Model-Based Testing of Software Product Lines: Mapping Study and Research Roadmap

Kleber L. Petry[a], Edson OliveiraJr[a,*], Avelino F. Zorzo[b]

[a]*Informatics Department (DIN)*
State University of Maringá (UEM)
Maringá-PR, Brazil
[b]*School of Technology*
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre-RS, Brazil

Abstract

Model-Based Testing (MBT) has been successfully applied to Software Product Lines (SPL). This paper provides a panorama of state-of-the-art on MBT of SPLs. We performed a systematic mapping for answering questions related with domains, approaches, solution types, variability, test case automation, artifacts, and evaluation. We built a roadmap from 44 selected studies. Main obtained results are: Software and Automotive domains are most considered; Black-box testing is widely performed; most studies have fully-automated support; variability is considered in most studies; Finite State Machines is the most used model to test SPLs; Behavioral-based and Scenario-based are the most used models; Case Studies and Experiments are used to evaluate MBT solutions and the majority is performed in industrial environments; traceability is not widely explored for MBT solutions. Furthermore, we provide a roadmap synthesizing studies based on used models, more formal artifacts, supporting tools, variability management, (semi-)automation, and traceability. The roadmap contributes to identify related primary studies based on given artifacts, variability management, tools, automation, and traceability techniques and to identify, from a given primary study, which artifacts, tools, variability management, automation and traceability techniques are related. Therefore, the roadmap serves as a guide to researchers and practitioners on how to model-based test SPLs.

Keywords: `Model-Based Testing, Software Product Line, Reuse of Test Cases, MBT Roadmap, Systematic Mapping Study, Variability`

*Corresponding author
Email addresses: `kleberpetry@gmail.com` (Kleber L. Petry), `edson@din.uem.br` (Edson OliveiraJr), `avelino.zorzo@pucrs.br` (Avelino F. Zorzo)

1. Introduction

Software testing is an essential activity to improve products quality [1]. Due to increasing and important research on software design, models are the main target on software testing on early stages of software development. Design models might be created, for instance, to model software requirements and to describe the software architecture of a system [2, 3]. Therefore, Model-Based Testing (MBT) becomes an alternative for software testing to discover faults in software design models, thus, reducing effort and costs in maintenance and evolution of software systems [4, 5, 6]. Fixing issues in source code, for example, might cost several times more than to fix something in early stage of software development, e.g. in models [7].

MBT provides test cases from a model that describes aspects, usually functional, of the system being tested. These test cases are known as the abstract test suites, and their level of abstraction is closely related to the model abstraction level [8]. Some of the advantages of MBT are: i) to begin the generation of test cases earlier in the software development lifecycle; and, ii) to create test cases automatically from such models.

Although MBT has demonstrated its effectiveness in testing single systems, it is a challenge to accomplish it to mass customization approaches, such as in Software Product Lines (SPL) [9, 10]. An SPL represents a set of systems sharing common and variable aspects for a given domain [11]. The SPL engineering is basically composed of two activities: domain engineering, which is responsible for creating reusable assets encompassing respective variabilities; and application engineering, in which such assets are instantiated by resolving variabilities and producing specific SPL products. SPLs are straightforward based on models, such as, features and variability, for representing their similarities and variable aspects [12]. Thus, models are inheriting parts of an SPL and they are a factor to determine its success. Low quality SPL models generate a higher cost of maintaining a software product [13]. In addition, testing all potential SPL products from the perspective of the domain engineering core assets is unfeasible, as widely discussed in the existing literature [8, 14, 15, 16, 17].

One of main challenges in MBT of SPL is related to the particularities of each model, in which it is necessary to provide models within the SPL domain engineering, for later generation of test cases in the SPL application engineering [8, 14, 13]. Another great challenge is the SPL variability [12, 13, 18]. This challenge is due to the large number of specific products an SPL might derive. Therefore, testing all potential SPL specific products might become unfeasible [8, 13]. In this case, MBT must take into account SPL models with variability for generating test cases, then converting them into artifacts that can be tested and reused later, preserving their characteristics [19, 20, 21]. For this reason, MBT on SPLs is a promising research area.

Although several secondary studies have been published within the SPL testing context [8, 1, 14, 13, 22, 16], they had a wider scope compared to MBT to SPL. For example, Razak et al. [23] provide a superficial Systematic Mapping Study (SMS) on SPL, with no further relevant discussions.

Therefore, this paper extends previous works, through an SMS, with a panorama of current literature on MBT for SPL, as well as a roadmap based on such a panorama. Furthermore, this SMS contributes to systematically identify, summarize and evaluate the existing literature in order to find gaps in the area and to position new research and practice activities. Hence, this SMS helps:

- new researchers in a structured understanding of the area by indexing the existing studies and by providing new research directions;
- practitioners to understand state-of-the-art artifacts tools, variability management techniques, automation, and traceability techniques and their appropriate usage to MBT of SPLs; and
- to provide researchers and practitioners a roadmap on MBT of SPL.

The roadmap, which graphically depicts the analyzed panorama from the SMS, provides readers with the ability to identify which artifacts, tools, techniques and traceability adopted in each of the MBT of SPL stages. Thus, it provides researchers and practitioners a guidance at:

- identifying which artifacts (class diagram, state machine, sequence diagram, use case diagram, etc), supporting tools, variability management techniques, automation (full, semi, or manual), and traceability techniques have been used for deriving SPL MBT test sequences;
- given specific artifacts, tools, variability management techniques, automation and traceability techniques, to identify related primary studies that use them; and
- given a specific primary study, how to identify the artifacts, tools, variability management techniques, automation and traceability techniques it uses.

The first systematic mapping study on SPL testing was performed in 2011 by Engström et al. [13] aimed at surveying existing research to identify useful approaches and needs for future research. In general, a majority of the analyzed studies are proposal research types (64%). System testing is the largest group with respect to research focus (40%), followed by management (23%). Method contributions are in majority. Although such a mapping study covered general approaches for SPL testing, we can observe ours partially confirmed evidence provided by Engström et al. [13] and add findings specific for MBT of SPLs.

On the contrary of the Engström et al.'s [13] complains on lack of evidence (in about 90% of studies), our SMS found majority (79.5%) of the studies provide case studies or experiments as proposals evidence methods for MBT of SPLs. From such evidenced studies, most of them (59%) are in the industry sets. This means, for the MBT of SPL specific topic, actual evidence-based works grown in the last years, which is widely desired in this area. In addition, as SPL testing

is costly, MBT of SPL researches are investing time and effort to provide more close-to-reality evidence.

Although the SMS of Engström et al. [13] claim testability issues, especially related to the product line architecture (PLA) have an underdeveloped potential to be researched, in our SMS we found most of solution types are focused on testing the PLA (40.9%). We understand the topic of MBT of SPL concentrates most of its activities in early stages of SPL development, thus focusing most on the PLA and its models.

Our SMS found most of studies propose testing approaches (75%) as in Engström et al.'s [13].

This paper is organized as follows. Section 2 discusses essential background and related work; Section 3 presents the research methodology adopted to conduct this study; Section 4 presents results of our SMS; Section 5 discusses results; Section 6 discusses validity evaluation of this study; Section 7 presents a roadmap of MBT applied to SPL; Section 8 discuss remaining challenges and open issues on MBT of SPL; and Section 9 presents final remarks.

2. Background and Related Work

This section presents essential concepts on Model-Based Testing of Software Product Lines (SPL). Related work is discussed at the end of this section.

2.1. Model-Based Testing of Software Product Lines

MBT creates test specifications and test cases of formal or semi-formal models of a software system. Such models might be state machines or other ones, such as, UML class, activity or sequence diagrams. Thus, more than one software model can be used to create tests [24].

MBT methods based on a formal specification language can be mapped to an executable language to generate tests. Informal or semi-formal models need to be complemented with additional information before they can be used to generate test specifications [24].

In MBT, a test model serves as a specification of the test implementation. Depending on the model-based testing practices applied, the test models can provide a comprehensive base for any activity during the test processes, including test case derivation, test coverage, test case execution, test result evaluation, and test report. In combination with appropriate test interfaces and tool support, MBT campaigns are executable in a more or less automated manner, since a (validated) test model specification and a well-defined test interface are available [25].

To be useful for SPL testing, the test models must capture the variability between products. Some papers advocated static analysis as a way to reduce costs, but few provided static analysis techniques [26]. The same held true for nonfunctional testing; most proposed approaches dealt only with functional requirements. Testing an SPL requires consideration of variability and commonality in overall testing levels [17].

The great challenge of applying MBT in SPL is the great derivation of products, which in turn results in a set of variabilities to be resolved, thus a larger test combination may arise. This is why MBT has been combined with variability management approaches, which allows the reuse of cases or test scenarios. MBT seems to be an extremely SPL-compatible approach, but variability challenges the adaptation of MBT for product family design [27]. The variability of a product family specifies the differences between the applications to be created and is defined during domain engineering. The feature is called a variant whenever it is not intended to be part of all applications [27]. Testing is applied throughout the entire SPL life cycle and addresses both core assets and product-specific software, along with their interactions [17].

Figure 1 depicts variability role in MBT of SPL. Variability is present in SPL models from domain engineering, whereas during application engineering product requirements support testing such models by resolving variabilities (binding variability) for specific SPL products. Reuse of MBT test cases is a straightforward good practice, as testing all potential specific products is unfeasible and it ignores the possibility of reuse benefits [17].

The use of MBT has several other advantages when compared to other test techniques. For example, it can reduce the probability of misinterpretation of system requirements by system analyst, test engineers, and testers in an SPL [27].

The effort to develop models for MBT can be significantly reduced if a systematic modeling and configuration of SPL products methodology is used [28]. For example, Wang et al. [28] present a methodology that captures the variability in UML state machine, generates the test cases from these artifacts, and, with help of a tool, generates the executable tests. Olimpiew [24] also provides MBT data and information in SPL, although she is focused on customized activity diagrams reusing test cases in SPL variant products.

2.2. Related Work

As far as we know, based on a non-systematic search, there is only one direct related work to ours, which is of Razak et al. [23]. Therefore, in this section, we firstly discuss Razak et al.'s work, then we present major contributions of other works, which are important information source on MBT tools and models, and overall SPL testing.

Razak et al. [23] (RW.1 - Table 1) carried out a systematic literature review to indicate strategies, models, and problems solved by applying MBT to SPL.

With regard to review study strategies, they use as a basis Kitchenham et al.'s [29], performing searches in electronic databases and related studies, whereas our study took into account Kitchenham et al.'s [29, 30] and Petersen et al's [31] as we performed a SMS and performed searches in electronic and several manual sources (conferences, workshops, and journals). They used a generic search string with two terms as in Table 1, thus returning 60 studies generating a final set with 25 studies, whereas we defined a broader search string with three main terms and several respective synonyms (Section 3.2),

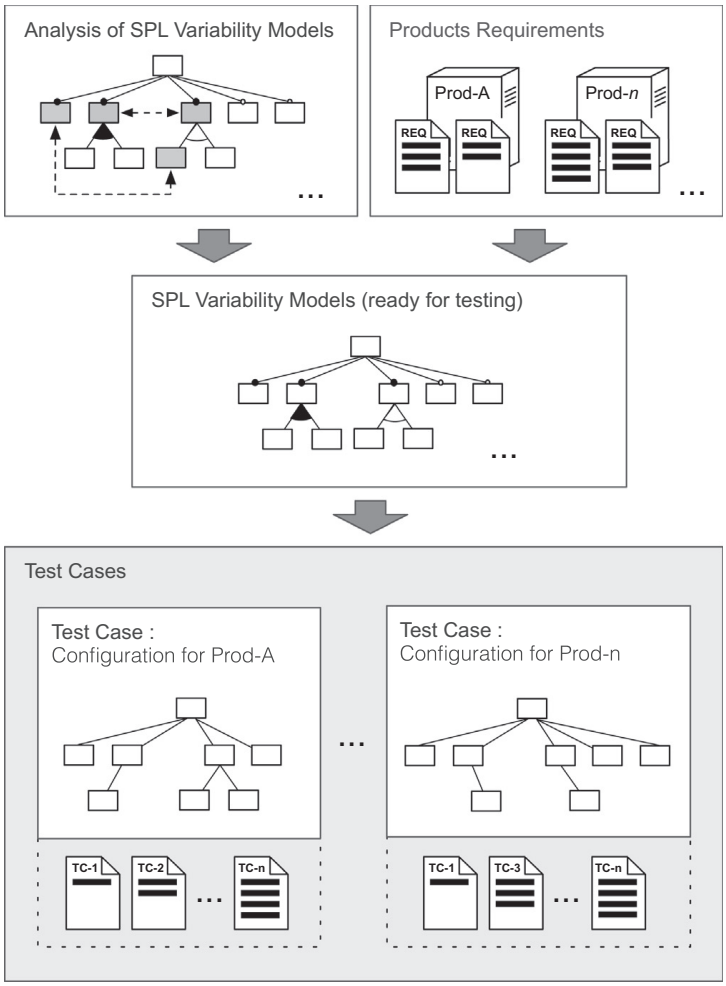

Figure 1: Model-based Testing of Software Product Lines [8]

thus returning 251 studies generating a final set of 44 studies. They defined inclusion and exclusion criteria and measured the quality of the studies, but they did not performed snowballing, whereas we performed forward snowballing, defined inclusion/exclusion criteria and analyzed the quality of the studies. Four research question compose their study and six compose ours. They defined four items to be extracted from the final set to compare such studies, whereas we established 20 items to be extracted.

As a matter of models, Razak et al. [23] found: feature diagrams, specific-domain models, sequence diagrams, use case diagrams, state diagrams, and transition diagrams. In our SMS, we found the same models as Razak et al.'s [23] and state machines, class diagrams, and ortogonal models.

The following works are secondary studies on broader SPL testing.

Machado et al. [8] (RW.2 - Table 1) discuss strategies that have the potential to scale SPL tests up, thus increasing error detection rates with less effort for quality assurance. Their results comprise two fundamental aspects: SPL testing, in which there is product selection for testing; and the product testing itself. Main contribution is to present techniques used in conjunction with the MBT, thus one can analyze all possibilities of use. Techniques are classified according to interests in SPL testing: (i) testing generation as a systematic selection of a representative set of product instances, comprising a subset of all possible configurations in a SPL; and (ii) focuses on performing testing on end-product functionalities. **Our SMS becomes a complement to this one, because we broadly visualize such interests as techniques for MBT SPL testing, to indicate to which models these techniques are used.**

Lamancha et al. [14] (RW.3 - Table 1) performed a systematic review of the literature on SPL testing, analyzing existing approaches, discussing the most significant issues and providing a state of the art updated to serve as a basis for other innovative research. They also perform an analysis of how SPL research can contribute to other research topics on software testing. Contributions are significantly targeted to technical test questions, presenting test aspects for future analysis. **Our SMS is a complement study seeking for finding approaches, issues, and more evidence specifically for MBT testing of SPLs.**

Silveira Neto et al. [16] (RW.4 - Table 1) performed a systematic mapping study with regard to process testing in SPL, aiming to investigate the state of the art of test practices, synthesize available evidence and identify gaps between required techniques and existing approaches available in the literature. They were able to identify what activities are addressed by the approaches encountered and what the researchers are developing as SPL testing study. **Such study contributed to ours as providing important keywords and inclusion and exclusion criteria.**

Engstrom and Runeson [13] (RW.5 - Table 1) performed a systematic mapping study to raise the state of the art in relation to testing of SPL. The main objective was to carry out a survey of the study being developed on testing in SPL to identify approaches and needs for future research. It presents scenarios that contribute to more assertive issues in SPL testing. **This study contributed to ours in the sense of defining important research questions to be answered specifically for MBT of SPLs.**

Bernardino et al. [4] (RW.6 - Table 1) present a general overview of MBT tools and models. They evaluated 1,197 studies, from which they selected 87 for a quanti-quali analysis. They classified 70 tools, as well as different domains to which MBT is applied. Forty tools are academic, fifteen commercial, and fifteen open-source. Most of such tools use UML and Finite State Machine (FSM) as modeling notations. All 70 tools are focused on functional testing. MBT is used in several different domains: critical systems 9.2%, automotive 8.05%, Web applications 6.9%, telecommunications 6.9%, health care 6.9%, and mobile 5.75%. Although Bernardino et al.'s study is not focused on SPL testing, they

discuss important criteria for selecting MBT tools and models. **Most of such models are also investigated and discussed in our SMS. Our research looks beyond tools and models to understand the reasons why MBT is adopted for SPLs, and if automated processes present some way to keep track of the test cases generated considering SPL variability.**

Lee et al. [22] performed a survey on SPL testing based on the fact that existing surveys and secondary studies did not deeply addressed the questions of what researches have been conducted in order to overcome the challenges posed by the two separate testing activities and their relationships: domain testing and application testing. They define a reference SPL testing processes and identify, based on them, key research perspectives that are important in SPL testing. In addition, they provide open research opportunities from each perspective. **Our SMS is not a survey. However, the proposed reference SPL testing process aids to understand these two SPL perspectives in terms of MBT testing of SPLs. Therefore, it directly contributed to establish a roadmap for MBT testing of SPLs (Section 7).**

Table 1 presents related works and respective number of studies and search strings.

3. Research Methodology

We elaborated the research methodology following guidelines by Petersen et al. [31] and Kitchenham et al. [32].

Figure 2 depicts the overview of our systematic mapping process.

Figure 2: Overview of our systematic mapping process

We started our study by defining research questions (Section 3.1). Then, we performed the search process (Section 3.2) towards gathering an initial set of primary studies from several different sources. With this set of studies, we performed the selection process (Section 3.3) by filtering studies according to different criteria. The extraction process (Section 3.4) was performed in the filtered studies to support the analysis process. In the analysis process (Section 3.5) we answered the defined research questions for this study.

3.1. Aim and Research Questions

This SMS **aims to** gathering literature evidence, **with the purpose of** characterizing Model-Based Testing, **with respect to** its application to Software Product Lines, **from the point of view of** SPL researchers, **in the context of** different primary study digital sources.

We formulated six research questions based on the main objective of this SMS and related works from Section 2.2, as follows:

Table 1: Related Works, Number of Studies, and Search Strings.

