Information and Software Technology 56 (2014) 1183–1199

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

On strategies for testing software product lines: A systematic literature review

Ivan do Carmo Machado [a,*], John D. McGregor [b], Yguaratã Cerqueira Cavalcanti [c], Eduardo Santana de Almeida [a]

[a] Computer Science Department, Federal University of Bahia, UFBA, Salvador, Brazil
[b] School of Computing, Clemson University, Clemson, SC, USA
[c] Federal Data Processing Service, SERPRO, Florianópolis, Brazil

ARTICLE INFO

Article history:
Received 4 December 2012
Received in revised form 5 March 2014
Accepted 2 April 2014
Available online 13 April 2014

Keywords:
Software product lines
Software testing
Software quality
Systematic literature review

ABSTRACT

Context: Testing plays an important role in the quality assurance process for software product line engineering. There are many opportunities for economies of scope and scale in the testing activities, but techniques that can take advantage of these opportunities are still needed.
Objective: The objective of this study is to identify testing strategies that have the potential to achieve these economies, and to provide a synthesis of available research on SPL testing strategies, to be applied towards reaching higher defect detection rates and reduced quality assurance effort.
Method: We performed a literature review of two hundred seventy-six studies published from the year 1998 up to the 1st semester of 2013. We used several filters to focus the review on the most relevant studies and we give detailed analyses of the core set of studies.
Results: The analysis of the reported strategies comprised two fundamental aspects for software product line testing: the selection of products for testing, and the actual test of products. Our findings indicate that the literature offers a large number of techniques to cope with such aspects. However, there is a lack of reports on realistic industrial experiences, which limits the inferences that can be drawn.
Conclusion: This study showed a number of leveraged strategies that can support both the selection of products, and the actual testing of products. Future research should also benefit from the problems and advantages identified in this study.

Contents

* Corresponding author. Tel.: +55 71 9183 9735.
E-mail addresses: ivanmachado@dcc.ufba.br (I.d.C. Machado), johnmc@cs.clemson.edu (J.D. McGregor), ycc@cin.ufpe.br (Y.C. Cavalcanti), esa@dcc.ufba.br (E.S. de Almeida).

http://dx.doi.org/10.1016/j.infsof.2014.04.002
0950-5849/

1. Introduction

Software product line (SPL) engineering has proved to be an efficient and effective strategy to achieve economies of scale and scope, by exploiting commonalities and managing variation among products [1]. It is based on the idea that assets, built on the basis of a common design, can be configured and composed in different ways, enabling the creation of a diversity of products, in a shortened building period. Such a software development paradigm leads companies to achieve remarkable results such as substantial cost savings, reduction of time to market, and large productivity gains [2].

The systematic variability management is a fundamental characteristic of a SPL. This practice is what mainly distinguishes this from other software development strategies such as single system development. Variation points identify places in design and code where individual variants can be inserted. When assets designed with variation points are composed, with specific variants selected for use at each variation point, a large number of products can be built leading to what is often referred to as a combinatorial explosion in the number of variant products [3]. Therefore, managing variations at different levels of abstraction and across all generic development assets might be a cumbersome and complex task [4].

In a SPL, features are basic building blocks for specifying products. They can be defined as the distinctive characteristics of a system [5]. Feature modeling has become the de facto standard variability model, as it provides a compact representation of all products of a SPL in terms of their features [6]. Feature models represent the products in a SPL by means of dependencies among features and interrelationships among variation points [7].

Guaranteeing that every feature in a SPL will work as expected, and ensuring that combinations of features will work in each product is often problematic because of the high costs involved. Exhaustive testing is seldom feasible in any development process. This is particularly infeasible in SPL due to the variability in features, due to the many input variables.

A scoping review, carried out as background to the present review [8], revealed two independent but complementary interests a SPL testing strategy should handle, as follows:

- Firstly, it is necessary to check the feature interaction coverage, i.e., when checking the properties or configurations of a SPL, every feature combination have to be consistent with the specification and must not violate the stated constraints.
- Secondly, it is necessary to check the set of correctness properties of each product. Given that a built software artifact can be used by a range of products, an uncovered defect may be propagated to the many products that include it.

The *first interest* considers testing generation as a systematic selection of a representative set of product instances, comprising a subset of all possible configurations in a SPL. The main idea is to reduce the testing space. Fig. 1 illustrates the first interest. It shows that test cases refer to product configurations, i.e., a test case is responsible for testing whether an instance of the feature model (or whatever represents the variability in a SPL) is valid or not.

There are two main inputs to consider for this *interest*: **a set of product requirements** and the **quality of the variability model under test**. The role of requirements is to establish what functionalities a product instance should encompass. Regarding the quality of the variability model, through the analysis of its consistency, we could ensure that the produced models are complete and correct, in the sense that there are no conflicting restrictions between features, and that all the important distinctions and differences in a domain are covered by the model.

The *second interest* focuses on performing testing on end-product functionalities. Such an interest deals with the systematic reuse of test assets and results, as a means to reduce the overall effort, and avoid retesting of already tested features, while being effective at revealing faults. Test assets are designed to test the functionalities of features that will compose the products.

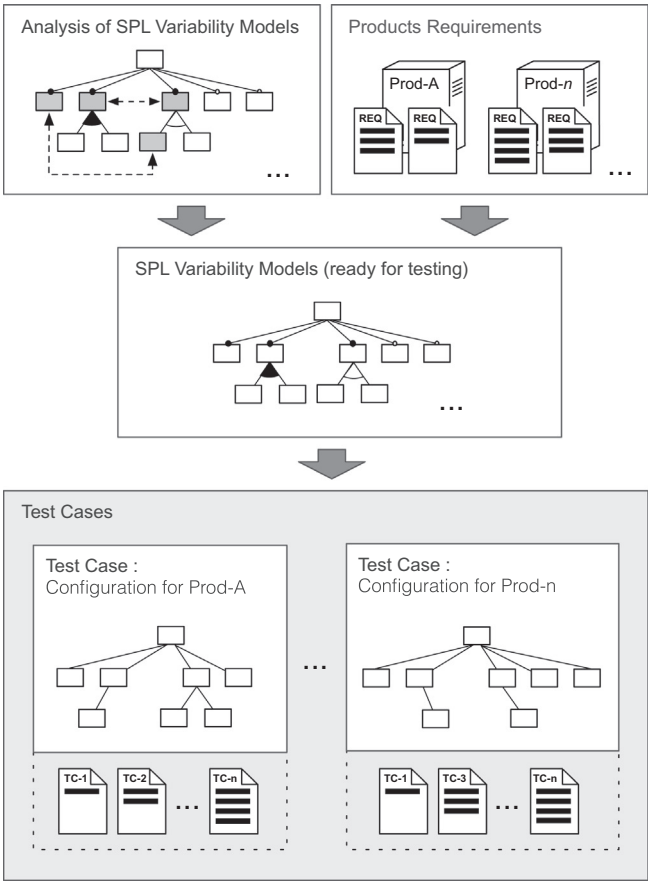

Fig. 1. SPL Testing interest: selection of product instances to test.

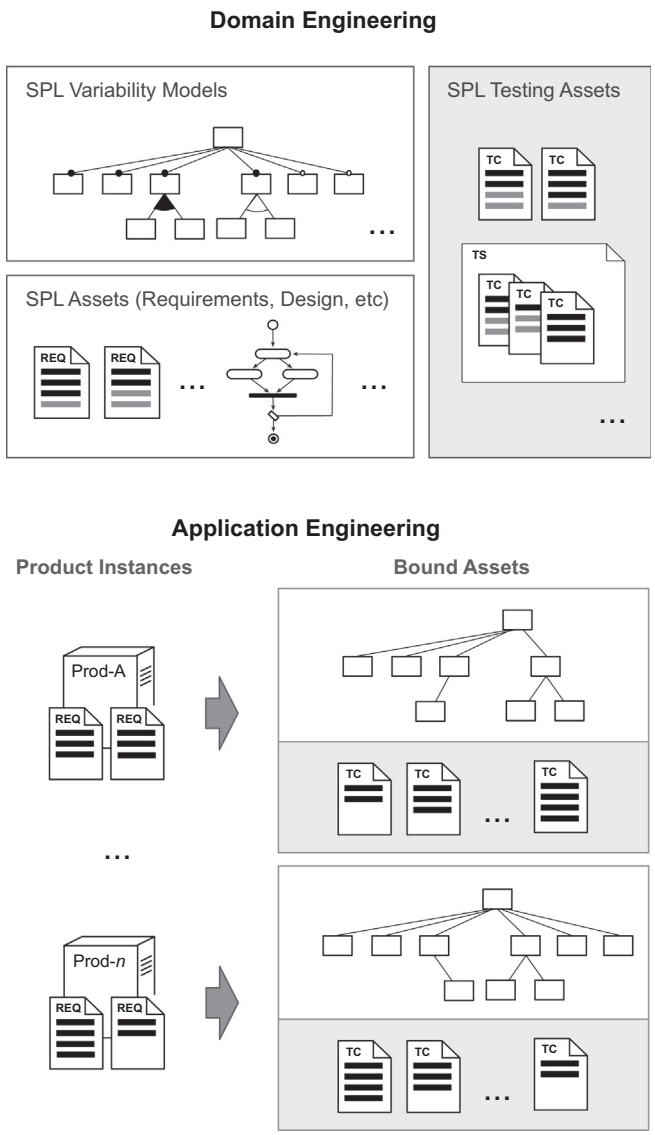

Fig. 2. SPL Testing interest: actual test of products.

Fig. 2 illustrates how this interest works. It comprises the two SPL processes [9]: *domain engineering* and *application engineering*. The former defines variable test assets (test cases, test scenarios, etc.), taking as input the variability defined for the SPL. In the later, when a product variant is instantiated, the variable assets are bound, according to the requirements expected for that particular product instance. Existing testing assets are also bound, as it is encouraged the reuse of test assets between product instances.

In both interests, a particular problem is the number of test inputs to consider, which can increase exponentially with the number of features that a SPL comprises [10]. Designing and/or selecting an effective[1] set of test cases, considering the likely amount of test inputs, play an important role in SPL testing.

Considering the importance of knowing which test case design and selection techniques the current SPL practice adopts, a detailed insight from the point of view of the interests aforementioned would be valuable. To this end, we conducted a *systematic literature review*, aiming at identifying, assessing, and interpreting available research evidence in the SPL testing research field, for the purpose of clear categorization. We investigated how studies address the generation of representative sets of products, from domain models, in order to sketch which techniques could be used and what their properties are. Yet, we investigated the existing support for testing end-product functions by taking advantage of the specific features of a product line, i.e., commonality and variability. We intend to collect evidence about current research that suggests implications for practice, and to identify open problems and areas that need attention.

In comparison with a previous review [8], this paper has been substantially extended, in terms of methodological details and findings, as follows: first, Section 2 is broadened and considerably detailed to provide a much fuller account of the research method employed. The results in Section 3 are considerably expanded with new material related to the studies that were included and with respect to the topics that were covered. Additionally, there is a discussion on appraisal methods employed in the studies, and their implications. Finally, Section 4 is expanded with a deeper discussion of the findings, their implications for research, and opportunities for future research.

The remainder of this article is structured as follows. Section 2 describes the research method used in this review. Section 3 presents the results of the review. The main findings are discussed in Section 4, together with the threats to the validity of this study. Section 5 discusses related work. Section 6 concludes the paper and presents directions for future investigation.

