Information and Software Technology 61 (2015) 33–51

Contents lists available at ScienceDirect

Information and Software Technology

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

A systematic mapping study of search-based software engineering for software product lines

Roberto E. Lopez-Herrejon *, Lukas Linsbauer, Alexander Egyed

Institute for Software Systems Engineering, Johannes Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria

ARTICLE INFO

Article history:
Received 28 August 2014
Received in revised form 16 January 2015
Accepted 17 January 2015
Available online 29 January 2015

Keywords:
Software product line
Systematic mapping study
Search based software engineering
Evolutionary algorithm
Metaheuristics

ABSTRACT

Context: Search-Based Software Engineering (SBSE) is an emerging discipline that focuses on the application of search-based optimization techniques to software engineering problems. Software Product Lines (SPLs) are families of related software systems whose members are distinguished by the set of features each one provides. SPL development practices have proven benefits such as improved software reuse, better customization, and faster time to market. A typical SPL usually involves a large number of systems and features, a fact that makes them attractive for the application of SBSE techniques which are able to tackle problems that involve large search spaces.
Objective: The main objective of our work is to identify the quantity and the type of research on the application of SBSE techniques to SPL problems. More concretely, the SBSE techniques that have been used and at what stage of the SPL life cycle, the type of case studies employed and their empirical analysis, and the fora where the research has been published.
Method: A systematic mapping study was conducted with five research questions and assessed 77 publications from 2001, when the term SBSE was coined, until 2014.
Results: The most common application of SBSE techniques found was testing followed by product configuration, with genetic algorithms and multi-objective evolutionary algorithms being the two most commonly used techniques. Our study identified the need to improve the robustness of the empirical evaluation of existing research, a lack of extensive and robust tool support, and multiple avenues worthy of further investigation.
Conclusions: Our study attested the great synergy existing between both fields, corroborated the increasing and ongoing interest in research on the subject, and revealed challenging open research questions.

Contents

* Corresponding author. Tel.: +43 732 2468 4380.
E-mail addresses: roberto.lopez@jku.at (R.E. Lopez-Herrejon), lukas.linsbauer@jku.at (L. Linsbauer), alexander.egyed@jku.at (A. Egyed).

http://dx.doi.org/10.1016/j.infsof.2015.01.008
0950-5849/

1. Introduction

Search Based Software Engineering (SBSE) is an established yet young discipline that focuses on the application of search-based optimization techniques to software engineering problems [1]. In this article, we follow Harman et al. who consider SBSE techniques to primarily include metaheuristic search based optimization techniques and classical operations research techniques [1,2]. Some examples of SBSE techniques are: evolutionary computation techniques[1] (e.g. genetic algorithms), basic local searches (e.g. hill climbing, simulated annealing or random search) [4], and integer programming [1].

Software Product Lines (SPLs) are families of related systems whose members are distinguished by the set of features they provide [5,6]. *Variability* is the capacity of software artifacts to vary and its effective management and realization lie at the core of successful SPL development [7]. *Feature models* are tree-like structures that establish the relations between features and have become the de facto standard for modeling variability [8,9]. Over the last decade, extensive research and practice both in academia and industry attest to the substantial benefits of applying SPL practices [6,10,11]. Among the benefits are better customization, improved software reuse, and faster time to market.

Typical SPLs have a large number of features that are combined in complex feature relations yielding a large number of individual software systems that must be effectively and efficiently designed, implemented and managed. Precisely this fact is what makes SPL-related problems suitable for the application of SBSE techniques which are generic, flexible, robust, and have been shown to scale to large search spaces such as those that typically characterize SPLs (e.g. [12]). This recent realization has sparked a surge of research and application at the intersection of SBSE and SPLs, which has manifested with an increasing number of articles at many of the publication outlets of both research communities. This is precisely what prompted us to perform a *systematic mapping study* to provide an overview of the research at the intersection of these two fields [13–15]. In contrast with a *systematic literature review* whose goal is primarily to identify best practice [13,15–17], our general goal was to identify the quantity and the type of research and results available, and thus highlight possible open research problems and opportunities, for both SBSE and SPL communities. More concretely we wanted to identify at what stages of the SPL development life cycle have SBSE techniques been used and which ones. We also wanted to find out the provenance, number and types of artifacts used as case studies as well as how they were empirically analyzed. And finally, which are the fora where the research work was published.

Our study corroborated the increasing and ongoing interest in applying SBSE techniques in SPLs. We found that the most common application is software testing, and the most common techniques are genetic algorithms and multi-objective evolutionary algorithms. Our study identified the need to improve the robustness of the empirical evaluation (e.g. more adequate statistical analysis) and the need for more extensive and robust tool support. We hope that this mapping study not only serves to highlight the main research topics at the intersection of SBSE and SPLs but that it also serves to encourage researchers to pursue work at the intersection of both areas.

