Model-based testing areas, tools and challenges: A tertiary study

Leonardo Villalobos-Arias, Christian Quesada-López,
Alexandra Martínez, Marcelo Jenkins
Universidad de Costa Rica, San José, Costa Rica
{leonardo.villalobosarias, cristian.quesadalopez,
alexandra.martinez, marcelo.jenkins}@ucr.ac.cr

Abstract

Context: Model-based testing is one of the most studied approaches by secondary studies in the area of software testing. Aggregating knowledge from secondary studies on model-based testing can be useful for both academia and industry. **Objective:** The goal of this study is to characterize secondary studies in model-based testing, in terms of the areas, tools and challenges they have investigated. **Method:** We conducted a tertiary study following the guidelines for systematic mapping studies. Our mapping included 22 secondary studies, of which 12 were literature surveys and 10 systematic reviews, over the period 1996–2016. **Results:** A 3-level hierarchy of model-based testing areas was built based on existing taxonomies as well as data that emerged from the secondary studies themselves. This hierarchy was then used to classify the studies, tools, challenges and their tendencies in a unified classification scheme. The most studied areas overall are two model paradigms: UML models and transition-based notations, and two test levels: unit and integration testing. Only five studies were found to compare and classify model-based testing tools, which motivated us to classify all tools found into our hierarchy of areas. Most tools fell within the model paradigm area. Over time, tools that test the functional behavior have prevailed, with a recent tendency to support executable tests. With regard to model-based testing challenges, most of them are associated to the model specification area. Besides, a grounded analysis was done on challenges, yielding six categories. Availability is the category where more challenges are reported. Over time, challenges have moved from complexity to lack of approaches for specific software domains. **Conclusions:** Only a few systematic reviews on model-based testing could be found, therefore some areas still lack secondary studies, particularly, test execution aspects, language types, model dynamics, as well as some model paradigms and generation methods. We thus encourage the 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 model-based testing.

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

1 Introduction

Model-based testing (MBT) is a commonly studied software testing approach in secondary studies. MBT attempts to automate parts of the software testing process, primarily the test design and the test case execution stages. It has been found that MBT detects at least as many errors as manual testing [1]. 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 meeting certain criteria such as code coverage [2]. Finally, tools for model-based testing can help with requirements traceability and evolution, both of which can be difficult to achieve in traditional testing processes [2].

Model-based testing suffers from some limitations. First, it requires a tester with a different skill set than that for manual testing [2]. In particular, the tester needs to be able to abstract and design the model from the requirements, which means she has to know and decide upon which aspects of the software under test (SUT) should be model or not. Second, training costs are high (due to the special skills needed) and therefore, there is a learning curve before MBT can be used [2]. Third, a considerable investment is required to build the model, since the quality of generated tests depends on the quality of the model.

Recently, a few tertiary studies have been conducted in software engineering [3, 4] and software testing [5].

According to Garousi et al. [5], MBT is the testing method with more secondary studies. There are no tertiary studies specific to MBT, to the best of our knowledge. A tertiary study that aggregates and synthesizes knowledge on MBT from secondary studies can be useful to both academia and industry. The academia benefits from tertiary studies by pointing knowledge areas in which further research is required, and the industry benefits from the information regarding novel tools and techniques uncovered by tertiary studies[4].

To address the lack of tertiary studies on model-based testing, our research goal was to identify and characterize secondary studies in model-based testing, in terms of the areas, tools, and challenges investigated ¹. We conducted a tertiary study based on the guidelines described in [7, 8] as well as the recommendations stated in [9, 10, 11]. The studies were characterized in terms of specific MBT areas, model types, test selection criteria, test generation and test execution techniques, testing tools, and challenges. 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 which specific MBT areas have been addressed (or not) 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 existing classifications for MBT tools, as well as the particular characteristics exhibited by current tools, according to these classifications. Furthermore, we will classify MBT tools according to the areas hierarchy derived from RQ1, as a unified classification scheme.
- RQ3. What are the challenges reported in the literature of model-based testing? This question will allow us to identify common challenges of current research on MBT, classify these challenges according to the MBT areas hierarchy derived from RQ1, and devise potential areas of future research.

The analysis derived from our research questions RQ1 to RQ3 will enable us to identify areas, tools and challenges in MBT research. It is also interesting to analyze tendencies of these three aspects over time, with the underlying motivation of recognizing which ones are booming, declining or stable in time.

In this paper, we extend our previous publication [6] by extending the analysis performed for each research question to include trends in areas, tools and challenges over time, which aims at discovering how research in model-based testing has evolved in time. In addition, we expanded the results to include counts and percentages of studies and tools per MBT area, which allowed us offer a richer discussion of the results. Moreover, we performed a new grounded analysis of challenges, resulting in six categories: Efficacy, Availability, Complexity, Professional Skills, Investment, Cost & Effort, and Evaluation & Empirical Evidence. Furthermore, each challenge was also classified according to the top-level areas of our hierarchy. We analyzed these data in three different ways: challenges by categories, challenges by areas, and the relationship between categories and areas.

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.

¹An earlier version of this paper was presented at the Ibero-American Conference on Software Engineering (CIBSE 2018) and was published in its Proceedings [6].

Figure 1: Model-based testing process (adapted from [2] and [13]).

2 Background

Software testing consists of verifying if a program matches its expected behavior [12]. 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 [12]. 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 four stages [13], as illustrated in Figure 1: (1) building the model of the SUT from the requirements, (2) choosing the test selection criteria, (3) generating test cases as either abstract test sequences or concrete test scripts, and (4) executing the test cases. The model should be built for testing purposes (i.e., simpler than the model used for development) [2]. An MBT tool may provide support for activities in any of these stages.

Utting et al. [14] 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 executing the test cases. Authors suggest that other taxonomies may be defined to express different aspects of MBT’s approaches or tools.

Several benefits of MBT over traditional testing have been identified [2]. First and foremost, MBT has proven to be more likely to find errors than traditional testing. While the quality of the testing still relies on the experience of the tester (who either creates the model or the actual test cases), its effectiveness is still unquestionable. Second, MBT reduces testing time and costs. Third, test cases generated by an MBT tool are typically of higher quality than those manually written, since the tools can generate large test sets with good coverage (more consistently than a tester). Fourth, MBT can help detect defects in software requirements, given that some failures occur due to unspecified, unknown or wrong behavior from

the requirements. Fifth, MBT makes it easier to implement new requirements because the model is usually smaller and simpler than a test suite. Sixth and last, MBT offers better traceability from test cases to informal requirements and test selection criteria.

Limitations of MBT have also been recognized [2]. MBT requires a different set of skills, compared to traditional testing. A skillful traditional tester may find MBT difficult to adopt because of this. Besides, MBT is normally recommended for organizations with certain level of maturity in their testing process, due to its complexity. Another shortcoming is that MBT is usually applied for functional testing only. Furthermore, MBT tends to have problems with outdated requirements, tends to consume a significant amount of time in manual analysis of failed tests, and can generate many useless test cases [2].

The International Software Testing Qualifications Board's Foundation Level Certified Model-Based Tester Syllabus [15] 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. [16] 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, such as Conformiq Designer, Smartsensing CertifyIt, and Spec Explorer.

Utting et al. [13] report on the updates on the literature of model-based testing in the last 10 years. In this chapter, they give an overview of the updated model-based testing process and its history, and the current state of MBT adoption in the industry. They also present the current modeling languages and test case generation technologies. At the end of the chapter, they suggest recommendations for the MBT research community and offer some ideas in order to assess the existing challenges.

We have found few tertiary studies that include model-based testing as one of the studied approaches, such as [5], but none in which MBT is the main topic. Our study will perform a deeper, more accurate mapping of model-based testing than the existing tertiary studies. We believe it will be useful for those new to MBT research, as a guide for other studies.

3 Methodology

Our tertiary study was conducted following the methodology stated in [7, 8] and the recommendations in [9, 10, 11, 17]. In particular, our search strategy was hybrid, as recommended in [11]. First, we conducted an automated search in Scopus and Web of Science engines to select a relevant seed set of secondary studies. Then, we performed a backward snowballing to allow the identification of new relevant papers.

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 of 2017. We looked for studies that contained the search terms anywhere in the title, abstract or keywords fields.

3.1.1 Inclusion and exclusion criteria

Articles meeting the following criteria were included:

- I1. Language: English
- I2. Type of study: Secondary study (survey, map or review)
- I3. Approach: Model based testing

Articles meeting the following criteria were excluded:

- E1. Papers not available in full text
- E2. Publications not related to the software engineering domain
- E3. Non peer reviewed publications

Figure 2: Overview of the selection process used in the study.

When a secondary study was published in more than one journal or conference, both versions of the study were compared for the purpose of data extraction.

3.1.2 Selection process

Our study selection process is depicted in Figure 2. The search string produced an initial set of 47 potentially relevant studies. The inclusion and exclusion criteria were applied to these studies, considering only the title, abstract and keywords fields. Any paper that was irrelevant became excluded. The following steps were performed during the inclusion and exclusion process: (i) all papers were evaluated by five members of the research group, (ii) each member independently screened the paper for inclusion and exclusion criteria, and decided whether or not to include the paper, offering a justification, (iii) 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. In general, we tended to include rather than exclude potentially relevant papers. After applying the exclusion criteria, we were left with 13 studies [S03, S04, S06, S08, S10, S11, S16, S17, S18, S19, S20, S21, S22].

To expand the scope of our automated search, backward snowballing was manually applied to the remaining secondary studies. We further applied backward snowballing to 3 relevant tertiary studies [5, 3, 4]. This expansion process yielded 9 more papers [S01, S02, S05, S07, S09, S12, S13, S14, S15]. Our automated search was not able to find these papers because they did not match our search string: some targeted specific models such as UML and FSMs; some used a name other than model-based testing, for example, a formal testing approach; and yet others focused on a specific aspect of MBT like test case generation.