2. The review methodology

A systematic literature review (SLR) is a rigorous, systematic, and transparent method to identify, appraise, and synthesize all available research relevant to a particular research question, or topic area, or phenomenon of interest, which may represent the best available evidence on a subject [12]. A SLR may serve as central link between evidence and decision making. They provide the decision-maker with best available evidence. This evidence, in combination with field expertise and the customer-related values, characteristics, and circumstances, are necessary ingredients for making good decisions.

The importance of SLR for software engineering has been addressed by a reasonable amount of studies, as deeply discussed in [12–15]. Kitchenham and Charters [16] describe a set of reasons for performing a SLR, as follows:

- to review the existing evidence concerning a treatment or technology,
- to identify gaps in the existing research, which may indicate areas for further investigation, and
- to provide a context/framework in order to properly position new research activities.

In this study, we followed Kitchenham's guidelines for performing SLRs in software engineering [16]. This SLR included the following steps: *development of a review protocol*, *conducting the review*, *analyzing the results*, *reporting the results* and *discussing the findings*. The review protocol specifies all steps performed during the review and increases its rigor, transparency, and repeatability, while establishing a means to reduce risk of bias. The protocol includes: (i) the strategy employed to define the research questions, based upon an explicit research objective; (ii) the systematic searching methods that reduce the risk of selective sampling of studies, which may support preconceived conclusions, thus reducing risk of bias, and (iii) the method employed to evaluate available information.

Hence, the methodology used in this SLR included formulation of research questions to attain the objective, as next detailed in

[1] By *effective* we mean the defect revealing ability of a test suite [11].

Table 1
Research questions as structured by the PICOC criteria.

Population	Software product line testing research
Intervention	Approaches, i.e., methods, strategies, techniques, and so on, that support testing in SPL engineering
Comparison	N/A
Outcome	The effectiveness of the testing approaches
Context	Within the domain of SPL engineering, with a focus on testing approaches

Section 2.1; identification of sources from where research papers were to be extracted, described in Section 2.2; and also deciding the search criteria and principles for selecting and assessing the relevant studies, described in Section 2.3.

2.1. Research questions

We followed the *Population, Intervention, Comparison, Outcome* and *Context* (PICOC) structure to define our research questions, as defined by Petticrew and Roberts [17]. Such a structure enlists the attributes to consider when defining the research question of a SLR. Considering that in this review we do not sketch any comparison of interventions, the attribute *comparison* is not applicable. Table 1 shows the PICOC structure.

Considering that the two SPL testing interests discussed in the introduction section, hold different overarching goals, studies from both categories should be analyzed in a proper manner. Hence, our SLR aims to answer the following research questions:

- **RQ1.** What SPL testing strategies are available to handle the selection of products to test?
- **RQ2.** What SPL testing strategies are available to deal with the test of end-product functionalities?
- **RQ3.** What is the strength of the evidence in support of these proposed SPL testing strategies?
- **RQ4.** What are the implications of these findings for the software industry and the research community?

We defined the RQ1 to get an in-depth view on how existing SPL testing techniques cope with the *selection of product instances for testing* (first SPL testing interest). It considers the configuration of features as the main input for the design of test cases.

In addition, we are interested in the SPL strategies used to handle the *actual testing of SPL assets* (second SPL testing interest). Such an issue, covered by RQ2, is aimed at carrying out tests of end-product functions. It considers core assets as the input for designing the test cases. Core assets are those assets that form the basis for the SPL. They often include, but are not limited to, the architecture, domain models, requirements statements, reusable software components, etc.

Within the set of strategies identified in RQ1 and RQ2, we observed whether tool support was available to practitioners.

Furthermore, RQ3 helps researchers assess the quality of existing research. The results of this question are critical for researchers to identify new topics for empirical studies, and for practitioners to assess the maturity of a proposed strategy. RQ4 help us outline directions for future research and identify areas that need work in order to make strategies more applicable in industrial practice.

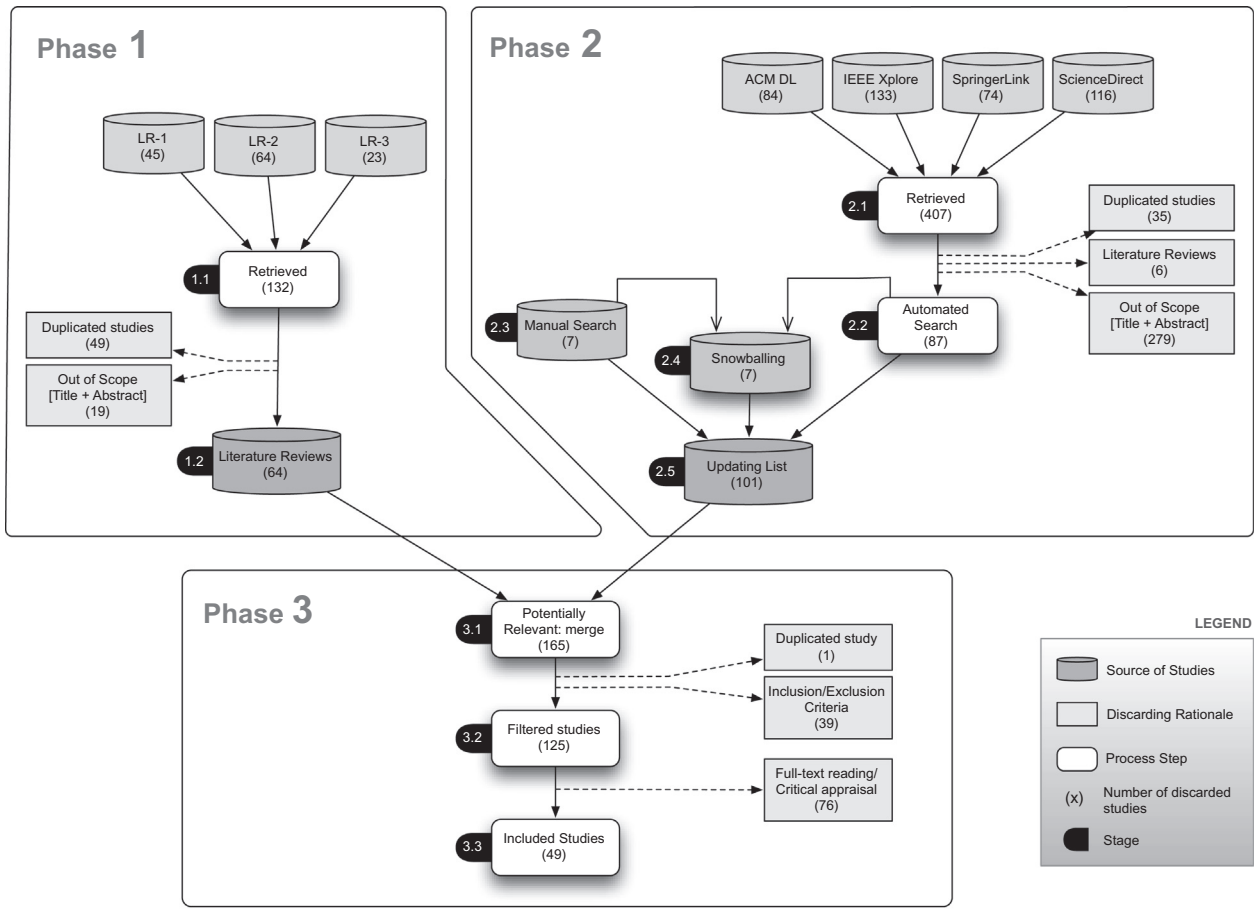

Fig. 3. Study selection procedure.

2.2. Identification of relevant literature

The process of gathering and selecting relevant studies involved three phases, as Fig. 3 shows. Our initial set of candidate papers was provided by previously published literature reviews on SPL testing [18–20]. Therefore, the first phase consisted of collecting the primary studies listed in each of these literature reviews, which include research papers published up to the year 2009. For the second phase, we performed a search for studies published from the year 2009 up to the year 2013. The third phase consisted of the screening process to exclude studies that are not relevant to answer the research questions.

Along this section, we expand on each gathering phase, and detail the study selection procedure.

2.2.1. Phase 1: analysis of existing reviews

In earlier research, our team reported on a systematic mapping study of SPL testing [18]. That is, most authors in this present paper are also authors of such a literature review.

By analyzing existing research, we could leverage state-of-the-art practices in the field, identifying *clusters* of studies that could support a fuller review. The systematic mapping study investigated 45 unique publications. The findings were in accordance with an analogous study [19], which also systematically mapped out the existing literature on SPL testing, in order to identify useful approaches and needs for future research. This latter analyzed a set of 64 unique publications. In both reviews, the overall focus was to enlighten researchers and practitioners with a broad picture of research and practice in the field of SPL testing, without providing in-depth analysis of any nature whatsoever.

A third study was performed with similar goals [20]. In a systematic literature review, a group of reseachers analyzed 23 unique publications. Unlike the two previously mentioned mapping studies, published as journal papers, this review was published as a conference paper, which, due to space constraints, might have limited the treatment of some required details, as a reader could expect.

These three studies complement each other in terms of research questions investigated. All of them have in common their overall goal, namely to provide an overview of the SPL testing field, pointing out achievements, opportunities, problems and available resources.

Thus, the initial set of primary studies for this SLR was gathered from these literature reviews, respectively *LR-1* [18], *LR-2* [19], and *LR-3* [20]. Given that they followed systematic processes of gathering, selecting, and assessing the studies [16,21], we acknowledge that they are a representative sampling of all primary studies in the SPL testing field, for studies published up to early 2009.

Within the reviews, we identified 132 potentially relevant papers, illustrated as **stage (1.1)** in Fig. 3. We read the *titles* and *abstracts* of the publications, to identify and exclude those which bear no relation to our investigation, i.e., studies that are not suitable to answer the RQs. This stage of screening was carried out by two independent reviewers (ICM and YCC) who screened and then met to compare their results. Any disagreement or uncertainty was discussed and arbitrated by a third independent reviewer (JdM).

Besides, as some studies were included in more than one literature review, we removed the overlap. In the end of such a screening, we had a set of 64 studies, illustrated as **stage (1.2)**.

2.2.2. Phase 2: gathering recent publications

The second phase of the *search process* consisted of an update on the list of primary studies. We analyzed the literature published between *2009* and *2013*.[2]

We performed an automated search in the following search engines and indexing systems: *ScienceDirect, ACM Digital Library, IEEE Xplore*, and *SpringerLink*. These are very useful online databases, since they index IEEE, ACM, Springer and Elsevier publications which, together, provide many of the leading publications in the Software Engineering field. Therefore, they are likely to include major SPL testing-related venues. Here we use the word *venue* as a generic name for journals or conferences (including workshops).

From the stated research questions, and known terms of the SPL testing domain, we identified the *keywords* to use in the search process. We applied variants of the terms *"software product lines"*, *"software product family*[3]*"*, *"software testing"* and *"testing techniques"*, to compose the search query. This was coded to fit the syntax requirements and capability of the search engines of each data source used. Table 2 lists the search strings applied in each search engine. The Table also shows the number of results retrieved from each.

From this task, we obtained a set of 407 publications, depicted as **stage (2.1)** in Fig. 3. As we considered studies retrieved from different search engines, 35 articles were excluded because they were duplicates, i.e., retrieved in more than one search engine. This search also retrieved related work, such as the literature reviews analyzed, and similar studies. We discarded these 6 publications, since they have been considered in other respects in this article.

Next task included reading the *title* and *abstract* of each remaining paper, similarly as in the previous phase. In this sense, a set of 279 articles were found to be irrelevant, as they did not consider testing from a SPL standpoint, but instead addressed issues from single-system software development. In the end, we had a pool of 87 publications from the automated search, depicted as **stage (2.2)** in Fig. 3.