Related Work (RW)/Year/ Aim/Result	#St.	Search String	Diff. from our SMS
RW.1 [23] / 2016 / MBT of SPL / strategies, models, and problems	25	"software product line" AND model-based	- we cover several studies from RW.1; - we found more strategies than RW.1; - we discovered a bunch of different problems to be solved with MBT to SPLs; - RW.1 covers MBT to SPL superficially; - RW.1 does not deeply map and discuss MBT to SPL findings as we do.
RW.2 [8] / 2016 / SPL Testing / testing interests	49	"software product lines test" OR "software product line test" OR "software product line testing" OR "software product lines testing" OR "software product family testing" OR "software product family test"	- our SMS is a complement to RW.2; - we broadly visualize techniques for MBT to SPL; - we indicate used models from RW.2 identified techniques.
RW.3 [14] / 2013 / SPL Testing / significant approaches	23	("product line" OR "software product lines" OR "product families" OR "product family" OR "software family" OR "system families") AND (testing OR test)	- our SMS is a complement to RW.3; - we defined approaches, issues, and more evidence then RW.3 for MBT of SPL.
RW.4 [16] / 2011 / SPL Testing / gaps between techniques and approaches	45	Eighteen strings summarized as: ((Verification AND validation) OR "Static analysis" OR "Dynamic analysis"OR (Test AND level) OR (Variability AND commonality AND testing) OR (Variability AND commonality AND testing) OR (Binding AND test) OR (Test AND "effort reduction") OR "Test effort" OR "Test effort reduction" OR "Test automation" OR "Regression test" OR "Non-functional test" OR (Measure AND test) OR "Testing framework" OR (Performance OR security) OR (Evaluation OR validation)) AND ("product line" OR "product family" OR "SPL")	- RW.4 contributed to ours with important keywords and inclusion and exclusion criteria.
RW.5 [13] / 2011 / SPL Testing / approaches and scenarios	64	product AND (line OR lines OR family OR families) AND (test OR testing)	- RW.5 contributed to ours for defining important research questions specifically for MBT of SPLs.
RW.6 [4] / 2017/ MBT Models and Tools / models and tools	87	(MBT OR "model-based testing" OR "model based testing" OR "model-based test" OR "model based test" OR "model-based software testing" OR "model based software testing") AND (approach OR method OR methodology OR technique) AND software	- most of RW.6 models are investigated and discussed in our SMS; - we look beyond tools and models to MBT of SPLs; - we analyze whether automated processes can track generated test cases considering SPL variability.

- **RQ.1: To which SPL application domains, solution types, and proposals MBT is used?** We are interested in gathering evidence of which SPL domains are most frequent as, for instance, software, aerospace, and automotive, as well as what kind of solutions have been proposed for MBT of SPLs;

- **RQ.2: Which MBT approaches, testing levels, and artifacts have been used to testing SPLs?** In this question the main focus is to identify MBT techniques, test levels, and test automation;

- **RQ.3: How variability and binding time are treated during MBT of SPLs?** In this question we would like to understand which primary studies take into account variability and how they perform variability management in MBT activities of SPL, as well as whether binding time affects MBT activities in SPLs;

- **RQ.4: Does MBT support testing of SPL non-functional requirements?** We are interested in gathering evidence whether primary studies rely on testing SPL non-functional requirements;

- **RQ.5: How are MBT of SPL Proposals Evaluated?** MBT of SPLs solutions should be assessed as a means to provide evidence of their feasibility. Therefore, we would like to understand what kind of assessment is performed, such as, controlled experiments and case studies;

- **RQ.6: How is traceability considered during MBT activities for SPLs?** Traceability is a fundamental concept of both MBT and SPL. In this question we gather evidence of how solutions consider such concept for MBT activities in SPL.

3.2. Search Process

Figure 3 depicts the SMS search process. The search strategy to find relevant primary studies defined for this SMS is based primarily on the **Selection of Terms and Synonyms** related to MBT applied to SPL. These terms and synonyms take into account some of the keywords from related works, and are presented as follows:

- **software**;

- **product line**: *product-line*, *product family*, *product-family*, *product-families*, *family of products*;

- **model-based testing**: *model based testing*, *MBT*.

Once we had such terms and their synonyms, **Linking Synonyms with OR** is performed, as well as **Linking Terms with AND**. Thus, we came up with our general search string, as presented in Table 2.

Table 2: SMS General Search String

software **AND** (“product line” **OR** “product-line” **OR** “product family” **OR** “product-family” **OR** “product-families” **OR** “family of products” **OR** variability) **AND** (“model-based testing” **OR** “model based testing” **OR** MBT)

In addition, we performed **Selection of Databases/Search Engines** (electronic and manual sources), defining language of primary studies, and publication type as follows:

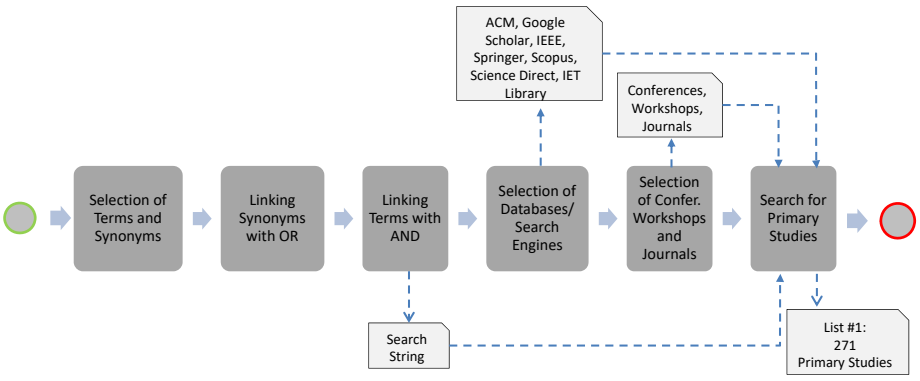

Figure 3: SMS Search Process

- **Search Sources:** indexed electronic databases (IEEE, ACM, ScienceDirect, Scopus, IET Digtial Library, Google Scholar, Springer) listed in Table 3 and manual search in specialized journals, conferences and workshops (**Selection of Confer., Workshops and Journals**) as in Table 4 and Table 5;
- **Language of Studies:** English, due to its broadening range in the Computer Science area;
- **Type of Publication:** peer-reviewed primary studies published in journals, conferences/workshops, or book chapters.

The reasons for choosing these search sources are listed:

- worldwide known computer science source;
- source provides an advanced search mechanism able to handle advanced queries;
- internationally indexed source;
- source indexes most relevant conferences and journals in Software Engineering;
- source indexes papers with high impact factor (JCR and/or H-index);
- source has not ceased publication.

As a last activity of the search process, we performed the **Search for Primary Studies** by applying the search string to electronic sources, and seeking primary studies manually in conferences, workshops and journals, as presented in the next sections. A total of 271 studies (List #1) were retrieved at the end of this process.

Table 3: Defined Electronic Search Sources

Electronic Source	URL
ACM Digital Library	http://dl.acm.org
IEEE Xplore	http://ieeexplore.ieee.org
ScienceDirect	http://www.sciencedirect.com
Scopus	http://www.info.sciverse.com/scopus
Springer	http://www.springer.com
Google Scholar	https://scholar.google.com
IET Digital Library	http://digital-library.theiet.org/content/journals/iet-sen

Table 4: Defined Conferences and Workshops for Manual Search

Acronym	Conference/Workshop
ICECCS	Engineering of Complex Computer Systems
AISE	Advanced Information Systems Engineering
ICST	International Conference on Software Testing
ICIS	Computer and Information Science
ACM SIGSOFT	Software Engineering Notes
MSIS	Workshop on Variability Modeling of Software-Intensive Systems
ISPL	International Software Product Line Conference
ICSTSS	International Conference on Testing Software and Systems
QA&TEST	International Conference on Software QA and Testing
IFIP	International Conference on Testing Software and Systems

Table 5: Defined Journals for Manual Search

Acronym	Journal
STVR	Software Testing, Verification & Reliability
ESE	Empirical Software Engineering
JSS	Journal of Systems and Software
IST	Information and Software Technology
ASC	Applied Soft Computing
SQJ	Software Quality Journal
SCP	Science of Computer Programming

3.2.1. Protocol Assessment

We elaborated a questionnaire[1] for assessing our SMS protocol to collect the opinion of researchers who worked with MBT and/or SPL, for facilitating the construction of the research. The questionnaire is composed of eight items, seven questions in a Likert scale-like [33] and one open question, as follows:

1. This study relies in an important research question on Model-Based Testing (MBT) of Software Product Lines (SPL).
2. The search string is adequately derived from the research questions.
3. The defined search sources are enough to cover relevant studies.
4. Inclusion and exclusion criteria are adequate and properly described.
5. The research can evaluate the quality/validity of the selected studies.
6. The extraction process adequately approaches the research questions.
7. The analysis process is appropriate to answer the research questions.

Each likert-like question has the following options as answers:

[1] https://forms.gle/Eji6S7go1Jeft1Rt6

- **I Fully Disagree:** when the protocol does not meet the criteria of the question in any way;
- **I Partially Disagree:** when the protocol does not meet some criteria of the question;
- **Neutral:** when the protocol does not make it clear whether or not it meets the criteria of the question;
- **I Partially Agree:** when the protocol meets some criteria of the question; and
- **I Fully Agree:** when the protocol fully meets the criteria of the question.

At the bottom of the questionnaire, an option was made available for free suggestions on protocol improvements. The main objective of this evaluation was to refine the protocol based on the responses and suggestions of the researchers.

We sent the evaluation questionnaire to seven researchers in the area of Information Systems, Software Engineering, and Electrical Engineering (see Table 6), thus five answered it. We made available the questionnaire for 20 days.

Table 6: Researchers Who Evaluated the SMS Protocol

ID	Education Level	Institution	Expert Area
R.1	Ph.D.	Federal University of Technology-Paraná	Information Systems
R.2	Ph.D.	Federal University of Pampa	Software Engineering
R.3	Ph.D.	Universidad Politècnica de València Espanha	Electrical Engineering
R.4	Ph.D.	Federal University of Paraná	Software Engineering
R.5	M.Sc.	SENAI Institute of Technology in Mechanics	Electrical Engineering

The invited researchers have made a very productive evaluation. All of them consider the research can generate satisfactory data based on the context presented on the topic. The majority also agree the Search String is adequate based on the research questions.

Another point of agreement is that the sources and types of research may cover relevant studies, but with relation to the inclusion and exclusion criteria, some disagreed, for example, that the selection of papers is limited to only the last 10 years of research. In the end, we did not constraint our searches for the last 10 years.

There was a question about the quality analysis of the works, in this case it was suggested the application of the quality criterion questionnaire.

One suggestion was to perform an evaluation based on the number of citations of a given study. This suggestion was applied in conjunction with the criteria questionnaire.

3.2.2. Electronic Search

We performed the electronic search in January/2017. We derived our general search string to each of the electronic sources, as presented in Table 7.

Table 7: Electronic Sources and Respective Adapted Search Strings

Electronic Source	Adapted Search String
ACM	(*"software" +"product line" "product-line" "product family" "product-family" "product-families" "family of products" "variability" +"model-based testing" "model based testing" "MBT"*)
Google Scholar	(*"Software") AND ("product line" or "product-line" or "product family" or "product-family" or "product-families" or "family of products" or "variability") AND ("model-based testing" or "model based testing" or "MBT"*)
IEEE	((*"software") AND ("product line" OR "product-line" OR "product family" OR "product-family" OR "product-families" OR "family of products" OR "variability") AND ("model-based testing" OR "model based testing" OR "MBT"*))
Springer	((*software) AND ("product line" or "product-line" or "product family" or "product-family" or "product-families" or "family of products" or "variability") AND ("model-based testing" or "model based testing" or MBT)*
Scopus	*(TITLE-ABS-KEY (software) AND TITLE-ABS-KEY (product line OR product-line OR product family OR product-family OR product-families OR family of products OR variability) AND TITLE-ABS-KEY (model-based testing OR model based testing OR MBT))*
Science Direct	(*"software") AND ("product line" OR "product-line" OR "product family" OR "product-family" OR "product-families" OR "family of products" OR "variability") AND ("model-based testing" OR "model based testing" OR "MBT")*
IET Digital Library	((*"software") AND ("product line" OR "product-line" OR "product family" OR "product-family" OR "product-families" OR "family of products" OR "variability") AND ("model-based testing" OR "model based testing" OR "MBT"))*

The electronic search returned 251 studies as we can see in Table 8, column "Returned". Science Direct was the most returning source with 125 studies followed by Scopus with 62 studies.

Table 8: Number of Studies from Electronic Sources

Electronic Source	Returned	Accepted	Duplicated	Rejected
Science Direct	125	07	00	118
Scopus	62	10	21	31
IEEE Xplore	25	00	00	25
ACM	20	16	01	03
IET Digital Library	09	00	00	09
Springer	07	05	01	01
Google Scholar	03	02	00	01
Total	**251**	**40**	**23**	**188**

3.2.3. Manual Search

Manual search returned 20 studies from conferences, workshops and journals, as presented in Table 9, column "Returned".

3.3. Selection Process

Figure 4 depicts the SMS selection process. We performed four main activities in this process: **Definition of Inclusion/Exclusion Criteria**, which describe criteria to maintain or remove studies from the initial set of studies;

Table 9: Number of Studies from Manual Sources

Source	Returned	Accepted	Rejected
Software & Systems Modeling	01	01	00
Model Driven Engineering Languages and Systems	01	00	01
Software Quality Journal	01	01	00
Software Testing, Verification and Validation (ICST)	01	01	00
Computer and Information Science (ICIS)	01	01	00
ACM SIGSOFT Software Engineering Notes	01	01	00
Workshop on Variability Modeling of Software-Intensive Systems	01	00	01
International Software Product Line Conference	01	00	01
IEEE Software	01	00	01
Google Scholar	11	06	05
Total	**20**	**11**	**09**

Screening Title and Abstract of Studies, in which we analyze title, abstract of each selected study in accordance to our SMS subject; **Full Reading of Studies**, in which we fully ready all remaining studies; and **Snowballing Studies**, in which we search for related studies by citations of each fully read study.

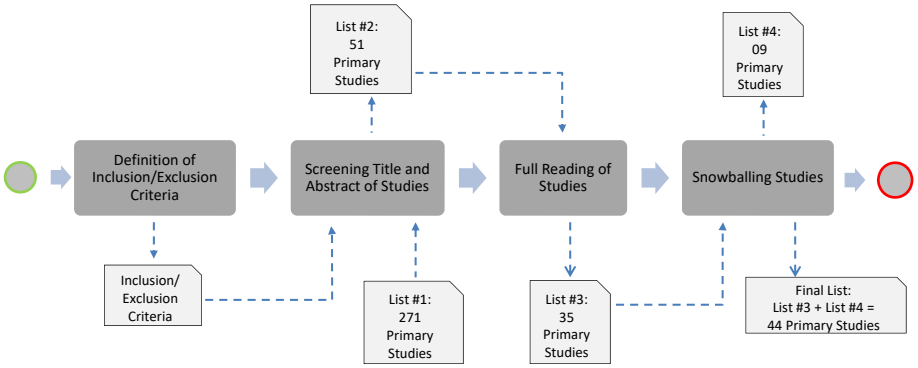

Figure 4: SMS Selection Process

Next sections present how we performed the selection process.

3.3.1. Definition of Inclusion and Exclusion Criteria

To select relevant studies and contribute to answer the research questions of this study, we defined the following inclusion and exclusion criteria:

- **Inclusion Criterion:**
  - IC.1 - studies discussing MBT activities performed exclusively to the SPL domain.
- **Exclusion Criteria:**

– EC.1 - studies do not addressing MBT activities performed exclusively to the SPL domain;

– EC.2 - studies with incomplete text;

– EC.3 - studies in a language other than English to promote international dissemination and reproducibility;

– EC.4 - studies with less than four pages, as we believe they have no space for relevant contributions;

– EC.5 - duplicated studies;

– EC.6 - unavailable studies, even in contact with authors.

3.3.2. Screening Titles and Abstracts of Studies

After defining inclusion and exclusion criteria, we applied them to studies by reading titles and abstracts of List #1 (271 studies). Such reading filtered List #1 for accepting 51 studies (List #2), where 40 studies come from electronic search (Table 8, column "Accepted") and 11 from manual search (Table 9, column "Accepted").

Table 10: Papers Selected for Full Reading

Title	Year	Venue	Pub. Type
Delta-Oriented Model-Based SPL Regression Testing [34]	2012	ACM	Event
Industrial Evaluation of Pairwise SPL Testing with MoSo-PoLiTe [35]	2012	ACM	Event
Model-Based Coverage-Driven Test Suite Generation for Software Product Lines [36]	2011	ACM	Journal
MoSo-PoLiTe - Tool Support for Pairwise and Model-Based Software Product Line Testing [37]	2011	ACM	Event
MPLM - MaTeLo Product Line Manager [38]	2014	ACM	Event
On the use of test cases in model-based software product line development [39]	2014	ACM	Event
Pairwise Feature-Interaction Testing for SPLs: Potentials and Limitations [40]	2011	ACM	Event
Deriving Usage Model Variants for Model- based Testing: An Industrial Case Study [41]	2014	IEEE	Event
Model-based Software Product Line Testing by Coupling Feature Models with Hierarchical Markov Chain Usage Models [42]	2016	IEEE	Event
Model-Based Test Design of Product Lines: Raising Test Design to the Product Line Level [43]	2014	IEEE	Journal
Requirements-Based Delta-Oriented SPL Testing [44]	2013	IEEE	Event
Using Feature Model to Support Model-Based Testing of Product Lines: An Industrial Case Study [28]	2013	IEEE	Journal
An automated Model-based Testing Approach in Software Product Lines Using a Variability Language [45]	2010	Politécnica Arquivo digital UPM	Event
Automated Product Line Methodologies to Support Model-Based Testing [46]	2013	CEUR Proceedings	Event
Behavioural Model Based Testing of Software Product Lines [47]	2014	ACM	Event
Feature Model-based Software Product Line Testing [48]	2012	TUprints Darmstadt publication service	Journal
Model-based pairwise testing for feature interaction coverage in software product line engineering [49]	2011	Springer	Journal
Model-based Test Generation for Software Product Line [50]	2013	IEEE	Event
Model-Based Testing for Software Product Lines [24]	2008	Springer	Event
PLETS - A Product Line of Model- Based Testing Tools [27]	2013	PUC-RS	Event