In the end, a total of 22 papers were used for the analysis. From these, 12 were literature surveys [S01, S02, S03, S07, S08, S09, S10, S11, S13, S15, S17, S21] and 10 were systematic reviews [S03, S04, S06, S12, S14, S16, S18, S19, S20, S22]. Figure 3 shows the number and type of our final set 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://tinyurl.com/yc29yx16>.

3.2 Quality assessment

The quality of the secondary studies was evaluated using the Database of Abstracts of Reviews of Effects (DARE) criteria², as in [4, 17]. The original DARE assessment is based upon four questions that aim to evaluate how well the systematic review process is reported. These questions are:

- Q1. Are the study's inclusion and exclusion criteria explicitly and clearly defined? Y (yes): the inclusion criteria are explicitly and clearly defined; P (partially): the inclusion criteria are implicit; N (no):

²<https://www.crd.york.ac.uk/CRDWeb/Homepage.asp>

Figure 3: Number and type of secondary studies in time.

Table 1: Quality assessment of secondary studies, based on DARE.

ID	Ref.	Type	Year	Guid.	Q1	Q2	Q3	Q4	Overall score
S01	[18]	LS	1996		0	0	0	0.5	0.5
S02	[19]	LS	2005		0	0	0	1	1
S03	[20]	LS	2006		0	0	0	1	1
S04	[21]	SR	2007	[8, 9]	1	1	0	1	3
S05	[22]	SR	2008	[8, 9]	1	1	0	1	3
S06	[23]	SR	2008	[8, 9]	1	1	0	1	3
S07	[24]	LS	2009		0	0	0	0	0
S08	[25]	LS	2009		0	0	0	0.5	0.5
S09	[26]	LS	2010		0	0	0	0	0
S10	[27]	LS	2010		0	0	0	1	1
S11	[28]	LS	2010		0	0	0	0	0
S12	[29]	SR	2010	[8, 9]	1	1	0.5	1	3.5
S13	[30]	LS	2012		0	0	0	1	1
S14	[31]	SR	2012		1	0.5	0	1	1.5
S15	[32]	LS	2013		0	0	0	1	1
S16	[33]	SR	2014	[8]	1	1	1	1	4
S17	[34]	LS	2015		1	1	0	1	3
S18	[35]	SR	2015	[7, 8, 9]	1	0.5	1	1	3.5
S19	[36]	SR	2015	[8]	1	1	0	1	3
S20	[37]	SR	2015	[3]	1	1	1	1	4
S21	[38]	LS	2016		0	0	0	0	0
S22	[39]	SR	2016	[8, 3]	0.5	1	0.5	1	3

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 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.

A commonly used convention in DARE is to score each question on the following 3-point scale: Yes = 1, Partially = 0.5, and No = 0, and then add the scores of individual questions for an overall score. A high overall DARE score is usually interpreted as representing a high quality secondary study, meaning we have confidence in its findings [17].

The quality assessment of our 22 papers is presented in Table 1. We observe that all but one of the systematic reviews (SR) exhibited high overall DARE scores: between 3 and 4 (4 being the maximum possible score). On the other hand, only one literature survey (LS) attained an overall DARE score of 3, while the others obtained overall scores of 1 or below. Studies with low quality scores were not excluded from our subsequent analysis. This decision was taken because DARE is inherently biased towards systematic literature reviews and their formal and rigorous process [17], usually granting low quality scores to less formal secondary studies such as surveys. Furthermore, DARE would assign high scores to secondary studies that actually report or describe the process steps in detail, yet it does not (and possibly cannot) assess whether the steps were correctly performed.

3.3 Data extraction and analysis

The extraction process was performed in four steps, which are described next. The first step consisted in randomly assigning each paper to a pair of researchers (from the research team, including the authors of this paper). In the second step, each researcher independently screened the assigned papers for extraction. The third step consisted in having each pair meet and compare their individual results, discussing about

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

any possible disagreements until a consensus was reached, and this decision was documented. In the fourth and final step, one author of this paper validated the data extraction of one of the research questions, and complemented any information as needed.

We analyzed all 22 secondary studies and extracted any relevant data that helped answering 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 the extracted data. No statistical data analysis (such as meta-analysis) was performed due to the heterogeneity found among secondary studies.

For RQ1, our initial hierarchy was the union of the categories in our reference taxonomies [14, 34] ([34] is S17 in Table 1). Then, we kept adding categories (areas) that emerged from the secondary 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 characteristic into an upper-level node/area. We tried to respect the original category names given by the authors, but sometimes we had to choose (whenever 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, model-based testing tools mentioned in secondary studies were identified. We extracted information reported about each tool, usually based on an existing classification/taxonomy. We then attempted to group them in common categories. Furthermore, we classified each tool based on the hierarchy of areas derived from RQ1 (Section 4.1), as a way to standardize the classification of all tools found. Finally, we also analyzed tendencies of tools over time, along the MBT areas discovered in RQ1.

For RQ3, challenges reported in secondary studies were identified. Then, a grounded analysis was performed to allow categories to emerge from the data. These categories were associated with challenges previously reported in the literature [21, 23, 22, 2, 6]. Afterwards, each challenge was classified based on the hierarchy of areas derived from RQ1 (see Section 4.1), in an attempt to standardize the classification of data and the presentation of results. Lastly, we analyzed tendencies of challenges in time, across the MBT areas discovered in RQ1.

3.4 Threats to validity

We briefly discuss here what are the main threats to the validity of our 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 engines, but to counteract, we applied snowballing, as recommended by [11]. The snowballing process was a backward process based only on titles, implying the risk of missing relevant studies. Thus, 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-based 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 the areas or categories used in RQ1 through RQ3 were based on previous MBT studies. Nevertheless, some subjective interpretation took place when merging categories and determining whether a study provided support for a specific category.

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

The RQ1 research question helped us identify MBT areas that have been investigated by secondary studies, as well as their tendencies over time. 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. [14] (aimed for MBT approaches) and other proposed by Marinescu et al. [34] (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 3-level hierarchy of MBT areas is shown in Figure 4. 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). The coloring scheme used in the third level of our hierarchy is as follows: red nodes are highly investigated areas (with 9 or more studies), blue nodes are moderately investigated areas (between 5 and 8 studies), uncolored (or white) nodes are low investigated areas (between 2 and 4 studies), and yellow nodes are scarcely investigated areas (with 1 or no studies).

Some collateral findings emerged during the construction of our hierarchy. Most of them were inconsistencies among classifications offered by different authors. For example, [S12] classified FSM and LTS as state-based models, while Utting et al. [14] and [S17] classified them as transition-based models in their taxonomies. Another example is regression testing, that is classified as a test level by [S04] and as a test type by us and Utting et al. [13]). Additionally, [S21] considered search-based algorithms for test case generation (also known as meta-heuristic search) as a category outside MBT, while other studies [S13, S15, S21][14] included it as a test generation method within MBT.

From Figure 4 we see that the two most studied areas throughout the hierarchy are UML models (with 16 studies) and transition-based notations (with 12 studies), both belonging to the model paradigm level-2 area and model specification level-1 area. Likewise, the third and fourth most studied areas throughout the hierarchy are unit testing and integration testing, both belonging to the test level level-2 area and test objectives level-1 area.

Table 3 shows the number of secondary studies found per area of our hierarchy (the specific studies mapped to each area can be found in Figure 4). A single study may provide support for one, multiple, or no areas at any level, hence, it could potentially be added to the paper counts of different areas. However, it would only be counted once for the higher-level (parent) area, to avoid over-counting. Therefore, paper counts of the subareas do not necessarily add up to the total paper count of their parent area, the same way that paper counts do not add up to our total number of secondary studies (22). Table 3a shows a top-level view of the number of studies per level-1 area of our hierarchy. We observe from this table that the level-1 area with most studies (19) is model specification, followed in second place by test objectives (with 14 studies), and third place by test generation (with 12 studies). The level-1 area with less studies is test execution (with just 5 studies). Next we describe the most and least investigated subareas (level-2 and level-3) for each of the top (level-1) areas of our hierarchy.

SUT. Table 3b shows the number of studies for level-2 and level-3 areas belonging to the SUT area. The level-2 area with more studies is domain (with 8 studies), while the level-2 area with least studies is software paradigm (with 4 studies). The most studied level-3 area of the software paradigm area is object-oriented, with 4 studies. Regarding the domain area, all its level-3 areas (embedded systems, real-time systems, reactive systems, and web applications) have 4 studies each.

Test objectives. Table 3c shows the number of studies for level-2 and level-3 areas of the test objectives area. The level-2 area with more studies is test level, with a total of 11 studies, whereas the level-2 area with least studies is test artifact, with 8 studies. Among the most studied level-3 areas of test artifact are functional

Figure 4: MBT areas investigated by secondary studies.

Figure 4: (Continued) MBT areas investigated by secondary studies.

Figure 4: (Continued) MBT areas investigated by secondary studies.

Table 3: Number of secondary studies per MBT areas.

(a) Number of studies per level-1 areas.

Level 1 area	Papers (22)	
	Count	Percentage
SUT	10	45%
Test objectives	14	64%
Model specification	19	86%
Test generation	12	55%
Test execution	5	23%

(b) Number of studies per level-2 and level-3 areas belonging to the **SUT** area.

Level 2 area	Level 3 area	Papers (22)	
		Count	Percentage
Software paradigm (4)	Object-oriented	4	18%
	Component-oriented	2	9%
	Cyber-physical-oriented	1	5%
	Aspect-oriented	1	5%
	Service-oriented	1	5%
Domain (8)	Embedded systems	4	18%
	Real-time systems	4	18%
	Reactive systems	4	18%
	Web applications	4	18%
	Safety-critical systems	2	9%
	Concurrent software	3	14%
	Distributed systems	3	14%
	Web services	2	9%
	Mobile applications	1	5%
	Software product line	1	5%

(c) Number of studies per level-2 and level-3 areas belonging to the **test objectives** area.