Upon completion of the automated search, we carried out a manual search aiming at increasing coverage and quality [22,23], considering the same time span as in the automated search. Some important journals and conference proceedings were individually searched. Table A.10 in Appendix A lists the venues, and their usual abbreviations. They are clearly representative of the Software Engineering field. Such a claim is grounded in observations in related work, and also in discussions with colleagues. The task retrieved an additional 7 publications, as shown as **stage (2.3)** in Fig. 3.

In addition, reference lists of all identified publications were manually scanned, in order to identify missing relevant papers. This is the process called *snowballing* [24]. This task resulted in an additional 7 articles, as Fig. 3 illustrates in **stage (2.4)**. With such a small number of additional papers, we are convinced that we have a representative sampling and, furthermore, that the pool of the papers we have are relevant from a SPL engineering perspective. At the end of Phase 2, we revealed a universe of 101 new publications.

2.2.3. Primary study selection strategy

By merging the results from Phases 1 and 2, the list of potentially relevant studies was then composed of 165 publications. **Stage (3.1)** in Fig. 3 shows such an amount. We identified one duplicate, a study listed in *LR-1* that was also retrieved in the automated search. This was due to the year 2009 was considered in both *LR-1* and in the automated search phase.

We established a set of *inclusion* and *exclusion criteria* to assess each potential primary study. They were applied to the titles and abstracts of identified articles.

The criteria were specified based on the analysis scope of found papers to guarantee that only works really related to the context of

[2] It is worth mentioning that we considered papers published in the year 2013 that were available by the time we performed the searches (early of September).

[3] Software product family is a commonly used synonym for SPL [1].

Table 2
Detailed search strings applied in the automated search engines.

Engine	URL	Search string	Results[a]	
			Raw	Refined
IEEE Xplore	http://ieeexplore.ieee.org	(((software product line) OR software product family) AND test)	158	133
Springer	http://www.springerlink.com	'software product line' AND '(test, OR testing)' within 2009–2013	77	74
ScienceDirect	http://www.sciencedirect.com	pub-date >2008 ((''software product line'' OR ''software product lines'' OR ''software product family'') AND (''testing'' OR ''test'')) AND LIMIT-TO (topics, ''software, product line'') AND LIMIT-TO (yearnav, ''2013, 2012, 2011, 2010, 2009'')[All Sources (Computer Science)]	129	116
ACM DL	http://dl.acm.org	(''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'')	92	84

[a] **Raw results** means the whole set of entries listed by the search engines. Inasmuch as not only research papers are retrieved, but also *cover letters, foreword, preface, guest introductions, etc.*, hence, we filtered out those entries, and list the total number of research papers retrieved in the column **refined results**.

testing approaches[4] for SPL should be selected as the primary studies of this SLR.

This task aims to ensure that relevant studies are included and no study is excluded without thorough evaluation. At the outset, studies are only excluded if they clearly meet one or more of the exclusion criteria.

To be included in the review, studies had to present an approach to cope with SPL testing and at least one empirical evaluation to demonstrate its feasibility. It is worth mentioning that only studies produced in English were included. Table 3 presents the exclusion criteria.

At the end of the screening process, we ended up with a pool of `125` studies, to be subject of full-text reading, depicted as **stage (3.2)** in Fig. 3.

2.3. Data extraction

After using the criteria to select relevant papers, a quality assessment was performed on the remaining papers. We undertook a comprehensive analysis of the `125` filtered studies, to collect data necessary to answer the research questions, and to measure their quality.

The data were extracted and stored in a spreadsheet after reading each paper using a data extraction form. The form included the following attributes: **title, authors, corresponding email address, year of publication, source of publication, publication type, notes**, and an additional set of attributes, as listed below:

- **Research result.** We analyzed the outcomes of each primary study, and classified their main findings according to the types of software engineering research results [25]: *procedure or technique, qualitative or descriptive model, empirical model, analytic model, notation or tool, specific solution, answer or judgment*, or *report*.
- **Evidence gathering method.** We evaluated the evidence level (Lev1–6) reported in the study. Such assessment might lead researchers to identify new topics for empirical studies, and for practitioners to assess the maturity of a particular approach. Kitchenham et al. classified six levels of study design, based on the evidence hierarchy suggested from medical research [16]. On the basis of such a proposal, Alves et al. [26] described a tailored classification that could be fully applicable to the interest of this review. The classification includes the following hierarchy (from weakest to strongest):
  1. No evidence.
  2. Evidence obtained from demonstration or working out toy examples.
  3. Evidence obtained from expert opinions or observations.
  4. Evidence obtained from academic studies.
  5. Evidence obtained from industrial studies.
  6. Evidence obtained from industrial practice.
- **Industry.** In case a study was evaluated in industry settings, matching evidence levels `5` and/or `6` above, this attribute was used to identify the application domain in which authors carried out the evaluation.
- **SPL testing interest.** As a means to analyze techniques matching either one of the SPL testing interests, earlier discussed in this article, we decide to verify which interest is the main concern in each analyzed primary study.

The procedure of reading and completing the extraction form for each paper was again conducted by two independent reviewers (ICM and YCC), and any discrepancies were resolved by calling upon a third reviewer (JdM).

2.4. Quality assessment

We also quality appraised each study remained for data extraction, using a set of quality criteria. We extracted the criteria mainly from the questionnaire for quality assessment proposed by Dybå and Dingsøyr [27], which is based on principles of good practice for conducting empirical research in Software Engineering [16], and also on the Critical Appraisal Skills Programme (CASP).[5]

The questionnaire used to critically appraise the quality of the selected studies contained 11 closed-questions, as listed in Table 4. Taken together, the criteria provided a measure of the extent to which we could be confident that findings of a particular study could provide the review with a valuable contribution. The criteria covered the four main issues pertaining to quality that need to be considered when appraising the studies identified in the review [27]:

- **Reporting.** Reporting of the study's rationale, aims, and context.
- **Rigor.** Has a thorough and appropriate approach been applied to key research methods in the study?
- **Credibility.** Are the findings well-presented and meaningful?
- **Relevance.** How useful are the findings to the software industry and the research community?

[4] We herein generalize the term *approaches* to include not only actual *approaches* but also *strategies, methods, techniques,* and *processes*, given that authors sometimes interchangeably use those terms to define their proposal.

[5] http://www.casp-uk.net/.

Table 3
Exclusion criteria and list of excluded studies.

Exclusion criteria	Rationale
Related work	Secondary studies were not considered in this review. Such a kind of study were analyzed as related work
Abstracts	We excluded prefaces, editorials, and summaries of tutorials, panels and poster sessions
Doctoral symposium	Studies published in *doctoral symposia* were also discarded. To the best of our knowledge, this kind of study do not bring information other than a *status report* on the doctoral thesis work, and usually make reference to more complete studies, published elsewhere
Extended studies	When several duplicated articles of a study exist in different versions that appear as books, journal papers, conference and workshop papers, we include only the most complete version of the study and exclude the others
Position papers	Position or philosophical papers, i.e., papers only presenting an anecdotal evidence of the SPL testing field, were excluded from the literature review, but some of the position papers showing future directions are mentioned in the conclusion and future work section
Comparative papers	Comparative papers, with no additional contribution, but rather only analyzing existing literature, that eventually were included in our primary studies list
Out of scope	By analyzing the introduction section of a study, it is possible to figure out what the topic under investigation is about. Based on this statement we discarded studies which did not deal directly with testing, but instead that consider SPL in a general viewpoint

From the criteria obtained from [27], we only made a few changes to customize the criteria for *relevance* assessment, as a means to evaluate the relevance of the study for the software industry at large and the research community.

Each of the 11 criteria was answered with either "yes" (1) or "no" (0). Then, a quality assessment score was given to a study by summing up the scores for all the questions for a study. The resulting total quality score for each study ranged from 0 (very poor) to 11 (very good).

We used quality assessment criteria for both synthesis purposes and filtering papers. For the second matter, the first criterion was used as the minimum quality threshold of the review to exclude non-empirical research papers. Hence, 76 papers were excluded as part of this screening process.

Upon completion of the quality evaluation assessment, the set of selected primary studies was composed of 49 studies, illustrated as **stage (3.3)** in Fig. 3. See Table B.11 in Appendix B for a list of selected primary studies.

3. Results

We used the extracted data to answer our research questions. In this section, we initially give an overview of the selected studies with respect to their publication venues. Then, we answer each research question, based on the extracted information.

3.1. Characteristics of the studies

Table 5 shows the temporal distribution of the selected 49 primary studies, encompassing the years 2003 through 2013. The table also shows the distribution of the studies based on their publication channels, along with the number of studies from each source: workshops, conferences, journals, and the gray literature.[6] We may observe that only 3 out of 49 studies were found in journals, whereas most was published in conferences.

Such a distribution gives us the initial impression that most relevant studies in the field were only found in recent publications, i.e., as of the year 2010. Regardless the number of studies removed from our final list for matching any of the exclusion criteria, we should recall that we only included studies which presented any kind of empirical evaluation. Thus, we may notice a trend curve in the data, showing an increasing attention on the use of scientifically rigorous evaluation methods as a means to assessing and making explicit the value of the proposed approaches for the SPL testing field.

The 49 selected primary studies were gathered from 20 conferences, 6 workshops, 3 journals, 3 book chapters, and 1 technical report. As expected, the greater amount of studies in a single vehicle was found in the SPLC[7] (10 studies), considered the most representative conference for the SPL engineering area, followed by the SPLiT[8] (4 studies), a workshop dedicated to the SPL testing topic, co-located with the SPLC. The workshop was held yearly for five years, between 2004 and 2008.

During the data extraction process, we collected the SPL testing interest each selected primary study addressed. As the both interests are not mutually exclusive, i.e., a comprehensive process for testing SPL might encompass both of them, in this analysis we could group the studies according to their central proposal. The histogram in Fig. 4 shows the number of studies addressing each interest by publication year, where *Interest 1* means *selection of product instances to test*, and the *Interest 2* means the *actual test of products*, as earlier discussed in this article. Table 6 shows what studies addressed what SPL testing interest.

This histogram shows a recent growing interest in the proposals towards overcoming the first SPL testing interest. Despite the observed trend, we can state that both interests holds the same importance. Regarding the second one, much has been proposed along the years, so as to enable a detailed analysis of existing practices and describe the inherent challenges.

Besides, despite of the importance of both interests working together in a same technique, only in the year 2013 we found studies dealing with both. Although this is a rather small number of studies, it might be some indication that the research community has realized the benefits of proposing approaches which encompass both interests.

3.2. Strategies to handle the selection of products to test (RQ1)

Variability in features may lead to diverse products composition possibilities [28,29]. Although it is necessary to test and analyze all possible feature combinations, this is unrealistic for variability models of a reasonable size. In this sense, the core problem of the **first SPL testing interest** is to reduce the set of possibilities to a reasonable and representative set of product configurations. While keeping a small sample is critical to limit the effort necessary for testing each selected configuration [30], this is particularly a combinatorial problem, that should be handled accordingly.

Optimization techniques may be used to prune redundant configurations that need not be explored. From the set of 24 selected studies that cope with this SPL testing interest, we found *combinatorial interaction testing (CIT)* to be the *de facto* standard to handle test selection in such an interest. CIT enables the selection of a small subset of products where the interaction faults are most likely to occur. This is a cost-effective strategy for SPL engineering,

[6] *Gray literature* herein includes technical reports and book chapters.

[7] SPLC stands for *International Software Product Line Conference*.

