, decisions need to be made based on what model (or set of models) are most suitable [S20].

In practice, evidence in determining whether MBT approaches are useful to different domains and contexts is necessary to support the selection process. Different factors such as type and level of testing, models, types of software projects, tool characteristics should be studied to objectively measure the performance, effectiveness, complexity, costs, effort, flexibility, feasibility, advantages and benefits [S03, S04, S05, S06, S13, S20]. Many MBT approaches have not been empirically evaluated or not transferred to the industrial environments [S04]. The understanding about the similarities and differences among approaches could lead to determining appropriate MBT techniques and tools in specific operating contexts in practice. For [S11], a real work that remains is fitting specific models to specific application domains because many models have implicit drawbacks. Also, the number of techniques proposed for test case generation is very large and a deeper insight on the techniques proposed is needed [S16]. In some cases, the generated test cases may get irreverent due to the disparity between a model and its corresponding code [S11]. A key step in any MBT methodology is the validation of the model, this is an essential step in resolving specification ambiguities and synchronizing the understanding of the requirements [S03].

It is still a challenge to expand modeling notations in complex systems and development environments [S20]. Models should consider concurrent sub states and events [S13]. Models state identification and verification are still a challenge in dealing with the state explosion problem [S01]. A prominent problem for state models is state space explosion, as it propagates to almost all MBT tasks such as model maintenance, checking and review, test case generation and achieving coverage criteria [S11]. Some approaches use genetic algorithms for test case generation [S13]. These techniques could provide feasible test sequences [S13] but also suffer on providing balanced transition coverage and unfeasible transition and looping problems [S13]. Many studies ignore concurrent and integration testing resulting in less efficient test cases generation [S14]. In some cases the test adaptation is manually implemented and need multiple test adaptations [S20]. Ensuring adequate traceability or coverage of test scenarios is needed [S14]. In terms of test case generation assessment, most of the studies are based on coverage criteria [S13, S16, S19]. Complexity levels depend on modeling language and algorithms necessary to perform each step in the MTB process [S04]. Models for nontrivial software functionality can grow beyond management even with tool support [S11]. Finding the right level of abstraction for the test model is one of the challenges in MBT [S20].

The choice of a model also depends on aspects of the system under test and skills of user [S10, S11]. One of the disadvantages of the languages for modelling is that they are usually complex to learn and use [S03]. Requirements to use a MBT approach include knowledge about the modeling language, testing coverage criteria, generated output format and supporting tools. These make the adoption of MBT difficult or unfeasible [S04]. Moreover, practitioners require different skills and expertise in modeling notations, test criteria, test metrics, and tools [S04, S11]. Most testers would probably prefer visual languages for understanding a model [S03]. There is a necessity of increasing automation using simple and known notations. MBT approach must be obtainable with low effort, time, and skill to perform the its steps [S04, S11]. For practitioners, the leap from traditional scripted testing to MBT seems as hard as moving from manual to automatic test [S03]. Creating models and keeping them updated demands require significant time and effort [S03, S06]. Practitioners also perceived that they perform system design twice, once for development and once for testing [S03]. It is necessary to be clear that MBT is an activity that requires investment, planning and knowledge [S11].

Some of the main risks associated to MTB are related to the quality assurance of the artifacts used for test generation, the testing schedule planning, the selection of MBT approaches, the behavior or structural model construction, the selection criteria for test cases, tracking and change managing, manual tasks in the MBT process, test generation and execution process controlling [S05]. To increase the practical application of MBT, tools have to support modeling and test generation steps. It is necessary to integrate automated and manual steps because the complex no automated steps makes any approach unfeasible. Thus, increasing the automation levels of the steps in a MBT approach and reducing requirements to use these approaches determines the feasibility of using MBT. The construction of models is not a trivial task due to the lack of a systematic methodology and of supporting tools for its automation [S20]. The MBT testing tools should be integrated with the software development processes [S04, S11, S14, S20].

There exist many MBT tools which vary significantly in their specific designs, testing target, tool support, and evaluation strategies [S08]. MBT tools generate many test cases and not having full automation support for managing could make the use of MBT infeasible [S19]. Practitioners faced problems related with the learning and understanding of the features of MBT tools [S19, S20]. Many of them can generate executable test cases but still some just generate abstract tests [S17]. In many cases, there are no tools available for modeling and generating tests in a particular domain and custom made tools are needed [S03]. In other cases, MBT tools are not available to the public [S18, S21]. Furthermore, most of open source tools have incomplete or outdated documentation [S19].

5 Conclusions and future work

In this paper, we have reported the results of a tertiary study that performs a systematic mapping of secondary studies in MBT, published between 1996 and 2016. From an initial set of 47 secondary studies, we ended up analyzing 22 papers (10 systematic reviews and 12 literature surveys), after applying the exclusion and inclusion criteria. The complete data extraction form is available online. We created a hierarchy of MBT areas, partially based on taxonomies from previous studies. We also categorized the MBT tools into this hierarchy. Finally, we grouped and classified the main challenges and limitations found in secondary studies.