Table 10 continued from previous page

Title	Year	Venue	Pub. Type
Top-Down and Bottom-Up Approach for Model-Based Testing of Product Lines [51]	2013	EPTCS	Event
A Product Line Modeling and Configuration Methodology to Support Model-Based Testing: An Industrial Case Study [52]	2012	Springer	Journal
Coverage Criteria for Behavioural Testing of Software Product Lines [53]	2014	Springer	Event
A Model Based Testing Approach for Model-Driven Development and Software Product Lines [19]	2010	Springer	Event
A Vision for Behavioural Model-Driven Validation of Software Product Lines [54]	2012	Springer	Event
Abstract Test Case Generation for Behavioural Testing of Software Product Lines [55]	2014	ACM	Event
Applying Incremental Model Slicing to Product-Line Regression Testing [56]	2016	Springer	Journal
Automated Testing of Software-as-a-Service Configurations using a Variability Language [57]	2015	ACM	Event
Delta-Oriented FSM-Based Testing [58]	2015	Springer	Event
Incremental Model-Based Testing of Delta-oriented Software Product Lines [59]	2012	Springer	Journal
Model Based Testing in Software Product Lines [21]	2011	Springer	Event
Model-Based Testing [60]	2014	Springer	Event
Parameterized Preorder Relations for Model-Based Testing of Software Product Lines [61]	2012	Springer	Event
Poster: VIBeS, Transition System Mutation Made Easy [62]	2015	IEEE	Event
Spinal Test Suites for Software Product Lines [63]	2014	EPTCS	Event
Automated model-based testing using the UML testing profile and QVT [20]	2009	ACM	Event
Relating Variability Modeling and Model-Based Testing for Software Product Lines Testing [64]	2012	ICTSS	Event
An Evaluation of Model-Based Testing in Embedded Applications [65]	2014	IEEE	Event
Assessing Software Product Line Testing Via Model-Based Mutation An Application to Similarity Testing [66]	2013	IEEE	Event
Automated and Scalable T-wise Test Case Generation Strategies for Software Product Lines [67]	2010	IEEE	Event
Model-based Testing of System Requirements using UML Use Case Models [68]	2008	IEEE	Event
Successive refinement of models for model-based testing to increase system test effectiveness [69]	2016	IEEE	Event
A Software Product Line for Model-Based Testing Tools [27]	2012	PUC-RS	Event
Reusing State Machines for Automatic Test Generation in Product Lines [70]	2008	MoTip	Event
A Novel Markov Boundary Based Feature Subset Selection Algorithm [71]	2010	Elsevier	Journal
A Novel Model-Based Testing Approach for Software Product Lines [72]	2016	Springer	Event
Abstract Test Case Generation for Behavioural Testing of Software Product Lines [55]	2014	ACM	Event
An Overview on Test Generation from Functional Requirements [73]	2011	Elsevier	Journal
Basic Behavioral Models for Software Product Lines: Expressiveness and Testing Pre-Orders [74]	2016	Elsevier	Journal
Bottom-up reuse for multi-level testing [75]	2010	Elsevier	Journal
Colored Model Based Testing for Software Product Lines [76]	2010	Tec. Univ. Ilmenau	PhD Thesis
IncLing: Efficient Product-Line Testing Using Incremental Pairwise Sampling [77]	2016	ACM	Event
Input-output conformance testing based on featured transition systems [78]	2014	ACM	Event
Model-based testing of automotive systems [79]	2008	IEEE	Event
Model-based system testing of software product families [80]	2005	Springer	Event
Model-based testing for applications derived from software product lines [81]	2005	ACM	Event
Reducing the Concretization Effort in FSM-Based Testing of Software Product Lines [82]	2016	IEEE	Event
Refinement-based testing of delta-oriented product lines [83]	2013	ACM	Event
Risk-based integration testing of software product lines [84]	2016	ACM	Event
Uncertainty-wise evolution of test ready models [85]	2016	Elsevier	Journal

3.3.3. Full Reading of Studies

Given List #2, we full read each of the 51 studies. We rejected 16 of them (see Table 11) based on exclusion criteria from Section 3.3.1, thus providing List #3 with 35 studies.

Table 11: Removed Studies According to Exclusion Criteria

Title	**EC.1**	**EC.2**	**EC.3**	**EC.4**	**EC.5**	**EC.6**
A Novel Markov Boundary Based Feature Subset Selection Algorithm [71]	X	-	-	-	-	-
A Novel Model-Based Testing Approach for Software Product Lines [72]	-	-	-	-	X	-
Abstract Test Case Generation for Behavioural Testing of Software Product Lines[55]	X	-	-	-	-	-
An Overview on Test Generation from Functional Requirements [73]	X	-	-	-	-	-
Basic Behavioral Models for Software Product Lines: Expressiveness and Testing Pre-Orders [74]	-	-	-	X	-	-
Bottom-up reuse for multi-level testing [75]	X	-	-	-	-	-
Colored Model Based Testing for Software Product Lines [76]	X	-	-	-	X	-
IncLing: Efficient Product-Line Testing Using Incremental Pairwise Sampling [77]	X	-	-	-	-	-
Input-output conformance testing based on featured transition systems [78]	X	-	-	-	-	-
Model-based testing of automotive systems [79]	-	-	-	X	-	-
Model-based system testing of software product families [80]	-	-	-	-	X	-
Model-based testing for applications derived from software product lines [81]	-	-	-	-	X	-
Reducing the Concretization Effort in FSM-Based Testing of Software Product Lines [82]	X	-	-	-	-	-
Refinement-based testing of delta-oriented product lines [83]	X	-	-	-	-	-
Risk-based integration testing of software product lines [84]	X	-	-	-	-	-
Uncertainty-wise evolution of test ready models [85]	X	-	-	-	-	-

3.3.4. Snowballing Studies

We performed reverse snowballing in List #3, where we evaluated the references of the studies. Ten studies returned from snowballing, and inclusion and exclusion criteria were applied to them. Therefore, one of them was duplicated as presented in Table 12. The nine remaining studies composed List #4.

Table 12: Studies from Snowballing

ID	Source Study	ID	Snowballing Study	Status
S24	A Model Based Testing Approach for Model-Driven Development and Software Product Lines [19]	S36	Automated model-based testing using the UML testing profile and QVT [20]	Accepted
S8	Deriving Usage Model Variants for Model-based Testing: An Industrial Case Study [41]	S37	Relating Variability Modeling and Model-Based Testing for Software Product Lines Testing [64]	Accepted
S10	Model-Based Test Design of Product Lines: Raising Test Design to the Product Line Level [43]	S38	An Evaluation of Model-Based Testing in Embedded Applications [65]	Accepted
S10	Model-Based Test Design of Product Lines: Raising Test Design to the Product Line Level [43]	S39	Assessing Software Product Line Testing Via Model-Based Mutation An Application to Similarity Testing [66]	Accepted
S5	MPLM-MaTeLo Product Line Manager [38]	S40	Automated and Scalable T-wise Test Case Generation Strategies for Software Product Lines [67]	Accepted
S20	PLETS - a Product Line of Model-Based Testing Tools [86]	S41	Model based testing of system requirements using UML use case models [68]	Accepted
S9	Model-based Software Product Line Testing by Coupling Feature Models with Hierarchical Markov Chain Usage Models [42]	S42	Successive refinement of models for model-based testing to increase system test effectiveness [69]	Accepted
S20	PLETS - a Product Line of Model-Based Testing Tools [86]	S43	A Software Product Line for Model-Based Testing Tools [27]	Accepted
S21	Top-Down and Bottom-Up Approach for Model-Based Testing of Product Lines [87]	S44	Reusing State Machines for Automatic Test Generation in Product Lines [70]	Accepted
S5	MPLM-MaTeLo Product Line Manager [38]	–	An approach to derive usage models variants for model-based [88]	Duplicate

Table 13 presents the final set of selected studies for this SMS, composed of 44 studies (Final List), from which: the first 35 are from electronic and manual searches and remaining nine are from snowballing.

Table 13: Final List of Studies

ID	Title	Year	Venue	Venue Type
S1	Delta-Oriented Model-Based SPL Regression Testing [34]	2012	PLEASE	Workshop
S2	Industrial Evaluation of Pairwise SPL Testing with MoSo-PoLiTe [35]	2012	VaMoS	Workshop
S3	Model-Based Coverage-Driven Test Suite Generation for Software Product Lines [36]	2011	MODELS	Conference
S4	MoSo-PoLiTe - Tool Support for Pairwise and Model-Based Software Product Line Testing [37]	2011	ACM	Event
S5	MPLM - MaTeLo Product Line Manager [38]	2014	VaMoS	Workshop
S6	On the use of test cases in model-based software product line development [39]	2014	SPLC	Conference
S7	Pairwise Feature-Interaction Testing for SPLs: Potentials and Limitations [40]	2011	SPLC	Conference
S8	Deriving Usage Model Variants for Model- based Testing: An Industrial Case Study [41]	2014	ICECCS	Conference
S9	Model-based Software Product Line Testing by Coupling Feature Models with Hierarchical Markov Chain Usage Models [42]	2016	QRS-C	Conference
S10	Model-Based Test Design of Product Lines: Raising Test Design to the Product Line Level [43]	2014	ICST	Conference
S11	Requirements-Based Delta-Oriented SPL Testing [44]	2013	PLEASE	Workshop
S12	Using Feature Model to Support Model-Based Testing of Product Lines: An Industrial Case Study [28]	2013	QSIC	Conference
S13	An automated Model-based Testing Approach in Software Product Lines Using a Variability Language [45]	2010	ECMDA	Workshop
S14	Automated Product Line Methodologies to Support Model-Based Testing [46]	2013	MODELS	Conference
S15	Behavioural Model Based Testing of Software Product Lines [47]	2014	Univ. Namur	PhD Thesis
S16	Feature Model-based Software Product Line Testing [48]	2012	Technische Univ.	PhD Thesis
S17	Model-based pairwise testing for feature interaction coverage in software product line engineering [49]	2011	SQJ	Journal
S18	Model-based Test Generation for Software Product Line [50]	2013	ICIS	Conference
S19	Model-Based Testing for Software Product Lines [24]	2008	G. Mason Univ.	PhD Thesis
S20	PLETS - A Product Line of Model- Based Testing Tools [27]	2013	PUCRS	OhD Thesis
S21	Top-Down and Bottom-Up Approach for Model-Based [87] Testing of Product Lines	2013	MBT	Conference
S22	A Product Line Modeling and Configuration Methodology to Support Model-Based Testing: An Industrial Case Study [52]	2012	MODELS	Conference
S23	Coverage Criteria for Behavioural Testing of Software Product Lines [53]	2014	ISoLA	Conference
S24	A Model Based Testing Approach for Model-Driven Development and Software Product Lines [19]	2010	ENASE	Conference
S25	A Vision for Behavioural Model-Driven Validation of Software Product Lines [54]	2012	ISoLA	Conference
S26	Abstract Test Case Generation for Behavioural Testing of Software Product Lines [55]	2014	SPLC	Conference
S27	Applying Incremental Model Slicing to Product-Line Regression Testing [56]	2016	ICSR	Conference
S28	Automated Testing of Software-as-a-Service Configurations using a Variability Language [57]	2015	SPLC	Conference
S29	Delta-Oriented FSM-Based Testing [58]	2015	ICFEM	Conference
S30	Incremental Model-Based Testing of Delta-oriented Software Product Lines [59]	2012	TAP	Conference
S31	Model Based Testing in Software Product Lines [21]	2011	ICEIS	Conference
S32	Model-Based Testing [60]	2014	SFM	Conference
S33	Parameterized Preorder Relations for Model-Based Testing of Software Product Lines [61]	2012	ISoLA	Conference
S34	Poster: VIBeS, Transition System Mutation Made Easy [62]	2015	ICSE	Conference
S35	Spinal Test Suites for Software Product Lines [63]	2014	EPTCS	Conference
S36	Automated model-based testing using the UML testing profile and QVT [20]	2009	MoDeVVa	Workshop

Table 13 continued from previous page

ID	Title	Year	Venue	Venue Type
S37	Relating Variability Modeling and Model-Based Testing for Software Product Lines Testing [64]	2012	ICTSS	Conference
S38	An Evaluation of Model-Based Testing in Embedded Applications [65]	2014	ICST	Conference
S39	Assessing Software Product Line Testing Via Model-Based Mutation An Application to Similarity Testing [66]	2013	ICSTW	Workshop
S40	Automated and Scalable T-wise Test Case Generation Strategies for Software Product Lines [67]	2010	ICST	Conference
S41	Model-based Testing of System Requirements using UML Use Case Models [68]	2008	ICST	Conference
S42	Successive refinement of models for model-based testing to increase system test effectiveness [69]	2016	ICSTW	Workshop
S43	A Software Product Line for Model-Based Testing Tools [27]	2012	JSS	Journal
S44	Reusing State Machines for Automatic Test Generation in Product Lines [70]	2008	MoTip	Workshop

3.4. Extraction Process

Figure 5 depicts the SMS extraction process. In this process, we performed three activities: **Definition of Classification Scheme**, in which we defined how selected studies are initially classified; **Quality of Studies Analysis**, in which we analyzed the quality of each selected study based on criteria we defined; and **Data Extraction**, in which we extract data from the selected studies to organize our mapping.

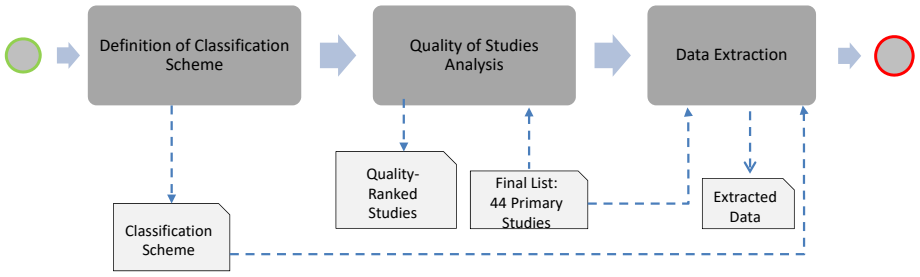

Figure 5: SMS Extraction Process

3.4.1. Definition of Classification Scheme

Once we extracted data from primary studies, we classify them mainly based on three MBT of SPL stages: **Early Stage**, in which an initial model from the SPL domain engineering activity, represented by an artifact, such as UML diagrams or state machines, is taken into consideration; **Second Stage (optional)**, which takes as input the initial artifact and then transform it usually a more formal one, with the optional support of a (semi-)automated tool to manage variability; and, **Final Stage**, in which test sequences are generated from the artifact with an automated support, and the option to manage traceability.

3.4.2. Quality of Studies Analysis

For quality evaluation of studies, we defined a questionnaire and applied it to each study. Therefore, we elaborated the following three questions:

- Is the study pair reviewed?
- Is the purpose of the study clear?
- Is the study proposal evaluated/validated?

For each question there were three alternatives, in which only one of them could be chosen: "No", "Yes", and "Partially". Studies with "No" for any questions is automatically discarded. We carefully read again studies with "Partially" as an answer to any questions.

We adopted this procedure, as well as the application of inclusion and exclusion criteria to make sure minimal contributions from selected studies to the extraction process.

We performed data extraction to collect information needed to answer the research questions, as well as to analyze the studies of the selection criteria. Therefore, we defined the following quality criteria (QC) to assure minimal quality of studies:

- QC.1 - Does the study clearly describe the purpose of the research?
- QC.2 - Is the field of action compatible with the research area?
- QC.3 - How study is evaluated?
- QC.4 - Was there an appropriate data collection?
- QC.5 - Was there an appropriate data analysis?
- QC.6 - Does the study present results consistent with its objectives?
- QC.7 - Does the results contribute to the process carried out for the research?

3.4.3. Data Extraction

Next items present the list of data we defined to be extracted from each primary study:

- Bibliometric data: Title, Author(s), Publication Year, Publication Source, Publication Type, Document Type;
- Solution Domain;
- Includes Variability?;
- Feature Interaction?;
- Method of Execution;

- Test Item Used;
- Test Level Applied;
- Test Approach Used;
- Level of Coverage;
- Treats Traceability?;
- Purpose of MBT;
- Non-Functional Requirement Test is Supported?;
- Artifact of Origin;
- Formal (Intermediate) Artifact;
- Use of Tools;
- Binding Time;
- MBT Activities Evaluated;
- Search Results;
- Method of Evidence Collection;
- Validation Venue;
- Type of Contribution.

3.5. Analysis Process

Figure 6 depicts the SMS analysis process.

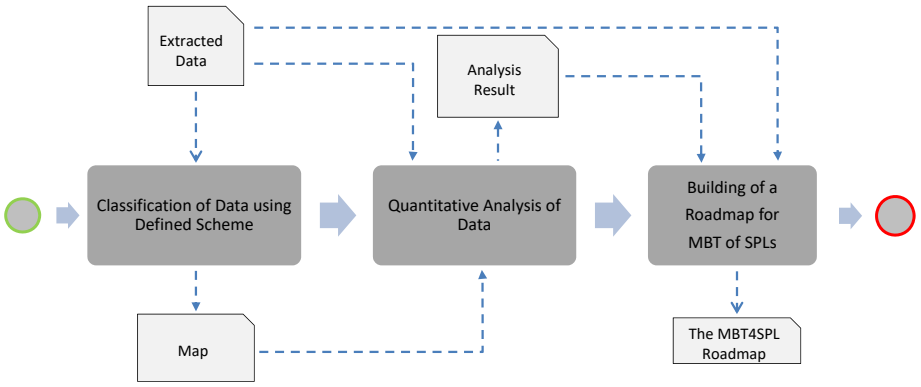

Figure 6: SMS Analysis Process

We performed three activities in this process: **Classification of Data using Defined Scheme**, using the defined classification scheme we organize the selected studies in terms of MBT of SPL stages compliance; **Quantitative Analysis of Data**, in which we analyze all extracted data in terms of quantities, frequencies, percentages, etc (see Section 4); and **Building of a Roadmap for MBT of SPLs**, in which we developed a roadmap of the studies mainly based on the classification scheme on MBT of SPL stages (see Section 7).

4. Results

This section presents our SMS results based on the characterization of the analyzed studies (Section 4.1) and the proposed research questions (Section 4.2).

4.1. Characterization of Studies

4.1.1. Studies per Year

It is important to understand how the research topic of MBT applied to SPL has behaved throughout the last years. Therefore, Table 14 presents the number of studies per year and Figure 7 depicts the distribution of such studies over the years.

Table 14: Number of studies per year

Venue Type	2008	2009	2010	2011	2012	2013	2014	2015	2016	Count
Conference	1	1	1	2	-	3	6	3	2	19
Workshop	2	1	1	1	4	3	1	-	-	13
Journal	-	-	-	2	3	1	1	-	1	8
Symposium	-	-	-	-	2	-	2	-	-	4
Total	3	2	2	5	9	7	10	3	3	44

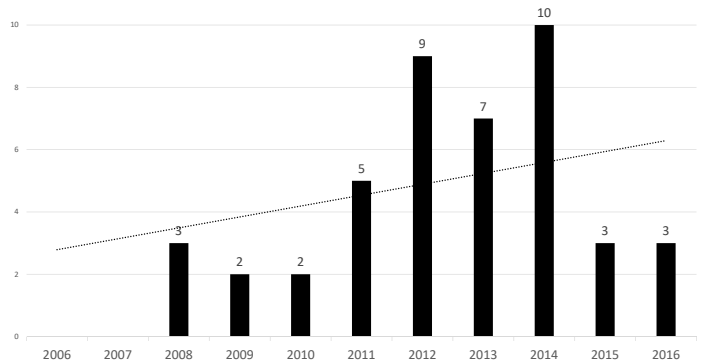

Figure 7: Distribution of Studies per Year

Based on Figure 7, we indeed observe an increasing number of studies over the years. Special attention is given to 2014 with 10 studies. In addition, excepting 2006 and 2007 with no occurrence, studies are well distributed in most years as in 2008, 2009, 2010, 2011, 2015, and 2016.