Level 2 area	Level 3 area	Papers (22)	
		Count	Percentage
Test artifact (8)	Functional behavior	6	27%
	Extra-functional behavior	5	23%
	Architectural description	2	9%
	Environment	2	9%
Test type (10)	Functional testing	5	28%
	Non-functional testing	4	18%
	Regression	5	28%
Test level (11)	Unit	10	45%
	Integration	9	41%
	System	5	23%
	Acceptance	0	0%

Table 3: (Continued) Number of secondary studies per MBT areas.

(d) Number of studies per level-2 and level-3 areas of the **model specification** area.

Level 2 area	Level 3 area	Papers (22)	
		Count	Percentage
Model scope (0)	Input-only	0	0%
	Input-output	0	0%
Language type (2)	Design	1	5%
	Test-specific	1	5%
	Domain-specific	2	9%
	Generic	1	5%
Model paradigm (19)	State-based	5	23%
	Transition-based	12	55%
	Stochastic	4	18%
	Data-flow	1	5%
	History-based	0	0%
	Functional	0	0%
	Operational	1	5%
	Attribute event grammar	1	5%
Model characteristics (2)	UML	16	73%
	Timing	2	9%
	Non-determinism	2	9%
	Dynamics	1	5%

(e) Number of studies per level-2 and level-3 areas belonging to the **test generation** area.

Level 2 area	Level 3 area	Papers (22)	
		Count	Percentage
Selection criteria (6)	Structural model coverage	6	28%
	Data coverage	3	14%
	Fault-based coverage	2	9%
	Requirements-based coverage	3	14%
	Ad-hoc test case specification	1	5%
	Stochastic & Random	2	9%
Generation method (9)	Manual	1	5%
	Random	4	18%
	Graph search	3	14%
	Search-based algorithms	4	18%
	Model-checking	1	5%
	Symbolic execution	3	14%
	Theorem proving	1	5%
	Constraint solving	0	0%
	UML-based	3	14%
	Axiomatic	1	5%
	Labeled transition system	1	5%
	Finite-state machines	2	9%

Table 3: (Continued) Number of secondary studies per MBT areas.

(f) Number of studies per level-2 and level-3 areas belonging to the **test execution** area.

Level 2 area	Level 3 area	Papers (22)	
		Count	Percentage
Technology (4)	Online	3	14%
	Offline	4	18%
Conformance check (3)	Conformance relation	3	14%
	Test oracle	2	9%
Mapping (1)	Abstract test	1	5%
	Executable test	1	5%
Scaffolding (1)	Test adapter	1	5%
	Test stubs	1	5%
	Test oracles	1	5%

behavior (or properties), with 6 studies, and extra-functional behavior (or non-functional aspects), with 5 studies. Among the least studied level-3 areas of test artifact are architectural description and environment, each with 2 studies. The most studied level-3 areas of test type are functional and regression testing, with 5 studies each; while the least studied is non-functional testing, with 4 studies. On the other hand, the most studied level-3 areas of test level are unit testing, with 10 studies, and integration testing, with 9 studies. Interestingly, we did not find any study on acceptance testing, and only 5 on system testing.

Model specification. Table 3d shows the number of studies for level-2 and level-3 areas belonging to the model specification area. The model specification's level-2 area with more studies is model paradigm (with a total 19 studies), while the least studied level-2 area is model scope (with 0 studies). None of the model scope's level-3 areas have studies. Regarding language type's level-3 areas, the most studied is domain-specific, with 2 studies. All the other level-3 areas of language type have 1 study each. Besides, we found that the most studied level-3 area within model paradigm is UML (with 16 secondary studies), followed by transition-based notation (with 12 studies). Within UML, behavioral models (including state machine, sequence and activity diagrams) are the most common, with 16 secondary studies in total. Within transition-based notations, finite state machines is the most common, with 9 secondary studies. The least studied level-3 areas of model paradigm are history-based and functional, with no studies at all. Furthermore, with regards to model characteristics' level-3 areas, the most studied ones are timing and non-determinism, with 2 studies each, whereas the least studied one is dynamics (i.e., discrete, continuous, hybrid), with 1 study only.

Test generation. Table 3e shows the number of studies for level-2 and level-3 areas of the test generation area. The test generation's level-2 area with more studies is generation method, with 9 studies, whereas the level-2 area with least studies is selection criteria, with 6 studies. We found that the most studied selection criteria's level-3 area is structural model coverage, with 6 studies, while the least studied is ad-hoc test case specification, with one study. Regarding the generation methods' level-3 areas, the most studied ones are random and search-based, with 4 studies each, and the least studied is constraint solving, with no studies.

Test execution. Table 3f shows the number of studies for level-2 and level-3 areas belonging to the test execution area. The test execution's level-2 area with more studies is technology, with 4 studies, while the least studied level-2 areas are mapping and test scaffolding, with 1 study each. The most studied technology's level-3 area is offline, with 4 studies, while the least studied is online, with 3 studies. With respect to conformance check's level-3 areas, the most studied area is conformance relation, with 3 studies, and the least studied is test oracle, with 2 studies. Additionally, we found that all level-3 areas belonging to mapping and scaffolding have only one study each.

4.1.1 Trends in MBT research areas

We performed an analysis of secondary studies by MBT area through time, in order to determine research trends. Figure 5 illustrates this by showing the number of studies by year, for (a) the top-level areas of our MBT hierarchy, along with three selected level-2 areas (that showed high study counts and interesting

trends): (b) model paradigm's level-3 areas, (c) test level's level-3 areas, and (d) test artifact's level-3 areas. The right-most column (Total) shows the total number of studies for each subarea.

Figure 5a shows the research trends for the top (level-1) areas of MBT. Initially (in 1996), the only studied area was model specification. There was a gap from 1997 to 2004, where no studies were reported. Later, between 2005 and 2016, the strongest areas were test objectives and model specification, with constant activity almost every year. There had been two peak years for all MBT areas: 2010 and 2015, when many studies were published. Particularly, in 2015 all areas were studied by at least two papers. More recently (in 2016), there has been activity in all but one area (test objectives).

Figure 5b shows the research trends for the model paradigm subareas. We chose to analyze this level-2 area because it contains two of the most studied areas overall. For model paradigm, we too observe a peak in 2015 across its level-3 areas. Areas with a steady development in time are UML models and transition-based notations. Areas with recent activity include: data-flow, operational, and attribute event grammar, each of them having one study in 2015. Stochastic models and state-based (pre-post) notations are also gaining popularity.

Figure 5c shows the research trends for the test level subareas. We chose to analyze this level-2 area because it contains the third and fourth most studied areas overall. Research on unit and integration testing has been steady through time, with a small increase in 2015. On the other hand, research on system testing has only been studied sporadically. No studies were found on acceptance testing level.

Figure 5d shows the research trends for the test artifact subareas. We see here two peak years for all level-3 areas: 2010 and 2015. Furthermore, we note that functional and extra-functional behavior are the strongest and earliest subareas, but there has been a recent emergence in architecture and environment artifacts.

4.1.2 Discussion on MBT areas

It is clear from Table 3 and Figure 5 that there have been a few well studied areas, which have dominated the research on MBT over time, but many other areas need to be studied further. Examples of well studied areas in MBT are popular model paradigms such as UML models and transition-based notations, and widespread test levels such as unit and integration testing. Other areas that have been moderately studied are common test artifacts such as functional and extra-functional behavior, popular test types like functional and regression testing, state-based notations system test level, and structural model coverage as selection criteria for test generation. Examples of areas that have been little studied fall under software paradigms (particularly, cyber-physical-oriented, aspect-oriented and service-oriented), SUT domain (mobile applications and software product lines), language type (design, test-specific, and generic), model paradigm (data-flow, operational, and attribute event grammar), model characteristics (dynamics), selection criteria (ad-hoc test case specification), test generation method (manual, model-checking, theorem proving, axiomatic and labeled-transition system), and text execution mapping and scaffolding. Finally, there are some areas that have not been studied at all: acceptance test level, model scope (input-only or input-output models), constraint solving generation method, and model paradigms such as history-based and functional. We recommend to expand MBT research to cover areas that have been either little studied or no studied at all.

We can also analyze what stages of the MBT process (refer to Figure 1) have been most studied. In order to do that, we first need to establish an association between the MBT stages and our areas. In particular, the model building *stage* maps directly to the model specification *level-1 area*, and indirectly to the SUT and test objectives *level-1 areas*. The test selection criteria *stage* maps to the selection criteria *level-2 area*. The test case generation *stage* maps to the generation method *level-2 area*. Lastly, the test case execution *stage* maps to the test execution *level-1 area*. Having established this mapping, we can now infer where the focus of each MBT stage has been, research-wise. For the model building stage of the MBT process, the focus has been on two modeling paradigms: UML models and transition-based notations, at the unit and integration levels. For the test selection criteria stage, the focus has been on structural model coverage. For the test case generation stage, both random and search-based algorithms have prevailed. For the test case execution stage, there has been a focus on offline technology.

To analyze the implications of our research for the industry, we compared our hierarchy of MBT knowledge areas with the ISTQB foundation level model-based tester syllabus [40], inspired by what Garousi et al. [5] did in their tertiary literature review. Next we discuss about important MBT areas that are not included in the ISTQB syllabus, as well as sections of the of the ISTQB syllabus that are not covered by secondary studies.

Knowledge areas that should be considered for inclusion in the ISTQB syllabus are: validity of a particular MBT approach for different SUT domains and paradigms, representational characteristics of models, methods or technologies for automated test case generation, methods for automated conformance checking (such as oracles), and scaffolding support of MBT tools. We also believe that the syllabus should provide more detail

(a) Trends of studies over time, for all level-1 areas.

