Information and Software Technology 53 (2011) 2–13

Contents lists available at ScienceDirect

Information and Software Technology

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

Software product line testing – A systematic mapping study

Emelie Engström [*], Per Runeson [1]

Department of Computer Science, Lund University, SE-221 00 Lund, Sweden

ARTICLE INFO

Article history:
Received 5 October 2009
Received in revised form 14 April 2010
Accepted 26 May 2010
Available online 29 September 2010

Keywords:
Software product line testing
Testing
Systematic mapping
Systematic literature review

ABSTRACT

Context: Software product lines (SPL) are used in industry to achieve more efficient software development. However, the testing side of SPL is underdeveloped.
Objective: This study aims at surveying existing research on SPL testing in order to identify useful approaches and needs for future research.
Method: A systematic mapping study is launched to find as much literature as possible, and the 64 papers found are classified with respect to focus, research type and contribution type.
Results: A majority of the papers are of proposal research types (64%). System testing is the largest group with respect to research focus (40%), followed by management (23%). Method contributions are in majority.
Conclusions: More validation and evaluation research is needed to provide a better foundation for SPL testing.

Contents

* Corresponding author. Tel.: +46 46 222 88 99.
E-mail addresses: emelie.engstrom@cs.lth.se (E. Engström), per.runeson@cs.lth.se (P. Runeson).
[1] Tel.: +46 46 222 93 25.

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.05.011

1. Introduction

Efficient testing strategies are important for any organization with a large share of their costs in software development. In an organization using software product lines (SPL) it is even more crucial since the share of testing costs increases as the development costs for each product decreases. Testing of a software product line is a complex and costly task since the variety of products derived from the product platform is huge. In addition to the complexity of stand-alone product testing, product line testing also includes the dimension of what should be tested in the platform and what should be tested in separate products.

Early literature on product lines did not spend much attention to testing [7] (pp. 278–279), but the issue is brought up after that, and much research effort is spent on a variety of topics related to product line testing. In order to get a picture of existing research we launched a systematic mapping study of product line testing. The aim is to get an overview of existing research in order to find useful results for practical use and to identify needs for future research. We provide a map over the existing research on software product line testing. Overviews of challenges and techniques are included in several earlier papers, as well as a couple of brief reviews. However no extensive mapping study has been reported on earlier.

Systematic mapping is a relatively new research method in software engineering, adapted from other disciplines by Kitchenham [31]. It is an alternative to systematic reviews and could be used if the amount of empirical evidence is too little, or if the topic is too broad, for a systematic review to be feasible. A mapping study is performed at a higher granularity level with the aim to identify research gaps and clusters of evidence in order to direct future research. Some reports on systematic mapping studies are published e.g. on object-oriented software design [3] and on non-functional search-based software testing [1]. Petersen et al. [58] describe how to conduct a systematic mapping study in software engineering. Our study is conducted in accordance with these guidelines. Where applicable, we have used the proposed classification schemes and in addition, we have introduced a scheme specific to our topic.

This paper is organized as follows: Section 2 describes how the systematic mapping methodology has been applied. Section 3 summarizes challenges discussed in literature in response to our first research question. In section 4 we compile statistics on the primary studies to investigate the second research question. Section 5 presents the classification schemes used and in Section 6 the actual mapping of the studies, according to research questions three and four, is presented together with a brief summary of the research. Finally, discussion and conclusions are provided in Sections 7 and 8, respectively.

2. Research method

2.1. Research questions

The goal of this study is to get an overview of existing research on product line testing. The overall goal is defined in four research questions:

RQ1 Which challenges for testing software product lines have been identified? Challenges for SPL testing may be identified in specific surveys, or as a bi-product of other studies. We want to get an overview of the challenges identified to validate the relevance of past and future research.

RQ2 In which fora is research on software product line testing published? There are a few conferences and workshops specifically devoted to SPL. However, experience from earlier reviews indicates that research may be published in very different for a [15].

RQ3 Which topics for testing product lines have been investigated and to what extent? As SPL is related to many different aspects, e.g. technical, engineering, managerial, we want to see which ones are addressed in previous research, to help identifying needs for complementary research.

RQ4 What types of research are represented and to what extent? Investigations on types of research in software indicate that the use of empirical studies is scarce in software engineering [21]. Better founded approaches are advised to increase the credibility of the research [69] and we want to investigate the status for the specific subfield of SPL testing.

2.2. Systematic mapping

In order to get an overview of the research on SPL testing, a systematic mapping study is carried through. A detailed description on how to conduct systematic mapping studies, and a discussion of differences between systematic mapping and systematic reviews, is presented by Petersen et al. [58]. The mapping process consists of three activities: (i) search for relevant publications, (ii) definition of a classification scheme, and (iii) mapping of publications.

In this study, search for publications is done in five steps of which the two last steps validate the search, see Fig. 1, using a combination of data base searches and reference based searches [67]. In the first step an initial set of papers was identified through exploratory searches, mainly by following references and links to citing publications, with some previous known publications as the starting point [42,72,47,60,52,59]. The result of this activity was 24 publications, which were screened in order to retrieve an overview of the area; frequently discussed challenges, commonly used classifications and important keywords.

The second step consisted in reading introduction sections and related works sections in the initial set of publications and extending the set with referenced publications relevant to this study. Only papers with a clear focus on the testing of a software product line published up to 2008 were included. This resulted in additional 33 publications. In order to avoid redundancy in research contributions and to establish a quality level of included publications we decided however to narrow down the categories of publications after this stage. Non-peer reviewed publications; such as technical reports, books and workshop descriptions, in total 23 publications, were excluded from the set of primary studies. Among those is an early technical report by McGregor [42] (cited in 70% of the publications) which is used to find relevant primary studies, but not included among the primary studies as such. Another result of this step was a summary of challenges in SPL testing identified by the community and a preliminary classification scheme for research contributions.

In the third step we screened titles in proceedings from the most frequent publication forum from the previous steps; the workshop on Software Product Line Testing (SPLiT), and from the corresponding main conference; the Software Product Line Conference (SPLC). The number of primary studies is 53 after this step.

The fourth and fifth steps are validating the first three. The fourth step includes automatic searches with Google Scholar and ISI Web of science. The search string was "product" and "line/lines/family/families" and "test/testing" and it was applied only to titles, which has shown to be sufficient in systematic reviews [12]. This search resulted in 177 hits in Google Scholar and 38 hits in ISI Web of science. The search in web of science did not result in any new unique contribution.

Excluded publications were, except for the above mentioned, tool demonstrations, talks, non-english publications, patent applications, editorials, posters, panel summaries, keynotes and papers from industrial conferences. In total 49 publications were relevant