4.1.2. Quality of Studies

To guarantee minimal quality of the studies, we performed a quality assessment as planned in Section 3.4, taking into consideration the application of quality criteria from Section 3.4.2.

All studies from the Final List have minimal quality based on such criteria, as we can see in Table 15 with most responses "Yes".

Table 15: Quality Assessment of Studies

Study ID	QC.1-Purpose	QC.2-Compatibility	QC.3-Evaluation	QC.4-Data Collection	QC.5-Data Analysis	QC.6-Results Consistency	QC.7-Contribution
S1	Yes	No	Yes	Yes	Yes	Yes	Yes
S2	Yes	No	Yes	Yes	Yes	Yes	Yes
S3	Yes	No	Yes	Yes	Yes	Yes	Yes
S4	Yes	No	Not Informed	Yes	Yes	Yes	Yes
S5	Yes	No	Not Informed	Not Informed	Not Informed	Yes	Yes
S6	Yes	No	Not Informed	Yes	Yes	Yes	Yes
S7	Yes	No	Yes	Yes	Yes	Yes	Yes
S8	Yes	No	Yes	Yes	Yes	Yes	Yes
S9	Yes	No	Yes	Yes	Yes	Yes	Yes
S10	Yes	No	Yes	Yes	Yes	Yes	Yes
S11	Yes	No	No	Yes	Yes	Yes	Yes
S12	Yes	No	Yes	Yes	Yes	Yes	Yes
S13	Yes	Yes	Not Informed	Yes	Yes	Yes	Yes
S14	Yes	No	Yes	Yes	Yes	Yes	Yes
S15	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S16	Yes	No	Not Informed	Yes	Yes	Yes	Yes
S17	Yes	No	Yes	Yes	Yes	Yes	Yes
S18	Yes	Yes	Not Informed	Yes	Yes	Yes	Yes
S19	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S20	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S21	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S22	Yes	No	Yes	Yes	Yes	Yes	Yes
S23	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S24	Yes	Yes	Not Informed	Yes	Yes	Yes	Yes
S25	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S26	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S27	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S28	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S29	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S30	Yes	No	Yes	Yes	Yes	Yes	Yes
S31	Yes	Yes	Yes	Not Informed	Not Informed	Yes	Yes
S32	Yes	Yes	Not Informed	Not Informed	Not Informed	Yes	Yes
S33	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S34	Yes	Yes	Not Informed	Not Informed	Not Informed	Yes	Yes
S35	Yes	Yes	Yes	Not Informed	Not Informed	Yes	Yes
S36	Yes	Yes	Yes	Not Informed	Not Informed	Yes	Yes
S37	Yes	Yes	No	Not Informed	Not Informed	Yes	Yes
S38	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S39	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S40	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S41	Yes	No	Not Informed	Not Informed	Not Informed	Yes	Yes
S42	Yes	No	Yes	Yes	Yes	Yes	Yes
S43	Yes	Yes	Yes	Yes	Yes	Yes	Yes
S44	Yes	No	Yes	Yes	Yes	Yes	Yes

4.2. Results on Research Questions

In this section we present results based on each pre-defined research questions.

Table 16 traces which studies contributed to answer the research questions. This is important when one wants to gather up studies for a given research question. For example, study S11 contributes to answer RQ.1, RQ.3 and RQ.5.

Table 16: Contributor Studies for Research Questions

Study ID/Ref.	RQ.1	RQ.2	RQ.3	RQ.4	RQ.5	RQ.6
S1 [34]	✓	✓			✓	
S2 [35]	✓	✓			✓	
S3 [36]	✓				✓	
S4 [37]	✓	✓				
S5 [38]	✓	✓				✓
S6 [39]	✓	✓				
S7 [40]	✓	✓	✓		✓	✓
S8 [41]	✓	✓	✓		✓	
S9 [42]	✓	✓	✓		✓	
S10 [43]	✓	✓	✓		✓	
S11 [44]	✓		✓		✓	
S12 [28]	✓	✓	✓	✓	✓	
S13 [45]	✓	✓	✓			
S14 [46]	✓	✓	✓		✓	
S15 [47]	✓		✓		✓	✓
S16 [48]	✓	✓	✓			✓
S17 [49]	✓	✓	✓		✓	✓
S18 [50]	✓		✓			
S19 [24]	✓	✓	✓		✓	✓
S20 [27]	✓	✓			✓	✓
S21 [87]	✓		✓		✓	
S22 [52]	✓	✓	✓		✓	
S23 [53]	✓				✓	
S24 [19]	✓		✓			✓
S25 [54]	✓		✓		✓	
S26 [55]	✓				✓	
S27 [56]	✓		✓		✓	
S28 [57]	✓	✓	✓		✓	
S29 [58]	✓		✓		✓	
S30 [59]	✓	✓	✓		✓	
S31 [21]	✓	✓	✓		✓	
S32 [60]	✓		✓			✓
S33 [61]	✓		✓		✓	
S34 [62]	✓	✓	✓			
S35 [63]	✓		✓			
S36 [20]	✓	✓				✓
S37 [64]	✓	✓	✓			✓
S38 [65]	✓	✓			✓	
S39 [66]	✓				✓	
S40 [67]	✓	✓	✓		✓	
S41 [68]	✓	✓	✓			✓
S42 [69]	✓	✓			✓	
S43 [27]	✓				✓	
S44 [70]	✓		✓		✓	
Count	44	27	30	01	31	12

For RQ.4, only study S12 mentions something on testing of non-functional requirements, however it does not provide any details.

4.2.1. RQ.1: To which SPL application domains, solution types, and proposals MBT is used?

For answering this question we take into consideration MBT applied to SPL domains (*e.g.* software, automotive), SPL research area solution types (*e.g.* feature modeling), and the type of the studies proposal (*e.g.* approach, technique, tool).

Application Domains. We identified seven distinct application domains in which MBT for SPL is applied to. Table 17 lists each domain and respective studies.

Table 17: MBT of SPL Application Domains

Application Domain	Study ID	Count
Aerospace	S5, S8	02
Automotive	S1, S3, S4, S7, S10, S11, S16, S17, S30	09
Electronic	S2, S9, S42, S44	04
Manufacturing	S6	01
Medicine	S41	01
Software	T13, S15, S18, S19, S20, S21, S23, S24, S25, S26, S27, S28, S29, S31, S32, S33, S34, S35, T36, S37, S38, S39, S40, S43	24
Telecommunication	S12, S14, S22	03
TOTAL		**44**

The `Software` application domain has most occurrences with 24 studies, followed by `Automotive` with nine, which uses software embedded in its products.

Aerospace. For the development of aerospace products it is required a process model that allows to evaluate variability factors and derivation of products, for example. In this domain experimental works were presented by companies such as for the Airbus [38, 41].

Automotive. Cars have increasing embedded systems or high technology devices, thus the process model or design must take into account such factors. As this domain increases every year, MBT provides a means to develop secure and safe cars and devices [34, 43, 44, 36, 48, 49, 37, 40, 59].

Electronic. The electronic domain uses models to generate subgroups or test cases of derived products. Issues on the interaction of resources with test coverage and management of the variabilities as crossed items should be addressed. Certain tools are proposed to assist in this process. Thus, MBT acts in the generation of test cases for subgroups of products, in addition to seeking to guarantee greater coverage of functionality errors, thus, quality assurance must be significant [35, 42, 69, 70].

Manufacturing. The manufacturing domain presents procedures associated with a set of tools to assign the result of a test case to an arbitrary member of an SPL. To do so, it uses variability models based on UML and CVL. Such models are used for the creation of state machines that are, then, converted into class diagram [39].

Medicine. Critical systems have a high level of complexity and must be rigorously tested. The focus on the generation of test for derivative products carries out the conversion of models into activity diagrams and use case. After this conversion, one creates test cases, in which variants of the models can be considered. When using MBT, the study of this domain does not mention whether variability is managed, simply used to create test cases from activity diagrams [68].

Software. Software is one of the most reported domains. Although other domains also use MBT in the process of developing software products, the software domain provides a lot of data and information. Works in this domain mainly focus on custom activity diagrams for reusing test cases on SPL variant product testing.

Telecommunication. MBT acts to aid in the derivation of test cases from state machines to analyze performance, as the use of media embedded in the communication devices is significant in this domain.

The vast majority of studies reports the creation of approaches or tools for MBT to aid in the development of SPL [45, 47, 50, 24, 27, 87, 53, 19, 54, 55, 56, 57, 58, 21, 60, 61, 62, 63, 20, 64, 65, 66, 67, 67]. Automated processes contribute to the selection of algorithms, however much of them are used to create a systemic view of variability selection [28, 46, 52].

Solution Types. Table 18 lists studies performed to testing different SPL solution types as: `Feature Testing`, `SPL Model Testing`, and `PLA Testing` (Product-Line Architecture - PLA). For this classification, we separated PLA from SPL general artifacts, because of the straightforward importance of the PLA for the success of SPLs.

Table 18: MBT Testing of SPL Solution Types

Solution Type	**Study ID**	**Count**
Feature Testing	S1, S8, S11, S14, S15, S17, S18, S26, S27, S29, S32	11
SPL Model Testing	T2, S3, S4, S5, S9, S12, S13, S19, S20, S21, S24, S30, S36, S37, S39	15
PLA Testing	T6, S7, S10, S16, S22, S23, S25, S28, S31, S33, S34, S35, S38, S40, S41, S42, S43, S44	18
TOTAL		**44**

We can observe in Table 18 most studies focused on testing of PLA. This occurs mainly because of the central role of the PLA for developing SPLs, as it is an abstraction of all potential single-products to be developed by an SPL. Therefore, 18 studies (40.9%) test PLA and its models. The PLA tests are in the level of input and output data, by means of black box tests and functionality tests level. By seeking reuse, there is a concern about the main architecture reliability and on derive SPL products.

With regard to testing of SPL models, we found 34% of the studies take into consideration any kind of testing level or type to SPLs for both domain engineering and application engineering. Testing SPL models is a means to

provide reuse of test cases for specific products derived from an SPL. As pointed out in the general testing literature, detecting defects in models might be several times less costly than in later SPL activities. Testing on SPL models relies heavily on which level it will be tested. In the majority of these studies, tests rely on functionalities, thus models are testing accordingly to the expected result of specified requirements.

Feature testing comprises 25% of studies. Although features are at the problem space level, their testings are essential for the success of an SPL since features are directly related with SPL models at all levels, including source code. Functional feature testing is according to the coverage of expected actions. Thus, this is a big challenge as with the derivation of new products their functionalities can undergo variations, which might cause an exponential combination of test sequences.

MBT of SPL Proposal Types. As a matter of studies contributions, we analyzed and present them in Table 19 in terms of their proposal types.

Table 19: MBT to SPL Main Proposals

Proposal Type	**Study ID**	**Count**
Approach	S1, S5, S6, S7, S8, S9, S10, S11, S13, S14, S15, S16, S17, S18, S20, S21, S22, S23, S24, S27, S28, S29, S30, S31, S35, S36, S37, S38, S39, S41, S42, S43, S44	33
Algorithm	S3, S25, S26	03
Tool kit	S40	01
Strategy	S33	01
Extension Approach	S12	01
Tool	S2, S4, S34	03
Methodology	S19	01
Tutorial	S32	01
TOTAL		**44**

We can observe the majority of studies (33) present as a proposal contribution an approach. Remaining studies report one to three proposals each.

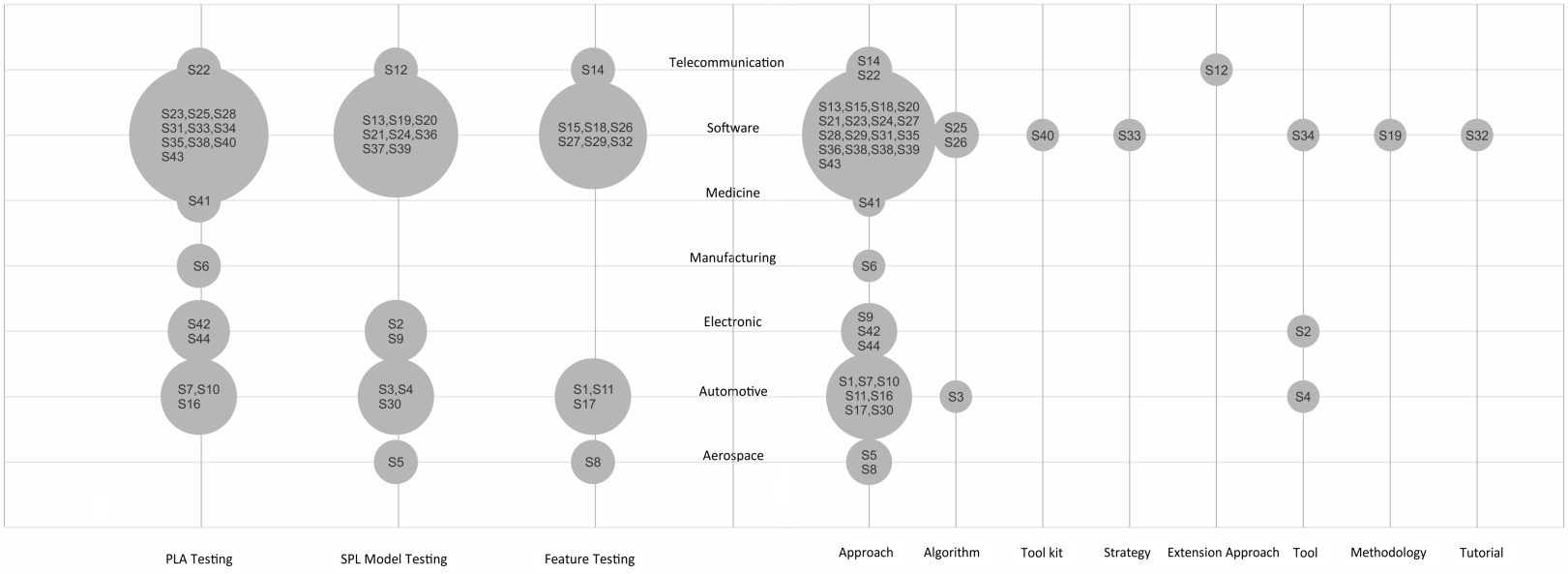

Figure 8: MBT to SPL: Solution Type per Domain and Proposal Type per Domain.

Approach. Approach is one of the most outstanding contributions among the studies of this work. Lackner et al. [43] and Lochau et al. [49] use generation of test models from general artifacts. The implementation of the framework takes into account reusing of product artifacts that contain variability using regression test principles. The majority of works seeks conversion of an initial artifact to another artifact usually more formal, or it does the direct conversion of an artifact in test cases. The main purpose is to generate the test cases directly from a model or starting from a formal artifact [38, 39, 40, 41, 42, 44, 45, 46, 47, 48, 50, 27, 87, 52, 53, 19, 56, 57, 58, 59, 21, 63, 20, 64, 65, 66, 68, 69, 27, 70].

Algorithm. Algorithms also aid in day-to-day software quality. Algorithms generate test cases based on use cases of products generated by an SPL [36]. Or, they are used to classify models according to pre-defined criteria. Thus, such models are used for derivation of test cases [55].

Tool kit. There are software tools to aid in the modeling and the testing process. In this case, Perrouin et al. [67] make use of a set of tools for the derivation of submodels to be used for test case generation.

Strategy. Lochau and Kamischke [61] propose a strategy set with a direct focus on the model and the conversion points based on SPL variation points.

Extension Approach. Enhancing an approach or expanding its coverage is what Wang et al. [28] address by using models of recursion after the transformation of them into state machines.

Tool. Steffens et al. [35], Oster et al. [37], and Devroey et al. [62] seek to create tools for the conversion of initial models to artifacts to be used for the generation of test cases. However, the great challenge is to maintain or treat the variabilities seeking reuse as the main purpose.

Methodology. The major challenge in methodologies is to creating them, thus any model can generate test cases as claimed by Olimpiew [24].

4.2.2. RQ.2: Which MBT approaches, testing levels, artifacts, tools, and models have been used to testing SPLs?

In this section we provide results on approaches, testing levels, artifacts, tools, and models used to apply MBT to SPLs.

Analyzed studies mostly perform design-time tests.

Approaches Used. From selected studies we came up with two "box"-based approaches: white-box and black-box. The former is when one has access to source code and might provide analysis with greater depth. The latter is performed when one has no access to the code, but only to input data and parameters, as well as the produced outcomes.

Table 20 presents the studies classified in each of the approaches.

Table 20: MBT of SPL Approaches

Testing Approach	Study ID	Count
Black-Box	S2, S3, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32, S33, S34, S35, S36, S37, S38, S39, S40, S41, S42, S43, S44	39
White-Box	S1, S4, S20	03
Not Informed	S5, S6	02
TOTAL		**44**

We can observe majority of studies (39/44) runs black-box testing, mostly because: tests are performed at a higher level abstraction as, for instance, functional testing; and SPL in domain engineering activity.

The vast majority of the studies seeks to generate test cases through models, thus this is one of the factors leading to the use of the black box approach. Steffens et al. [35] use models to generate test cases. This practice is done by almost all the other studies [36, 40, 41, 43, 45].

Studies considering white-box testing sought to use the generation of test scripts for automation and testing of source code [34]. Both mentioned in Table 20 are tools to automate a significant part of the process, in this case they use the templates to create test cases, generate the script and apply to the code [37, 27].

Testing Levels. We found different levels and types of MBT testing. From Final List, four studies stand out most (Table 21[2]), which are: `Integration` testing, `System` testing, `Regression` testing, and `Functional`. Figure 9 depicts levels and types of MBT of SPLs.

Table 21: MBT of SPL Testing Levels and Types

Testing Level/Type	Study ID	Count
Combination	S4	01
Structural	S20	01
Functional	S6, S9, S10, S22, S24, S28, S31	07
Performance	S20	01
Regression	S1, S11, S13, S14, S15, S17, S18, S27, S30,	09
System	S25, S26, S33, S34, S35, S36, S38, S39, S40, S41, S42, S43, S44	13
Integration	S3, S5, S7, S8, S11, S13, S14, S15, S16, S17, S18, S19, S21, S23, S29, S37	16
Unit	S2	01
Not Informed	S12, S32	02
TOTAL		**51**

We can observe integration testing level is one of the most considered as most tests are performed in the SPL domain engineering. In addition, for reusing many test cases, derivation of new products requires an integration check among models [89, 40, 41, 48, 90, 36, 53, 58, 64, 44, 45, 46, 91, 50].