(b) Trends of studies over time, for model paradigm's level-3 areas.

Figure 5: Trends of secondary studies in time.

(c) Trends of studies over time, for level-3 areas belonging to the test level area.

(d) Trends of studies over time, for test artifact's level-3 areas.

Figure 5: (Continued) Trends of secondary studies in time.

for the knowledge areas it currently covers. As an example, the syllabus mentions the concept of modeling notations and some of their existing types, with no further analysis or comparison among them. Lastly, the syllabus should somehow integrate the broad spectrum of existing MBT tools, either as a separate section or as a subsection of ‘tool support’ for existing sections.

On the other hand, there are some sections from the syllabus that need to be covered by secondary studies: the integration of model-based testing into the software development cycle, the creation and maintenance of an MBT model, best practices for modeling or test selection, and the deployment of model-based testing approaches. In general terms, we see that the research community has geared towards secondary studies mainly related to MBT techniques, lacking important aspects from the industry-perspective such as how to integrate, deploy, and adopt MBT in industrial contexts.

4.2 RQ2: Model-based testing tools

The RQ2 research question led us to identify classifications that have been used in secondary studies for MBT tools. This research question also motivated us to classify all MBT tools according to the areas hierarchy derived from RQ1, in an attempt to provide a single unified classification scheme. Finally, tendencies of MBT tools over time were also addressed with RQ2.

4.2.1 Existing classifications of MBT tools

In what follows, we present the results with respect 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 dimensions. The other 6 simply mentioned the tools. Several studies hinted the existence of tools, but offered no name for them. For the purpose of this study, only tools with a name were considered.

Studies that classified tools used similar criteria, which can be generalized as support for the following five dimensions: (i) model specification [S03, S10, S19, S20], (ii) test selection criteria [S10, S17, S19], (iii) test generation [S10, S17], and (iv) test execution [S10, S17, S19]. The most common criterion for tool classification was (support for) model specification, with 4 studies. This criterion corresponds, in most cases, to the model paradigm level-2 area of our hierarchy (in Figure 4), but in some cases it also includes the model scope, language type and model characteristics areas. The next two most common criteria for tool classification were (support for) test selection criteria and test execution, with 3 studies each. The test selection criteria corresponds to a level-2 area with the same name in our hierarchy, while the test execution criterion corresponds to a level-1 area with the same name. The least common criteria of tool classification was (support for) test generation, with just 2 studies. This criterion corresponds to the generation method level-2 area of our hierarchy.

Marinescu et al. [S17] performed 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 authors approached this by proposing a taxonomy based on the one by Utting et al. [14]. Many tools were characterized according to this new taxonomy. Shafique and Labiche [S19] studied the support of MBT tools that rely on state-based models. They characterized and compared these tools along several criteria, particularly, their support for test, test scaffolding, and related activities. These criteria were further divided into more specific classes like model-flow, script-flow, and data-flow coverage criteria. This is one of the studies that perform a more in-depth tool classification. Saifan and Dingel [S10] conducted a survey on how MBT has been used for testing the quality attributes of software. In their work, they extended the taxonomy of Utting et al. [14] by adding three additional criteria, and characterized existing tools based on these categories. Siavashi and Truscan [S20] performed a study on the use of environment models in MBT. They gathered a list of tools that use these models, and grouped them by model language. Mahdian et al. [S07] conducted a survey on regression testing that employs UML models. In their study, they cited studies that investigate regression approaches, types of UML diagrams, and proposed tools.

Studies that did not classify MBT tools were also considered for the purpose of our study. In general terms, they either mentioned the tools as examples of particular approaches/models, or commented on primary studies that proposed and experimented with those tools. The information found in these type of studies was: the description of the tool, the model or language supported by the tool, how the tool works, the studies that proposed or employed the tool, and its usage in industry. The information found in these studies was very heterogeneous, but it allowed us to identify tools that were not cited in other papers.

We found a total of 98 MBT tools in the analyzed papers. Approximately 38 of these tools had little or no information, so they could not be associated to MBT areas. The complete list of tools and their characterization can be found at <https://tinyurl.com/yc29yx16>. Such table is sparse for most of the tools and some areas, indicating that no study covers all the dimensions found.

4.2.2 Classification of tools based on the hierarchy of MBT areas

We classified these tools with respect to the hierarchy of areas depicted Figure 4, in order to find out which areas are supported by tools and which ones are not. For each low-level (level-3) area of the hierarchy tree (in Figure 4), we list the specific tools that support that area, and highlight in bold the secondary studies from which tools were extracted. By doing this we homogenize the tool data reported by all secondary studies.

Tables 4 through 8 shows the quantity of tools that support each area of our hierarchy. A tool may support one, multiple, or no areas at any level, and so the total counts does not align with the total number of tools (98). In what follows, we summarize the results of this classification, describing the most and least tool-supported areas. Table 4 shows the number of tools reported for each of the five top-level (level-1) areas of our hierarchy. We found that most tools provided support for the model specification area. Test objectives is the second most supported area, with test execution and test generation following closely behind. No tools were reported to support the SUT area. It is important to notice that tools may support multiple areas in a particular level. We count tool support for a particular area as the number of tools that support at least one of its sub-areas. This means that the number tools that support a particular area may be less than the sum of tool support across all its sub-areas. For each level-2 area of the taxonomy, the amount of tool that support this area is the number in parenthesis beside the name.

In what follows, we present and discuss the amount of tools that support each area. For each top-level area, we present all of its level-2 areas, and all of its level-3 areas if at least one tool supports it. In other words, we do not enter a level-2 area if no tools support this area.

Table 4: Tools by level-1 areas

Area	Tool (98)	
	Count	Percentage
SUT	0	0%
Test objectives	39	40%
Model specification	60	61%
Test generation	36	37%
Test execution	37	38%

Table 5: Tools per level-2 and level-3 areas belonging to the **test objectives** area.

Level 2 area	Level 3 area	Tool (98)		Names
		Count	Percentage	
Test artifact (35)	Functional behavior	34	35%	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	3	3%	CompleteTest, MISTA, TestOptimal
	Architectural description	0	0%	
	Environment	1	1%	AETG
Test type (10)	Functional testing	5	5%	AETG, Conformiq, LTG, SpecExplorer, TorX

(Continued)

Level 2 area	Level 3 area	Tool (98)		
		Count	Percentage	Names
	Non-functional testing	2	2%	JUMBL, TAF
	Regression	3	3%	DeJavu, RTSTool, Statechart base
Test level (1)	Unit	0	0%	
	Integration	1	1%	SeDiTeC
	System	0	0%	
	Acceptance	0	0%	

Table 6: Tools per level-2 and level-3 areas belonging to the **model specification** area.

Level 2 area	Level 3 area	Tool (98)		
		Count	Percentage	Name
Model scope (0)	Input-only	0	0%	
	Input-output	0	0%	
Language type (7)	Design	1	1%	AGEDIS
	Test-specific	4	4%	AETG, Conformiq, SpecExplorer, TorX
	Domain-specific	2	2%	Genesys-Pro, HOTTest
	Generic	0	0%	
Model paradigm (48)	State-based	5	5%	AsmL, BZ, Isabelle/HOL, ProTest, SpecExplorer
	Transition-based	35	36%	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	3	3%	JUMBL, Lurette, Markov model
	Data-flow	4	4%	AETG, CompleteTest, GATel, Simulink
	History-based	0	0%	Verification and Validation
	Functional	0	0%	
	Operational	0	0%	
	Attribute event grammar	1	1%	AEG

(Continued)

Level 2 area	Level 3 area	Tool (98)		
		Count	Percentage	Name
	UML	17	17%	AGEDIS, Smarttesting CertifyIt, DeJavu, EFM, LTG, MISTA, MoCAT, ParTeG, PrUDE, RTSTool, SeDiTeC, Statechart base, Tedeso, TestCast, TestUML, UMLTest, UTG
Model characteristics (8)	Timing	8	8%	AETG, Conformiq, JUMBL, LTG, MoCAT, SpecExplorer, TAF, TorX
	Non-determinism	5	5%	Conformiq, LTG, SpecExplorer, TAF, TorX
	Dynamics	6	6%	AETG, JUMBL, LTG, SpecExplorer, TAF, TorX

Table 7: Tools per level-2 and level-3 areas belonging to the **test generation** area.

Level 2 area	Level 3 area	Tool (98)		
		Count	Percentage	
Selection criteria (30)	Structural model coverage	26	27%	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	11	11%	AETG, BZ, Smarttesting CertifyIt, Conformiq, NModel, PyModel, SpecExplorer, TestCast, TestMaster, TestOptimal, TorX
	Fault-based coverage	0	0%	
	Requirements-based coverage	10	10%	Smarttesting CertifyIt, Conformiq, GOTCHA, GraphWalker, NModel, Reactis, SpecExplorer, TestCast, TestMaster, TestOptimal

(Continued)

Level 2 area	Level 3 area	Tool (98)		
		Count	Percentage	Name
Generation method (29)	Ad-hoc test case specification	0	0%	
	Stochastic & Random	2	2%	JUMBL, TestUML
	Manual	1	1%	Reactis
	Random	4	4%	ModelJUnit, Reactis, SpecExplorer, TestUML
	Graph search	9	9%	AGEDIS, GraphWalker, ModelJUnit, PyModel, Simulink Verification and Validation, Tedeso, TEMA, TestCast, TestOptimal
	Search-based algorithms	1	1%	AETG
	Model-checking	11	11%	AsmL, CompleteTest, Conformiq, CoVer, GOTCHA, JUMBL, NModel, ProTest, SpecExplorer, TorX, UPPAAL-Tron
	Symbolic execution	2	2%	Conformiq, LTG
	Theorem proving	2	2%	Isabelle/HOL, PrUDE
	Constraint solver	2	2%	BZ, GATel
	UML-based	0	0%	
	Axiomatic	0	0%	
	Labeled transition systems	0	0%	
	Finite-state machines	0	0%	

Table 8: Tools per level-2 and level-3 areas belonging to the **test execution** area.