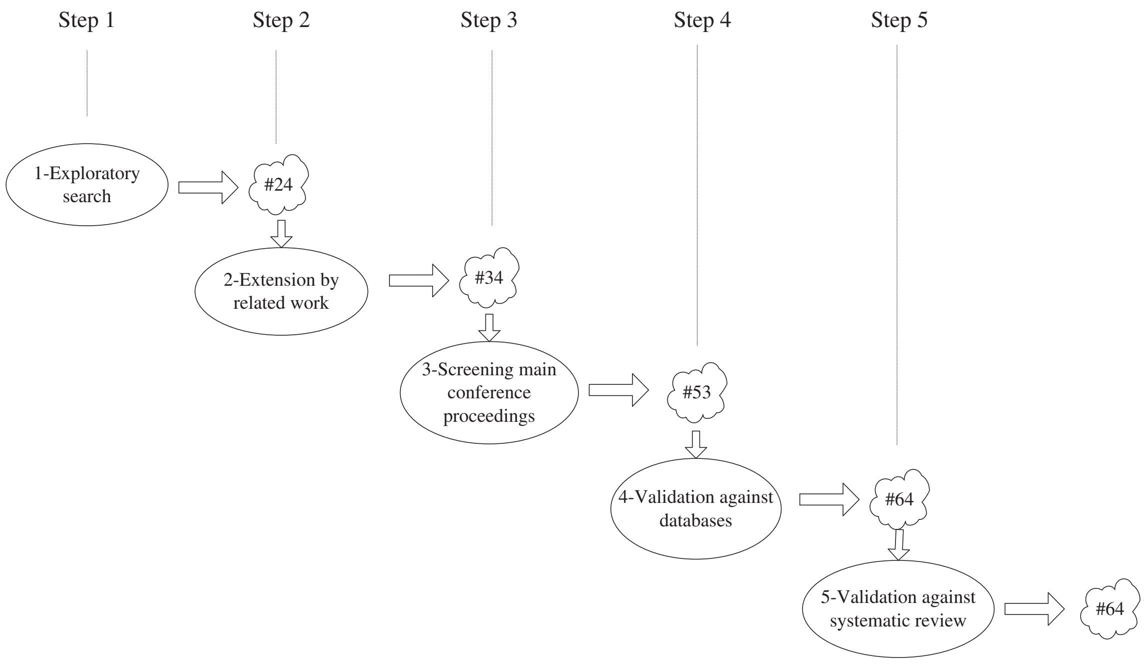

Step 1
Step 2
Step 3
Step 4
Step 5
1-Exploratory search
#24
2-Extension by related work
#34
3-Screening main conference proceedings
#53
4-Validation against databases
#64
5-Validation against systematic review
#64

Fig. 1. Search for publications on software product line testing.

for this study according to our selection criteria. This set was compared to our set of 53 papers from step three and 38 papers were common. The differing 11 publications were added to the study. In the fifth step the set of papers was compared to a set of paper included in a systematic review on product line testing by Lamancha et al. [38]. Their study included 23 papers of which 12 passed our criteria on focus and publication type. All of these were already included in our study. Thus we believe that the search for publications is sufficiently extensive and that the set of publications gives a good picture of the state of art in SPL testing research.

A summary of the inclusion and exclusion criteria is:

- Inclusion: Peer reviewed publications with a clear focus on some aspect of software product line testing.
- Exclusion: Publications where either testing focus or software product line focus is lacking. Non-peer reviewed publications.

The answer to RQ1 was retrieved through synthesising the discussions in the initial 24 publications until saturation was reached. Several publications are philosophical with a main purpose to discuss challenges in SPL testing and almost all papers discuss the challenges to some extent in the introductory sections. All challenges mentioned were named and grouped. A summary of the challenges is provided in Section 3. Answers to questions RQ2, RQ3 and RQ4 are retrieved through analysing the 64 primary studies. A preliminary classification scheme was established through *keywording* [58] abstracts and positioning sections. Classifications of the primary studies were conducted by the first author and validated by the second. Disagreements were resolved through discussions or led to refinement of the classification scheme, which in turn led to reclassification and revalidation of previously classified publications. This procedure was repeated until no disagreements remained.

2.3. Threats to validity

Threats to the validity of the mapping study are analyzed according to the following taxonomy: construct validity, reliability, internal validity and external validity.

Construct validity reflects to what extent the phenomenon under study really represents what the researchers have in mind and what is investigated according to the research questions. The terms product lines, software product lines and family/families are rather well established, and hence the terms are sufficiently stable to use as search strings. Similarly for testing, we consider this being well established. Another aspect of the construct validity is assurance that we actually find all papers on the selected topic. We have searched broadly in general publication databases which index most well reputed publication fora. The long list of different publication fora indicates the width of the searching is enough. The snowball sampling procedure has been shown to work well in searching with a specific technical focus [67]. We also validated our searches against another review, and found this review covering all papers in that review.

Reliability focuses on whether the data are collected and the analysis is conducted in a way that it can be repeated by other researchers with the same results. We defined search terms and applied procedures, which may be replicated by others. The non-determinism of one of the databases (Google scholar) is compensated by also using a more transparent database (ISI Web of Science). Since this is a mapping study, and no systematic review, the inclusion/exclusion criteria are only related to whether the topic of SPL testing is present in the paper or not. The classification is another source of threats to the reliability. Other researchers may possibly come up with different classification schemes, finer or more course grained. However, the consistency of the classification is ensured by having the classifications conducted by the first author and validated by the second.

Internal validity is concerned with the analysis of the data. Since the analysis only uses descriptive statistics, the threats are minimal. Finally, *external validity* is about generalization from this study. Since we do not draw any conclusions about mapping studies in general, but only on this specific one, the external validity threats are not applicable.

3. Challenges in testing a software product line

Software product line engineering is a development paradigm based on common software platforms, which are customized in order to form specific products [59]. A *software platform* is a set of *generic components* that form a common structure, from which a set of derivative products can be developed [46]. The process of developing the platform is named *domain engineering*, and the process of deriving specific products from the platform is named *application engineering* [59]. We refer to domain testing and application testing, accordingly. The variable characteristics of the platform, are referred to as *variability*; the specific representations of the variability in software artifacts are called *variation points*, while the representation of a particular instance of a variable characteristic is called a *variant* [59].

A number of challenges regarding testing of software product lines have been identified and discussed in the literature, which are identified in this mapping study (RQ1). They can be summarized in three main challenges concerning: (i) how to handle the large number of tests, (ii) how to balance effort for reusable components and concrete products, and (iii) how to handle variability.

3.1. Large number of tests

A major challenge with testing a software product line regards the large number of required tests. In order to fully test a product line, all possible uses of each generic component, and preferably even all possible product variants, need to be tested. The fact that the number of possible product variants grows exponentially with the number of variation points, makes such thorough testing infeasible. Since the number of products actually developed also increases, there is an increased need for system tests as well.

The main issue here is how to reduce redundant testing and to minimize the testing effort through reuse of test artefacts. The close relationship between the developed products and the fact that they are derived from the same specifications indicates an option to reduce the number of tests, due to redundancy. A well defined product line also includes a possibility to define and reuse test artefacts.

3.2. Reusable components and concrete products

The second major challenge, which of course is closely related to the previous, is how to balance effort spent on reusable components and product variants. Which components should be tested in domain (platform) engineering, and which should be tested in application (product) engineering? [59] A high level of quality is required for the reusable components but still it is not obvious how much the testing of reusable components may help reducing testing obligations for each product. There is also a question of how to test generic components, in which order and in how many possible variants. The planning of the testing activities is also further complicated by the fact that software process is split and testing may be distributed across different parts of the organizations.

3.3. Variability

Variability is an important concept in software product line engineering, and it introduces a number of new challenges to testing. Variability is expressed as variation points on different levels with different types of interdependencies. This raises a question of how different types of variation points should be tested. A new goal for testing is also introduced in the context of variability: the verification of the absence of incorrect bindings of variation points. We have to be sure that features not supposed to be there are not included in the end product. The binding of variation points is also important. Complete integration and system test are not feasible until the variation points are bound. It is also possible to realize the same functionality in different ways and thus a common function in different products may require different tests.

4. Primary studies

Following the method defined in Section 2.2, we ended up in 64 peer reviewed papers, published in workshops, conferences, journals and in edited books (RQ2). The papers are published between 2001 and 2008, and summarized by publication fora in Table 1.

Tables 2 and 3, the distribution over time is reported for the 64 primary studies. Note that one paper spans two research foci according to our classification scheme. Hence the total number of classification items in Table 2 is 65.

5. Classification schemes

Publications are classified into categories in three different dimensions: *research focus, type of contribution and research type*. This structure is presented by Petersen et al. [58]. However the different categories are adapted to this particular study. Establishing the scheme and mapping publications was done iteratively as new primary studies were added. When the scheme was finally set, all classifications were reviewed again.