[2]One study might support more than one testing level or type.

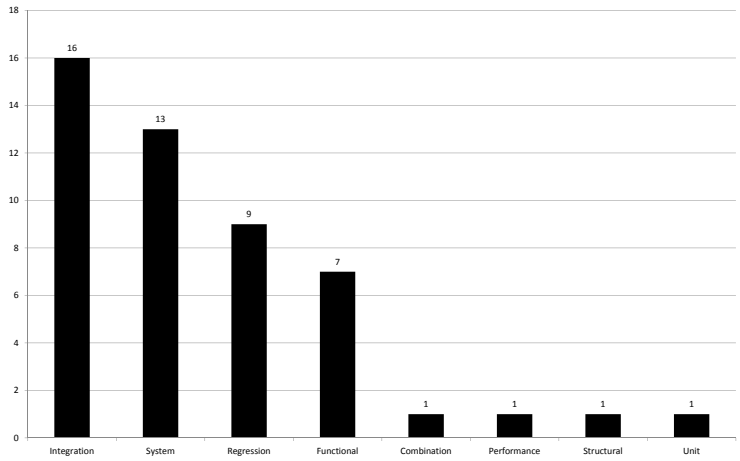

Figure 9: MBT to SPL Levels and Types Comparison

Other studies visualize something similar, but do not work in a modular way. They seek test at system level as some of them claim it is very costly to deal with a great quantity of test sequences in the SPL domain engineering. Therefore, they only focus in the structure of the system [54, 55, 61, 62, 63, 43, 65, 66, 67, 68, 86].

Different test levels are covered by these studies, such as, Combination, Structural, Functional, Performance, Regression, and Unit. These studies have a concern with the level of coverage of the functionalities, and the generation of the test cases in particular ways [56, 39, 42, 43, 44, 92, 46, 49, 50, 52, 57].

Testing Automation. By analyzing the selected studies, we identified the use of fully-automated and semi-automated tools to aid in the MBT testing of SPLs. We also identified studies with no aid of tools, manually developing testing activities, as we can observe in Table 22.

Table 22: MBT of SPL Automation

Automation Approach	**Study ID**	**Count**
Fully-Automated	S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S20, S22, S23, S24, S26, S27, S28, S31, S40, S41	27
Semi-Automated	S19, S21, S25, S29, S30, S34, S36, S37, S38, S39, S42, S43, S44	13
Manual	S1, S33, S35	03
Not Informed	S32	01
TOTAL		**44**

In the selected studies we verified the use of `fully-automated` tools in testing activities. According to selected studies, the majority of them use any kind of automation for performing such activities. Used tools are largely implemented by the authors of studies [37, 35, 62, 67], but such authors also provide comparisons among tools of similar approaches [37, 35, 62, 67, 86].

Another set of selected studies uses `semi-automated` tools, in which only an excerpt of the testing process is automated, with or without manual activities. Olimpiew [90] creates use cases for allowing reusable test case specifications by the software architect. According to Weissleder and Lackner [51], models are used jointly to modeling features and behavior for the generation of state and behavioral machine diagrams by software engineers using semi-automated approaches.

Purely manual studies do not adopt any kind of automation of the testing activities. Such studies discuss theoretical approaches related with MBT of SPL, and thus were classified as `Manual`. In general, manual activities would be the conversion of an initial model into a secondary model with greater detail, then to test cases [56, 61, 63]. Manual approaches tend to present only theoretical concepts of the developed process model, or seek to present the essence regarding an specific context.

Bubble plot of Figure 10 depicts the automation of approaches. As we can observe, black-box testing has greater automation, both by fully-automated and semi-automated tools, totalizing 39 studies. White-box approaches are less performed with the aid of (semi-)automated tools, only three studies.

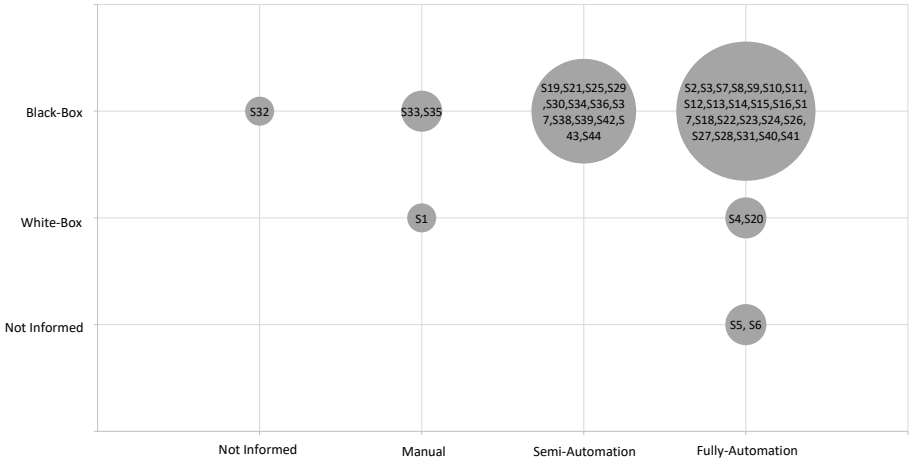

Figure 10: MBT to SPL Automation of Testing Approaches

Bubble plot of Figure 11 depicts the automation of test levels and types.

Thirteen studies are less accomplished with the aid of (semi-)automated tools. Only three studies do not present any kind of automation, whereas one study does not report it.

Artifacts Used. For generating test cases, MBT studies use several different artifacts, such as State Machine, Class Diagram, Sequence Diagram, and Feature Model, as we can observe in Table 23. A comparison of such artifacts is provided in Figure 12.

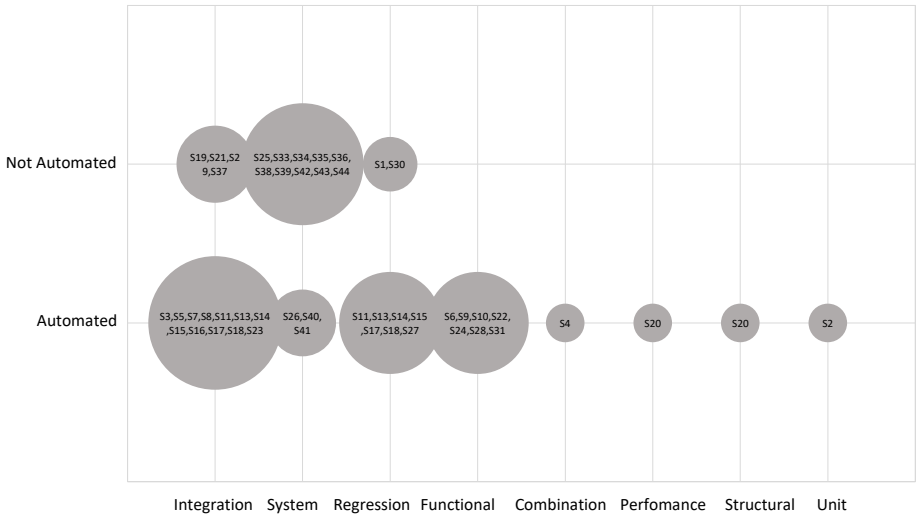

Figure 11: MBT to SPL Automation of Testing Levels

Table 23: Artifacts Used During MBT of SPL

Artifact	**Study ID**	**Count**
State Machine	S1, S6, S7, S10, S11, S12, S13, S14, S15, S17, S21, T22, S23, S27, S29, S30, S34, S38, S44	19
Original Model	S8, S9, S42	03
OVM Variability Model	S8	01
Class Diagram	S6, S12, S14, S31	04
Activity Diagram	S18, S19, S41	03
Use Case Diagram	S20, S28, S41, S43	04
Sequence Diagram	S24, S31, S36	03
Transition Diagram	S7, S15, S25, S26, S33, S35, S37	07
Function Diagram	S2, S3, S4, S5, S9, S16, S39, S40	08
Not Informed	S32	01
TOTAL		**53**

Some of these artifacts play the role of the original (initial) artifact under test, which are the artifacts from which test cases must be generated. Other artifacts are used as formal elements for providing MBT a means to test a given original model.

State Machine. Analyzing the studies that make use of the state machines, we can observe their use is due to the fact of a more formal process compared to other models of artifact. Its use is performed in processes with primary artifacts and even in those converting artifacts to a formal one, then, to a state machine [34, 39, 40, 43, 44, 28, 45, 46, 47, 49, 87, 52, 53, 56, 58, 59, 62, 65, 70].

Original Model. Studies which use the original model to generate test cases seek to maintain its initial properties and characteristics, independent of the

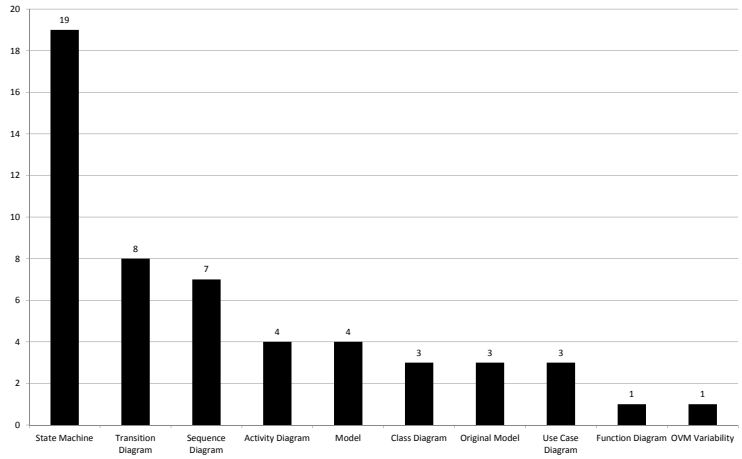

Figure 12: MBT to SPL Automation of Testing Approaches

used artifact. A model can be an initial structure that represents characteristics, realizing a generation of direct test cases as in [41, 42, 69].

OVM Variability Model. Orthogonal variability model (OVM) makes use of a specific model, similar to a function model. In addition, it adopts a modeling language used for the management of variability and creation of multiple views [41].

Class Diagram. The use of class diagrams is more linked to issues of attribute mapping characteristics, in which class relationship issues can be analyzed [39, 28, 46, 21]. Knapp et al. [39] and Wang et al. [46], for instance, convert class diagrams to a second model before generating test cases.

Activity Diagram. With higher abstraction level, the activity diagram is also one of the choices that can be used to generate state machines [50, 24, 68].

Use Case Diagram. Use case is a high level model, commonly related to business rule. Certain works make use of this initial model and later make the conversion to another type of artifact, usually an state machine, as in [27, 57, 68, 27]. As the generation of test cases is based on expected functionalities, we cannot determine the level of coverage compared to a more formal model.

Sequence Diagram. Sequence diagrams are rich in detail, which enables a large amount of iterations and test sequence combinations. Lamancha et al. [19], Reales et al. [21], and Lamancha et al. [20] perform conversions from sequence diagrams to state machines.

Transition Diagram. Transition diagram or state diagram are similar to finite state machines, but with peculiarities that differ from one another. Studies [40, 47, 54, 55, 61, 63, 64] make use of such diagrams to directly generate test cases.

Function Diagram. Works as [35, 36, 37, 38, 42, 48, 66, 67] focus on the expected behavior of function diagrams, to perform conversion to other models, then used for the generation of generic test cases.

Table 24 presents tools used to perform MBT of SPLs related with artifacts. Some of such tools are more frequent as they are used in more than one study. We can mention certain tools with a greater number of citations, such as, the Rational IBM series, MaTeLo Product Line Manager, PURE variants, CADeT Tools, and Eclipse [24, 38, 34, 63, 93].

CADeT Tools [90] assist in the generation of test cases from models, as well as the specification of reusable test cases. Making use of Delta for extracting the regression-focused test cases, this tool, as well as IBM Rational tools, Eclipse [59] and MoSo-PoLiTe were considered for solving the same problem.

Table 24: Comparison of tools and primary or secondary artifacts

Tools	State Machine	Direct from Model	Class Diagram	Statechart	Activity Diagram	Sequence Diagram	Transition Diagram	Functionality Diagram	Use Case
IBM Rational Rhapsody	S1 [34]								
MoSo-PoLiTe	S30 [59]			S7 [40]				S2 [35] - S4 [37]	
MaTeLo Product Line Manager		S8 [41] - S42 [69]						S5 [38]	
SPLOT tool		S9 [42]							
MOFLON - SDM								S16 [48]	
SPLTestbench	S10 [43]								
pure::variants	S12 [28]			S17 [49]					
UML2 Eclipse	S13 [45]								
CVL-Too	S13 [45]								
Rational Software Architect	S12 [28] - S14 [46]								
Rhapsody				S17 [49]					
CADeT Tools					S19 [24]				
TRUST tool	S22 [52]								
Selenium									S28 [57]
AspectOPTIMA								S40 [67]	
Quick Test Pro									S28 [57]
Rational Functional Tester									S28 [57]
Pralíntool			S31 [21]			S31 [21]			
VIBeS	S34 [62]								
Eclipse	S30 [59]					S36 [20]			
UML2 Tools						S36 [20]			
ViTAL							S37 [64]		
visio	S38 [65]								
ATG	S30 [59] - S38 [65]								
Excel	S38 [65]								

Most of the SPL model types are Behavioral-based, which is characterized as rich in details and facilitate the interpretation, even if later converted into a different artifact. In Table 25 it can be observed that several works use the Scenario-based models model, the relation is due to the fact that these works make use of formal artifacts, working with usage scenarios instead of behavior direct.

Table 25: MBT of SPL Types of Models

SPL Model Considered	**Reference**	**Count**
Scenario-based	S3, S11, S19, S20, S24, S31, S38, S40, S41, S42, S43, S44	12
Class-oriented	S6	01
Behavioral-based	S1, S2, S4, S7, S8, S9, S10, S12, S13, S14, S15, S16, S17, S18, S21, S22, S23, S25, S26, S27, S28, S29, S30, S33, S34, S35, S36, S37, S39	29
Not Identified	S5, S32	02
TOTAL		**44**

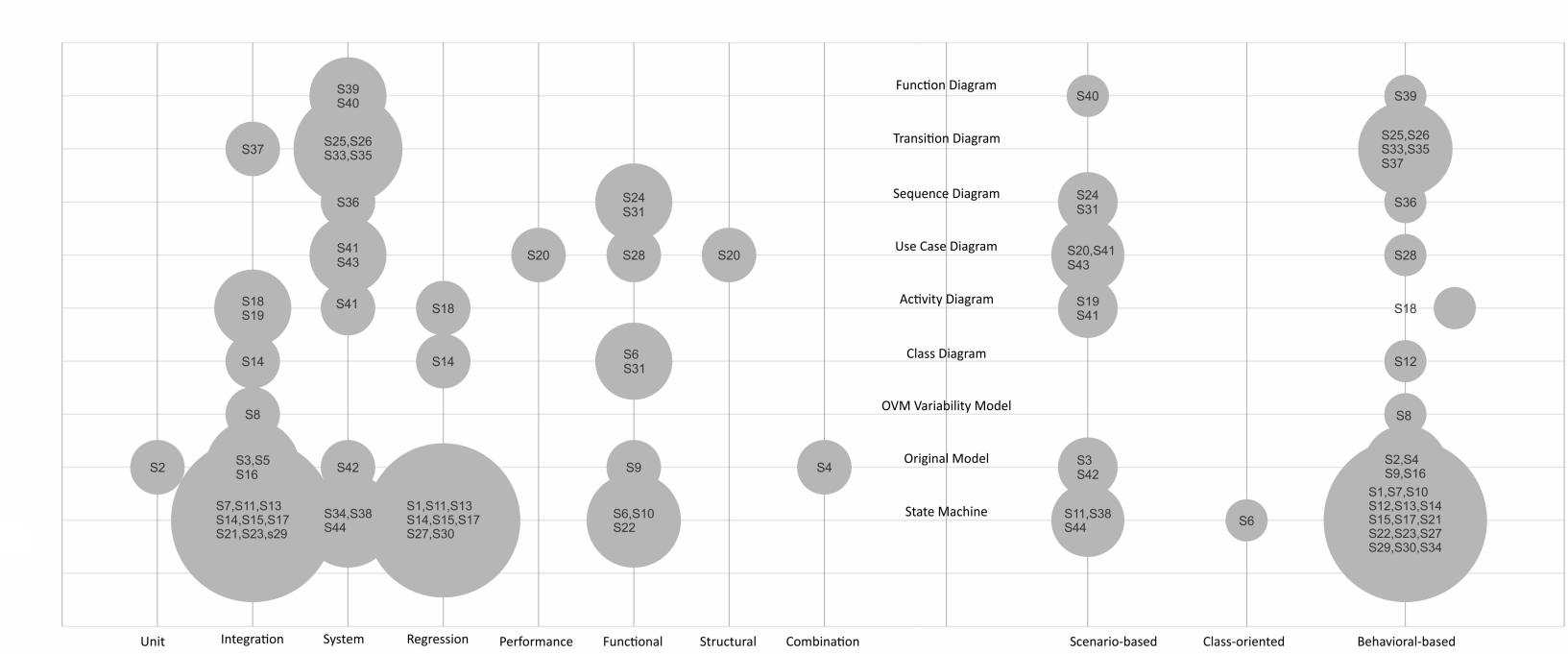

Figure 13: Artifacts X levels x models

4.2.3. RQ.3: How variability and binding time are treated during MBT of SPLs?

As variability is the essence of SPL, as well as it is directly related with binding time, in this section we present results on which studies take into account such concepts and how they treat them.

Table 26 classifies studies which: consider variability (‘‘Yes’’ row) during MBT of SPL and do not consider variability (‘‘No’’ row). In several studies we could not identify whether variabilities are treated.

It is reported in Table 26 studies dealing explicitly with main variability activities, such as, identification and representation, during MBT of SPLs, as well as studies not dealing with it and studies in which we could not identify whether they deal with variability.

Table 26: Studies Treating Variability During MBT Activities

Treat Variability?	Study ID	Count
Yes	S1, S2, S4, S5, S6, S7, S8, S9, S10, S12, S13, S14, S15, S16, S17, S18, S19, S21, S22, S24, S25, S27, S28, S29, S30, S31, S33, S34, S37, S40	30
No	S39, S43	02
Not Identified	S3, S11, S20, S23, S26, S32, S35, S36, S38, S41, S42, S44	12
TOTAL		**44**

Legend:
Yes = explicitly inform how variability is treated.
No = explicitly inform variability is not treated.
Not Identified = we could not identify whether they treat variability.

Only two studies (S39, S43) do not treat variability during MBT activities. This is because they apply tests on SPL specific products at the application engineering.

As we can observe in Table 26, 68.1% (30/44) of studies treat variability during any MBT of SPLs activities.