Level 2 area	Level 3 area	Tool (98)		
		Count	Percentage	Name
Technology (35)	Online	15	15%	AsmL, Conformiq, GraphWalker, JSXM, JTorX, MISTA, ModelJUnit, NModel, PyModel, SpecExplorer, TEMA, TestOptimal, TGV, TorX, UPPAAL-Tron
	Offline	30	31%	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 (17)	Conformance relation	2	2%	SpecExplorer, TorX

(Continued)

Level 2 area	Level 3 area	Tool (98)		Name
		Count	Percentage	
	Test oracle	14	14%	AETG, AGEDIS, Conformiq, GOTCHA, GraphWalker, JUMBL, NModel, ParTeG, PyModel, TAF, TestCast, TestMaster, TestOptimal, UPPAAL-Tron
Mapping (33)	Abstract test	9	9%	BZ, CoVer, GOTCHA, Isabelle/HOL, ProTest, Reactis, SeDiTeC, SpecExplorer, TGV
	Executable test	24	24%	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 (13)	Test adapter	9	9%	AGEDIS, Smarttesting CertifyIt, GOTCHA, GraphWalker, NModel, ParTeG, SpecExplorer, TestMaster, TestOptimal
	Test stubs	2	2%	Smarttesting CertifyIt, SeDiTeC
	Test oracles	4	4%	AGEDIS, Conformiq, GOTCHA, GraphWalker

SUT. No tools were reported as supporting the SUT area. The only study that reviews MBT tools and mentions the SUT area is Saifan and Dingel's [27].

Test objectives. Table 5 shows the amount of tools supporting the level-2 and level-3 areas belonging to the test objectives area. Of all level-2 areas, test artifact has the most support, totaling 35 tools. The functional behavior level-3 area is highly supported by MBT tools, with a total of 34 tools. In contrast, the extra-functional behavior, architectural and environment level-3 areas are scarcely supported, with only 3, 0 and 1 tools respectively. Interestingly, only 10 tools are reported to support the test type level-2 area, even though it is closely related with the test artifact area. Among the level-3 areas of test type, functional testing is the most supported one, with 5 tools, followed by regression testing, with 3 tools.

The support for the test level area is only reported for one tool at the integration level-3 area.

Model specification. Table 6 shows the number of tools supporting the level-2 and level-3 areas belonging to the model specification area. Studies that report on this area focus mainly on the model paradigm level-2 area, with a total of 48 tools. Of these, the majority (35) support transition-based models in level-3. The second most supported level-3 area is UML models, with a total 17 tools. The remaining level-3 areas of model paradigm have 5 or less supporting tools (some have none). This shows a dominance of transition-based models for MBT tools. In the model characteristics area, 8 tools were reported to support it. For the 8 tools that support the timing level-3 area, all of these tools support timing in models, and 6 support untimed models. For the 5 tools that support the non-determinism level-3 area, 5 tools allow determinism in models, and 3 allow non-determinism. For the 6 tools that support the dynamics level-3 area, 5 tools use continuous models, 1 uses hybrid models, and no tools support continuous models. 7 tools are reported in the language type level-2 area. Of these, 4 support the test-specific language, 2 support the domain-specific, 1 support the design languages, and no tools support the generic languages level-3 area. No tools are reported in the model scope level-2 area.

Test generation. Table 7 shows the quantity of tools supporting the level-2 and level-3 areas belonging to the test generation area. The selection criteria level-2 area is supported by 30 tools. Of these, 26 support structural model coverage, 11 support data coverage, 10 support requirements based coverage, 2 support stochastic and random criteria, and none supports fault-based coverage nor ad-hoc test case specifications. On the other hand, 29 tools support the generation method level-2 area. From these, 11 tools support model-checking, and 9 support graph search; while the remaining areas have 4 or less supporting tools. No tools were reported to support the UML, axiomatic, LTS and FSM generation methods.

Test execution. Table 8 shows the number of tools supporting the level-2 and level-3 areas belonging to the test execution area. The technology level-2 area is supported by 35 tools. Of these, 30 perform offline testing and 15 perform online testing. The conformance check level-2 area is supported by 17 tools, 14 of which belong to test oracles level-3 area and 2 belong to conformance relation level-3 area. The mapping level-2 area is supported by 33 tools, of which 24 support executable test cases and 9 support abstract test cases. The scaffolding level-2 area is supported by 13 tools. For the scaffolding's level-3 areas, 9 tools support the creation of adapters, 4 support test oracles, and 2 support test stubs.

4.2.3 Trends in MBT tools

We performed an analysis of MBT tools through time, in order to determine trends in tool research along our MBT areas. Figure 6 shows the quantity of MBT tools released by year the (a) top-level areas of our hierarchy, (b) the modeling notation area, (c) the mapping area, and (d) the test artifact area. The right-most column (Total) shows the total number of tools reported to support each subarea. This quantity may not match the aggregated sum over all years, as the release date was not available for all tools.

Figure 6a shows the research trends for the top-level MBT areas. The greatest activity in tools was between the years 2002 and 2007, as this period contains peak years for all MBT areas. From 2008 onward, the activity in MBT research decreased, but was still above that of the period before 2002. The year 2007 has been the peak year in all MBT research areas, and 2002 has been the second largest research peak.

Figure 6b shows the research trends in the model paradigm level-2 area and each of its level-3 areas. Research on state-based notations peaked on the year 2004. Research on transition-based notations peaked in 2007, and has been steady through the years. Research on stochastic notations started on 2003 but didn't continue until the year 2011. Research on data-flow notations started on 1997 and continued up to 2004, where a gap occurred until 2014. Research on event grammar notations only happened on the year 2006. For UML, tools first appeared in the year 1997, and have had a steady activity since then.

Figure 6c shows the research trends in the level-2 mapping area its level-3 areas. Research on tools for abstract tests occurred between the years 2002 and 2007, and peaked on its initial year, 2002. On the other hand, research on tools for executable tests peaked in the year 2007, and has been steady over time. While tool support for abstract tests was more popular during the 2002–2006 period, tool support for executable tests became the main research focus in the following years.

Figure 6d shows the research trends in the test artifact level-2 area and its level-3 areas. Research on tools for functional behavior peaked on 2007, and has been active over time. This level-3 area had also a large activity in the years 2004 and 2005. Research on tools for extra-functional behavior started in 2009 but is still an area that needs more research. We did not find any research on tools that support the modeling of architectural descriptions. Regarding tools for environment modeling, only one was found in the year 1997.

We believe that research in model-based testing tools will probably keep the current trend, focusing on modeling of functional behavior, using transition-based notations, to generate executable tests.

Incorporating non-functional characteristics of the SUT to the models will require expanding the capacity of existing modeling notations. Thus, research on non-functional MBT tools will first require research on new modeling notations. We believe more tools with such support will slowly rise in the coming years, as they have in recent years. Lastly, it is possible that tool support for test scaffolding remains low, as this research area has only been active recently.

4.2.4 Discussion on MBT tools

The tools we found with this literature mapping were not explicitly classified in terms of their SUT domain or software paradigm. The absence of a tool classification by software paradigm may be because the term was recently coined. The key paper related to software paradigms was published on 2014 [S16], and the majority of studies that classified tools were published up to 2015 [S03, S10, S17, S19, S20]. Thus, it is possible that the software paradigm dimension was too new to be considered in the secondary studies of our mapping. On the other hand, the lack of tool classifications by SUT domain draws attention since early

(a) Trends of tools over time, for all level-1 areas.

(b) Trends of studies over time, for model paradigm's level-3 areas.

Figure 6: Trends of tools in time.

(c) Trends of studies over time, for mapping's level-3 areas.

(d) Trends of studies over time, for test artifact's level-3 areas.

Figure 6: (Continued) Trends of tools in time.

MBT studies (from 2007) had already discussed used this dimension (although not for tools) [S04, S05]. It then seems that this dimension has not been adopted by MBT researchers to classify tools yet.

The current state of MBT tools allows testing of mainly functional behavior of a system. It was not until recently that research on MBT tools started to shift towards extra-functional behavior, which happened to be one of the main gaps in MBT tool research. Future research should consider the representation of non-functional aspects of systems, and their corresponding supporting tools.

Most of the MBT tools found in our study use either transition-based or UML notations. This is a trend that has continued over time, in favor of less diversity of modeling notations. We believe this trend will continue due to the popularity and ease of use of those notations. A possible reason for the lack of tools on other modeling notations may be that common generation techniques, such as graph search and model-checking, are applicable only to UML and transition-based notations. This would imply that research on different model notations requires research on new generation methods as well.

One of the most important advances in MBT tools has been the shift from abstract test sequences to concrete test cases. This has allowed an increase in the automation of the tools, and a corresponding decrease in the amount of work testers need to do. However, some tasks must still be performed by a tester: they need to provide scaffolding—in the form of adapters, stubs, and oracles—for the tool to be able to generate executable test scripts. Automation of these tasks is an open area of research.

4.3 RQ3: Challenges

The RQ3 research question helped us identify the challenges related to MBT that have been reported in secondary studies. In particular, it allowed us to identify common challenges in MBT research, classify them into our hierarchy of MBT areas (derived from RQ1), and devise potential areas of future research.