Six categories of research focus (RQ3) were identified through the keyword method described by Petersen et al. [58]: (i) test organization and process, (ii) test management, (iii) testability, (iv) system and acceptance testing (ST and AT), (v) integration testing (IT), (vi) unit testing (UT), and (vii) automation. *Test organization and process* includes publications with a focus on the testing framework, seeking answers to how the testing activities and test assets should be mapped to the overall product line development and also how product line testing should be organized overall. Papers on product line testing in general are also mapped into this category. *Test management* includes test planning and assessment, fault prediction, selection of test strategies, estimates of the extent of testing and test coverage. Papers on how to distribute resources (between domain engineering process and application engineering process, between different test activities, and between different products) are included as well. *Testability* includes papers with a focus on other aspects of product line engineering rather than the testing, but still with the goal of improved testing. The test levels used in the classification are *system and acceptance testing, integration testing,* and *unit testing.* Paper topics cover both design of new test cases and selection of already existing test cases. Test cases could be designed from requirements or from generic test assets. Some papers focus on the *automation* of testing.

Contribution type is classified into five categories: *Tool, Method, Model, Metric,* and *Open Items*. *Tools* refer to any kind of tool support for SPL testing, mostly in the form of research prototypes. *Methods* include descriptions of how to perform SPL testing, both as general concepts and more specific and detailed working procedures. *Models* are representations of information to be used in SPL testing. *Metrics* focus on what to measure to characterize certain properties of SPL testing. Finally, *open items* are identified issues that need to be addressed.

The classification of research types (RQ4) is based on a scheme proposed by Wieringa et al. [78]. Research is classified into six categories: (i) *validation research, (ii) evaluation research, (iii) solution proposals, (iv) conceptual proposals, (v) opinion papers,* and (vi) *experience papers. Validation research* focuses on investigating a proposed solution which has not yet been implemented in practice. Investigations are carried out systematically and include: experiments, simulation, prototyping, mathematical systematically analysis, mathematical proof of properties, etc. *Evaluation research* evaluates a problem or an implemented solution in practice and includes case studies, field studies, field experiments, etc. A *Solution proposal* is a novel or significant extension to an existing technique. Its benefits are exemplified and/or argued for. A *Conceptual pro-*

Table 1
Distribution of publication fora.

Publication Fora	Type	#
International Workshop on Software Product Line Testing (SPLiT)	Workshop	23
International Workshop on Software Product-family Engineering (PFE)	Workshop	3
Software Product Lines – Research Issues in Engineering and Management	Book chapter	3
Software Product Line Conference (SPLC)	Conference	2
ACM SIGSOFT Software Engineering Notes	Journal	1
Communications of the ACM	Journal	1
Concurrency: Specification and Programming Workshop	Workshop	1
Conference on Composition-Based Software Systems	Conference	1
Conference on Quality Engineering in Software Technology (CONQUEST)	Industry Conference	1
Development of Component-based Information Systems	Book chapter	1
European Conference on Information Systems, Information Systems in a Rapidly Changing Economy, (ECIS)	Conference	1
European Workshop on Model Driven Architecture with Emphasis on Industrial Application	Workshop	1
Fujaba days	Workshop	1
Fundamental Approaches to Software Engineering (FASE)	Conference	1
Hauptkonferenz Net. ObjectDays	Industry Conference	1
International Computer Software and Applications Conference	Conference	1
International Conference on Advanced Information Systems (CAiSE)	Conference	1
International Conference on Automated Software Engineering (ASE)	Conference	1
International Conference on Computer and Information Technology (ICCIT)	Conference	1
International Conference on Engineering of Complex Computer Systems (ICECCS)	Conference	1
International Conference on Software Engineering and Formal Methods (SEFM)	Conference	1
International Conference on Software Reuse (ICSR)	Conference	1
International Symposium on Computer Science and Computational Technology (ISCSCT)	Conference	1
International Symposium on Empirical Software Engineering (ISESE)	Conference	1
International Symposium on Software Reliability Engineering (ISSRE)	Conference	1
International Symposium on Software Testing and Analysis (ISSTA)	Conference	1
International Workshop on Requirements Engineering for Product Lines (REPL)	Workshop	1
International Workshop on Software Product-Family Engineering (PFE)	Workshop	1
International Workshop on Product Line Engineering The Early Steps: Planning, Modeling, and Managing (PLEES)	Workshop	1
International Workshop on Software Product Lines	Workshop	1
International Workshop on Test and Analysis of Component-Based Systems (TaCOS)	Workshop	1
Journal of Software	Journal	1
Nordic Workshop on Programming and Software Development Tools and Techniques (NWPER)	Workshop	1
The European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)	Conference	1
The Role of Software Architecture for Testing and Analysis (ROSATEA)	Workshop	1
Workshop on Advances in Model-Based Testing (A-MOST)	Workshop	1
Workshop on Model-based Testing in Practice	Workshop	1
Total		64

Table 2
Distribution over research focus.

Research focus	2001	2002	2003	2004	2005	2006	2007	2008	Total
Test organization and process	1	1	1	2	1	1	1	2	10
Test management			2	3	1	3	2	4	15
Testability				1		1			2
System and acceptance testing		1	4	4	3	7	2	5	26
Integration testing				1		1	2		4
Unit testing			2				1		3
Automation				4	1				5
Total	1	2	9	15	6	13	8	11	65

Table 3
Distribution over publication types.

Type of publication	2001	2002	2003	2004	2005	2006	2007	2008	Total	
Book chapter						4			4	6%
Conference paper			4	1	2	3	4	5	19	30%
Journal paper				1		1		1	3	5%
Workshop paper	1	2	5	13	4	4	4	5	38	59%
Total	1	2	9	15	6	12	8	11	64	100%

posal sketches a new way of looking at things, but without the preciseness of a solution proposal. *Opinion papers* report on the authorś opinions on what is good or bad. *Experience papers* report on personal experiences from one or more real life projects. Lessons learned are included but there is no systematic reporting of research methodology.

6. Mapping

Fig. 2 shows a map over existing research foci related to software product line testing, distributed over type of research and type of contribution. The number of publications on each side differs, since some publications provide multiple contributions e.g. both a model and a method. Most research effort is spent on system testing with contributions such as proposed methods for test case design, sketched out in detail but not yet evaluated, i.e. solution proposals. An overview of research presented by focus is given in Sections 6.1.1–6.1.7.

6.1. Research focus

Fig. 3 shows the distribution of research foci. A paper is assigned to several foci if it has a clear contribution to more than one area. Each of the focus areas is discussed below.

6.1.1. Test organization and process

Table 4 lists all papers on test organization and process. McGregor points out the need for a well designed test process, and discusses the complex relationships between platforms, products and different versions of both platforms and products in his technical report [42]. He argues there and elsewhere [41] for a structure of test assets and documentation in alignment with the structure of the constructed products. This is further concretized by Knauber and Hetrick [32]. Kolb and Muthig [35,37] discuss the importance and complexity of testing a software product line and component-based systems. They pinpoint the need for guidelines and comprehensive and efficient techniques for systematically testing product lines. They also promote the idea of creating generic test cases.