[8] SPLiT stands for *International Workshop on Software Product Line Testing*.

Table 4
Quality assessment questions.

No.	Question	Issue
QC1.	Is the paper based on research and it is not merely a "lessons learned" report based on expert opinion?	Reporting
QC2.	Is there a clear statement of the aims of the research?	Reporting
QC3.	Is there an adequate description of the context in which the research was carried out?	Reporting
QC4.	Was the research design appropriate to address the aims of the research?	Rigor
QC5.	Was there a control group with which to compare treatments?	Rigor
QC6.	Was the data collected in a way that addressed the research issue?	Rigor
QC7.	Was the data analysis sufficiently rigorous?	Rigor
QC8.	Has the relationship between researcher and participants been considered to an adequate degree?	Credibility
QC9.	Is there a clear statement of findings?	Credibility
QC10.	Is the study value for research or practice?	Relevance
QC11.	Are there any practitioner-based guidelines?	Relevance

Table 5
Summary of selected primary studies by publication type and publication year.

Venue	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
Workshop	1	–	1	3	2	–	–	1	–	–	3	11
Conference	–	1	1	1	–	2	1	8	7	4	6	31
Journal	–	–	–	–	–	1	–	1	–	1	–	3
Gray literature	–	–	–	3	–	1	–	–	–	–	–	4
Total	1	1	2	7	2	4	1	10	7	5	9	49

aimed at reducing the test set by selecting the possible combinations of features that will be present in most products. That is, the representative subset of sample products under test. Within the selected studies, this selection is usually based on *combinatorial criteria on feature models*, *coverage criteria on variable test models*, and *coverage criteria on feature interactions*.

A prevalence of formal approches could be noted in those selected studies. For instance, *t-wise* feature coverage has been applied in conjunction with *SAT solvers*, by dividing the set of clauses, transformed from a feature diagram, into solvable subsets. The idea is to automatically generate the test products from a feature diagram that satisfy the *t-wise* SPL test adequacy criteria. *Alloy* can be applied to generate the test cases, and works by capturing all dependencies between features in a feature diagram as well as the interactions that should be covered by the test cases. Some algorithms to translate feature diagrams into Alloy specifications have been proposed, e.g., *FeatureDiagram2Alloy* – P31 –, *Kesit* – P36. Alloy specifications are used to examine the semantics of features and their interactions, as a means to cope with the problem of scale.

A number of studies – P05, P18, P19, P20, P23, P24, P28, P29, P30, P38, P043, P044 (50% out of the total) – applied *pairwise testing*, a specialized notion of *t-wise* coverage, as the main heuristic both for designing and selecting test cases. Pairwise has proven to be most effective when solving problems of large complexity. The underlying assumption is that, by describing the input model as a combination of two features each other, obeying the constraints between them, it might be easier to find inconsistencies, rather than trying to combine all features at once. Establishing pairwise as a test case reduction heuristic may lead to a practical means of sampling all the relevant combination in features models. As a test minimization technique, it aims at identifying and eliminating redundant test cases from test suites in order to reduce the total number of test cases to execute, thereby improving the efficiency of testing.

Pairwise feature interactions serve as an input for the test generation based on *constraint programming*, i.e., a paradigm which enables the design of a tailor-made pairwise constraint. This paradigm is well suited for optimization problems such as those related

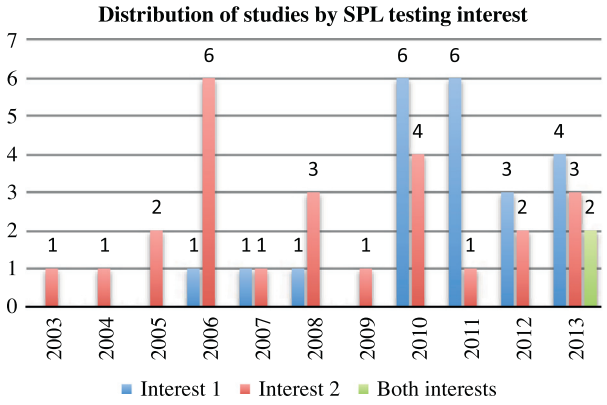

Fig. 4. Distribution of studies by SPL testing interest and publication year.

to the minimization of the size of test sets. It maps a feature model into a finite domain constraint model.

All selected studies provide either a process or an algorithmic implementation aimed at improving test coverage while reducing the overall test effort. However, the observed heterogeneity prevents us from sketching any kind of categorization.

We analyzed the tool support in the proposed approaches. In 17 studies, which represents 71% out of the total amount of selected studies handling this first SPL testing interest, we found a descriptive information about automated tool support. Among them, there are 8 studies – P23, P24, P30, P33, P36, P37, P42, P44 – which provide details about the proposed tool support, in terms of their goals and implemented requirements. They are aimed at analyzing feature models and automatically generate a set of test configurations that cover interactions (usually *pairwise*) between features in a feature model. Furthermore, the study P42 handles test case selection. In all of these studies it is possible to check the algorithm that implement the approach.

While this was expected to be a relevant aspect to enhance transparency in their proposals, an important downside might be observed, that is, from such a set, only 2 studies make reader aware

Table 6
Selected studies vs. SPL testing interest addressed.

Interest	Studies	Count
1	[P05], [P15], [P18], [P19], [P20], [P21], [P23], [P24], [P25], [P26], [P28], [P29], [P30], [P31], [P33], [P36], [P37], [P38], [P40], [P43], [P44], [P47]	22
2	[P01], [P02], [P03], [P04], [P06], [P07], [P08], [P09], [P10], [P11], [P12], [P13], [P14], [P16], [P17], [P22], [P27], [P32], [P34], [P35], [P39], [P41], [P45], [P48], [P49]	25
Both	[P42], [P46]	2

of how the tool could be obtained. The main reason is that, the proposed algorithms and tools are often developed solely to demonstrate a particular nuance of a methodology, mostly within the context of a research group.

Furthermore, in the remaining studies, authors from P05, P20, P25, P28, P29, P31, and P38 claimed that their approaches should be implemented in a tool support; however, they only described the algorithms associated to the proposal. No other information could be found. In one study – P18 –, carried out as an industrial case study, authors stated that an *in-house* tool was used to support their proposal. For this reason, they could not provide readers with further details. In P21, authors stated that their proposal works in conjunction with a tool that was developed beforehand, and discussed in a preceding publication.

3.3. Strategies to handle the test of end-product functionalities (RQ2)

Whereby research on configuration-aware software testing for SPL focuses on the combinatorial problems during interaction testing by detecting valid and invalid combinations of configuration parameters, the SPL testing interest covered in the RQ2 reveals testing practices and problems for the actual testing of functionalities.

The leveraged techniques work either by performing testing at the *domain engineering*, or testing the concrete assets at *application engineering*. Domain engineering is by definition the SPL process where commonality and variability analysis take place, leading to the definition of reusable test assets, i.e., by representing the points in which assets will accommodate different properties, that will be further exploited during application engineering.

As software reuse plays a fundamental role in SPL engineering, we analyzed the selected primary studies in the light of a set of characteristics a technique should cover. Table 7 presents the characteristics, and next Table 8 sketches a relationship between the selected studies and the addressed characteristics.

Regarding the characteristic *variability*, we were interested in understanding which, and how the approaches define test cases (TC) and test scenarios (TS) by expressing variability from domain models in such artifacts. We hypothesize that, for each feature, there should be test cases present in the test suite to validate whether the feature has been correctly implemented. We identified 12 out of 27 studies (44%) proposing strategies that use the variability models to define reusable test assets.

In the following characteristic, *asset reuse*, the focus was to understand whether the study explicitly provided a technique to reuse test cases (TC), test scenarios (TS), test results (TR), and test data (TD), either between products, or from a core asset base. This was found to be the most common characteristic within the selected studies. The amount of 23 out of 27 studies (85%) made any contribution to this group of characteristics, with a larger amount dedicated to establish strategies to handle TC and TS reuse.

There is an initiative to improve test asset reuse by focusing on the differences between product instances, in a so-called delta-oriented testing technique, based on regression testing principles and delta modeling concepts. The idea behind the technique is that a SPL can be represented by a core module and a set of delta modules. While the core modules will provide an implementation of a valid product, the delta modules specify changes to be applied to the core module to implement further products, by adding, modifying and removing code [31]. In this effect, test models for a new product instance will be generated by considering the deltas between this and the preceding product instances. That is, testing will focus on the deltas, what enables an increased reuse of test assets and test results between products. The tecnhique was investigated in both P41 and P48.

Next, we analyzed how the studies addressed the automated generation of test cases (TC), test case selection (TCS), and test inputs (TI). A small number of studies provided any description on how they could automate such important tasks so as to make SPL testing a feasible practical approach. Only 5 out of 27 studies (19%) explicitly provide tool support to handle the listed characteristics, as summarized in Table 8. They are: P01, P11, P27, P42, P49. In all of them, the studies only state they developed a tool, but they are not transparent in providing a means to make their tool available. Another 5 studies – P03, P12, P14, P17, P22 – point out the need of tool support to handle those characteristics. However, instead of proposing new tools, they use already established ones.

In another 7 studies (26%) – P06, P08, P32, P35, P39, P41, P45 – authors state their proposed approaches are supported by automated tools. However, they do not provide readers with detailed information about the tool, neither make clear whether the tool was built for the particular purpose of the investigation.

The last characteristic observed in which SPL process the approaches fit, domain engineering, application engineering, or both. Testing the common aspects early in domain engineering is essential, since an undetected error in the "framework" assets can be spread to all instances depending on those assets. However, as it is unfeasible to test all possible input values for the domain assets, since different variants might depend on specific applications, testing must also be performed in application engineering. Besides testing input data not covered yet in domain engineering, even the common assets previously tested might not work properly when bound in a specific product instance. Thus, some of domain assets should be retested again in application engineering, considering the particular behavior of a product instance. In this effect, we found 10 approaches working at *domain engineering*, while 9 approaches handle testing at *application engineering*. Finally, we found 8 approaches that can be applied in both SPL processes.

3.4. Strength of evidence in support of available strategies (RQ3)

According to the evidence evaluation scheme described in Section 2.3, Table 9 presents detailed results on how much evidence is available to adopt the proposed approaches. Fig. 5 provide a summarized data representation of the results.

Data showed that all selected studies present any kind of preliminary evaluation. However, this apparent benefit is diminished when we consider the low level of evidence of the proposed approaches. The most commonly employed evaluation methods are *academic studies* (Lev4) – 53% out of the total, followed by *demonstration* (Lev2) – 33%. Only a very small number of the approaches claimed to be evaluated in industry – 14%, what lead

Table 7
Leveraged SPL testing characteristics.