In previous studies, Utting et al. [2] stated that MBT requires different skills than manual testing, and that it is normally used for functional testing only. Dias-Neto et al. [21, 23, 22] pointed out that the main challenges are the cost, effort, complexity and skill required to use MBT approaches. Besides, they found that more empirical evidence is needed to support the selection of MBT approaches and tools. There is little evidence of experiences in the industry, thus more case studies on the application of different MBT approaches are needed. In our previous study [6], we found that the main challenges of MBT were (i) the need for more empirical evidence that supports the selection of MBT approaches and tools, (ii) a better understanding of similarities and differences among MBT approaches, that help in choosing the most appropriate technique and tool for specific operating contexts, and (iii) the need for considering both functional and non-functional testing in MBT studies. Additionally, other challenges found in [6] were related to the efficacy of applying MBT approaches and tools, the availability of MBT tools for different domains and contexts, the skills required from testers to use MBT, as well as the complexity and feasibility of adopting MBT approaches in practice.

We identified a total of 171 challenges related to MBT approaches and tools from secondary studies. In order to categorize challenges in MBT research, we performed a grounded analysis that identified the following six emerging categories: Efficacy, Availability, Complexity, Professional skills, Investment, cost & effort, and Evaluation & empirical evidence. Section 4.3.1 describes the challenges and their mapping, for each of the emerging categories. In addition, challenges were classified according to the top-level areas of the hierarchy presented in Section 4.1, namely: SUT, Test objectives, Model specification, Test generation, and Test execution. A ‘General’ category was added for challenges that apply to MBT approaches overall. Section 4.3.2 presents the challenges and its corresponding mapping, classified by the top-level MBT areas. Table 9 summarizes the results of challenge counts and percentages, by emerging category (Table 9a), and the top-level area (Table 9b). Furthermore, the relationship between challenges in these two classification schemes is discussed. Finally, an analysis of trends in MBT challenges is presented.

4.3.1 Challenges by emerging categories

Efficacy. The efficacy of the existing MBT approaches is addressed in 16 (9%) challenges [S03, S04, S05, S11, S13, S14, S22]. The quality and characteristics of the model affect the quality of the generated test cases [S04, S11, S13, S22], as well as the criteria used to generate the test cases [S13]. It is a key factor to define languages to for the model and the test objectives [S03]. Test generation algorithms have model exploration problems, in particular when exploring cycles [S12, S13]. Generating test data for test cases is also reported as a challenge for complex software systems [S13]. Regarding the generated test cases, these may not be feasible [S13], and do not consider traceability with test scenarios [S14]. For non-deterministic models, it is difficult to determine the amount of test cases required to reach certain coverage [S22].

Table 9: Challenges of model-based testing.

(a) Challenges grouped by emerging categories.

Category	Count	Percentage
Efficacy	16	9%
Availability	50	29%
Complexity	33	19%
Professional Skills	26	15%
Investment, Cost & Effort	13	8%
Evaluation & Empirical Evidence	33	19%

(b) Challenges grouped by top-level MBT areas.

Area	Count	Percentage
SUT	6	4%
Test objectives	27	16%
Model specification	43	25%
Test generation	31	18%
Test execution	8	5%
General	56	33%

Availability. The availability of existing MBT approaches is addressed in 50 (29%) challenges [S03, S04, S05, S08, S10, S11, S13, S14, S15, S16, S17, S18, S19, S20, S21]. There is a lack of MBT approaches that support non-functional testing [S04, S05, S10, S14, S16, S20], and those that do are restricted to only certain software paradigms [S16]. Existing MBT approaches are often limited to a particular SUT domain or programming language [S04, S16], and often do not consider incorporation into the software development process. Research in MBT approaches have focused on only a subset of all the existing modeling notations [S17], and often consider only a single behavior model over a multiple of these [S15]. Regarding MBT tools, there are studies that present tools but don't make them available [S18, S20] or they are available and badly documented [S19]. There is a lack of MBT tools for certain software domains [S03, S18, S20]. Existing MBT tools offer limited support for test generation criteria [S13, S19], test scaffolding [S19, S20], and for manual steps of the MBT process, such as modeling [S04].

Complexity. The complexity of existing MBT approaches is addressed in 33 (19%) challenges [S01, S03, S04, S05, S06, S11, S13, S16, S19, S20, S22]. Adapting model-based testing is a difficult task [S04], as hard as moving from manual to automatic testing [S03]. In particular, MBT requires knowledge on the existing approaches [S04]. Selecting a MBT approach is a complex task [S05, S06], and it requires knowledge on the domain of the software system [S04]. Creating a model for a system is a complex activity [S04, S11, S20], as modeling languages can prove to be hard to use [S03]. It is necessary to validate the model, as well as other artifacts as part of the MBT process [S04, S16]. Models suffer from implicit complexity in the form of the state explosion problem [S01, S11], which is more apparent with non-deterministic models [S22]. The selection of adequate test criteria has also been reported as a complex activity [S06, S11]. In order to reduce complexity, it has been reported that MBT could automate the most complex parts of its process [S04, S19].

Professional skills. The demand of professional skills for existing MBT approaches is addressed in 26 (15%) challenges [S03, S04, S05, S06, S10, S11, S13, S16, S19, S22]. Model-based testing requires knowledge and skills in order to be applied [S04], as well as expertise in the existing tools and languages [S11]. This requirement is especially necessary to choose an approach to apply to a particular software system [S04, S05, S06, S22]. It has also been reported that the MBT process is difficult to incorporate to the existing software process [S04, S13]. Models in MBT should be validated [S03], as errors performed in modeling impact the quality of the generated tests [S05]. These challenges make learning MBT a complex task overall [S19].

Investment, cost & effort. The investment, cost and effort of existing MBT approaches is addressed in 13 (8%) challenges [S03, S04, S05, S06, S11, S22]. The selection of the correct MBT approach is important, as it leads to reduction of effort and costs of MBT [S06]. This leads to the need of empirical knowledge in costs and effort in MBT [S04, S05]. Moreover, MBT requires planning in order to minimize costs and

efforts [S11]. Part of the cost of MBT comes from the learning overhead [S04, S11], and the creation of test oracles, which may be as complex as the SUT itself [S03]. The benefits of MBT should be attainable with low investment [S04].

Evaluation & empirical evidence. The evaluation and empirical evidence of existing MBT approaches is addressed in 33 (19%) challenges [S02, S03, S04, S05, S06, S08, S11, S16, S17, S19, S20]. There is a need for comparison of existing MBT approaches, both against traditional testing [S05], and other MBT approaches [S03, S06, S08, S16]. There is also need for evidence of costs, effort and quality of the existing MBT approaches [S04, S06]. It has also been reported that the efficacy of models in different SUT domains should be evaluated [S11]. Problems in existing MBT evaluations include the lack of evaluation on test criteria [S02], usage of small systems [S05], and not using test verdicts [S16]. The complexity of modeling makes MBT approaches less comparable [S19, S20].

4.3.2 Challenges by top-level MBT areas

Test objectives. The test objectives area is addressed in 27 (16%) challenges [S03, S04, S05, S10, S16, S17, S20]. Testing of both functional and non-functional characteristics of software systems is desirable for MBT [S04]. However, the support for non-functional testing in MBT approaches is limited, and is still an open field of research [S04, S05, S06, S10, S16, S17, S20]. MBT approaches should also support testing of architecture [S20] and environment [S20]. Studies on environment modeling don't report on how was this process performed [S20]. Selection of an MBT approach depends on the objectives of testing, such as test level and type [S05].

General. The entirety of MBT, without regard for a particular area, is addressed in 56 (33%) challenges. Adoption of MBT is a difficult and complex process [S03], and should be simplified in order to increase its feasibility [S04]. This can be performed by automating the most complex parts of the process [S04], or reducing its skill and knowledge requirements [S04, S06]. Model-based testing has been reported as being costly [S03, S05]. MBT requires the system design to be performed twice—once for development and once for testing—[S03], and the correct approach of MBT must be chosen in order to reduce the risk of increased efforts and cost [S05]. MBT is a labor intensive task [S11], including maintenance of the model [S05]. Regarding evidence on MBT, many studies have reported a lack of evidence on the existing approaches [S03, S04, S05, S06, S08]. These evaluations should consider performing testing on real software systems [S05], and should use a more thorough methodology [S19]. Regarding the existing MBT tools, they are lacking in support for both manual and automatic steps [S04], usability [S19], documentation [S19], and availability [S12]. MBT as a process has only been considered in the context of testing, and research on MBT as a part of the development process is necessary [S04, S13, S14].

Model specification. The model specification area is addressed in 43 (25%) challenges [S01, S03, S04, S05, S06, S08, S10, S11, S13, S15, S16, S17, S19, S20, S22]. Modeling has been reported as a complex task [S11]. Selection of a modeling notation demands knowledge and skills in MBT [S04, S11, S22], as does the selection of which aspects of a system to model [S10, S11]. Modeling languages are complex to learn and use [S03, S04, S11], and model visualization tools are desirable [S03]. Models become increasingly more complex as the SUT grows [S04, S11]. In particular, state- and transition-based models suffer from the state explosion problem [S01, S11]. Models that incorporate oracles grow in complexity, and may become as complex as the SUT [S03]. Model-based testing tools offer no modeling support for particular domains [S03, S15], and only support a subset of all the existing modeling notations [S17]. Validation of the model is vital [S03, S04, S06, S16], as mistakes performed during modeling lead to errors in test generation [S05]. Regarding open research areas, there is no evidence in which models are better for certain domains [S11], exploration of approaches outside UML [S05, S08], and consideration of sub-states and events in models [S13].

Test generation. The test generation area is addressed in 31 (18%) challenges [S02, S04, S05, S06, S11, S13, S14, S16, S19, S22]. Generation approaches suffer from problems such as feasibility of paths [S13] and looping problems [S13]. Existing generation techniques employ only single generation criteria [S13, S16, S19], and do not support certain, more powerful generation criteria [S16, S19]. The selection of which of these criteria to use impacts the final quality of the test cases [S06]. There is a lack of evidence of efficacy of test generation criteria [S02], and on the existing test generation techniques [S16].