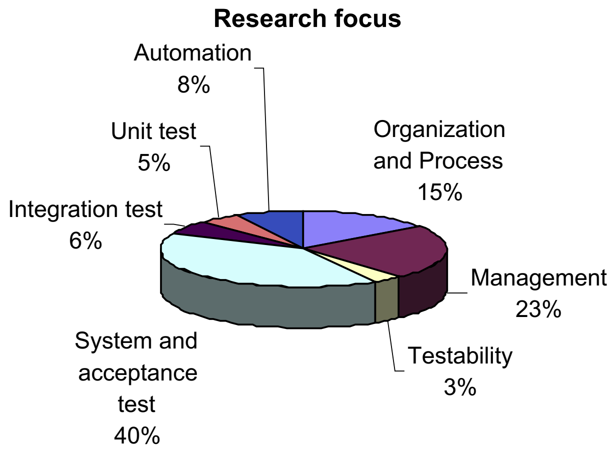
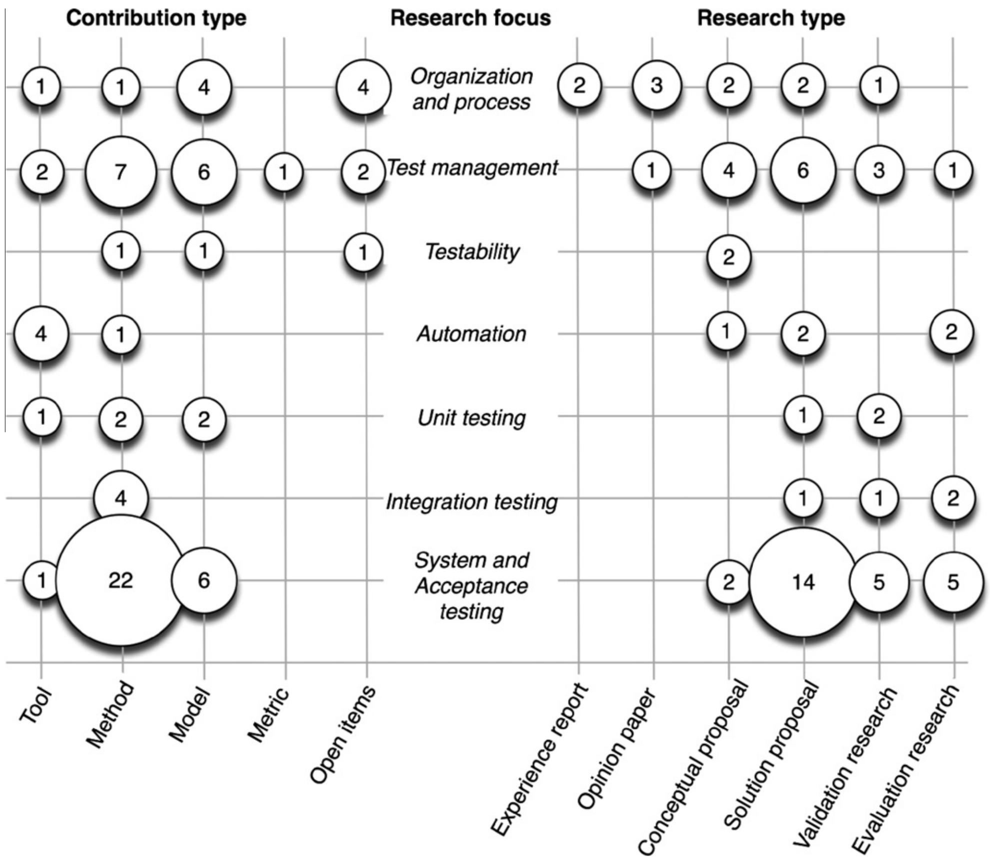


Fig. 3. Distribution of research foci.

Tevalinna et al. address the problem of dividing product line testing into two distinct instantiations of the V-model; testing is product oriented and no efficient techniques for domain testing exist [73]. Two problems are pointed out: First, complete integration and system testing in domain engineering is not feasible, and second, it is hard to decide how much we can depend on domain testing in the application testing. They also discuss four different strategies to model product line testing: testing product by product, incremental testing of product lines, reusable asset instantiation and division of responsibilities [73]. Weingärtner discusses the application of product-family engineering in an environment



Research focus	Tool	Method	Model	Metric	Open items	Experience report	Opinion paper	Conceptual proposal	Solution proposal	Validation research	Evaluation research
Organization and process	1	1	4		4	2	3	2	2	1	
Test management	2	7	6	1	2		1	4	6	3	1
Testability		1	1		1			2			
Automation	4	1						1	2		2
Unit testing	1	2	2						1	2	
Integration testing		4							1	1	2
System and Acceptance testing	1	22	6					2	14	5	5

Fig. 2. Map of research focus on software product line testing. Research focus on the *Y* axis; contribution type on the left side of the *X* axis, and research type on the right side of the *X* axis.

Table 4
Papers on test organization and process.

Author	Title	Paper type	Contribution type
Shaulis [68]	Salion's confident approach to testing software product lines	Experience report	Tool
Knauber and Hetrick [32]	Product line testing and product line development – variations on a common theme	Solution proposal	Method
McGregor [41]	Structuring test assets in a product line effort	Conceptual proposal	Model
Weingärtner [76]	Product-family engineering and testing in the medical domain-validation aspects	Opinion	Model
Ganesan et al. [17]	Comparing costs and benefits of different test strategies for a software product line: a study from Testo AG	Validation research	Model
Jin-hua et al. [24]	The W-model for testing software product lines	Solution proposal	Model
Kolb and Muthig [35]	Challenges in testing software product lines	Opinion paper	Open items
Tevanlinna et al. [73]	Product Family Testing - a Survey	Opinion paper	Open items
Kolb and Muthig [37]	Techniques and strategies for testing component-based software and product lines	Experience report	Open items
Ghanam et al. [19]	A test-driven approach to establishing and managing agile product lines	Conceptual proposal	Open items

where development was previously done according to the V-model [76]. Jin-hua et al. proposes a new test model for software product line testing, the W-model [24]. Ganesan et al. [17] compare cost benefits of a product focused test strategy contra an infrastructure focused test strategy and introduces a cost model to be able to quantify the influences on test costs from a given product variant. Ghanam et al. [19] discuss testing in the context of agile PL and highlights challenges in applying test driven development (TDD) in SPL. Shalius reports on positive experiences of agile testing in the context of XP and RUP [68].

6.1.2. Test management

The research on test management contains several proposals and a few evaluated research statements, see Table 5. Tevanlinna proposes a tool, called RITA (fRamework Integration and Testing Application) to support testing of product lines [72]. Kolb presents a conceptual proposal that sets focus on test planning and test case design, based on risks [34]. McGregor and Im make a remark that product lines vary both in space and in time, and outline a conceptual proposal to address this fact [43]. Oster et al. proposes a story driven approach to select which features to be tested in different product instances [57].

McGregor discusses, in his technical report, the possibility of product line organizations to retrieve a high level of structural coverage by aggregating the test executions of each product variant in the product line [42]. Schneidemann optimized product line testing by minimizing the number of configurations needed to verify the variation of the platform [70]. Gustafsson worked on algorithms to ensure that all features of a product line are covered in at least one product instance [22]. Cohen et al. [9] define a family of cumulative coverage criteria based on a relational model capturing variability in the feasible product variants, e.g. the orthogonal variability model. Kauppinenen et al. propose special coverage criteria for product line frameworks [29].

In order to reduce the test effort, McGregor proposes a combinatorial test design where pairwise combinations of variants are systematically selected to be tested instead of all possible

Table 5
Papers on test management.