Characteristic	Rationale
Variability	SPL test assets are explicitly designed and modeled using variation points, that express their behavioral diversity
Test reuse	Test assets from domain engineering may be systematically reused in application engineering, as assets from a product instance may serve as input for a next instance
Automation	Employing an automated strategy for generating SPL test assets leads to significant effort reduction
SPL process	Tests can be performed in domain engineering, application engineering, or both

Table 8
Relationship between selected studies and characteristics

Study	Variability		Asset reuse				Automated gen.			Process		
	TC	TS	TC	TS	TR	TD	TC	TCS	TI	DE	AE	Both
P01	·	·	●	·	·	·	●	·	·	·	·	●
P02	●	·	●	·	·	·	·	·	·	●	·	·
P03	⊖	●	·	●	·	·	⊖	·	·	·	·	●
P04	●	·	●	·	·	·	·	·	·	·	·	●
P06	·	·	●	·	·	·	·	·	·	·	●	·
P07	●	·	●	·	·	·	·	·	·	·	·	●
P08	·	·	·	·	●	·	·	·	·	·	●	·
P09	·	●	·	·	·	·	·	·	·	·	·	●
P10	·	·	●	·	·	·	·	·	·	●	·	·
P11	·	·	●	·	·	·	●	·	●	·	●	·
P12	·	·	●	·	·	·	⊖	·	·	·	●	·
P13	·	●	·	●	·	·	·	·	·	●	·	·
P14	·	●	·	●	·	·	⊖	⊖	·	●	·	·
P16	·	·	·	●	·	·	·	·	·	●	·	·
P17	·	●	⊖	●	⊖	·	⊖	·	·	·	·	●
P22	●	·	·	·	·	·	⊖	·	·	●	·	·
P27	·	●	·	●	·	·	·	·	●	●	·	·
P32	●	·	·	·	·	·	·	·	·	●	·	·
P34	●	·	●	·	⊖	·	·	·	·	·	·	●
P35	·	·	·	·	●	●	·	·	·	·	·	●
P39	·	·	●	·	·	·	·	·	·	·	●	·
P41	·	·	●	·	●	·	·	·	·	·	●	·
P42	·	·	·	·	·	·	·	●	·	●	·	·
P45	·	·	·	·	●	●	·	·	·	·	●	·
P46	·	·	·	·	●	●	·	·	●	●	·	·
P48	·	·	●	·	●	·	·	·	·	·	●	·
P49	·	·	●	·	·	·	·	●	·	·	●	·

Legend: [●] Characteristic clearly addressed by the study, [⊖] the study encourages the use of such characteristic, but do not provide any implementation (e.g., it states an external tool is used, but no detail is provided), [·] characteristic not mentioned in the study, [TC] test case, [TI] test input, [TD] test data, [TCS] test case selection, [TR] test result, [DE] Domain Engineering, [AE] Application Engineering.

us to consider an overall low-level of evidence in the SPL testing field.

The scenario is even more worrying as data from the SPL testing interests are considered in a separate way. Figs. 6 and 7 show data from interests 1 and 2, respectively. Only one study from the first interest set was evaluated in industry settings. We found no studies with both academic and industrial evidence.

It is worth mentioning that we found no studies applying more than one evaluation method. The combination of empirical evaluation methods to assess the feasibility of an approach is desirable because it increases external validity for findings.

3.5. Implications for research and practice (RQ4)

To determine the implications of the approaches found in the literature for both research and practice, we have to initially figure out the limitations of selected studies. We used the quality criteria established for appraising the selected studies, presented in Table 4, to determine the strength of inferences.

The criteria were categorized in four groups of issues: reporting (QC1–3), rigor (QC4–7), credibility (QC8–9), and relevance (QC10–11).

Quality assessment results show that most of the selected studies perform fairly well on *reporting* issues. We recall that, as the first criterion (QC1) was used as the basis for either including or excluding a study, meaning that all selected studies are based on research and not merely a "lessons learned" report based on expert opinion. The same positive result could be observed in the QC2, while roughly 98% of the selected studies clearly state the aims of the research. While excelent results were obtained in the first two criteria, only 71% of the studies provided an adequate description of the context in which the research was carried out. That is, authors provide readers with limited information on the application domain(s) in which the approach was used, or the software process(es) involved, or even the skills of involved engineers, necessary to seamlessly use the proposed approach, etc.

The *rigor* was analyzed in terms of four criteria. In roughly 49% of the studies, the research design is appropriate to address the aims of the research. In only 29% out of the total, there was a control group with which to compare treatments, so as to enable authors to compare the outcomes of the proposed approach against an external entity. In 43% of the studies, data was collected in a way that addressed the research issue, and in only 16% of the studies, data analysis was sufficiently rigorous. The results point out to a frustrating lack of rigor, as it is likely that some of the findings from the selected studies are probably accurate and usefully generalisable. However, the apparent shortcomings in methodology seriously limit their usefulness.

Table 9
Evidence level of selected studies.

Study	Lev1	Lev2	Lev3	Lev4	Lev5	Lev6
P01				•		
P02		•				
P03		•				
P04						•
P05				•		
P06		•				
P07		•				
P08		•				
P09						•
P10				•		
P11				•		
P12		•				
P13				•		
P14		•				
P15		•				
P16					•	
P17						•
P18		•				
P19				•		
P20		•				
P21		•				
P22		•				
P23				•		
P24				•		
P25				•		
P26		•				
P27		•				
P28				•		
P29				•		
P30				•		
P31				•		
P32		•				
P33				•		
P34				•		
P35				•		
P36				•		
P37				•		
P38		•				
P39				•		
P40				•		
P41				•		
P42				•		
P43				•		
P44					•	
P45				•		
P46				•		
P47				•		
P48					•	
P49					•	

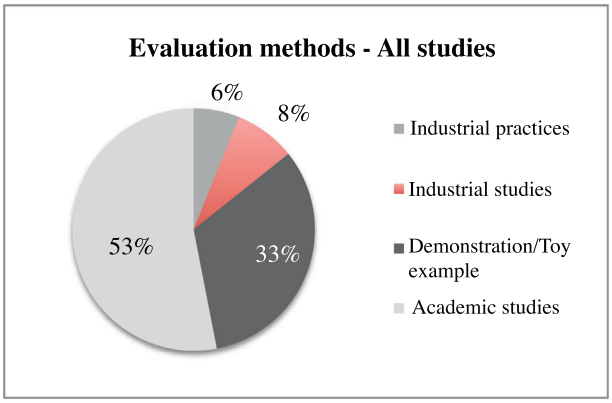

Fig. 5. Evidence available to adopt the proposed methods – all studies.

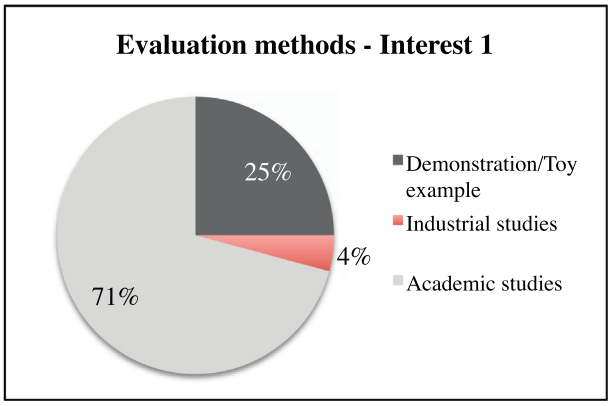

Fig. 6. Evidence available to adopt the proposed methods – interest 1.

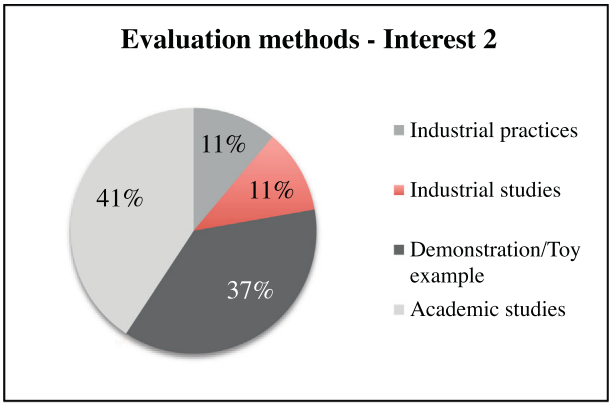

Fig. 7. Evidence available to adopt the proposed methods – interest 2.

In relation to *credibility* of the study methods, for ensuring that the findings are valid and meaningful, we found a rather small amount of studies (12%), in which the relationship between researcher and participants was considered in the evaluation. However, for 65% of the studies the findings were explicitly stated, in which they discussed the results on the basis of the research questions, and an adequate discussion of the evidence, thus including the limitation of the approach, was provided.

In terms of *relevance* for both research and practice, we considered that 84% of the selected studies as valuable, for they describe the strengths and weaknesses of their proposal, and point out open rooms for improvement. As opposed to this high value, only 16% of the studies present any practitioner-based guidelines, which acts as a barrier to providing optimal usage of the proposed approach, especially for industry-scale application.

To understand the impact of the selected studies for the research field, we gathered their citation counts. A simple way to gather the citations is using the Google Scholar Citations.[9] While some academics have been very critical of Google Scholar [32], as it may lack the quality control needed for its use as a bibliometric tool, other authors [33] argue that its coverage is very comprehensive, and as such it might provide a less biased comparison across disciplines than tools such as Web of Science or Scopus. These present an insufficient coverage, as they barely include conferences and/or workshops, but are focused on journals instead. Besides, Google Scholar has displayed considerable stability over time, increasing the coverage for disciplines that have traditionally been poorly represented.

Figs. 8 and 9 shows scatterplots for the citation counts and quality scores, respectively for the set of included studies published

[9] http://scholar.google.com.

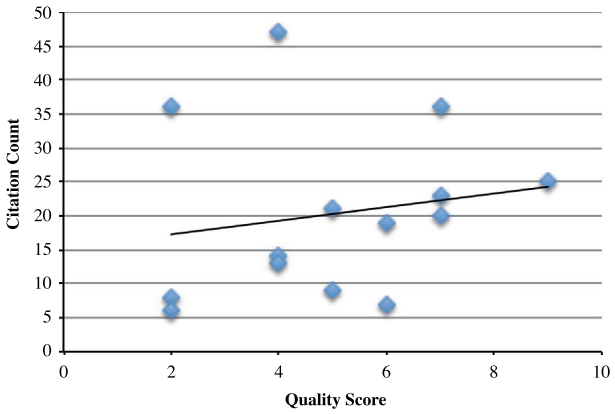

Fig. 8. Studies published between 2003 and 2009.

between 2003 and 2009, and the studies published between 2010 and 2012. Such a division was arbitrarily established as a means to mitigate the likely *timing* effect in the analysis, i.e., as time passes, it is likely that more citations a paper will hold. Hence, it would not be fair to use observations from all studies in a same sample.

It is worth to mention that we excluded selected studies published in the year 2013. The reason is that it usually takes time for any automated citation engine to index all publications and the citations thereof. In a briefly performed search, we barely found a citation to any paper published in the year 2013 that was considered in this systematic review.

The *y*-coordinate corresponds to the number of citations of a paper. The *x*-coordinate corresponds to the quality score achieved in our assessment. The point representing the observation for a given paper is placed at the intersection of the two coordinates. In the first set we list 14 observations (we found no results for 4 studies). In the latter, we list 23 observations.

In both sets we observe an uphill linear pattern, as we move from left to right. It is a brief indication of a positive relationship between the citation count and the quality score attributed to the selected studies, especially in the second set. However, it is hard to infer whether any cause-and-effect relationship exists, due to the rather limited number of observations.

4. Analysis and discussion

This systematic review identified a set of testing approaches that are relevant for the SPL engineering field. The main observation in this study is the emerging need of supporting future development of an evidence-based and effective testing practice designed to address the special SPL engineering quality needs and thereby improve the overall SPL practice. Along with such this observation, we describe throughout this section other important aspects that may arise from this research.

This review has to do with software testing in SPL engineering. We should recall that testing is a validation technique employed to measure the product correctness and efficiency, identifying early in the development life cycle points in the program that need fixes, before delivering it to the final customer. An ideal purpose for SPL testing is to evaluate the correctness of features' assembly by gathering information on the sources of faults and statistics on how many errors are generated with certain volumes.