Our main findings regarding the areas of MBT that have been investigated are the following: (1) the two most studied areas in MBT are model paradigms, particularly UML and Transition-based notations, (2) the most studied test case selection criteria is Structural model coverage, (3) the most studied test case generation methods are Random and Search-based algorithms, and (4) the most studied test execution area is Technology: online/offline.

With respect to the characterization of MBT tools, we found 99 MBT tools which were classified according to model specification, test selection criteria, test generation criteria, and test execution. We found that there are many MBT areas not supported by tools. These include testing of non-functional characteristics, modeling of architecture and environment, multiple modeling paradigms (history-based, functional, and operational), several coverage criteria (data-flow, data, and fault-based), specific test generation techniques (axiomatic and UML-, LTS-, and FSM-based), and test scaffolding support. Very few studies categorize tools by test level, model scope, modeling language, and model characteristics. Also, no studies reported the SUT domain or paradigm targeted by the tool.

The main challenges and limitations stated by secondary studies were: (1) more empirical evidence is needed to support the selection of MTB approaches and tools, (2) an understanding of the similarities and differences among MBT approaches could lead to choose the appropriate technique and tool in specific operating contexts, and (3) both functional and non-functional testing should be considered in MBT studies.

Since there are few secondary studies on MBT, we consider important to develop studies encompassing the following areas: test execution aspects such as scaffolding and mapping of test cases, languages types (e.g., design, test-specific and generic languages), model dynamics (e.g., discrete, continuous, hybrid), model paradigms like history-based and functional, generation methods like constraint solving, axiomatic and LTS, as well as different software paradigms.

A recommendation for the MBT researchers is to use a systematic approach when performing secondary studies, and follow standard reporting procedures. Systematic reviews scored high in our quality assessment, and tended to offer information in a more structured and useful way. We also encourage the use of existing taxonomies for MBT (including the one presented in this work) when performing primary or secondary studies. Classifying tools and approaches under known classification schemes makes information easier to extract for others, and more comparable across papers. MBT is gaining popularity as a research area, and we believe that the adoption of a systematic methodology is important in order to increase the quality of the studies.

In the future, we plan on expanding this tertiary study to cover recently published secondary studies in MBT (2017-2018) and analize additionally aspects of these studies. A recent search yielded a number of interesting and relevant systematic studies that we could be interested in reviewing.

6 Acknowledgements

This work was supported by University of Costa Rica (UCR)'s project No. 834-B7-749. We thank the SE Research Group at UCR for the valuable feedback and help.

References

1. Bourque, P., Fairley, R.E., et al.: Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press (2014)
2. Utting, M., Legeard, B.: Practical model-based testing: a tools approach. Morgan Kaufmann (2010)
3. Pretschner, A., Prenninger, W., Wagner, S., Kühnel, C., Baumgartner, M., Sostawa, B., Zölch, R., Stauner, T.: One evaluation of model-based testing and its automation. In: Proceedings of the 27th international conference on Software engineering, ACM (2005) 392–401
4. Garousi, V., Mäntylä, M.V.: A systematic literature review of literature reviews in software testing. Information and Software Technology **80** (2016) 195–216
5. Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S.: Systematic literature reviews in software engineering–a systematic literature review. Information and software technology **51** (2009) 7–15
6. Kitchenham, B., Pretorius, R., Budgen, D., Brereton, O.P., Turner, M., Niazi, M., Linkman, S.: Systematic literature reviews in software engineering–a tertiary study. Information and Software Technology **52** (2010) 792–805
7. Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology **64** (2015) 1–18
8. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. In: Technical report, Ver. 2.3 EBSE Technical Report. EBSE. School of Computer Science and Mathematics, Keele University (2007)
9. Biolchini, J., Mian, P.G., Natali, A.C.C., Travassos, G.H.: Systematic review in software engineering. System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES (**679**)
10. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering, ACM (2014) 38
11. Mourao, E., Kalinowski, M., Mendes, E., Wohlin, C.: Investigating the use of a hybrid search strategy for systematic reviews. (In: Proceedings of the 11th international symposium on empirical software engineering and measurement)
12. Utting, M., Pretschner, A., Legeard, B.: A taxonomy of model-based testing approaches. Software Testing, Verification and Reliability **22** (2012) 297–312
13. ISTQB®: Foundation level certified model-based tester. (2015)
14. Anand, S., Burke, E.K., Chen, T.Y., Clark, J., Cohen, M.B., Grieskamp, W., Harman, M., Harrold, M.J., Mcminn, P., Bertolino, A., Li, J., Zhu, H.: An orchestrated survey of methodologies for automated software test case generation. Journal of Systems and Software **86** (2013) 1978–2001
15. Budgen, D., Brereton, P., Drummond, S., Williams, N.: Reporting systematic reviews: Some lessons from a tertiary study. Information and Software Technology (2017) 62–74
16. Marinescu, R., Seceleanu, C., Le Guen, H., Pettersson, P.: Chapter three-a research overview of tool-supported model-based testing of requirements-based designs. Advances in Computers **98** (2015) 89–140
17. Utting, M., Legeard, B., Bouquet, F., Fourneret, E., Peureux, F., Vernotte, A.: Chapter two - recent advances in model-based testing. Volume 101 of Advances in Computers. Elsevier (2016) 53 – 120