Author	Title	Paper type	Contribution type
Tevanlinna [72]	Product family testing with RITA	Solution proposal	Tool
Kolb [34]	A risk-driven approach for efficiently testing software product lines	Solution proposal	Method
Scheidemann [70]	Optimizing the selection of representative configurations in verification of evolving product lines of distributed embedded systems	Solution proposal	Method
Gustafsson [22]	An approach for selecting software product line instances for testing	Validation research	Method
McGregor and Im [43]	The implications of variation for testing in a software product line	Conceptual proposal	Method
Oster et al. [57]	Towards software product line testing using story driven modeling	Conceptual proposal	Method
Cohen et al. [9]	Coverage and adequacy in software product line testing	Solution proposal	Model, Method
Al Dallal and Sorenson [2]	Testing software assets of framework-based product families during application engineering stage	Validation research	Model, method, tool
Zeng et al. [80]	Analysis of testing effort by using core assets in software product line testing	Solution proposal	Model
Dowie et al. [14]	Quality assurance of integrated business software: an approach to testing software product lines	Solution proposal	Model
Jaring et al. [25]	Modeling variability and testability interaction in software product line engineering	Evaluation research	Model
McGregor [44]	Toward a fault model for software product lines	Conceptual proposal	Model
Kauppinen et al. [29]	Hook and template coverage criteria for testing framework-based software product families	Conceptual proposal	Metric
Denger and Kolb [11]	Testing and inspecting reusable product line components: first empirical results	Validation research	Open items
Muccini and van der Hoek [48]	Towards testing product line architectures	Opinion paper	Open items

combinations [42]. Muccini and van der Hoek [48] propose a variant of this approach for integration testing, "core first then big bang", and emphasize the need for a combination of heuristic approaches to combine in order to effectively perform integration testing. Cohen et al. [9] propose application of interaction testing and connect this to the combinatorial coverage criteria.

Al Dallal and Sorenson present a model that focuses on framework testing in application engineering [2]. They identify uncovered framework use cases and select product test cases to cover those. The model is empirically evaluated on software, some 100 LOC in size.

Zeng et al. identify factors that influence SPL testing effort, and propose cost models accordingly [80]. Dowie et al. evaluate different approaches to SPL testing, based on a theoretical evaluation framework [14]. They conclude that the customer's perspective is missing in SPL testing, and must be included to make the approach successful.

Jaring et al. propose a process model, called VTIM (Variability and Testability Interaction Model) to support management of trade-offs on the binding point for a product line instance [25].

Table 6
Papers on testability.

Author	Title	Paper type	Contribution type
Kolb and Muthig [36]	Making testing product lines more efficient by improving the testability of product line architectures	Conceptual proposal	Model, method
Trew [74]	What design policies must testers demand from product line architects?	Conceptual proposal	Open items

They illustrate the model on a large-scale industrial system. Denger and Kolb report on a formal experiment, investigating inspection and testing as means for defect detection in product line components [11]. Inspections were shown to be more effective and efficient for that purpose. McGregor [44] discusses the need for more knowledge on faults likely to appear in a product line instance, and outlines a fault model. Fault models may be used as a basis for test case design and as help in estimating required test effort to detect a certain class of faults.

6.1.3. Testability

McGregor discusses testability of software product lines in his technical report [42]. This refers to technical characteristics of the software product that helps testing. We identified two papers on testability, see Table 6. Trew [74] identifies classes of faults that cannot be detected by testing and claim the need for design policies to ensure testability of an SPL. Kolb and Muthig [36] discuss the relationships between testability and SPL architecture and propose an approach to improve and evaluate testability.

6.1.4. System and acceptance testing

Table 7 lists paper on system and acceptance testing. Most research effort is spent on system and acceptance testing, 40%. The most frequent goal is automatic generation of test cases from requirements. Requirements may be model based, mostly on use cases [62], formal specifications [47] or written in natural language [8].

Hartman et al. present an approach based on existing UML based tools and methods [23]. Bertolino and Gnesi introduce PLUTO, product line use case test optimization [4,6], which is further elaborated by Bertolini et al. [5]. Kamsties et al. propose test case derivation for domain engineering from use cases, preserving the variability in the test cases [27].

Nebut et al. propose an algorithm to automatically generate product-specific test cases from product family requirements, expressed in UML [51,50], more comprehensively presented in [52]. They evaluate their approach on a small case study. Reuys et al. defined the ScenTED approach to generate test cases from UML models [64], which is further presented by Pohl and Metzger [60]. Olimpiew and Gomaa defined another approach using diagrams, stereotypes and tagged values from UML notations [55,54] which was illustrated in a student project [53]. Dueñas et al. propose another approach, based on the UML testing profile [13] and Kang et al. yet another process, based on UML use cases and a variability model [28]. Weißleder et al. specifically reuse state machines and generate sets suites, using OCL expressions [77].

Mishra [47] and Kahsai et al. [26] present test case generation models, based on process algebra formal specifications. Uzuncanova et al. introduce an incremental approach to test generation, using Alloy [75]. Bashardoust-Tajali and Corriveau extract tests for product testing, based on a domain model, expressed as generative contracts [8].

Stephensen et al. propose a test strategy to reduce the search space for test data, although without providing any reviewable details [71]. Geppert et al. present a decision model for acceptance testing, based on decision trees [20]. The approach was evaluated on a part of an industrial SPL. Li et al. utilize the information in execution traces to reduce test execution of each product of the SPL [39].

6.1.5. Integration testing

Table 8 lists papers on integration testing. The ScenTED method is proposed also for integration testing in addition to system and

Table 7
Papers on system and acceptance testing.

Author	Title	Paper type	Contribution type
Hartmann et al. [23]	UML-based approach for validating product lines	Solution proposal	Tool
Bertolino and Gnesi [6]	Use case-based testing of product lines	Solution proposal	Method
Bertolino and Gnesi [4]	PLUTO: a test methodology for product families	Validation research	Method
Kamsties et al. [27]	Testing variabilities in use case models	Solution proposal	Method
Nebut et al. [50]	Automated requirements-based generation of test cases for product families	Validation research	Method
Stephenson et al. [71]	Test data generation for product lines – a mutation testing approach	Solution proposal	Method
Geppert et al. [20]	Towards generating acceptance tests for product lines	Validation research	Method
Olimpiew and Gomaa [55]	Model-based testing for applications derived from software product lines	Solution proposal	Method
Reuys et al. [64]	Model-based system testing of software product families	Evaluation research	Method
Mishra [47]	Specification based software product line testing: a case study	Solution proposal	Method
Olimpiew and Gomaa [53]	Customizable requirements-based test models for software product lines	Evaluation research	Method
Pohl and Metzger [60]	Software product line testing	Conceptual Proposal	Method
Reis et al. [62]	A reuse technique for performance testing of software product lines	Evaluation research	Method
Reuys et al. [66]	The ScenTED method for Testingsoftware product lines	Evaluation research	Method
Li et al. [39]	Reuse execution traces to reduce testing of product lines	Evaluation research	Method
Bashardoust-Tajali and Corriveau [8]	On extracting tests from a testable model in the context of domain engineering	Solution proposal	Method
Kahsai et al. [26]	Specification-based testing for software productlines	Solution proposal	Method
Olimpiew and Gomaa [54]	Model-based test design for software product lines	Solution proposal	Method
Uzuncaova et al. [75]	Testing software product lines using incremental test generation	Validation research	Method
Weißleder et al. [77]	Reusing state machines for automatic test generation in product lines	Solution proposal	Method
Dueñas et al. [13]	Model driven testing in product family context	Solution proposal	Model
Nebut et al. [52]	System testing of product lines: from requirements to test cases	Validation research	Model
Olimpiew and Gomaa [56]	Reusable system tests for applications derived from software product lines	Conceptual proposal	Model
Kang et al. [28]	Towards a formal framework for product line test development	Solution proposal	Model, Method
Nebut et al. [51]	Reusable test requirements for UML-model product lines	Solution proposal	Model, Method
Bertolino et al. [5]	Product line use cases: scenario-based specification and testing of requirements	Solution proposal	Model, Method

Table 8
Papers on integration testing.