Much of the work on reducing test effort for SPLs has been performed from the *feature-based* (first interest) or *product-based* (second interest) perspectives. An overall impression we had is that, while approaches in both interests are concerned about minimizing the test effort, there is little discussion on the subject of achieving higher defect detection rates. There is rather limited evidence describing which are the commonly found faults in either interest. As each interest has a specific goal in the SPL testing task, knowing common faults can be helpful. It might guide an engineer to better identify the faults that are more likely to occur, and provide him/her with recommendations for the correct repair actions.

As our work attempts to provide research directions to SPL testing engineers to design and develop more effective SPL testing approaches, besides the previously stated observation, we next discuss the main open issues we found under each of the interests.

- *First SPL testing interest*

Within the challenges in this SPL testing interest, the main concern is in how to establish a strategy to test all the feature interactions that result in testing of all variations across all dimensions of variation. As pointed out in P46, current techniques handling the combinatorial problem can be categorized as *sampling* and *exhaustive* exploration techniques. While the former emphasizes oh the selection of configurations, likewise in pairwise coverage, to reduce the combinatorial space involved, the latter intends to consider all possible configurations, with the aid of specialized techniques to eliminate – or even to minimize – redundant configurations that do not need to be explored.

Within the selected studies, that fit into either one or another category, we observed the use of diverse optimization algorithms. Although the proposed solutions are usually claimed to achieve better levels of effectiveness than others, it is usually hard to figure out which can be suitable for an average scenario so as to enable generalizations. Despite the large amount of algorithms, we could not identify which could yield better results for a SPL project, due to the lack of reliable comparative assessment.

Another important highlight is that comparative studies of existing approaches are barely published. Perrouin et al. [3] empirically analyzed the capabilities of studies P29 and P31, in an investigation that can be considered one of the few and relevant reports providing comparisons between SPL testing techniques. Such a nice endeavor in highlighting the strengths and weaknesses of their approaches should be followed by other researchers, as a means to make practitioners aware of which testing approaches could be feasible to a given application domain and/or scenario.

A task strongly related to the automated generation of product configurations involves determining the *quality* of the product line variability models. For instance, in a feature model, it is necessary to analyze the relationships among the features that determine the composition rules and the cross-tree constraints, and some additional information, such as trade-offs, rationale, and justifications for feature selection [34,35]. Such analysis determines which configurations are feasible in practice.

SPL modeling approaches seek to better and more effective strategies to ensuring consistency of the variability models. This verification effort concerns the technical quality of the variability structure, such as checking that two features that are mandatory are not also mutually exclusive. This type of check usually employs some forms of logic or rule checker on the constructs provided by the methodology.

Fig. 1, earlier presented in Section 1, illustrates the role of the analysis of variability models in the first SPL testing interest. The *analysis* layer represents an input activity to the selection of product configurations to test. That is, before selecting the test cases – the valid and representative products – the SPL testing approaches should perform a series of analysis operations on feature models, such as finding if a product is valid, obtaining all products, calculating commonality and variability, and detecting dead and false optional features [36], through the extraction and understanding of the feature model semantics.

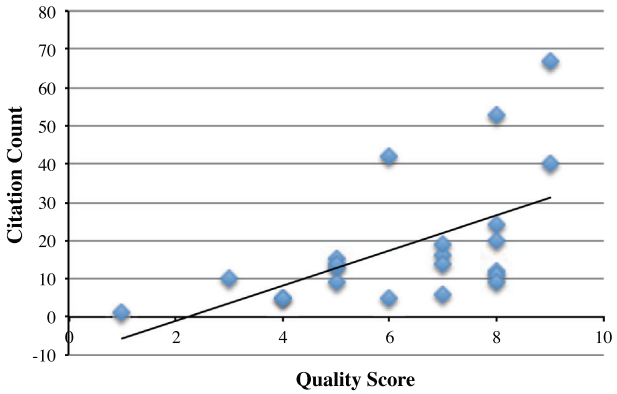

Fig. 9. Studies published between 2010 and 2012.

There are more complex analyses that can be made over feature models. Benavides et al. [37] provide state-of-the-art information on the analysis of feature models, leveraging the operations employed in model verification. The operations are used to identify anomalies in the models, and to detect structural and integrity constraint violations as well. The operations both help verifying the ability of the product line model to generate all the desired products, and only them, and measure the quality of the model, as a means to identify and correct the defects in the model, a vital task for efficient management and exploitation of the SPL [38].

Despite of the importance of model verification for the SPL testing field, the set of primary studies analyzed in this review, that discuss fundamental and practical aspects of testing in SPL engineering, do not provide details about how they perform the model verification task. They usually use the operations on feature models aimed at generating sets of valid products from them, instead of using the ones aimed at measuring the quality of the models. Therefore, model verification aspects are not subject to discussion in this systematic review. Salinesi and Mazo [38] surveyed the literature on verification of product line models, and as such their work can be used as a means to understand the gaps and challenges ahead.

- *Second SPL testing interest*

Most of the approaches dealing with specification of variability in the test assets work by annotating variability in test models. UML activity and sequence diagrams have been used as the standard models. The variable scenarios include specifications to the whole set of admissible products, plus other test cases which instead will vary for each specific product, depending on how the variant characteristics are instantiated. Indeed, not all admissible test cases is derived, but rather they derive the SPL test specification and leave it unfolded. The test cases will actually be derived for a specific product after having instantiated the tags in each SPL use case to the appropriate values.

While those approaches seems to work to any SPL project, improving testing practice, no discussions on the maintenance effort for those variable test assets is provided. A variable test suite is typically developed to test the whole SPL and the test suite will be modified as new products come into play, or the current products need to be improved. However, as the number of products increases, the number of test cases for testing the SPL will also increase. Therefore, it becomes practically impossible to execute all the test cases of the product line due to limited available time and resources for each new product. Therefore, it is essential to seek a solution to minimize test suites for a specific product efficiently before execution to reduce the cost of testing. Besides, an efficient testing process should systematically exploits the reusability of test artifacts between the products under test.

Besides, we observed a strong concern about handling test design at a very high level of abstraction, regardless the importance of also coping with variability at lower levels, such as at source code. While the existing approaches can be applied to any SPL project, with only minor adjustments, satisfying system testing of end-product functionalities, no studies could be found that consider the likely particularities of unit testing, performed in conjunction (either before, or right after) with the actual implementation of features. We have no indications about the impact of employing different mechanisms to implement variability in the domain assets, and its consequence for the unit testing, the lowest abstraction level for testing. This observation proceeds from the fact that testing techniques relevant to single-system engineering might not deal with the variability intrinsic to the SPL domain. Thus, a deeper evidence-based discussion on such an issue would be valuable.

Another important aspect to consider is the potential a SPL testing strategy has to cope with traceability issues. In a SPL, establishing traceability between feature models to actual implementation artifacts is a key task to a number of tasks, such as program comprehension, maintenance, requirement tracing, impact analysis, and reuse opportunities [39]. From a SPL testing perspective, the links between problem space and solution space entities enable both the definition of the test assets, and the capability to cope with evolution, in a sense that it is possible to identify the work products affected by a proposed change.

The selected primary studies address part of the aforementioned tasks. The use of annotated UML models (activity and sequence diagrams), in which features are associated to stereotypes, makes it possible to establish the relationship between a given feature and its corresponding test models and implementation – P07, P09, P11, P12, P13, P14, P17, P34, P35. It is also possible to use traceability-mining algorithms [40] to recover the tracing information, even though no primary study discussed about this feasible opportunity.

Traceability information between features and test source code should also be established. For each product, it is important to know the functionalities it provides and hence the features that it contains and its source code [41]. However, such low-level traceability information is barely mentioned in the analyzed primary studies. The studies P45 and P46 propose a means to handle low-level variability in test cases. Variability is implemented as simply Java Boolean variables, i.e., a SPL is an ordinary program with if-conditions. This known limitation prevents us to generalize such a technique to scenarios in which other variability implementation mechanisms are employed.

Furthermore, in practice, as systems evolve, traceability links between the models and the test code artifacts may become broken or outdated. Thus, it is necessary to keep the consistency between the artifacts. The study P22 contributes an approach to provide traceability links in a way that ensures consistency between the feature model and the code artifacts, enables the evolution of variability in the feature model. The approach provides information on the artifacts that are actually impacted by a change, and provides immediate feedback on the actual impact of that change, reflecting on the tracing between feature and code assets. In addition, in P49, test cases are augmented with an attribute to associate it to the feature, or to a set of features. This is a promising strategy that makes it possible to keep the tracing between both entities updated.

As a matter of fact, SPL testing strategies still have lots to do in terms of traceability and evolution. Currently, external tools handle most traceability and evolution aspects, as mentioned in P09. Just a few research efforts could be retrieved that incorporate those aspects into their proposals.

Another real concern in this SPL interest refers to the strength of evidence in the selected studies, as earlier discussed in RQ3 and reinforced hereinafter.

There is ample evidence that many SPL testing approaches are methodologically weak. Despite the amount of approaches, authors never benchmarked their results against other approaches. It may hinder any inference on highlighting the benefits of selecting and/or using an approach over another. Good research requires not only a result, but also clear and convincing evidence that the result is sound. This evidence should be based on experience or systematic analysis, not simply persuasive argument or small examples [25].

Empirical studies comparing the effectiveness of existing approaches, in a range of scenarios, thus involving project of different sizes and domain, is encouraged. Furthermore, there are some obstacles in the current tool support that should be addressed before further empirical studies can be conducted. As earlier mentioned, although some studies attempt to describe their tool support, no tools are readily available. When some algorithms describing how the approach would work in an automated scenario, readers are not provided with any discussion on their real benefits, and the required effort for its implementation.

The observed lack of empirical studies does not allow to evaluate what are the best ways to test a SPL. This is an important research issue for the area.

Besides, no guideline or methodology is provided to train test engineers to handle test design and selection. This means the current practice of test selection largely depends on the expertise of test engineers and it might not scale when more components are developed and are to be tested.

This analysis has considered individual research reports, but major results that influence practice rely on accumulation of evidence from many projects. Each individual paper thus provides incremental knowledge, and collections of related research projects and reports provide both confirming and cumulative evidence.

4.1. Limitations of this study

Our systematic review has some limitations. To the extent that we performed a systematic literature review, the potential for incomplete identification of relevant studies and publication bias are always consideration.

A potential risk that we might have missed relevant papers is due to lack of agreed terminology in the SPL field, leading to the possible existence of relevant papers that do not explicitly mention the keywords we specified. Hence, there is always a risk that important studies are omitted. To minimize this possibility, the search for potentially relevant studies encompassed a bibliographic search of published literature reviews on the topic under investigation, a search with multiple databases, and also bibliographic searches of the reviewed articles to identify additional studies. Thus, by combining the list of retrieved papers, we might assume a good coverage of publications and venues in the SPL testing field.

With respect to publication bias, the heterogeneity in studies' design, interventions, and outcome measures, and the absence of statistically reliable effects preclude any strong claims for the effectiveness of the analyzed approaches. In most cases, data collection and analysis were poorly described. Overall, empirical studies are not supported by rigorous evidence. Hence, due to there being insufficient data in the papers, necessary for the computation of effect sizes, we could not assess publication bias through meta-analysis. However, we acknowledge that, given the goals of this study, meta-analysis would not have been appropriate.

The effect of publication bias might have affected the data extraction, in terms of inaccuracy and bias. We had some difficulties to extract relevant information from the selected papers. For instance, several papers do not explicitly mention in which domain, the proposed approaches can be used, or there was a lack of information regarding the empirical methods employed by the studies to carry out their evaluation. In situations like these, we had to make subjective interpretations of information. Therefore, the researcher's bias could affect the final extracted data. Hence, we acknowledge that there is a possibility of having misunderstandings in the way we have extracted data from the primary studies. The data extraction form was designed to obtain consistent and relevant information for answering the research questions. Besides, we performed quality assessment on relevant studies to ensure that the identified findings and implications came from a credible basis.