Works as [34, 41, 42, 43, 46, 47, 48, 87, 54, 57, 58, 59, 21] consider the variability only in domain engineering, in which they are focused directly on the modeling. A significant part of these studies does not make clear the frontier between domain engineering and application engineering. Thus, we took into account all the aspects addressed by each of the works.

Other works [35, 37, 38, 39, 40, 28, 45, 49, 50, 24, 52, 19, 56, 61, 62, 64, 67] consider the variability in both domain and application engineering.

The way variability is treated is often reported directly.

Lity et al. [34] incrementally develop SPL artifacts for each product variant, explicitly considering the variability between two subsequent products being tested. Such approximation guarantees an stable product configuration and allows reduction redundancy derivation on test coverage.

In the work of Steffens et al. [35], they present the results of applying the MoSo-PoLiTe structure in Danfoss Power Electronics to calculate a representative set of product configurations for black-box testing. Within this evaluation they used peer-testing with MoSo-PoLiTe configuration selection component based on a model resource. Such component implements a heuristic to find

minimum subset of configurations covering 100% of resource interaction.

Samih and Bogusch [38] use the relationship approach between variability models, establishing formal correspondences between characteristics, requirements and use case, and external variability. Samih et al. [41] propose formal correspondences between characteristics, requirements and a usage model, in order to formally document what may vary from a usage model.

Gebizli et al. [42] document optional and alternative as feature models and a set of transitions in such model. Then, these transitions are modified according to the selected features, thus generated test cases focus only on such choices.

Wang et al. [28, 46] treat variability by selecting a set of relevant features and configuring products using class diagrams and state machines, generating behavior models.

Devroey [47] uses a variability specification language (TVL) for specifying the test cases. Such work does not provide any further details on it.

Cai et al. [50] makes variability explicitly modeled by variation point and variants limited to the three most common types: optional, coexisting, and alternative.

Olimpiew [24] treats and analyzes variability during the execution of its tool, in which variability is classified by interests based on criteria defined in the initial parameterization.

Weissleder and Hartmut [87] treat variability with one of the techniques that it addresses, because the approach is a combination of other approaches, however it is not clear how it is modeled, only that they are manageable in the generation of test cases.

Lity et al. [56], Varshoaz et al. [58], and Lochau et al. [59] deal with Delta-oriented modeling, in which a common core is created and the variability is treated as variants of this core.

Devroey et al. [62] propose a framework where the mutation is analyzed and present all the existing variables in a single model.

[19, 21, 64] Lamancha et al, uses the OVM (Orthogonal Variability Model) for variability modeling, are modeled using the notation of this approach.

[67] Perrouine et al, by using a range of tools the variabilities are treated in the generation of the test cases where the models originated in submodels of the main models.

Several works [37, 39, 40, 43, 48, 49, 52, 54, 57] do not make clear at what point variability is treated and resolved.

Figure 14 summarizes which studies do or do not treat variability by solution type. We observe how many works treat variability by the following solution types: feature testing, SPL model testing, and PLA testing.

For the solution space of an SPL, testing the PLA (11 works) together with testing SPL models (11 works) concentrate most of works dealing explicitly with variability. On the other hand, in the problem space, testing of features concentrates 10 works dealing with variability.

Analyzing the solution types separated, we observe a fair distribution of works which deal with variability during MBT activities.

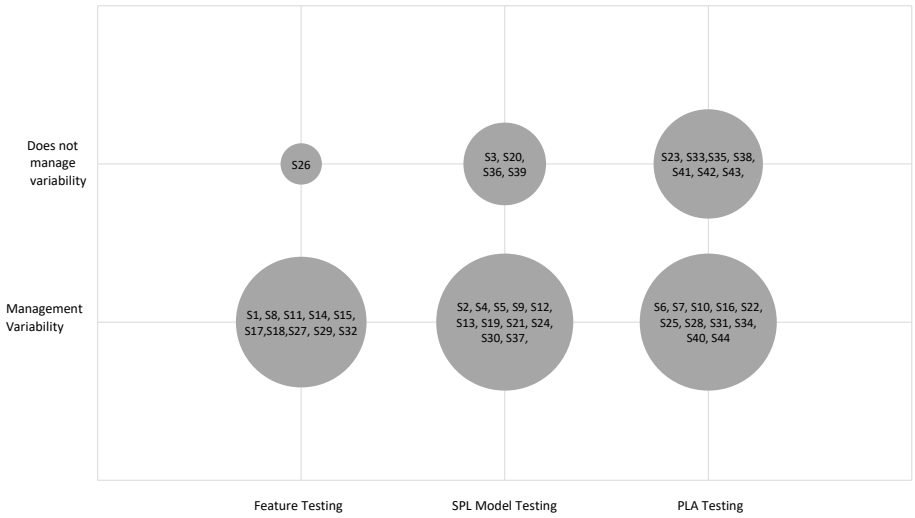

Figure 14: Variability treatment by solution type

Binding time is the ability to resolve variability at any point in the SPL lifecycle. It is mandatory as the significantly increased levels of variability [94].

The binding time in SPL is traditionally at design-time, pre-compile, compile, or linking-time, dealing with static variability. Whereas in systems that deal with dynamic variability it can be at load-time when a system is deployed and loaded into memory, and at runtime after the system has begun executing [95].

Table 27 lists studies from which 86.3% (38/44) deal with binding variability at design-time. Remaining studies do not inform their binding times. Although several papers did not mention explicitly main variability activities, such as identification and representation, as in Table 26, we could identify in several of them explicitly binding time, as shown in Table 27.

Table 27: Variability Binding Time in MBT of SPL

Binding Time	**Study ID**	**Count**
Design Time	S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32, S33, S34, S35, S36, S37, S38, S39, S40, S41, S42, S43, S44	38
Not Identified	S1, S2, S3, S4, S5, S6	06
TOTAL		**44**

Each product in the SPL is derived from a selection of features. For such derivation, features are then bound to a specific product. We can observe in Table 27 several studies bound features at design time, as we expected for MBT of SPL. In addition, few works do not provide such information.

4.2.4. RQ.4: Does MBT support testing of SPL non-functional requirements?

None of the 44 studies selected from this SMS present or mention any type of non-functional SPL requirements test. Although the SPL and tested products are generally at design time, nothing prevents the creation of non-functional requirements from the test plan. Nonfunctional requirements are important in the case-by-case derivation of the domain environment factor that will act.

4.2.5. RQ.5: How are MBT of SPL Proposals Evaluated?

We analyzed MBT of SPL studies proposal evaluation based on the type of evaluation and the environment where evaluations are performed.

Table 28 lists studies according to their evaluation types, which are: `Experiment` and `Case Study`. From the selected studies, 79.5% (35/44) of them performed one of these kinds of evaluation. In nine studies neither evaluation was performed nor they could be identified.

Table 28: Performed MBT to SPL Solution Evidence Method

Evaluation Method	Study ID	Count
Experiment	S1, S3, S4, S16, S20, S21, S23, S24, S25, S28, S39, S40, S42	13
Case Study	S2, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S17, S19, S22, S26, S27, S29, S30, S33, S38, S43, S44	22
Not Informed	S5, S18, S31, S32, S34, S35, S36, S37, S41	09
TOTAL		**44**

Experiments are responsible for 29.5% (13/44) of the evaluations performed. We did not perceive any experimental replication as a means to enforce original results on a certain proposal. In addition, few papers present essential experimental elements, such as, validity of the study, profile of participants, hypotheses formulation and test. Although most industry-related studies do not use human participants, sample size is not reported in most studies. About 25% of the studies [38, 39, 41, 42, 45, 46, 47, 48, 27, 55, 62] make their experimental package or part of it available, including, for instance, diagrams and algorithms, via a public URL. However, most of URLs are no longer available. We believe this occurs because of most evaluations are performed in industry sets (see Table 30).

Case studies are the most performed type of evaluation of MBT to SPL proposals, with 50% (22/44) of the studies and they might have confidential issues.

One important aspect related to proposals evaluation is the environment where such evaluations take place. Therefore, we analyzed which studies are performed in academia and which are performed in the industry. Table 30 lists studies per evaluation environment.

Most of studies take place at the industry, 59% (26/44), which is of great importance due to its realism and participation of actual testing practitioners. In addition, the industry set provides actual data from SPL projects. On the other hand, in studies performed in an academic environment usually recruits

Table 29: Studies that provide extra information

Study ID	Title	URL
S5	MPLM - MaTeLo Product Line Manager [38]	http://people.irisa.fr/Hamza.Samih/mplm and http://www.mbat-artemis.eu
S6	On the use of test cases in model-based software product line development [39]	http://www.cs.swan.ac.uk/~csmarkus/caseStudy.pdf
S8	Deriving Usage Model Variants for Model- based Testing: An Industrial Case Study [41]	http://www.mbat-artemis.eu and http://www.sse.uni-essen.de/varmod
S9	Model-based Software Product Line Testing by Coupling Feature Models with Hierarchical Markov Chain Usage Models [42]	http://www.vestel.com.tr and http://www.splot-research.org/
S13	An automated Model-based Testing Approach in Software Product Lines Using a Variability Language [45]	http://www.omgwiki.org/variability/doku.php and http://itea-mosis.org/
S14	Automated Product Line Methodologies to Support Model-Based Testing [46]	http://www.pure-systems.com/Home.142.0.html and http://jmetal.sourceforge.net/
S15	Behavioural Model Based Testing of Software Product Lines [47]	http://directory.unamur.be/staff/xdevroey/
S16	Feature Model-based Software Product Line Testing [48]	http://www.es.tu-darmstadt.de/mitarbeiter/sebastian-oster
S20	PLETS - A Product Line of Model- Based Testing Tools [27]	http://www.cepes.pucrs.br/plets/
S26	Abstract Test Case Generation for Behavioural Testing of Software Product Lines [55]	http://webcampus.unamur.be
S34	Poster: VIBeS, Transition System Mutation Made Easy [62]	https://projects.info.unamur.be/vibes/mutation.html

Table 30: MBT to SPL Solution Evidence Environments

Environment	Study ID	Count
Industry	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S20, S22, S29, S30, S38, S41, S42, S43, S44	26
Academia	S19, S23, S24, S25, S26, S27, S28, S31, S32, S33, S34, S35, S36, S37, S39, S40	16
Not Informed	S18, S21	02
TOTAL		**44**

students as participants and use pedagogical or non-commercial SPLs. Note the use of students in evidence-based software engineering is not a threat as widely discussed [96].

Figure 15 depicts a bubble plot of the types of evaluation and their environments. As we can observe in Figure 15, most studies, nine experiments and 15 case studies, were evaluated in the industry set, which contributes to increase the reliability of the provided proposals evidence.

4.2.6. RQ.6: How is traceability considered during MBT activities for SPLs?

Traceability is still one of widely research gaps on software development, especially for SPLs. As traceability is in charge of identifying important elements for deriving SPL specific products and for the SPL evolution, it is straightforward necessary during MBT activities [4, 93]. In addition, traceability provides essential support for quality assurance of SPLs.

In this section we analyzed which studies take into consideration any kind of traceability during MBT activities. Thus, Table 31 lists studies and whether they provide any support for traceability.

Few studies (12/44) report or mention traceability supporting. In addition, none of them state how they use or treat traceability. They only claim traceability is fundamental to the generation of test cases, which should be as automated as possible.

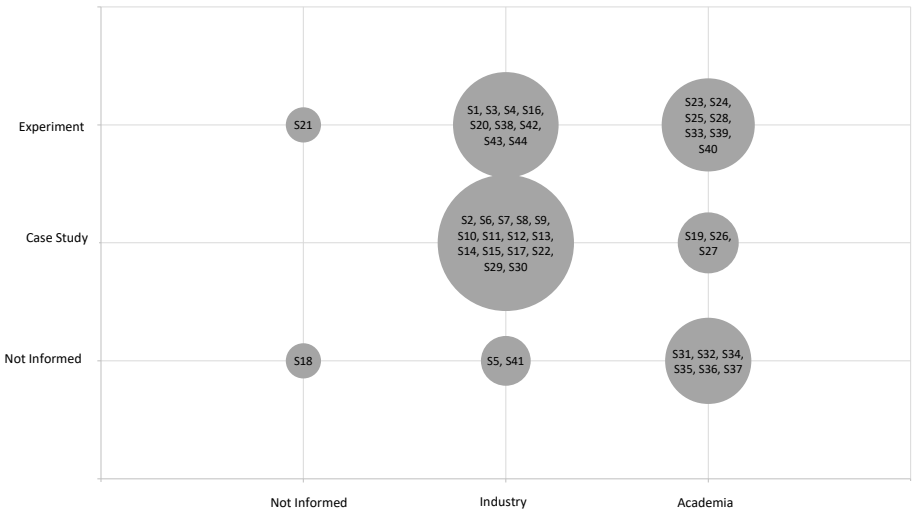

Figure 15: Proposals Evaluation Types and Environments

Table 31: Studies with Traceability Support for MBT of SPLs

Traceability Supporting?	**Study ID**	**Count**
Yes	S5, S7, S15, S16, S17, S19, S20, S24, S31, S36, S37, S41	12
No	S1, S2, S3, S4, S6, S8, S9, S10, S11, S12, S13, S14, S18, S21, S22, S23, S25, S26, S27, S28, S29, S30, S32, S33, S34, S35, S38, S39, S40, S42, S43, S44	32
TOTAL		**44**

4.3. SMS Data Sharing Procedures

We perform procedures to promote openness and reproducibility of this study. Therefore, we:

- provide BIBTEX with references of the 44 studies from the Final List to be used in different bibliographic tools;
- store data set in a public and permanent open data sharing trusted repository, named Zenodo[3], under DOI `https://doi.org/10.5281/zenodo.3712084`.

5. Discussion of Results

As we observed in Section 4, most of the analyzed studies are published from 2011 to 2014. In addition, we understand an increasing number of studies over the years and a fair distribution of them per year.

[3] `zenodo.org`

Majority of studies has high quality according to our defined criteria, thus significantly contributing to answer our research questions.

Each study contributed to answering at least two research questions, thus covering all research questions. Excepting RQ.4, which was answered by only S12.

5.1. Domains, Solution Types and Proposals

The variety of domains which MBT is applied provides us an insight of how the MBT approach to SPL is usable and is not restricted to the software domain.

As we already expected and according to Silveira Neto et al. [16], MBT is widely applied to the SPL software domain.

Nevertheless, other domains are growing with respect to the application of MBT to testing SPLs, such as: automotive and electronic.

With regard to MBT to SPL solution types, we highlight three of them: feature testing, SPL model testing, and the SPL architecture (PLA) testing.

PLA testing is more performed as it is one of the most important SPL core assets. It is corroborated by the increasing number of PLA quality evaluation research over the years at ICSE, ECSA, and WICSA conferences and specialized journals.

SPL model testing is also widely performed as it focuses on the solution space by testing formal models, such as, UML, CVL, OVM, and variability resolution models.

On the other hand, in the problem space, testing of features is also performed. This is an essential solution type as features are crosscutting and directly related to models at the solution space at any level (SPL models and PLA models).

We also analyzed main MBT to SPL proposals. Approaches of MBT to SPL are most proposed as a way to systematize the testing process of an SPL, especially for the software and automotive domains.

Algorithms and tools are the second most published proposals. Algorithms are proposed for the software and automotive domains only, whereas tools encompass software, automotive, and electronic domains. The remaining proposals focus on tool kits, strategies, methodology, and tutorials for MBT of SPLs.

5.2. Approaches, Testing Levels and Artifacts

Black-box and white-box are use approaches to testing SPLs, with the majority of studies in the context of black-box testing. Such studies aim at generating test cases from SPL models at a high level of abstraction. On the other hand, white-box testing seeks to generate testing scripts for the automation of the MBT process.

With regard to MBT levels, we observed integration test, system test, and regression test are the most performed to SPL.

As expected, integration test plays a central role in MBT of SPLs as SPLs are aimed at composing different specific products, thus testing the integration of components from the core assets is essential, especially for its PLA. System

testing is also important as SPL composes whole products and such products consistency must be tested before their launching. Regression testing of SPLs and products must be performed as new requirements and features evolve an SPL, thus impacting the derived specific products.

The good news in the analyzed studies is most of them provide fully-automation of the testing process. Such automation contributes to speed up the testing process, as well as to improve quality throughout consistency of SPL and its derived products. Majority of these studies provide automation for black-box testing approaches. In addition, a significant subset of the analyzed studies provides semi-automated tools and its majority is also for black-box testing approaches. Automation of the testing process mainly focuses in the integration, regression, and functional levels.

For MBT of SPL several different artifacts are used. As expected, state machines are most used to aiding the testing of SPLs. We understand this artifact is the most formal one among all found in this SMS, thus it directly contributes to the quality and consistency of generated test cases.

Transition diagrams, sequence diagrams, and activity diagrams are also outstanding artifacts used to MBT of SPLs. Transition diagrams are similar to finite state machines, thus it has as a main characteristic a certain level of formalization. Thus, such characteristic corroborates its wide usage. We believe sequence diagrams are more used than activity diagrams as the former allows modeling at a low level abstraction, whereas the latter it is more generally abstract.

Remaining artifacts found are: general models, original models, class diagrams, use case diagrams, function diagrams, and OVM variability models.

Most of artifacts used for testing SPLs are behavioral-based models, followed by scenario-based and class-oriented models. This is expected as behavioral and scenario-based models express states of a system, thus directly facilitating the generation of more accurate test cases as compared to class-oriented models, which are more structural.

5.3. Variability Treatment and Binding Time

We observed most studies take into account variability in models to be tested. These studies are straightforward focused on taking variability from domain engineering to application engineering testing of products.

To allow such treatment of variabilities, these studies aim at testing products based on features, SPL models (UML, CVL, etc), and PLAs. We understand such variability is well treated as these are the most used for models for representing variations in an SPL. Therefore, variability is carried out to and resolved to testing specific products, as testing all potential products of an SPL is unfeasible.

In addition, as expected, majority of studies focuses on representing and resolving variabilities at design-time. We strongly believe this is due to the lack of studies on MBT of dynamic SPLs (DSPL) in this mapping study, in which variability is usually resolved at runtime.

5.4. Testing of Non-Functional Requirements

Another factor that has not been addressed and can be observed in this mapping is the lack of non-functional testing of SPLs. Non-functional requirements are essential to aid identifying and specifying variability and SPL constraints, which differs one product from another. In addition, such kind of test influences on the performance and quality of a derived product.

Another aspect that impacts the testing of non-functional requirements is the domain. For instance, electronics and embedded systems require depth testing on their constraints. Most of software from the embedded systems domain is critical, thus demanding further testing.

5.5. Evaluation of Proposals

MBT for SPL proposals are mainly evaluated based on experiments and case studies, with majority of them as case studies. A few studies do not evaluate their proposals, thus it is a huge threat to their adoption and evolution.