Author	Title	Paper type	Contribution type
Reuys et al. [66]	The ScenTED method for testing software product lines	Evaluation research	Method
Kishi and Noda [30]	Design testing for product line development based on test scenarios	Solution proposal	Method
Li et al. [40]	Automatic integration test generation from unit tests of eXVantage product family	Evaluation research	Method
Reis et al. [63]	Integration testing in software product line engineering; a model-based technique	Validation research	Method

acceptance testing, and hence mentioned here [66]. Reis et al. specifically validated its use for integration testing in an experimental evaluation [63]. Kishi and Noda propose an integration testing technique based on test scenarios, utilizing model checking techniques [30]. Li et al. generate integration test from unit tests, illustrated in an industrial case study [40].

6.1.6. Unit testing

Table 9 lists papers on unit testing. Different approaches to create test cases based on requirements including variabilities, are proposed with a focus on how to cover possible scenarios. In ScenTED, [65], UML-activity diagrams are used to represent all possible scenarios. Nebut et al. [49] use parameterized use cases as contracts on which testing coverage criteria may be applied. Feng et al. use an aspect-oriented approach to generate unit tests [16].

6.1.7. Test automation

Table 10 lists papers on test automation. McGregor et al. [45] propose and evaluate an approach to design test automation software which is based on correspondence between variability in product software and in test software. Condron [10] proposes a domain approach to automate PL testing, combining test automation frameworks from various locations in the entire product line where test is needed. Knauber and Schneider [33] explore how to combine aspect-oriented programming and unit testing and thus reach traceability between implementation of variability and its test. Ganesan et al. [18] focus on performance testing, reporting on a realization of an environment for testing response time and load of an SPL. Williams presents an approach to integrating test automation in an existing development environment for control systems [79].

6.2. Research type

Fig. 4, shows the distribution of research types in the area of software product line testing. The most frequent research type is solution proposals 41%. Adding solution, conceptual proposals and opinion papers sum up to 64% of the papers. 14% of the papers report on evaluation of the proposals and 3% are experience reports. 19% present other types of validation, primarily off-line approaches.

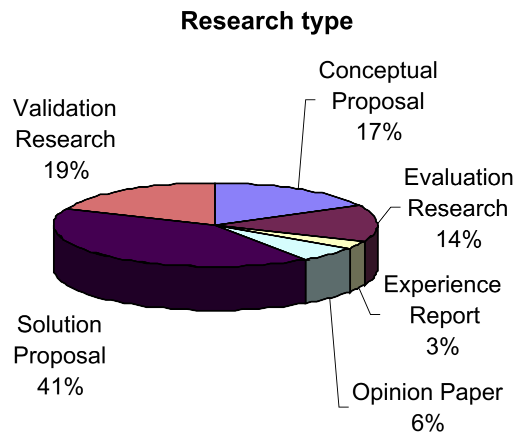


Fig. 4. Distribution of research type.

7. Discussion

The surveyed research indicates software product line testing being a rather immature area. The seminal paper is presented in 2001 [42], and most papers are published in workshops and conferences; only one has reached the maturity of a journal publication.

Software product line testing seems to be a "discussion" topic. There is a well established understanding about challenges, as summarized in Section 5. However, when looking for solutions to these challenges, we mostly find proposals. The mapping shows that 64% of the papers found include proposals, which contain ideas for solutions of the identified challenges, but only 17% of the research report actual use and evaluation of proposals.

Table 9
Papers on unit testing.

Author	Title	Paper type	Contribution type
Feng et al. [16]	A product line based aspect-oriented generative unit testing approach to building quality components	Validation research	Method
Reuys et al. [65]	Derivation of domain test scenarios from activity diagrams	Solution proposal	Model
Nebut et al. [49]	A Requirement-based approach to test product families	Validation research	Model, method, tool

Table 10
Papers on test automation.

Author	Title	Paper type	Contribution type
Knauber and Schneider[33]	Tracing variability from implementation to test using aspect-oriented programming	Conceptual proposal	Tool
Williams [79]	Test case management of controls product line points of variability	Solution proposal	Tool
Condron [10]	A domain approach to test automation of product lines	Solution proposal	Tool
Ganesan et al. [18]	Towards testing response time of instances of a web-based product line	Evaluation research	Tool
McGregor et al. [45]	Testing variability in a software product line	Evaluation research	Method

This is not unique for the SPL testing. Ramesh et al. reviewed publications in 13 computer science journals, and found less than 3% being case studies, field studies or experiments [61]. Close to 90% were of research type "conceptual analysis", which is close to our "proposals" categories. In software engineering, the case is somewhat better. Glass et al. reported 2002 that "conceptual analysis" also dominates in software engineering (54%), while case study, field study and experiment sum up to less than 10% [21].

Product line testing is a large scale effort and evaluations are costly [73], which is one of the explanations behind the limited share of empirical studies. However, extensive experience in PL engineering exist within companies (Philips, Nokia, Siemens, etc. [59]) but no studies on testing can be found [73].

The distribution across the research foci, with its major share on system testing is natural. This is where product line testing may gain a lot from utilizing the fact that it is a software product line. Testability issues, especially related to the product line architecture have an underdeveloped potential to be researched. Approaches that help isolate effects of variability to limited areas of the software would help improve the efficiency of product line testing. Test management issues have a reasonable proportion of the studies, although issues of balancing e.g. domain vs. product testing are not treated. Some sketched out proposals and many high-level opinions on how this should be done are reported on but none of them has been evaluated empirically.

Almost all of the proposed strategies for product line testing are idealistic in the sense that they put specific requirements on other parts of the development process than the testing. Hence, it is hard to find "useful approaches", since they require major changes to the whole software engineering process, e.g. formal models for requirements and variability. In a majority of the publications the handling of variability is in focus. Different approaches for test case derivation are based on specific ways of documenting and handling variation points. This is natural since variability is the core concept in product line development. However from the perspective of system testing the main challenge is how to deal with the large number of required tests of a range of product variants which are more or less similar. How variability is handled may not always be possible to affect or even visible at that stage. There is a need for strategies for test case design and selection, which are feasible for incremental introduction and applicable in a testing context regardless of the maturity of the product line organization.

The contribution type is mostly of "method" type. Product line engineering in general, and testing in particular, need new methodological approaches. However, methods need to be supported by underlying models for their theoretical foundation, tools for their practical use and metrics for their management and evaluation.

8. Conclusions

We launched a systematic mapping study to get an overview of existing research on software product line testing. We identified 64 papers published between 2001 and 2008.

The picture of research needs and challenges is quite clear and unanimous, enabling a focused research endeavor. In response to RQ1, the main challenges are: (i) the large number of tests, (ii) balance between effort for reusable components and concrete products, and (iii) handling variability. Still, there is a need to address different focus: process and organization, management, testability, test case design as well as test automation. To respond to RQ2, we conclude that the research is mostly published in workshops (59%) and conferences (30%), with only four book chapters and three journal publications issued so far. The research topics identified are (RQ3): (i) test organization and process, (ii) test management, (iii) testability, (iv) system and acceptance testing, (v) integration testing, (vi) unit testing, and (vii) automation, with high-level test case derivation as the most frequent topic followed by test management. Research methods (RQ4) are mostly of proposal type (64%) with empirical evaluations and experience as a minor group (17%).

With a clear picture of needs and challenges, we encourage the research community to launch empirical studies that use and evaluate the proposals, in order to give a solid foundation for software product line testing in industry. Further, trade-off management issues seem to be in need of deeper understanding and evaluation.

References

[1] W. Afzal, R. Torkar, R. Feldt, A systematic mapping study on non-functional search-based software testing, in: 20th International Conference on Software Engineering and Knowledge Engineering (SEKE), 2008.

[2] J. Al Dallal, P. Sorenson, Testing software assets of framework-based product families during application engineering stage, Journal of Software 3 (5) (2008) 11–25.

[3] J. Bailey, D. Budgen, M. Turner, B. Kitchenham, P. Brereton, S. Linkman, Evidence relating to object-oriented software design: a survey, in: First International Symposium on Empirical Software Engineering and Measurement, 2007.