We next discuss the potential threats to the validity of this systematic literature review, in accordance with the following taxonomies: *construct validity*, *internal validity*, *external validity*, and *reliability* [42].

- *Construct validity* concerns establishing a relationship between the theory behind the study and the observations. It covers issues that are related to the design of the study, to analyze its ability to represent what reseachers have in mind, and what is investigated according to the research questions. To avoid threats to construct validity, we applied the concepts of "testing in SPL" and "systematic literature review" as the main constructs. In the former, we leveraged the key characteristics in the studies, following the division of interests. As for the second construct, we followed the guidelines to design the research questions, search and assessment criteria. Another important aspect is to ensure the discovering of all relevant studies in the topic under investigation. For this purpose, we combined automated and manual searches, to increase the coverage.
- *Internal validity* concerns establishing a causal relationship, whereby certain conditions are shown to lead to other conditions. As threats to the internal validity we can consider the subjective decisions that might have occurred during primary studies selection and data extraction, and individual bias in assessment. The decision process for including a study into the analysis may be considered the most important influential factor on the resulting conclusions. Given that some primary studies did not provide a clear description or proper objectives and results, it was difficult the objective application of the eligibility criteria or the impartial data extraction. The strategy to minimize selection and extraction mistakes was to consider several stages in the review in order to incorporate the most complete primary studies possible to increase reliability of the conclusions. Besides, to minimize threats regarding data analysis, that arise from individual researchers bias when assessing her assigned primary studies, we followed a pre-defined protocol, carrying out several dry runs individually, and consolidating the differences collaboratively. However, it is possible that some papers that do not contribute to the understanding of the study goals are included, as it is possible that some excluded papers might present useful characteristics that might affect the review conclusions.
- *External validity* concerns establishing the extent to which results of the studies provide a correct basis for generalization to other scenarios and application domains. Most of the papers included in this investigation refers to approaches that have not been used in industry. Even most of the evaluation and assessment performed on the approaches do not refer to real-world practice. This prevents us from asserting that the classification provided in this analysis is fully generalizable.
- *Reliability* is concerned with issues that affect the ability to draw that the operations of a study can be repeated with the same results. To attain this goal, we defined the search strings and procedures in such a way that other researchers could directly and objectively replicate this study. However, we cannot

guarantee that other researchers could achieve the exact same outcomes, as subjectivity is a major criticism levelled at primary studies analysis. However, we believe that the underlying trends should remain unchanged.

5. Related work

Early in 2003, Kolb and Muthig [43] published one of the initial analysis of existing SPL testing practices, and pointed out a set of directions for further improvement. Although the study was not organized as a literature survey, the authors discussed about the use of techniques that were commonly used to test a SPL, and also highlighted the common problems of applying techniques from traditional software development in the SPL scenario. Despite the importance of such a study to the field, since its publication, much more investigation has been carried out.

One year later, Tevanlinna et al. [44] published a survey on product family testing. In the light of available studies at the time of publication, the authors provided research community with an overview on the established practices and the challenges surrounding the SPL testing field. Authors focused on discussing methods developed for or that could be applied to test product families, while disregarding particular characteristics of a given method. The paper served for a long time as a good roadmap for researchers intended to investigate the field.

Johansen et al. [45] presented a survey of product line testing, focusing on the investigation of strategies employed towards developing test suites for a SPL. They followed a formalized literature review process [46]. After analyzing existing publications, the authors focused on the analysis of three studies that contained empirical evaluations on their data. The authors claimed that the reason to only include and analyze empirically assessed studies was to perform a more reliable assessment of SPL testing practices. However, the small amount of studies prevents the generalization of research findings.

There are some other related research we could include in this section, such as a series of literature reviews on the topic [18–20]. Such studies provide state-of-the-art evidence on the SPL testing field. In a systematic way, these surveyed existing research trying to identify useful approaches, and synthesize the achievements, identify gaps, and propose research directions, based on studies published up to the year 2009. They complement each other well in terms of research questions addressed.

As a next step, in this present study, we go further, and analyze the current research on strategies for handling testing in SPL, by assessing provided evidence about current research regarding how far it can convince practitioners, and also try to identify open problems and areas for improvement.

6. Concluding remarks

The goal of SPL engineering is to optimize effectiveness and efficiency by capitalizing on the commonality and managing the variation that exists between multiple software systems. In order to achieve the benefits of a software product line, testing of the assets that will compose the products is a critical activity. Given that an asset can be reused by multiple products, it is not economically meaningful to test every interaction between assets in every product instance derived from the product line. A new paradigm demands new strategies that result in new improvements.

This paper reports on the results of a systematic literature review of testing in software product line engineering. It aims at understanding how products are selected from the very large set of possible products for asset testing, and how each selected product is tested. Twenty-four papers were found to provide the material for the discussion surrounding the first research question, while twenty-seven papers described strategies that matched the second.

Despite an observed increasing interest in the SPL testing topic, the study led us to claim the need for more effective methods and techniques for testing SPL, issue stated a decade ago by Kolb and Muthig [43], that still holds true as probably the main rationale for current research. Given the current available evidence, we noted that research has advanced in terms of strategies to handle testing each selected product, but it is still in its initial stages when considering strategies to cope with the selection of representative products. Additionally, in either topic, we observed a lack of generalization of existing techniques, which demands further investigation.

Based on the findings of this systematic literature review we suggest that research be undertaken to expand on the strategies that have been investigated. Further research into this topic should include some form of empirical assessment of existing strategies, as a means to improve their accuracy, and enable generalizations. This may entail investigating more than just small usage scenarios, but rather large-scale and industry-side scenarios.

Acknowledgements

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES)[10], funded by CNPq and FACEPE, Grants 573964/2008- 4 and APQ-1037-1.03/08 and CNPq Grants 305968/2010-6, 559997/2010-8, 474766/2010-1 and FAPESB.

Appendix A. Venues manually searched

See Table A.10.

Table A.10
Venues subject to manual search.

Journals
ACM TOSEM – Transactions on Software Engineering and Methodology
ASE – Automated Software Engineering
IEEE SW – Software
IEEE TSE – Transactions on Software Engineering
IET SW – Software
IST – Information & Software Technology
JSEP – Software: Evolution and Process
JSS – Systems & Software
SPE – Software: Practice and Experience
SQJ – Software Quality Journal
STVR – Software Testing, Verification & Reliability
Conferences
AOSD – Aspect-Oriented Software Development
ASE – Automated Software Engineering
CSMR – Software Maintenance and Reengineering
ENASE – Evaluation of Novel Approaches to Software Engineering
FASE – Fundamental Approaches to Software Engineering
ICSE – Software Engineering
ICSM – Software Maintenance
ICSR – Software Reuse
ICST – Software Testing
ICTSS – Testing Software and Systems
ISSRE – Software Reliability Engineering
ISSTA – Software Testing and Analysis
MODELS – Model-Driven Engineering and Software Development
SEFM – Software Engineering and Formal Methods
SPLC – Software Product Line Conference
QSIC – Quality Software
Workshops
A-MOST – Advances in Model Based Testing
AST – Automation of Software Test
FOSD – Feature Oriented Software Development
PLEASE – Product Line Approaches in Software Engineering
SPLiT – Software Product Lines Testing
VaMoS – Variability Modelling of Software-intensive Systems

[10] INES - http://www.ines.org.br.

Table B.11
Selected primary studies.

ID	Title	Author(s)	Venue
P01	Testing software assets of framework-based product families during application engineering stage	J. Al-Dallal, P.G. Sorenson	JSW 3 (5): 11–25, 2008
P02	On extracting tests from a testable model in the context of domain engineering	S. Bashardoust-Tajali, J.-P. Corriveau	ICECCS'08: 98–107
P03	Product line use cases: Scenario-based specification and testing of requirements	A. Bertolino, A. Fantechi, S. Gnesi, G. Lami	Book Chapter: 425–445, 2006
P04	Towards generating acceptance tests for product lines	B. Geppert, J.J. Li, F. Roessler, D.M. Weiss	ICSR'04: 35–48
P05	An approach for selecting software product line instances for testing	T. Gustafsson	SPLC'07: 81–86
P06	Specification-based testing for software product lines	T. Kahsai, M. Roggenbach, B.-H. Schlingloff	SEFM'08: 149–158
P07	Testing variabilities in use case models	E. Kamsties, K. Pohl, S. Reis, A. Reuys	PFE'03: 6–18
P08	Reuse execution traces to reduce testing of product lines	J.J. Li, B. Geppert, F. Roessler, D. Weiss	SPLiT'07: 1–8
P09	A reuse technique for performance testing of software product lines	A.P.K. Reis, S. Metzger	SPLiT'06: 5–10
P10	Specification based software product line testing: A case study	S. Mishra	CS&P'06: 243–254
P11	System testing of product lines: From requirements to test cases	C. Nebut, Y. Traon, J.-M. Jézéquel	Book Chapter: 447–477, 2006
P12	Model-based testing for applications derived from software product lines	E.M. Olimpiew, H. Gomaa	A-MOST'05: 1–7
P13	Customizable requirements-based test models for software product lines	E.M. Olimpiew, H. Gomaa	SPLiT'06: 17–22
P14	Reusable model-based testing	E. M. Olimpiew, H. Gomaa	ICSR'09: 76–85
P15	Towards software product line testing using story driven modeling	S. Oster, A. Schürr, I. Weisemöller	T.Report: 48–51, 2008
P16	Production-testing of embedded systems with aspects	J. Pesonen, M. Katara, T. Mikkonen	HVC'05: 90–102
P17	The SCENTED method for testing software product lines	A. Reuys, S. Reis, E. Kamsties, K. Pohl	Book Chapter: 479–520, 2006
P18	Optimizing the selection of representative configurations in verification of evolving product lines of distributed embedded systems	K. Scheidemann	SPLC'06: 75–84
P19	Improving the testing and testability of software product lines	I. Cabral, M. B. Cohen, G. Rothermel	SPLC'10: 241–255
P20	Model-based coverage-driven test suite generation for software product lines	H. Cichos, S. Oster, M. Lochau, A. Schürr	MODELS'11: 425–439
P21	Goal-oriented test case selection and prioritization for product line feature models	A. Ensan et al.	ITNG'11: 291–298
P22	Linking feature models to code artifacts using executable acceptance tests	Y. Ghanam, F. Maurer	SPLC'10: 211–225
P23	PACOGEN: Automatic generation of pairwise test configurations from feature models	A. Hervieu, B. Baudry, A. Gotlieb	ISSRE'11: 120 –129
P24	Properties of realistic feature models make combinatorial testing of product lines feasible	M. F. Johansen, Ø. Haugen, F. Fleurey	MODELS'11: 638–652
P25	Reducing combinatorics in testing product lines	C.H.P. Kim, D.S. Batory, S. Khurshid	AOSD'11: 57–68
P26	Testing product generation in software product lines using pairwise for features coverage	B.P. Lamancha, M.P. Usaola	ICTSS'10: 111–125
P27	A model based testing approach for model-driven development and software product lines	B.P. Lamancha, M.P. Usaola, M.P. Velthius	ENASE'10: 193–208
P28	Model-based pairwise testing for feature interaction coverage in software product line engineering	M. Lochau, S. Oster, U. Goltz, A. Schürr	SQJ 20(3): 567–604, 2012
P29	Automated incremental pairwise testing of software product lines	S. Oster, F. Markert, P. Ritter	SPLC'10: 196–210
P30	Pairwise feature-interaction testing for SPLs: potentials and limitations	S. Oster, M. Lochau, M. Zink, M. Grechanik	SPLC'11: 1–8
P31	Automated and scalable *t-wise* test case generation strategies for software product lines	G. Perrouin et al.	ICST'10: 459–468
P32	Modelling requirements to support testing of product lines	C. Robinson-Mallett et al.	A-MOST'10: 11–18
P33	Integration testing of software product lines using compositional symbolic execution	J. Shi, M. Cohen, M. Dwyer	FASE'12: 270–284
P34	A regression testing approach for software product lines architectures	P.A.M.S. Neto et al.	SBCARS'10: 41–50
P35	Avoiding redundant testing in application engineering	V. Stricker, A. Metzger, K. Pohl	SPLC'10: 226–240
P36	Incremental test generation for software product lines	E. Uzuncaova, S. Khurshid, D. Batory	TSE 36 (3): 309–322, 2010
P37	Model-Driven Software Product Line Testing: An Integrated Approach	A. Schürr, S. Oster, F. Markert	SOFSEM'10: 112–131
P38	Combinatorial Testing for Feature Models Using CitLab	A. Calvagna, A. Gargantini, P. Vavassori	IWCT'13: 338–347
P39	Continuous test suite augmentation in software product lines	Z. Xu, M.B. Cohen, W. Motycka, G. Rothermel	SPLC'13: 52–61
P40	Evolutionary Search-Based Test Generation for Software Product Line Feature Models	F. Ensan, E. Bagheri, and D. Gašević	CAiSE'12: 613–628
P41	Incremental Model-Based Testing of Delta-Oriented Software Product Lines	M. Lochau, I. Schaefer, J. Kamischke, S. Lity	TAP'12: 67–82
P42	Minimizing test suites in software product lines using weight-based genetic algorithms	S. Wang, S. Ali, A. Gotlieb	GECCO'13: 1493–1500
P43	Multi-objective test generation for software product lines	C. Henard et al.	SPLC'13: 62–71
P44	Practical pairwise testing for software product lines	D. Marijan, A. Gotlieb, S. Sen, A. Hervieu	SPLC'13: 227–235
P45	Shared Execution for Efficiently Testing Product Lines	C.H. Kim, S. Khurshid, D. Batory	ISSRE'12: 221–230
P46	SPLat: lightweight dynamic analysis for reducing combinatorics in testing configurable systems	C.H. Kim et al.	ESEC/FSE'13: 257–267
P47	Towards efficient SPL testing by variant reduction	M. Kowal, S. Schulze, I. Schaefer	VariComp'13: 1–6
P48	Requirements-based Delta-oriented SPL Testing	M. Dukaczewski et al.	PLEASE'13: 49–52
P49	Automated Test Case Selection Using Feature Model: An Industrial Case Study	S. Wang, A. Gotlieb, S. Ali, M. Liaaen	MODELS'13: 237–253