Test execution. The test execution area is addressed in 8 (5%) challenges [S06, S17, S19, S20]. There is a higher support for offline technology and executable test cases, and less for online technology and abstract

test cases [S17]. Most of the existing MBT tools do not support test scaffolding activities [S19]. In particular, no tools fully support test stubs [S19], and most tools require test adapters to be implemented manually [S20].

SUT. The SUT area is addressed in 6 (4%) challenges [S04, S05, S16]. Software systems are becoming more complex each year, which rises the need for MBT evaluations with large SUTs [S04, S05]. Existing MBT approaches don't specify to which software systems they are applicable [S04], and those that do focus on component-oriented systems [S16]. It is necessary to have knowledge on the SUT before applying an MBT approach [S04].

4.3.3 Relationship between top-level MBT areas and emerging challenge categories

Table 10 shows the amount of challenges of each combination of emerging categories and top-level MBT areas. Figure 7 shows this relation in the form of a bubble graph. In the bottom axis the MBT areas are shown and in the left axis the categories of challenges associated with these areas. In this case, the diagram shows a bubble whose size is proportional to the quantity of challenges in the intersection of each challenge category and area combination. For each area, we will study the type of challenges that are related to it.

Figure 7: Relationship between MBT top-level areas and challenge categories.

SUT. For the SUT area, 3 challenges belong in the availability category. This relation is due to the lack of MBT approaches and tools for particular types of software systems or paradigms. The 2 challenges related to complexity, and the challenge belonging in to evaluation, deal with the difficulty of applying MBT to larger software systems.

Test objectives. For the test objectives area, 20 of the challenges belong in the availability category. This relation is due to the lack of MBT approaches that incorporate non-functional, environment, and architecture testing. The remaining 7 challenges belong in the other categories: professional skills, evaluation, complexity, and efficacy. These report on the importance of correctly selecting the adequate MBT approach that adapt to the specific software project.

Model specification. For the model specification area, 14 of the challenges belong in the complexity category. This indicates that modeling as an activity, and the existing model notations, are one of the most difficult aspects of model-based testing. The 7 challenges that relate to professional skills follow this same trend, reporting that testers should have sufficient knowledge and skills to be able to apply MBT. 4 of the challenges are related to the investment, cost, & effort of MBT; which report on the risk of incorrect modeling, which results in increased costs and effort. The 7 challenges related to evaluation & empirical evidence, as well as the 5 about efficacy, suggest that it is necessary to test and improve the efficacy of the existing notations, as they have important drawbacks. 6 of the challenges are related to availability, which report on the lack of availability of tools for specific model notations.

Test generation. For the test generation area, 9 of the challenges belong in the availability category. These report mainly on the current lack of support for more complex generation criteria. The 7 challenges related to efficacy, as well as the 6 related to evaluation, state that there is no evidence on many of the existing approaches, which results on problems on the efficacy of this challenges, such as the ability to find faults and the relevance of the generated test cases. The remaining 9 challenges belong in the complexity, professional skills, and investment categories. These describe that testers must be familiar with the complex generation techniques in order to correctly apply MBT.

Test execution. For the text execution area, 6 of the challenges belong in the availability category. This are mainly related to the lack of support of test scaffolding for existing MBT tools, especially the lack of support for the creation of test oracles. The 2 challenges related to evaluation & empirical evidence report on the lack of evidence on the benefits of online and offline approaches.

General. For the challenges that do not belong in a specific area of MBT, 15 of the challenges belong in the evaluation & empirical evidence category, and 3 belong in the efficacy category. These report on one of the largest challenges in MBT: the lack of empirical evidence of the existing approaches. The 6 challenges in the availability area report the lack of availability of certain tools and approaches reported in studies. The challenges in the the professional skills, complexity, and investment areas are related to the increased complexity in MBT approaches, which impedes their adoption by the industry.

4.3.4 Trends in MBT challenges

We performed an analysis of challenges by emerging categories through time, in order to determine trends. The results of this analysis are summarized in Table 11 as well as in Figure 8. Table 11 in particular shows the challenge count per year and emerging category. Figure 8 depicts the cross relationship between categories and years, where the left axis shows the categories of challenges while the bottom axis is a time line. The size of each bubble in this figure is proportional to the quantity of challenges at the intersection of challenge category and year.

The earliest challenges in MBT (in 1996) dealt with complexity of models. The year 2006 had important activity for challenges, having at least two studies in each challenge category. Complexity was the most reported challenge until 2007. In 2007, the largest number of challenges on professional skills was reported. During the first years of MBT, challenges seem to be related to the complexity of both models and tools, as well as to the professional skills required to use MBT approaches. For the year 2008, challenges related to empirical evidence emerged. Evidence was required to support the selection of MBT approaches and tools, and experiences from industry were needed. Challenges in evaluation and empirical evidence spiked in 2008 and 2009. In the year 2010, all challenge types were reported, the trend being complexity challenges. This trend shifted to efficacy in 2012. From 2012 onward, the prevalence of efficacy and availability challenges started. The efficacy challenge is related to the correctness of the models to describe SUT's behavior, as well as the limitations of the models to represent the SUT. Additionally, efficacy also refers to the relevance of the generated test cases, and how these test cases make sense according to the source code of the SUT. The availability challenge refers to approaches, techniques, or tools for specific areas not being available for the public, and being very limited in their automation of MBT stages. In particular, the years 2014 and 2015 exhibited the largest activity in one challenge category: availability. In 2016, the amount of challenges in MBT decreased considerable, with attention moving back to the efficacy of the approaches.

Overall, in the early years of MBT, from 2005 to 2010, the research on MBT reported a lack of empirical evidence, a high complexity of the existing approaches, and the requirement of knowledge and skills to apply them. From the year 2012 onward, these problems were reported in lesser degree, as the importance of availability of tools for a particular MBT area and the complexity of the existing MBT approaches rose. Through the years, challenges related to investment, cost & effort were not the trend in MBT research. This may be due to MBT being still in early stages, with entities being more interested in having tools and approaches that correctly function, rather than focusing on resource efficacy.

4.3.5 Discussion on MBT Challenges

In general, 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

Table 10: Challenge count by top-level MBT area and category.

Area	Category	Count	Percentage
Test objectives	Availability	20	12%
Test objectives	Professional Skills	3	2%
Test objectives	Evaluation & Empirical Evidence	2	1%
Test objectives	Complexity	1	1%
Test objectives	Efficacy	1	1%
Test objectives	Investment, Cost & Effort	0	0%
General	Evaluation & Empirical Evidence	15	9%
General	Professional Skills	14	8%
General	Complexity	10	6%
General	Investment, Cost & Effort	8	5%
General	Availability	6	4%
General	Efficacy	3	2%
Model specification	Complexity	14	8%
Model specification	Evaluation & Empirical Evidence	7	4%
Model specification	Professional Skills	7	4%
Model specification	Availability	6	4%
Model specification	Efficacy	5	3%
Model specification	Investment, Cost & Effort	4	2%
Test generation	Availability	9	5%
Test generation	Efficacy	7	4%
Test generation	Complexity	6	4%
Test generation	Evaluation & Empirical Evidence	6	4%
Test generation	Professional Skills	2	1%
Test generation	Investment, Cost & Effort	1	1%
Test execution	Availability	6	4%
Test execution	Evaluation & Empirical Evidence	2	1%
Test execution	Efficacy	0	0%
Test execution	Complexity	0	0%
Test execution	Professional Skills	0	0%
Test execution	Investment, Cost & Effort	0	0%
SUT	Availability	3	2%
SUT	Complexity	2	1%
SUT	Evaluation & Empirical Evidence	1	1%
SUT	Efficacy	0	0%
SUT	Professional Skills	0	0%
SUT	Investment, Cost & Effort	0	0%

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 and usability [S04, S06, 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

Table 11: Count of challenges per year, by category.

Year	Category	Count	Percentage
1996	Complexity	1	1%
2005	Evaluation & Empirical Evidence	1	1%
2006	Complexity	4	2%
2006	Efficacy	2	1%
2006	Investment, Cost & Effort	2	1%
2006	Professional Skills	2	1%
2006	Availability	2	1%
2006	Evaluation & Empirical Evidence	2	1%
2007	Professional Skills	10	6%
2007	Complexity	10	6%
2007	Availability	6	4%
2007	Investment, Cost & Effort	3	2%
2007	Evaluation & Empirical Evidence	2	1%
2007	Efficacy	1	1%
2008	Evaluation & Empirical Evidence	10	6%
2008	Complexity	6	4%
2008	Investment, Cost & Effort	4	2%
2008	Professional Skills	4	2%
2008	Availability	2	1%
2008	Efficacy	1	1%
2009	Evaluation & Empirical Evidence	3	2%
2009	Availability	1	1%
2010	Complexity	6	4%
2010	Investment, Cost & Effort	3	2%
2010	Professional Skills	3	2%
2010	Availability	3	2%
2010	Evaluation & Empirical Evidence	3	2%
2010	Efficacy	1	1%
2012	Efficacy	9	5%
2012	Availability	3	2%
2012	Professional Skills	1	1%
2012	Complexity	1	1%
2013	Availability	1	1%
2014	Availability	11	6%
2014	Evaluation & Empirical Evidence	4	2%
2014	Professional Skills	3	2%
2014	Complexity	1	1%
2015	Availability	20	12%
2015	Evaluation & Empirical Evidence	8	5%
2015	Complexity	3	2%
2015	Professional Skills	2	1%
2016	Efficacy	2	1%
2016	Investment, Cost & Effort	1	1%
2016	Professional Skills	1	1%
2016	Availability	1	1%
2016	Complexity	1	1%

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

Figure 8: Challenge categories over time.