Industrial sets play an essential role as they take place in most of case studies and experiments evaluation types. Academia also takes part in such evaluations, but with less frequency than industry, which is an outstanding finding.

Only two studies were performed in the industry, but they do not reveal the type of evaluation performed. The same for academia with six studies.

Overall, authors have been evaluated their proposals in an industry set, which is straightforward recommended.

5.6. Traceability

Traceability usually refers to the capacity of linking product requirements to corresponding design artifacts, such as test cases [97].

Unfortunately, in this SMS, the majority of studies do not present or discuss any traceability resources to provide such capacity from requirements to MBT test artifacts, such as, test sequences or test cases.

This fact jeopardizes MBT of SPL in many aspects, such as: decrease of reusing test cases from the variability perspective to specific products testing; make it difficult to trace back defects from MBT artifacts to respective used models (use case diagrams, activity diagrams, etc.); and provide no round-trip capability and models evolution when testing new features of an SPL.

6. Validity Evaluation

This section discusses main threats to validity according to Ampatzoglou et al. [98].

6.1. Study Selection Validity

We **built our search string** taking into consideration our research group's expertise on SMS, as well as on pilot searches at IEEE and ACM digital libraries. We **chose these libraries and publication venues** as: they are worldwide sources on computer science and software engineering; they have boolean expression-based search mechanism able to handle advanced queries; they are internationally indexed sources; and they index mostly journals with high impact factor (CiteScore/JCR).

We **built specific search strings for the chosen digital libraries** based on the general one (Table 2). Thus, due to **inefficiencies of search engines**, relevant papers might not get included in this SMS.

To avoid a **reduced number of journals and conferences** we performed a broad search on electronic and manual sources.

We defined as one of the inclusion criteria only **papers written in English** to promote reproducibility and potential updating of our study by external groups. Therefore, missing non-English papers may be a threat as there must exist important papers on this matter in other languages.

To **handling duplicate studies**, we adopted the following strategy: (i) load all bibtex of the retrieved studies into the StArt[4] tool, thus it automatically identifies duplicates; and (ii) check whether any papers were missed by StArt by alphabetically sorting them, then manual verifying each one.

At first, in the initial selection phase, we decided to include every grey literature study. After filtering studies and coming up with 44 to fully reading, we decided to **include/exclude grey literature** based on analyzed quality of each study. Although this is an SMS, we understand it may mitigate the threat of less important grey literature being included in the analysis of this study.

To decide **which study to include or exclude**, we defined inclusion/exclusion criteria. To define such criteria, we took into account our pilot searches, as well as the experience of our group.

6.2. Data Validity

One candidate threat to data validity in this SMS may be the **sample size** (N=44). However, taking into consideration Table 16, we believe our sample is enough to draw initial conclusions based on the reported results.

To mitigate potential threats on **choosing variables to be extracted**, we analyzed the defined goal and research questions, as well as considered performed pilot searches.

To reduce **publication bias**, we included grey literature as this is an SMS, as well as manually analyzed additional venues and performed snowballing.

To **validate the returned primary studies**, we selected only quality venues, as discussed in source definition, and we defined criteria to assess the quality of each study in Section 3.4.2. We believe this approach mitigates the potential threat of invalid studies.

[4] `http://lapes.dc.ufscar.br/tools/start_tool`

We performed a pilot data extraction based on expert inputs to train data extraction and classification schema to mitigate **data extraction bias**. However, a potential threat may be **misclassification of a given study** as we did not perform random study screening.

Although we did not **use basic statistical analysis** as our data is mainly qualitative, we believe our result analysis, interpretation and discussion suffer little from lack of further statistical verification.

We did not **use an existing classification schema**. In fact, we used the main concepts of MBT and SPL to provide a classification/grouping schema for our extracted data. We, thus, understand this schema may be reused in prospective related works.

6.3. Research Validity

We believe the **repeatability threat** was mitigated by following a standard protocol by Petersen et al. [31] and Kitchenham et al. [32]. The second action taken also contributed to mitigate a potential threat of **review process deviation** and **chosen research method**. We also had various discussions during our SMS.

Lack of comparable studies is not a threat to our study, as we further discussed related work in Section 2.2.

We believe **unfamiliarly with research field** was not a threat for us, as we have already developed research on this matter.

We tried to guarantee **generalizability** of the study by using a broad search and a well-structured search string, as well as comparing existing primary studies on this subject. We believe we reached a general view based on the results obtained in this study.

7. MBT4SPL: a Roadmap for Model-Based Testing Research Applied to SPL

In this section we present a roadmap[5], named MBT4SPL, as a result of mapping the Final List of studies with relation to the MBT process for SPLs.

To do so, we used results from Section 4 to build MBT4SPL. Such roadmap is composed of three main general stages as the main strategy to classify the retrieved studies as in Section 3.4.1: **Early Stage** based on the original artifacts; a **Second Stage (optional)**, in which original high-level abstract artifacts (i.e. UML diagrams) are transformed into a more formal (lower-level abstract, such as a finite state machine) artifact by using or not a supporting tool and provide variability; and a **Final Stage** with automated/semi-automated or manual support generate test cases providing or not traceability.

Figure 16 depicts MBT4SPL on how the 44 analyzed studies of the SMS are organized according to the stages and activities of MBT4SPL.

[5]Images available at `https://doi.org/10.5281/zenodo.3712084`

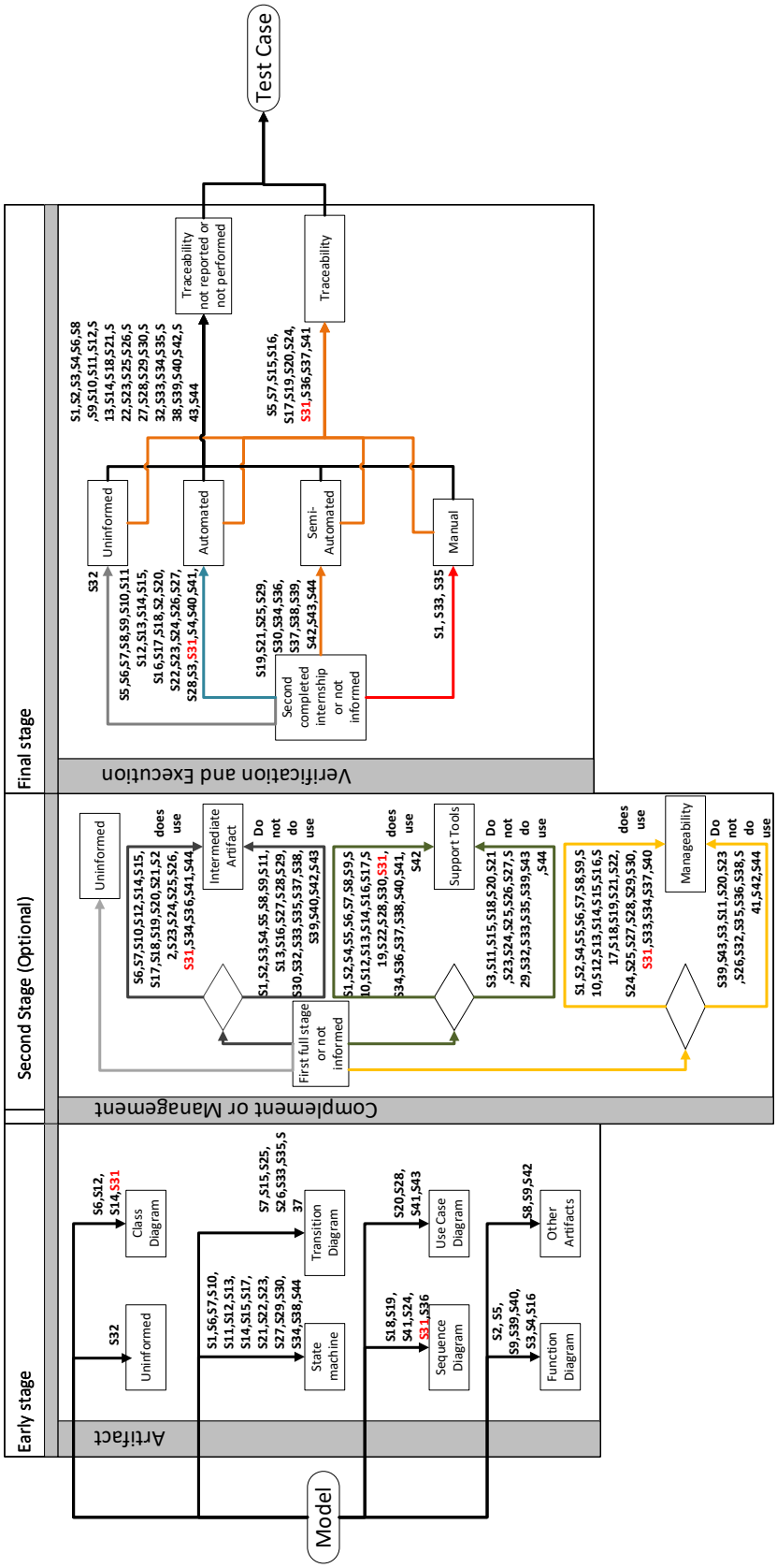

Figure 16: MBT4SPL: a Roadmap for MBT of SPLs

Such roadmap allows researchers and practitioners to identify: (i) the path taken by a given study (Primary Study-based Route); and (ii) given a specific path, which studies taken it (Path-based Route).

The following sections present application examples on how to use MBT4SPL based on the two kinds of routes.

7.1. Primary Study-based Route

Let's say that when selecting a study from Table 13 one might want to check which path such study traverses by identifying its stages and activities. Thus, we initially analyze the **Early Stage** in Figure 16.

One might take, for example, study S31. From a class diagram, S31 is transformed into a sequence diagram. With the **Early Stage** completed, the **Second Stage** is then started to S31. S31 takes into account variability management (Manageability), tool support, and is transformed into a more formal (intermediate) artifact, for instance, an FSM.

In the **Final Stage**, S31 makes use of an automated process and provides any kind of traceability.

We, then, demonstrated which path S31 took and the details of the MBT of SPL process overview for such study.

7.2. Path-based Route

The second kind of route provides a way to identify which studies took a same given path.

One might, for example, be interested on which studies are candidate ones to his/her research on MBT of SPL. To do so, one's MBT of SPL hypothetically process might be as described next: a process which starts with a sequence diagram, then it is converted to a finite state machine by using supporting tools, also managing variabilities, using a semi-automated tool to generate test cases with no support to traceability.

By checking Figure 16, we observe the following related studies to such given path:

- start with a sequence diagram in **Early Stage**: S18, S19, S24, S31, S36, S41;
- convert a sequence diagram to a finite state machine in **Second Stage**: S6, S7, S10, S12, S14, S15, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S31, S34, S36, S41, S44;
- use supporting tools in **Second Stage**: S1, S2, S4, S5, S6, S7, S8, S9, S10, S12, S13, S14, S16, S17, S19, S22, S28, S30, S31, S34, S36, S37, S38, S40, S41, S42;
- manage variability in **Second Stage**: S1, S2, S4, S5, S6, S7, S8, S9, S10, S12, S13, S14, S15, S16, S17, S18, S19, S21, S22, S24, S25, S27, S28, S29, S30, S31, S33, S34, S37, S40;

- use semi-automated support to generate test cases in **Final Stage**: S19, S21, S25, S29, S30, S34, S36, S37, S38, S39, S42, S43, S44.

By taking the intersection of such sets of studies we came up with the following directly possible related study to the given path: **S19**. Thus, S19 might be further analyzed and compared to the potential research or practice one is performing.

S6, for instance, does not fit the given path as it does not start with a sequence diagram. S18 does not use any supporting tool to convert the sequence diagram into an intermediate artifact. S36 does not manage variability.

Another example might an MBT of SPL process which starts with use case diagrams, does not convert to an intermediate artifact, use a supporting tool, manage variability, use a fully automated tool, and does not provide traceability:

- start with a use case diagram in **Early Stage**: S20, S28, S41, S43;
- do not convert an use case diagram to an intermediate artifact in **Second Stage**: S1, S2, S3, S4, S5, S8, S9, S11, S13, S16, S27, S28, S29, S30, S32, S33, S35, S37, S38, S39, S40, S42, S43;
- use supporting tools in **Second Stage**: S1, S2, S4, S5, S6, S7, S8, S9, S10, S12, S13, S14, S16, S17, S19, S22, S28, S30, S31, S34, S36, S37, S38, S40, S41, S42;
- manage variability in **Second Stage**: S1, S2, S4, S5, S6, S7, S8, S9, S10, S12, S13, S14, S15, S16, S17, S18, S19, S21, S22, S24, S25, S27, S28, S29, S30, S31, S33, S34, S37, S40;
- use automated support to generate test cases in **Final Stage**: S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S20, S22, S23, S24, S26, S27, S28, S31, S40, S41.

S28 is the exactly match for the given MBT of SPL process. Therefore, one might take a further look at S28 to analyze how the process occurs, then decide to adopt it or even use it as a basis for its own new proposal.

In fact, any activity of each stage of the roadmap might be combined with the logical operators "**OR**" and "**AND**". For instance, one might analyze the roadmap searching for studies which: convert function diagrams to FSMs **AND** do not provide any kind of tool assistance. Another example might be studies which: manage variability **OR** adopt any kind of tool support to convert an initial artifact to an intermediate one.

8. MBT of SPL Agenda

In this section, we discuss a research and practice agenda on MBT of SPL. Therefore, we list the main points that should be taken into consideration, such as, open issues and remaining challenges for future research.

Throughout this paper, we showed how models have been used during the main phases of MBT of SPLs. The main reported issue, in the examined literature, is still how to demonstrate the particularities of each used model. For example, several studies use sequence diagrams as an initial diagram for generating test cases. However, they do not provide, for instance, how such model source (XML, XMI, etc) is generated, which tools are used, which UML version is adopted, and the state of OCL constraints. We know these details are vital for model-driven studies. Therefore, we understand that providing these models particularities is still an open issue in MBT of SPL corroborating the claims from other researchers [8, 14, 13], especially for the automation of the MBT of SPL testing process.

Automation of MBT of SPL has grown in the last years as it is shown in the analysed 44 studies, only four of them do not provide any automation support. This corroborates the claims of Bernardino et al. [4] evidencing the concern of the community on automation of the MBT of SPL testing process. However, the automation process itself is poorly discussed on the analyzed studies, especially those providing full automation towards making the automation process externally reproduced due to several issues: lack of good documentation, insufficient environment (platform and software) details, and unavailable or discontinued old software versions. Automation process setup, thus, is still an important remaining challenge.

Another open issue that MBT of SPL still faces is how to consider variability during the SPL testing phases. Although we found many studies that deal with variability during the SPL testing process, most of them do not [12, 13, 18, 27, 17]: (i) provide relevant and interchangeable information on how variability is considered in the SPL Domain Engineering activity, except those handling feature models as such models have well-defined systematic notations; (ii) characterize open and close variabilities as a way to include variants to specific variation points; (iii) provide any information on the need of an auxiliary variability resolution model; (iv) inform whether variability is reused in the SPL Application Domain activity for testing SPL specific products; (v) show how variability is resolved during specific products testing; and (vi) discuss how variability deals with changing SPL requirements and how such changes impact the reuse of pre-generated test cases, which is an important issue on SPL testing.

An important live challenge on MBT of SPL is how to test software on different domains. A large proportion of the studies analysed in this paper (45%) focus on MBT of SPL for specific domain, such as, electronic, telecommunication, and aerospace. These domains require several different certifications compliance, thus making it difficult to publish generic testing models. We, then, understand this is an open issue that the MBT of SPL community might explore to provide new insights.

Traceability on MBT of SPL is still a premiere and relevant open issue. Only 27% of studies superficially inform some kind of traceability concerns during SPL testing. However, none of them present and discuss in details how models and respective variabilities are dealt with. For instance, during changes in SPL requirements and, consequently, tracing impacted models and pre-generated test

cases, or in the case of an extractive or reactive SPL approach development.

9. Concluding Remarks

This systematic mapping was performed to gather together an overview on Model-Based Testing of Software Product Lines. We analyzed 44 studies with regard to six research questions and answered each of these questions based on a set of pre-defined data extracted and schema. Then, we discussed the findings of this SMS mainly based on what the studies provide and what is missing for a complete MBT process at SPL level.

From the results, we built up a roadmap for researchers and practitioners to analyze the MBT process for SPL based on two main routes: primary study-based and path-based. Therefore, one might consult such map to understand the path a given study took or to identify which studies took a given path. We compose such path in three main phases: early phase, in which an initial model is considered, a second (optional) phase in which the original model is converted to a more formal model, and the final phase in which test cases are generated from such models. We understand that the roadmap is an important contribution as it traces the path throughout performed studies.

We also understand there are big challenges when addressing MBT to SPL, especially with regard to variability management, process automation, and traceability. Variability management is the most present considered issue in the analyzed studies, however how it occurs is not explicit in such studies. They lack which approaches are used to manage variabilities, as well as how variability is resolved in a test sequence to test a specific product. The automation process is the most cited challenge, but poorly described during all testing process phases. Traceability seems a chaos at testing SPL. It is neither cited or described. In addition, although there are dozens of traceability implementation techniques, even for SPL, they are poorly exploited, especially for SPLs, which tend to evolve with the incorporation of new features over time.

As we could observe, MBT of SPL has growing in the last years as an important factor to the quality of the SPL and its specific products. Thus, we strongly encourage the community to keep researching on this matter, especially in the identified gaps we reported in this SMS.

References

[1] V. Garousi, M. V. Mäntylä, A systematic literature review of literature reviews in software testing, Information and Software Technology 80 (2016) 195 – 216. doi:https://doi.org/10.1016/j.infsof.2016.09.002.

[2] F. Ciccozzi, I. Malavolta, B. Selic, Execution of UML models: a systematic review of research and practice, Software & Systems Modelingdoi:10.1007/s10270-018-0675-4.

[3] T. Kosar, S. Bohra, M. Mernik, Domain-specific languages, Inf. Softw. Technol. 71 (C) (2016) 77–91. doi:10.1016/j.infsof.2015.11.001.

[4] M. Bernardino, E. M. Rodrigues, A. F. Zorzo, L. Marchezan, Systematic mapping study on MBT: tools and models, IET Software 11 (4) (2017) 141–155.

[5] H. G. Gurbuz, B. Tekinerdogan, Model-based testing for software safety: a systematic mapping study, Software Quality Journal 26 (4) (2018) 1327–1372. doi:10.1007/s11219-017-9386-2.

[6] N. M. Minhas, S. Masood, K. Petersen, A. Nadeem, A systematic mapping of test case generation techniques using UML interaction diagram, Journal of Software: Evolution and Process (2018) 2047–7481.

[7] R. van Megen, D. B. Meyerhoff, Costs and benefits of early defect detection: experiences from developing client server and host applications, Software Quality Journal 4 (4) (1995) 247–256. doi:10.1007/BF00402646.