Appendix B. Primary studies

See Table B.11.

References

[1] P. Clements, L. Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley, Boston, MA, USA, 2001.
[2] P. Clements, J.D. McGregor, Better, faster, cheaper: Pick any three, Bus. Hor. 55 (2) (2012) 201–208.
[3] G. Perrouin, S. Oster, S. Sen, J. Klein, B. Baudry, Y. le Traon, Pairwise testing for software product lines: comparison of two approaches, Soft. Qual. J. (2011) 1–39.
[4] L. Chen, M.A. Babar, A systematic review of evaluation of variability management approaches in software product lines, Inform. Soft. Technol. 53 (4) (2011) 344–362.
[5] K. Lee, K.C. Kang, J. Lee, Concepts and guidelines of feature modeling for product line software engineering, in: Proceedings of the 7th International Conference on Software Reuse (ICSR), Springer-Verlag, 2002, pp. 62–77.
[6] D. Beuche, H. Papajewski, W.S. Preikschat, Variability management with feature models, Sci. Comp. Program. 53 (3) (2004) 333–352.
[7] M.F. Johansen, Ö. Haugen, F. Fleurey, Properties of realistic feature models make combinatorial testing of product lines feasible, in: 14th International Conference on Model Driven Engineering Languages and Systems (MODELS), Springer, 2011, pp. 638–652.
[8] I.C. Machado, J.D. McGregor, E.S. Almeida, Strategies for testing products in software product lines, ACM SIGSOFT Soft. Eng. Notes 37 (6) (2012) 1–8.
[9] K. Pohl, G. Böckle, F.J.v.d. Linden, Software Product Line Engineering: Foundations, Principles and Techniques, Springer-Verlag New York, Inc., 2005.
[10] 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: Proceedings of the Third International Conference on Software Testing, Verification and Validation (ICST), IEEE, 2010, pp. 459–468.
[11] K. Naik, P. Tripathy, Software Testing and Quality Assurance: Theory and Practice, John Wiley & Sons, Inc., 2008.
[12] D.S. Cruzes, T. Dybå, Research synthesis in software engineering: a tertiary study, Inform. Soft. Technol. 53 (5) (2011) 440–455.
[13] B. Kitchenham, P. Brereton, D. Budgen, M. Turner, J. Bailey, S.G. Linkman, Systematic literature reviews in software engineering – a systematic literature review, Inform. Softw. Technol. 51 (1) (2009) 7–15.
[14] T. Dybå, T. Dingsøyr, Strength of evidence in systematic reviews in software engineering, in: Proceedings of the Second International Symposium on Empirical Software Engineering and Measurement (ESEM), ACM, 2008, pp. 178–187.
[15] P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner, M. Khalil, Lessons from applying the systematic literature review process within the software engineering domain, J. Syst. Soft. 80 (4) (2007) 571–583.
[16] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.
[17] M. Petticrew, H. Roberts, Systematic Reviews in the Social Sciences: A practical guide, Blackwell Publishing, Oxford, 2006.
[18] P.A.M.S. Neto, I.C. Machado, J.D. McGregor, E.S. Almeida, S.R.L. Meira, A systematic mapping study of software product lines testing, Inform. Soft. Technol. 53 (5) (2011) 407–423.
[19] E. Engström, P. Runeson, Software product line testing – a systematic mapping study, Inform. Softw. Technol. 53 (1) (2011) 2–13.
[20] B.P. Lamancha, M.P. Usaola, M.P. Velthius, Software product line testing – a systematic review, in Proceedings of the 4th International Conference on Software and Data Technologies (ICSOFT), INSTICC Press, Sofia, Bulgaria, 2009, pp. 23–30.
[21] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), University of Bari, Bari, Italy, 2008.
[22] H. Zhang, M.A. Babar, P. Tell, Identifying relevant studies in software engineering, Inform. Soft. Technol. 53 (6) (2011) 625–637.
[23] B.A. Kitchenham, P. Brereton, M. Turner, M. Niazi, S.G. Linkman, R. Pretorius, D. Budgen, Refining the systematic literature review process – two participant-observer case studies, Emp. Soft. Eng. 15 (6) (2010) 618–653.
[24] J. Webster, R.T. Watson, Analyzing the past to prepare for the future: writing a literature review, MIS Quart. 26 (2) (2002) xiii–xxiii.
[25] M. Shaw, What makes good research in software engineering?, Int J. Soft. Tools Technol. Transf. 4 (1) (2002) 1–7.
[26] V. Alves, N. Niu, C. Alves, G. Valena, Requirements engineering for software product lines: a systematic literature review, Inform. Soft. Technol. 52 (8) (2010) 806–820.
[27] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inform. Soft. Technol. 50 (9–10) (2008) 833–859.
[28] F. Dordowsky, R. Bridges, H. Tschope, Implementing a software product line for a complex avionics system, in: Proceedings of the 15th International Conference on Software Product Lines (SPLC)., ACM, Munich, Germany, 2011, pp. 241–250.
[29] C. Tischer, A. Muller, T. Mandl, R. Krause, Experiences from a large scale software product line merger in the automotive domain, in: Proceedings of the 15th International Conference on Software Product Lines (SPLC), ACM, Munich, Germany, 2011, pp. 267–276.
[30] A. Hervieu, B. Baudry, A. Gotlieb, PACOGEN: automatic generation of pairwise test configurations from feature models, in: 22nd IEEE International Symposium on Software Reliability Engineering (ISSRE), IEEE Computer Society, Hiroshima, Japan, 2011, pp. 120–129.
[31] I. Schaefer, L. Bettini, F. Damiani, N. Tanzarella, Delta-oriented programming of software product lines, in: Proceedings of the 14th International Conference on Software Product Lines (SPLC), Springer-Verlag, Berlin, Heidelberg, 2010, pp. 77–91.
[32] I.F. Aguillo, Is google scholar useful for bibliometrics? a webometric analysis, Scientometrics 91 (2) (2012) 343–351.
[33] A.-W. Harzing, A preliminary test of google scholar as a source for citation data: a longitudinal study of nobel prize winners, Scientometrics 94 (3) (2013) 1057–1075.
[34] C. Thörn, A quality model for evaluating feature models, in: 11th International Conference on Software Product Lines, SPLC, vol. 2 (Workshops), Kyoto, Japan, 2007, pp. 184–190.
[35] R. Pohl, K. Lauenroth, K. Pohl, A performance comparison of contemporary algorithmic approaches for automated analysis operations on feature models, in: Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE Computer Society, Washington, DC, USA, 2011, pp. 313–322.
[36] L. Rincón, G. Giraldo, R. Mazo, C. Salinesi, An ontological rule-based approach for analyzing dead and false optional features in feature models, Electron. Notes Theoret. Comp. Sci. 302 (2014) 111–132.
[37] D. Benavides, S. Segura, A. Ruiz-Cortés, Automated analysis of feature models 20 years later: a literature review, Inform. Syst. 35 (6) (2010) 615–636.
[38] C. Salinesi, R. Mazo, Software Product Line – Advanced Topic, InTech, April 2012, ch. Defects in Product Line Models and How to Identify Them, pp. 97–122.
[39] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, E. Merlo, Recovering traceability links between code and documentation, IEEE Trans. Soft. Eng. 28 (10) (2002) 970–983.
[40] L. Linsbauer, E.R. Lopez-Herrejon, A. Egyed, Recovering traceability between features and code in product variants, in: Proceedings of the 17th International Software Product Line Conference (SPLC), ACM, 2013, pp. 131–140.
[41] I.C. Machado, A.R. Santos, Y.a.C. Cavalcanti, E.G. Trzan, M.M.a. Souza, E.S. Almeida, Low-level variability support for web-based software product lines, in: Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS), ACM, 2014, pp. 15:1–15:8.
[42] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, Experimentation in Software Engineering, Springer, 2012.
[43] R. Kolb, D. Muthig, Challenges in testing software product lines, in: Proceedings of the 7th Conference on Quality Engineering in Software Technology (CONQUEST), Fraunhofer Publica, 2003, pp. 81–95.
[44] A. Tevanlinna, J. Taina, R. Kauppinen, Product family testing: a survey, ACM SIGSOFT Soft. Eng. Notes 29 (2) (2004) 12.
[45] M. Johansen, O. Haugen, F. Fleurey, A survey of empirics of strategies for software product line testing, in: IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, IEEE Computer Society Press, Berlin, Germany, 2011, pp. 266–269.
[46] T. Dyba, B.A. Kitchenham, M. Jorgensen, Evidence-based software engineering for practitioners, IEEE Soft. 22 (1) (2005) 58–65.