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 found in secondary studies.

Our main findings regarding the **areas of MBT** that have been investigated are the following: (1) the most studied modeling paradigm is UML, (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. Research on MBT over time has had two strong areas with constant activity: test objectives and model specification.

With respect to **model-based testing tools**, we found 98 MBT tools which were classified according to model specification, test selection criteria, test generation criteria, and test execution. There are many MBT areas not supported by tools, such as 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. Research of MBT tools over time has focused on the testing of functional behavior of systems, supporting only a subset of all the existing modeling notations. Tools have also shifted into supporting the generation of executable test cases from abstract ones. We believe this trend will continue until more powerful modeling notations are investigated.

In regards to **challenges in model-based testing**, six common categories emerged from a grounded analysis: (1) efficacy, (2) availability, (3) complexity, (4) professional skills, (5) investment, cost & effort, and (6) evaluation & empirical evidence. We found that most challenges were related to availability. Besides, we also classified challenges according to our hierarchy of MBT areas, and found that most challenges fell in the model specification area. Challenges in MBT research have evolved in time. At the beginning, the largest problems in MBT were related to the complexity of the approaches, and their lack of empirical evidence. More recently, the greatest challenge identified is the lack of approaches for specific software domains. Furthermore, a challenge that has been present over time is that MBT is regarded as difficult, demanding, and hard to adapt.

Since there are few secondary studies on MBT, we consider important to develop studies encompassing the following areas: model scoping (e.g., input-only vs. input-output models), model dynamics (e.g., discrete, continuous, and hybrid), uncommon model paradigms (e.g., history-based, functional, data-flow, operational, and attribute event grammars), language types (e.g., design, test-specific, and generic languages), selection criteria like ad-hoc test case specification, test generation methods (e.g., constraint solving, manual, model-checking, theorem proving, axiomatic, and labeled-transition system), specific SUT domains (e.g., mobile applications and software product lines), specific software paradigms (e.g., cyber-physical-oriented, aspect-oriented, and service-oriented), and test execution aspects such as mapping (e.g., abstract vs. executable tests) and scaffolding of test cases (e.g., test adapters, stubs and oracles).

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 analyze 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 ESE Research Group at UCR for the valuable feedback and help.

References

- [1] A. Pretschner, W. Prenninger, S. Wagner, C. Kühnel, M. Baumgartner, B. Sostawa, R. Zölch, and T. Stauner, “One evaluation of model-based testing and its automation,” in *Proceedings of the 27th international conference on Software engineering*. ACM, 2005, pp. 392–401.
- [2] M. Utting and B. Legeard, *Practical model-based testing: a tools approach*. Morgan Kaufmann, 2010.
- [3] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering—a systematic literature review,” *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.
- [4] B. Kitchenham, R. Pretorius, D. Budgen, O. P. Brereton, M. Turner, M. Niazi, and S. Linkman, “Systematic literature reviews in software engineering—a tertiary study,” *Information and Software Technology*, vol. 52, no. 8, pp. 792–805, 2010.
- [5] V. Garousi and M. V. Mäntylä, “A systematic literature review of literature reviews in software testing,” *Information and Software Technology*, vol. 80, pp. 195–216, 2016.
- [6] L. Villalobos-Arias, C. Quesada-López, A. Martinez, and M. Jenkins, “A tertiary study on model-based testing areas, tools and challenges: Preliminary results,” in *21st Ibero-American Conference on Software Engineering*, 2018. [Online]. Available: <http://hdl.handle.net/10669/76953>
- [7] K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering: An update,” *Information and Software Technology*, vol. 64, pp. 1–18, 2015.
- [8] B. Kitchenham and S. Charters, “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] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos, “Systematic review in software engineering,” *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES*, vol. 679, no. 05, 2005. [Online]. Available: <http://www.cin.ufpe.br/~in1037/leitura/systematicReviewSE-COPPE.pdf>
- [10] C. Wohlin, “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, p. 38.
- [11] E. Mourao, M. Kalinowski, E. Mendes, and C. Wohlin, “Investigating the use of a hybrid search strategy for systematic reviews,” in *Proceedings of the 11th international symposium on empirical software engineering and measurement*. IEEE Computer Society, 2017.
- [12] P. Bourque, R. E. Fairley *et al.*, *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.
- [13] M. Utting, B. Legeard, F. Bouquet, E. Fournieret, F. Peureux, and A. Vernotte, “Chapter two - recent advances in model-based testing,” ser. *Advances in Computers*, A. Memon, Ed. Elsevier, 2016, vol. 101, pp. 53 – 120.

- [14] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," *Software Testing, Verification and Reliability*, vol. 22, no. 5, pp. 297–312, 2012.
- [15] ISTQB®, "Foundation level certified model-based tester," 2015.
- [16] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold, P. McMinn, A. Bertolino, J. Li, and H. Zhu, "An orchestrated survey of methodologies for automated software test case generation," *Journal of Systems and Software*, vol. 86, no. 8, pp. 1978–2001, 2013.
- [17] D. Budgen, P. Brereton, S. Drummond, and N. Williams, "Reporting systematic reviews: Some lessons from a tertiary study," *Information and Software Technology*, pp. 62–74, 2017.
- [18] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines-a survey," *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090–1123, 1996.
- [19] J. A. McQuillan and J. F. Power, "A survey of uml-based coverage criteria for software testing," *Department of Computer Science. NUI Maynooth, Co. Kildare, Ireland*, 2005.
- [20] A. Hartman, M. Katara, and S. Olvovsky, "Choosing a test modeling language: A survey," in *Haifa Verification Conference*. Springer, 2006, pp. 204–218.
- [21] A. C. Dias Neto, R. Subramanyan, M. Vieira, and G. H. Travassos, "A survey on model-based testing approaches: a systematic review," in *Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*. ACM, 2007, pp. 31–36.
- [22] A. D. Neto, R. Subramanyan, M. Vieira, G. H. Travassos, and F. Shull, "Improving evidence about software technologies: A look at model-based testing," *IEEE software*, vol. 25, no. 3, pp. 10–13, 2008.
- [23] A. C. Dias Neto and G. H. Travassos, "Supporting the selection of model-based testing approaches for software projects," in *Proceedings of the 3rd international workshop on Automation of software test*. ACM, 2008, pp. 21–24.
- [24] A. Mahdian, A. A. Andrews, and O. J. Pilskalns, "Regression testing with uml software designs: a survey," *Journal of Software: Evolution and Process*, vol. 21, no. 4, pp. 253–286, 2009.
- [25] M. Mussa, S. Ouchani, W. Al Sammane, and A. Hamou-Lhadj, "A survey of model-driven testing techniques," in *Quality Software, 2009. QSIC'09. 9th International Conference on*. IEEE, 2009, pp. 167–172.
- [26] R. Dorofeeva, K. El-Fakih, S. Maag, A. R. Cavalli, and N. Yevtushenko, "Fsm-based conformance testing methods: A survey annotated with experimental evaluation," *Information and Software Technology*, vol. 52, no. 12, pp. 1286–1297, 2010.
- [27] A. Saifan and J. Dingel, "A survey of using model-based testing to improve quality attributes in distributed systems," *Advanced Techniques in Computing Sciences and Software Engineering*, pp. 283–288, 2010.
- [28] S. K. Swain, S. K. Pani, and D. P. Mohapatra, "Model based object-oriented software testing," *Journal of Theoretical & Applied Information Technology*, vol. 14, 2010.
- [29] A. T. Endo and A. Simao, "A systematic review on formal testing approaches for web services," in *Brazilian Workshop on Systematic and Automated Software Testing, International Conference on Testing Software and Systems*, 2010.
- [30] M. Aggarwal and S. Sabharwal, "Test case generation from uml state machine diagram: A survey," in *Computer and Communication Technology (ICCCT), 2012 Third International Conference on*. IEEE, 2012, pp. 133–140.
- [31] A. Kaur and V. Vig, "Systematic review of automatic test case generation by uml diagrams," *International Journal of Engineering Research and Technology*, vol. 1, no. 7, 2012.
- [32] M. Shirole and R. Kumar, "Uml behavioral model based test case generation: a survey," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 4, pp. 1–13, 2013.

- [33] F. Häser, M. Felderer, and R. Breu, “Software paradigms, assessment types and non-functional requirements in model-based integration testing: a systematic literature review,” in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2014.
- [34] R. Marinescu, C. Seculeanu, H. Le Guen, and P. Pettersson, “Chapter three-a research overview of tool-supported model-based testing of requirements-based designs,” *Advances in Computers*, vol. 98, pp. 89–140, 2015.
- [35] A. Méndez-Porras, C. Quesada-López, and M. Jenkins, “Automated testing of mobile applications: a systematic map and review,” in *XVIII Ibero-American Conference on Software Engineering, Lima-Peru*, 2015, pp. 195–208. [Online]. Available: <http://hdl.handle.net/10669/77115>
- [36] M. Shafique and Y. Labiche, “A systematic review of state-based test tools,” *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 1, pp. 59–76, 2015.
- [37] F. Siavashi and D. Truscan, “Environment modeling in model-based testing: Concepts, prospects and research challenges: A systematic literature review,” in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2015.
- [38] Vishawjyoti and P. Gandhi, “A survey on prospects of automated software test case generation methods,” in *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on*. IEEE, 2016, pp. 3867–3871.
- [39] V. Arora, R. Bhatia, and M. Singh, “A systematic review of approaches for testing concurrent programs,” *Concurrency and Computation: Practice and Experience*, vol. 28, no. 5, pp. 1572–1611, 2016.
- [40] ISTQB (International Software Testing Qualifications Board), “ISTQB Worldwide Software Testing Practices Report 2015-2016,” 2016. [Online]. Available: <http://www.istqb.org/references/surveys/istqb-worldwide-software-testing-practices-report.html>