[4] A. Bertolino, S. Gnesi, PLUTO: a test methodology for product-families, in: 5th International Workshop Software Product-Family Engineering, Siena, Italy, 2003.

[5] A. Bertolino, A. Fantechi, S. Gnesi, G. Lami, Product line use cases: scenario-based specification and testing of requirements, in: T. Käkölä, J.C. Duenas (Eds.), Software Product Lines Research Issues in Engineering and Management, Springer, 2006. Chapter 11.

[6] A. Bertolino, S. Gnesi, Use case-based testing of product lines, in: Proceedings of ESEC/FSE, ACM Press, 2003, pp. 355–358.

[7] J. Bosch, Design and use of software architectures, in: Adopting and Evolving a Product-line Approach, Addison-Wesley, 2000.

[8] S. Bashardoust-Tajali, J-P. Corriveau, On extracting tests from a testable model in the context of domain engineering, in: 13th IEEE International Conference on Engineering of Complex Computer Systems, 2008, pp. 98–107.

[9] M.B. Cohen, M.B. Dwyer, J. Shi, Coverage and adequacy in software product line testing, in: Proceedings of the ISSTA 2006 Workshop on Role of Software Architecture for Testing and Analysis, ACM, New York, 2006, pp. 53–63.

[10] C. Condron, A domain approach to test automation of product lines, in: International Workshop on Software Product Line Testing, 2004.

[11] C. Denger, R. Kolb, Testing and inspecting reusable product line components: first empirical results, in: Proceedings 5th International Software Metrics Symposium, 2006.

[12] O. Dieste, A. Grimán, N. Juristo, Developing search strategies for detecting relevant experiments, Empirical Software Engineering (2008), doi:10.1007/s10664-008-9091-7.

[13] J.C. Dueñas, J. Mellado, R. Cerón, J.L. Arciniegas, J.L. Ruiz, R. Capilla, Model driven testing in product family context, in: First European Workshop on Model Driven Architecture with Emphasis on Industrial Application, 2004.

[14] U. Dowie, N. Gellner, S. Hanssen, A. Helferich, G. Herzwurm, S. Schockert, Quality assurance of integrated business software: an approach to testing software product lines, in: Proceedings of the 13th European Conference on Information Systems, Information Systems in a Rapidly Changing Economy, (ECIS), 2005.

[15] E. Engstrom, P. Runeson, M. Skoglund, A systematic review on regression test selection techniques, Information and Software Technology 52 (1) (2010) 14–30.

[16] Y. Feng, X. Liu, J. Kerridge, A product line based aspect-oriented generative unit testing approach to building quality components, in: Proceedings of the 31st Annual international Computer Software and Applications Conference (COMPSAC), 2007.

[17] D. Ganesan, J. Knodel, R. Kolb, U. Haury, G. Meier, Comparing costs and benefits of different test strategies for a software product line: a study from Testo AG, in: Proceedings of Software Product Line Conference (SPLC), 2007.

[18] D. Ganesan, U. Maurer, M. Ochs, B. Snoek, M. Verlage, Towards testing response time of instances of a web-based product line, in: Proceedings of International Workshop on Software Product Line Testing (SPLiT 2005), Rennes, France, September 2005, pp. 23–34.

[19] Y. Ghanam, S. Park, F.A. Maurer, A test-driven approach to establishing & managing agile product lines, in: The 5th SPLiT Workshop – SPLC, Ireland, 2008.

[20] B.J. Geppert, J. Li, F. Rossler, D.M. Weiss, Towards generating acceptance tests for product lines, in: 8th International Conference on Software Reuse, Madrid, Spain, 2004.

[21] R.L. Glass, I. Vessey, V. Ramesh, Research in software engineering: an analysis of the literature, Information and Software Technology 44 (2002) 491–506.

[22] T. Gustafsson, An approach for selecting software product line instances for testing, in: International Workshop on Software Product Line Testing, 2007.