[8] I. Carmo Machado, J. D. Mcgregor, Y. C. Cavalcanti, E. S. De Almeida, On strategies for testing software product lines: A systematic literature review, Information and Software Technology 56 (10) (2014) 1183–1199.

[9] K. Pohl, G. Böckle, F. J. van der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, Springer, Berlin, 2005. doi:10.1007/3-540-28901-1.

[10] F. J. v. d. Linden, K. Schmid, E. Rommes, Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering, Springer-Verlag, Berlin, Heidelberg, 2007.

[11] P. C. Clements, L. Northrop, Software Product Lines: Practices and Patterns, SEI Series in Software Engineering, Addison-Wesley, 2001.

[12] M. Raatikainen, J. Tiihonen, T. Männistö, Software product lines and variability modeling: A tertiary study, Journal of Systems and Software 149 (2019) 485 – 510. doi:https://doi.org/10.1016/j.jss.2018.12.027.

[13] E. Engström, P. Runeson, Software product line testing: a systematic mapping study, Information and Software Technology 53 (1) (2011) 2–13.

[14] B. P. Lamancha, M. Polo, M. Piattini, Systematic review on software product line testing, in: International Conference on Software and Data Technologies, Springer, 2013, pp. 58–71.

[15] E. Engstrom, P. Runeson, I. do Carmo Machado, E. Almeida, S. de Lemos Meira, Testing software product lines, IEEE Software 28 (5) (2011) 16–20. doi:10.1109/MS.2011.90.

[16] P. A. d. M. Silveira Neto, I. C. Machado, J. D. McGregor, E. S. Almeida, S. R. L. Meira, A systematic mapping study of software product lines testing, Information and Software Technology 53 (5) (2011) 407–423.

[17] P. A. M. Silveira Neto, I. d. C. M. Per Runeson, E. S. Almeida, S. R. de Lemos Meira, E. Engström, Testing software product lines, IEEE Software 28 (5) (2011) 16–20. doi:10.1109/MS.2011.90.

[18] R. Capilla, J. Bosch, K.-C. Kang, Systems and Software Variability Management: Concepts, Tools and Experiences, Springer Publishing Company, Incorporated, 2013.

[19] B. P. Lamancha, M. P. Usaola, M. P. Velthius, A model based testing approach for model-driven development and software product lines, in: International Conference on Evaluation of Novel Approaches to Software Engineering, Springer, 2010, pp. 193–208.

[20] B. P. Lamancha, P. R. Mateo, I. R. de Guzmán, M. P. Usaola, M. P. Velthius, Automated model-based testing using the uml testing profile and qvt, in: International Workshop on Model-Driven Engineering, Verification and Validation, ACM, 2009, p. 6.

[21] P. Reales, M. Polo, D. Caivano, Model based testing in software product lines, in: International Conference on Enterprise Information Systems, Springer, 2011, pp. 270–283.

[22] J. Lee, S. Kang, D. Lee, A survey on software product line testing, in: International Software Product Line Conference, SPLC '12, ACM, New York, NY, USA, 2012, pp. 31–40. doi:10.1145/2362536.2362545.

[23] S. A. Razak, M. A. Isa, D. N. A. Jawawi, O. L. Fuh, Model-based testing for software product line: A systematic literature review, International Journal of Software Engineering and Technology 2 (2) (2017) 27–34.

[24] E. M. Olimpiew, Model-based testing for software product lines, Ph.D. thesis, George Mason University, Fairfax, VA, USA, aAI3310145 (2008).

[25] M. Lochau, S. Peldszus, M. Kowal, I. Schaefer, Model-based testing, in: International School on Formal Methods for the Design of Computer, Communication and Software Systems, Springer, 2014, pp. 310–342.

[26] K. Pohl, A. Metzger, Software product line testing, Commun. ACM 49 (12) (2006) 78–81. doi:10.1145/1183236.1183271.

[27] E. M. Rodrigues, A. F. Zorzo, E. Y. Nakagawa, I. M. S. Gimenes, E. Nakagawa, F. M. Oliveira, J. C. Maldonado, A software product line for model-based testing tools, Tech. rep., PUCRS (2012).

[28] S. Wang, S. Ali, T. Yue, M. Liaaen, Using feature model to support model-based testing of product lines: an industrial case study, in: International Conference on Quality Software, IEEE, 2013, pp. 75–84.

[29] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report (2007).
URL http://www.dur.ac.uk/ebse/resources/Systematic-reviews-5-8.pdf

[30] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 28 (2) (2002) 721–734.

[31] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Information and Software Technology 64 (2015) 1 – 18. doi:https://doi.org/10.1016/j.infsof.2015.03.007.

[32] B. A. Kitchenham, D. Budgen, P. Brereton, Evidence-Based Software Engineering and Systematic Reviews, Chapman & Hall/CRC, 2015.

[33] Q. Li, A novel likert scale based on fuzzy sets theory, Expert Systems with Applications 40 (5) (2013) 1609–1618.

[34] S. Lity, M. Lochau, I. Schaefer, U. Goltz, Delta-oriented model-based SPL regression testing, in: International Workshop on Product LinE Approaches in Software Engineering, IEEE Press, 2012, pp. 53–56.

[35] M. Steffens, S. Oster, M. Lochau, T. Fogdal, Industrial evaluation of pairwise SPL testing with moso-polite, in: Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems, ACM, 2012, pp. 55–62.

[36] H. Cichos, S. Oster, M. Lochau, A. Schürr, Model-based coverage-driven test suite generation for software product lines, in: International Conference on Model Driven Engineering Languages and Systems, Springer, 2011, pp. 425–439.

[37] S. Oster, I. Zorcic, F. Markert, M. Lochau, MoSo-PoLiTe: tool support for pairwise and model-based software product line testing, in: Workshop on Variability Modeling of Software-Intensive Systems, ACM, 2011, pp. 79–82.

[38] H. Samih, R. Bogusch, MPLM-MaTeLo product line manager: relating variability modelling and model-based testing, in: International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools-Volume 2, ACM, 2014, pp. 138–142.

[39] A. Knapp, M. Roggenbach, B.-H. Schlingloff, On the use of test cases in model-based software product line development, in: Proceedings of the 18th International Software Product Line Conference-Volume 1, ACM, 2014, pp. 247–251.

[40] S. Oster, M. Zink, M. Lochau, M. Grechanik, Pairwise feature-interaction testing for spls: potentials and limitations, in: Proceedings of the 15th International Software Product Line Conference, Volume 2, ACM, 2011, p. 6.

[41] H. Samih, H. Le Guen, R. Bogusch, M. Acher, B. Baudry, Deriving usage model variants for model-based testing: an industrial case study, in: Engineering of Complex Computer Systems (ICECCS), 2014 19th International Conference on, IEEE, 2014, pp. 77–80.

[42] C. S. Gebizli, H. Sözer, Model-based software product line testing by coupling feature models with hierarchical markov chain usage models, in: Software Quality, Reliability and Security Companion (QRS-C), 2016 IEEE International Conference on, IEEE, 2016, pp. 278–283.

[43] H. Lackner, M. Thomas, F. Wartenberg, S. Weißleder, Model-based test design of product lines: Raising test design to the product line level, in: Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference on, IEEE, 2014, pp. 51–60.

[44] M. Dukaczewski, I. Schaefer, R. Lachmann, M. Lochau, Requirements-based delta-oriented spl testing, in: Product Line Approaches in Software Engineering (PLEASE), 2013 4th International Workshop on, IEEE, 2013, pp. 49–52.

[45] B. García Gutiérrez, R. Garcia Carmona, A. Navas Baltasar, H. A. Parada Gélvez, F. Cuadrado Latasa, J. C. Dueñas López, An automated model-based testing approach in software product lines using a variability language, in: Workshop on Model-Driven Tool and Process Integration, 2010, pp. 1–10.

[46] S. Wang, S. Ali, A. Gotlieb, Automated product line methodologies to support model-based testing, in: Demos/Posters/StudentResearch@ MoDELS, 2013, pp. 56–60.

[47] X. Devroey, Behavioural model-based testing of software product lines, Ph.D. thesis, Université de Namur (2017).

[48] S. Oster, Feature model-based software product line testing, Ph.D. thesis, Technische Universität (2012).

[49] M. Lochau, S. Oster, U. Goltz, A. Schürr, Model-based pairwise testing for feature interaction coverage in software product line engineering, Software Quality Journal 20 (3-4) (2012) 567–604.

[50] X. Cai, H. Zeng, Model-based test generation for software product line, in: Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on, IEEE, 2013, pp. 347–351.

[51] S. Weißleder, H. Lackner, Top-down and bottom-up approach for model-based testing of product lines, arXiv preprint arXiv:1303.1011.

[52] S. Ali, T. Yue, L. Briand, S. Walawege, A product line modeling and configuration methodology to support model-based testing: an industrial case study, in: International Conference on Model Driven Engineering Languages and Systems, Springer, 2012, pp. 726–742.

[53] X. Devroey, G. Perrouin, A. Legay, M. Cordy, P.-Y. Schobbens, P. Heymans, Coverage criteria for behavioural testing of software product lines, in: International Symposium On Leveraging Applications of Formal Methods, Verification and Validation, Springer, 2014, pp. 336–350.

[54] X. Devroey, M. Cordy, G. Perrouin, E.-Y. Kang, P.-Y. Schobbens, P. Heymans, A. Legay, B. Baudry, A vision for behavioural model-driven validation of software product lines, in: International Symposium On Leveraging Applications of Formal Methods, Verification and Validation, Springer, 2012, pp. 208–222.

[55] X. Devroey, G. Perrouin, P.-Y. Schobbens, Abstract test case generation for behavioural testing of software product lines, in: Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools-Volume 2, ACM, 2014, pp. 86–93.

[56] S. Lity, T. Morbach, T. Thüm, I. Schaefer, Applying incremental model slicing to product-line regression testing, in: International Conference on Software Reuse, Springer, 2016, pp. 3–19.

[57] S. Patel, V. Shah, Automated testing of software-as-a-service configurations using a variability language, in: Proceedings of the 19th International Conference on Software Product Line, ACM, 2015, pp. 253–262.

[58] M. Varshosaz, H. Beohar, M. R. Mousavi, Delta-oriented fsm-based testing, in: International Conference on Formal Engineering Methods, Springer, 2015, pp. 366–381.

[59] M. Lochau, I. Schaefer, J. Kamischke, S. Lity, Incremental model-based testing of delta-oriented software product lines, in: International Conference on Tests and Proofs, Springer, 2012, pp. 67–82.

[60] M. Lochau, S. Peldszus, M. Kowal, I. Schaefer, Model-based testing, in: Advanced Lectures of the 14th International School on Formal Methods for Executable Software Models - Volume 8483, Springer-Verlag New York, Inc., New York, NY, USA, 2014, pp. 310–342. doi:10.1007/978-3-319-07317-0_8.
URL http://dx.doi.org/10.1007/978-3-319-07317-0_8

[61] M. Lochau, J. Kamischke, Parameterized preorder relations for model-based testing of software product lines, in: International Symposium On

Leveraging Applications of Formal Methods, Verification and Validation, Springer, 2012, pp. 223–237.

[62] X. Devroey, G. Perrouin, P.-Y. Schobbens, P. Heymans, Vibes, transition system mutation made easy, in: Proceedings of the 37th International Conference on Software Engineering-Volume 2, IEEE Press, 2015, pp. 817–818.

[63] H. Beohar, M. R. Mousavi, Spinal test suites for software product lines, arXiv preprint arXiv:1403.7260.

[64] H. Samih, B. Baudry, Relating variability modeling and model-based testing for software product lines testing, in: ICTSS, 2012, pp. 18–22.

[65] S. Weißleder, H. Schlingloff, An evaluation of model-based testing in embedded applications, in: Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference on, IEEE, 2014, pp. 223–232.

[66] C. Henard, M. Papadakis, G. Perrouin, J. Klein, Y. Le Traon, Assessing software product line testing via model-based mutation: An application to similarity testing, in: Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on, IEEE, 2013, pp. 188–197.

[67] G. Perrouin, S. Sen, J. Klein, B. Baudry, Y. Le Traon, Automated and scalable t-wise test case generation strategies for software product lines, in: Software Testing, Verification and Validation (ICST), 2010 Third International Conference on, IEEE, 2010, pp. 459–468.

[68] B. Hasling, H. Goetz, K. Beetz, Model based testing of system requirements using uml use case models, in: Software Testing, Verification, and Validation, 2008 1st international conference on, IEEE, 2008, pp. 367–376.

[69] C. S. Gebizli, H. Sözer, A. Ö. Ercan, Successive refinement of models for model-based testing to increase system test effectiveness, in: Software Testing, Verification and Validation Workshops (ICSTW), 2016 IEEE Ninth International Conference on, IEEE, 2016, pp. 263–268.

[70] S. Weißleder, D. Sokenou, B.-H. Schlingloff, Reusing state machines for automatic test generation in product lines, in: 1st workshop on model-based testing in practice (MoTiP), 2008, p. 19.

[71] S. R. de Morais, A. Aussem, A novel markov boundary based feature subset selection algorithm, Neurocomputing 73 (4-6) (2010) 578–584.

[72] F. Damiani, D. Faitelson, C. Gladisch, S. Tyszberowicz, A novel model-based testing approach for software product lines, Software & Systems Modeling 16 (4) (2017) 1223–1251.

[73] M. Escalona, J. J. Gutierrez, M. Mejías, G. Aragón, I. Ramos, J. Torres, F. Domínguez, An overview on test generation from functional requirements, Journal of Systems and Software 84 (8) (2011) 1379–1393.

[74] H. Beohar, M. Varshosaz, M. R. Mousavi, Basic behavioral models for software product lines: Expressiveness and testing pre-orders, Science of Computer Programming 123 (2016) 42–60.

[75] A. M. Pérez, S. Kaiser, Bottom-up reuse for multi-level testing, Journal of Systems and Software 83 (12) (2010) 2392–2415.

[76] M. Farrag, Colored model based testing for software product lines (cmbt-swpl), Ph.D. thesis, Technical University of Ilmenau (2010).

[77] M. Al-Hajjaji, S. Krieter, T. Thüm, M. Lochau, G. Saake, Incling: efficient product-line testing using incremental pairwise sampling, in: ACM SIGPLAN Notices, ACM, 2016, pp. 144–155.

[78] H. Beohar, M. R. Mousavi, Input-output conformance testing based on featured transition systems, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM, 2014, pp. 1272–1278.

[79] E. Bringmann, A. Krämer, Model-based testing of automotive systems, in: 2008 1st international conference on software testing, verification, and validation, IEEE, 2008, pp. 485–493.

[80] A. Reuys, E. Kamsties, K. Pohl, S. Reis, Model-based system testing of software product families, in: International Conference on Advanced Information Systems Engineering, Springer, 2005, pp. 519–534.

[81] E. M. Olimpiew, H. Gomaa, Model-based testing for applications derived from software product lines, in: ACM SIGSOFT Software Engineering Notes, ACM, 2005, pp. 1–7.

[82] V. H. Fragal, A. Simao, A. T. Endo, M. R. Mousavi, Reducing the concretization effort in fsm-based testing of software product lines, in: 2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), IEEE, 2017, pp. 329–336.

[83] F. Damiani, C. Gladisch, S. Tyszberowicz, Refinement-based testing of delta-oriented product lines, in: Proceedings of the 2013 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools, ACM, 2013, pp. 135–140.

[84] R. Lachmann, S. Beddig, S. Lity, S. Schulze, I. Schaefer, Risk-based integration testing of software product lines, in: Proceedings of the Eleventh International Workshop on Variability Modelling of Software-intensive Systems, ACM, 2017, pp. 52–59.

[85] M. Zhang, S. Ali, T. Yue, R. Norgre, Uncertainty-wise evolution of test ready models, Information and Software Technology 87 (2017) 140–159.

[86] E. d. M. Rodrigues, et al., Plets: a product line of model-based testing tools, Ph.D. thesis, PhD thesis, Pontifical Catholic University of Rio Grande do Sul, Porto ... (2013).

[87] S. Weißleder, H. Lackner, Top-down and bottom-up approach for model-based testing of product lines, arXiv preprint arXiv:1303.1011.

[88] H. Samih, H. Le Guen, R. Bogusch, M. Acher, B. Baudry, An approach to derive usage models variants for model-based testing, in: Testing Software and Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 80–96.

[89] H. Samih, R. Bogusch, Mplm–matelo product line manager, in: 18th International Software Product Line Conference (2014), 2014, pp. 138–142.

[90] E. M. Olimpiew, Model-based testing for software product lines, Ph.D. thesis (2008).

[91] X. Devroey, Behavioural model based testing of software product lines.

[92] B. García Gutiérrez, R. Garcia Carmona, A. Navas Baltasar, H. A. Parada Gélvez, F. Cuadrado Latasa, J. C. Dueñas López, An automated model-based testing approach in software product lines using a variability language, in: Third Workshop on Model-Driven Tool & Process Integration, Fraunhofer Institute for Open Communication Systems, 2010, pp. 1–10.

[93] L. T. Costa L. T., Split-mbt: A model-based testing method for software product lines, Ph.D. thesis, Pontifícia Universidade Católica do Rio Grande do Sul (2016).

[94] L. Chen, M. Ali Babar, N. Ali, Variability management in software product lines: A systematic review, in: International Software Product Line Conference, Carnegie Mellon University, Pittsburgh, PA, USA, 2009, pp. 81–90.

[95] V. Alves, D. Schneider, M. Becker, F. Iese, F. Platz, N. Bencomo, P. Grace, Comparative study of variability management in software product lines and runtime adaptable systems, in: International Workshop on Variability Modelling of Software-Intensive Systems, 2009, pp. 9–17.

[96] R. Feldt, T. Zimmermann, G. R. Bergersen, D. Falessi, A. Jedlitschka, N. Juristo, J. Münch, M. Oivo, P. Runeson, M. Shepperd, D. I. K. Sjøberg, B. Turhan, Four commentaries on the use of students and professionals in empirical software engineering experiments, Empirical Software Engineering 23 (6) (2018) 3801–3820. doi:10.1007/s10664-018-9655-0.

[97] T. Vale, E. S. Almeida, V. Alves, U. Kulesza, N. Niu, R. Lima, Software product lines traceability: A systematic mapping study, Information and Software Technology 84 (2017) 1 – 18. doi:https://doi.org/10.1016/j.infsof.2016.12.004.

[98] A. Ampatzoglou, S. Bibi, P. Avgeriou, M. Verbeek, A. Chatzigeorgiou, Identifying, categorizing and mitigating threats to validity in software engineering secondary studies, Information and Software Technology 106 (2019) 201 – 230. doi:https://doi.org/10.1016/j.infsof.2018.10.006.