[23] J. Hartmann, M. Vieira, A. Ruder, UML-based approach for validating product lines, in: International Workshop on Software Product Line Testing (SPLiT), Avaya Labs Technical Report, Boston, USA, August 2004, pp. 58–64.
[24] L. Jin-hua, L. Qiong, L. Jing, The W-model for testing software product lines, in: International Symposium on Computer Science and Computational Technology (ISCSCT), 2008.
[25] M. Jaring, R.L. Krikhaar, J. Bosch, Modeling variability and testability interaction in software product line engineering, in: Seventh International Conference on Composition - Based Software Systems, ICCBSS, 2008, pp. 120–129.
[26] T. Kahsai, M. Roggenbach, B.-H. Schlinglof, Specification-based testing for software product lines, in: Sixth IEEE International Conference on Software Engineering and Formal Methods, SEFM 2008, Cape Town, South Africa, November 2008, pp. 10–14.
[27] E. Kamsties, K. Pohl, S. Reis, A. Reuys, Testing variabilities in use case models, in: F. van der Linden, (Ed.), Proceedings of the 5th International Workshop on Software Product-Family Engineering, PFE-5 (Siena, Italy, November 2003), Springer, Heidelberg, 2003, 6–18.
[28] S. Kang, J. Lee, M. Kim, W Lee, Towards a formal framework for product line test development, in: Proceedings of the 7th IEEE international Conference on Computer and information Technology (October 16–19, 2007), CIT, IEEE Computer Society, Washington, DC, 2007, pp. 921–926.
[29] R. Kauppinen, J. Taina, A. Tevanlinna, Hook and template coverage criteria for testing framework-based software product families, in: Proceedings of the International Workshop on Software Product Line Testing, August 2004, 7–12.
[30] T. Kishi, N. Noda, Design testing for product line development based on test scenarios, in: Presented at Software Product Line Testing Workshop (SPLiT), Boston, MA, 2004.
[31] B.A. Kitchenham, Guidelines for performing systematic literature reviews in software engineering version 2.3, Technical Report S.o.C.S.a.M. Software Engineering Group, Keele University and Department of Computer Science University of Durham, 2007.
[32] P. Knauber, W. Hetrick, Product line testing and product line development - variations on a common theme, in: Proceedings of International Workshop on Software Product Line Testing (SPLiT 2005), 2005.
[33] P. Knauber, J. Schneider, Tracing variability from implementation to test using aspect-oriented programming, in: International Workshop on Software Product Line Testing SPLiT, 2004.
[34] R. Kolb, A risk driven approach for efficiently testing software product lines, in: 5th GPCE Young, Researches Workshop, Erfurt, Germany, September 2003.
[35] R. Kolb, D. Muthig, Challenges in testing software product lines, in: Proceedings of CONQUEST'03, Nuremberg, Germany, September (2003), pp. 81–95.
[36] R. Kolb, D. Muthig, Making testing product lines more efficient by improving the testability of product line architectures, in: Proceedings of the ISSTA 2006 Workshop on Role of Software Architecture for Testing and Analysis (Portland, Maine, July 17–20, 2006), ROSATEA '06, ACM, New York, NY, 22–27, 2006. <http://doi.acm.org/10.1145/1147249.1147252>.
[37] R. Kolb, D. Muthig, Techniques and strategies for testing component-based software and product lines, development of component-based information systems, Advances in Management Information Systems 2 (2006) 123–139. Chapter 7.
[38] B.P. Lamancha, M.P. Usaola, M.P. Velthius, Software product line testing – a systematic review, in: 4th International Conference on Software and Data Technologies (ICSOFT), 2009, pp. 23–30.
[39] J.J. Li, B. Geppert, F. Roessler, D.M. Weiss, Reuse execution traces to reduce testing of product lines, in: Proceedings of the International Workshop on Software Product Line Testing, 2007.
[40] J.J. Li, D.M. Weiss, J.H. Slye, Automatic integration test generation from unit tests of EXvantage product family, in: Proceedings of the International Workshop on Software Product Line Testing, 2007.
[41] J.D. McGregor, Structuring test assets in a product line effort, in: Proceedings of the Second International Workshop on Software Product Lines: Economics, Architectures, and Implications, May 2001, pp. 89–92.
[42] J.D. McGregor, Testing a software product line, Technical Report, CMU/SEI-2001-TR-022, ESC-TR-2001-022.
[43] J.D. McGregor, K. Im, The implications of variation for testing in a software product line, in: International Workshop on Software Product Line Testing (SPLiT 2007), 2007.
[44] J.D. McGregor, Toward a fault model for software product lines, in: Proceedings Fifth International Workshop on Software Product Line Testing, (SPLiT 2008) – informatik.fh-mannheim.de, 2008, p. 27.
[45] J.D. McGregor, P. Sodhani, S. Madhavapeddi, Testing variability in a software product line, in: Proceedings of the International Workshop on Software Product Line Testing, Avaya Labs, ALR-2004-031, 2004, pp. 45–50.
[46] M. Meyer, A. Lehnerd, The Power of Product Platforms, Free Press, New York, 1997.
[47] S. Mishra, Specification based software product line testing: a case study, in: Proceedings of the Concurrency: Specification and Programming Workshop, 2006, pp. 243–254.
[48] H. Muccini, A. van der Hoek, Towards testing product line architectures, Electronic Notes in Theoretical Computer Science 82 (6) (2003).
[49] C. Nebut, F. Fleurey, Y.L. Traon, J.-M. Jézéquel, A requirement-based approach to test product families, in: International Workshop on Product Family Engineering (PFE), 2003.
[50] C. Nebut, S. Pickin, Y. Le Traon, J.M. Jezequel, Automated requirements-based generation of test cases for product families, in: Proceedings 18th IEEE International Conference on Automated Software Engineering, 2003.
[51] C. Nebut, S. Pickin, Y. Le Traon, J.M. Jezequel, Reusable test requirements for UML-model product lines, in: International Workshop on Requirements Engineering for Product Lines (REPL), 2002.
[52] C. Nebut, Y. Le Traon, J.M. Jézéquel, System testing of product lines: from requirements to test cases software product lines, Engineering and Management (2006) 447–477. Chapter.
[53] E.M. Olimpiew, H. Gomaa, Customizable requirements based test models for software product lines, in: International Workshop on Software Product Line Testing, Baltimore, MD, August 2006.
[54] E.M. Olimpiew, H. Gomaa, Model-based test design for software product lines, in: International Workshop on Software Product Line Testing (SPLiT 2008), 2008.
[55] E.M. Olimpiew, H. Gomaa, Model-based testing for applications derived from software product lines, in: 1st Workshop on Advances in Model-Based Software Testing, A-MOST'05, ACM Press, 2005.
[56] E.M. Olimpiew, H Gomaa, Reusable system tests for applications derived from software product lines, in: International Workshop on Software Product Line Testing (SPLiT 2005), 2005, pp. 8–15.
[57] S. Oster, A. Schürr, I. Weisemöller, Towards software product line testing using story driven modeling, in: Proceedings of 6th International Fujaba Days, 2008, pp. 48–55.
[58] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE). University of Bari, Italy, 26–27 June 2008.
[59] K. Pohl, G. Böckle, F. van der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, Springer, Heidelberg, 2005. August.
[60] K. Pohl, A. Metzger, Software product line testing, Communications of ACM 49 (12) (2006) 78–81.
[61] V. Ramesh, R.L. Glass, I. Vessey, Research in computer science: an empirical study, The Journal of Systems and Science 70 (1–2) (2004) 165–176.
[62] S. Reis, A. Metzger, K. Pohl, A reuse technique for performance testing of software product lines, in: Proceedings of the International. Workshop on Software Product Line Testing, Mannheim University of Applied Sciences, Report No. 003.06, 2006, 5–10.
[63] S. Reis, A. Metzger, K. Pohl, Integration testing in software product line engineering: a model-based technique, in: M.B. Dwyer, A. Lopes (Eds.), Proceedings Fundamental Approaches to Software Engineering, 10th International Conference, FASE 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007, Braga Portugal, LNCS 4422, March 24–April 1, 2007.
[64] A. Reuys, E. Kamsties, K. Pohl, S. Reis, Model-based system testing of software product families, in: O. Pastor, J. Falcao e Cunha, (Eds.), Proceedings of the 17th Conference on Advanced Information Systems Engineering, CAiSE (Porto, Portugal, June 2005), Springer, Heidelberg, 2005, 519–534.
[65] A. Reuys, S. Reis, E. Kamsties, K. Pohl, Derivation of domain test scenarios from activity diagrams, in: Proceedings of the International Workshop on Product Line Engineering the Early Steps: Planning, Modeling, and Managing (PLEES'03), 2003.
[66] A. Reuys, S. Reis, E. Kamsties, K. Pohl, The scented method for testing software product lines, in: Software Product Lines, Springer, Heidelberg, 2006, pp. 479–520.
[67] P. Runeson, M. Skoglund, Reference-based search strategies in systematic reviews, in: 13th International Conference on Empirical Assessment & Evaluation in Software Engineering, Durham University, UK, 2009.
[68] C. Shaulis, Salion's confident approach to testing software product lines, in: Proceedings of International Conference on Product Line Testing, Boston, Massachusetts, USA (SPLiT 04), 2004.
[69] M. Shaw, What makes good research in software engineering?, International Journal on Software Tools for Technology Transfer (STTT) 4 (1) (2002) 1433–2779
[70] K.D. Scheidemann, Optimizing the selection of representative configurations in verification of evolving product lines of distributed embedded systems, in: Proceedings of the 10th International Software Product Line Conference (SPLC'06), 2006, pp. 75–84.
[71] Z. Stephenson, Y. Zhan, J. Clark, J. McDermid, Test data generation for product lines - a mutation testing approach, in: Nord, R.L. (Ed.), SPLC 2004, LNCS, vol. 3154, Springer, Heidelberg, 2004.
[72] A. Tevanlinna, Product family testing with RITA, in: Proceedings of the Eleventh Nordic Workshop on Programming and Software Development Tools and Techniques (NWPER), 2004, pp. 251–265.
[73] A. Tevanlinna, J. Taina, R. Kauppinen, Product family testing: a survey, ACM SIGSOFT Software Engineering Notes 29(2) (2004) pp. 12–17. doi: 10.1145/979743.979766.
[74] T. Trew, What design policies must testers demand from product line architects? in: Proceedings of International Workshop on Software Product Line Testing, 2004.
[75] E. Uzuncaova, D. Garcia, S. Khurshid, D. Batory, Testing software product lines using incremental test generation, in: ISSRE, 2008.
[76] J. Weingärtner, Product family engineering and testing in the medical domain—validation aspects, in: Software Product-Family Engineering, 4th International Workshop (PFE) (Bilbao, Spain, October 3–5 2001), Revised Papers, LNCS 2290/2002, 2002, 56–77.

[77] S. Weißleder, D. Sokenou, B.H. Schlingloff, Reusing state machines for automatic test generation in product lines, in: 1st Workshop on Model-based Testing in Practice, 2008.
[78] R. Wieringa, N. Maiden, N. Mead, C. Rolland, et al., Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, Requirements Engineering 11 (1) (2006) 102–107.
[79] J.J. Williams, Test case management of controls product line points of variability, in: International Workshop on Software Product Line Testing, SPLiT, 2004.
[80] H. Zeng, W. Zhang, D. Rine, Analysis of testing effort by using core assets in software product line testing, in: International Workshop on Software Product Line Testing, SPLiT, 2004.