The paper is structured as follows. Section 2 presents the process we followed for our systematic mapping study. It details the research questions addressed, how the search was performed, the classification scheme used, and how the data was extracted and analyzed. Section 3 presents the results we obtained for each research question. Section 4 contains a description of the results found along with open questions and avenues worth of further investigation. Section 5 summarizes the threats to validity we identified in our work and how they were addressed. Section 6 concisely describes the existing review studies and surveys of SPLs and SBSE. Section 7 summarizes the conclusions of our study and future work.

2. Systematic mapping study

Evidence-Based Software Engineering (EBSE) is an emerging software engineering area whose goal is *"to provide the means by which current best evidence from research can be integrated with practical experience and human values in the decision making process regarding the development and maintenance of software"* [18]. One of the approaches advocated by EBSE is systematic mapping studies whose goal is to provide an overview of the results available within an area by categorizing them along criteria such as type, forum, frequency, etc. [14]. For performing our mapping study, we followed the protocol proposed by Petersen et al. [14], whose main stages are shown in Fig. 1. Next we describe each of the processes

[1] Evolutionary computation is an area of computer science, artificial intelligence more concretely, that studies algorithms that follow Darwinian principles of evolution [3].

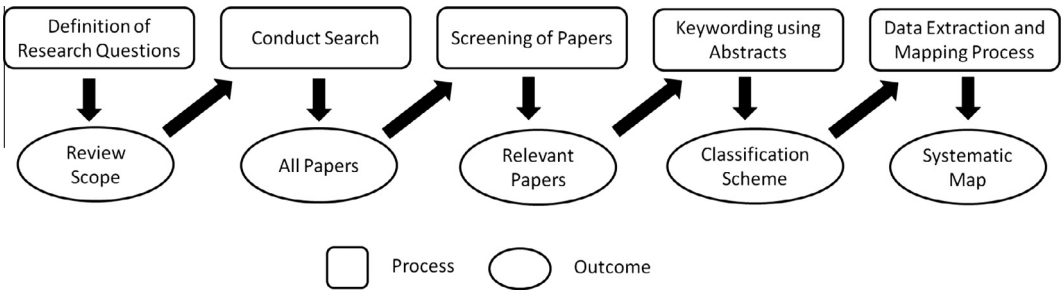


Fig. 1. Systematic mapping study process [14].

and how they were performed for our mapping study. In Section 3 we present the results we obtained, which we analyze in Section 4.

2.1. Definition of research questions

We again reiterate that throughout this paper we follow Harman and colleagues' common understanding of SBSE techniques to primarily include metaheuristic search based optimization techniques and classical operations research techniques [1,2,19].

Recall that the main goal underlying our work is to provide an overview of research that applies SBSE techniques to tackle SPL problems. Hence our main objective is to summarize and characterize the existing research evidence at the intersection of these two fields. There are additional aspects that can also provide further insight on the existing evidence such as types of comparison analysis performed, number and nature of case studies, publication fora, etc. Our mapping study then focuses on the following research questions:

- ***RQ1. In what phases of the SPL life cycle have SBSE techniques been used?***
  Rationale: SBSE has been applied throughout the entire life cycle of single systems [1], so our interest is finding out if SBSE has been applied also throughout the entire life cycle of SPLs [6].
- ***RQ2. What SBSE techniques have been used?***
  Rationale: There are a vast number of SBSE techniques available in literature. Our goal here is cataloging their use for SPL problems and analyze if there are common trends in their application.
- ***RQ3. What type of comparative analysis is used?***
  Rationale: Search-based techniques commonly rely on randomness, so adequate statistical analysis is necessary for the results to be useful and meaningful [20]. Here our objective is to identify which statistical tests and measures are used to compare algorithms. Further details are described in Section 2.4.3.
- ***RQ4. What evaluation case studies are used?***
  Rationale: Here our focus is on cataloging the type, number, and provenance of the case studies analyzed. We believe that identifying common case studies and their sources could lead to establishing community-wide benchmarks for certain problems.
- ***RQ5. What are the publication fora used?***
  Rationale: SBSE and SPL research appears in multiple outlets and in different research communities. We believe that by identifying publication fora researchers can keep abreast with research developments as well as target future publications.

2.2. Conduct search for primary sources

In this step of the systematic mapping the strings of terms to be used for the search are defined. In our study we selected two sets of terms, one for SPLs and a second for SBSE. Table 1 shows the list of all search terms we used.[2] In order to gather a list of SPL search terms as comprehensive as possible, we collected the relevant search terms of twelve systematic mapping and literature review studies in SPLs [21–32]. We should remark that none of these studies focuses on SBSE techniques, hence we took from them only their SPL-specific search terms and excluded those terms of the domains of their studies (e.g. testing, service orientation, agile methods).

For gathering the list of SBSE terms we found three studies [1,33,34]. The terms used by these studies are SBSE algorithms. We complemented this list by including more generic terms (e.g. mutation testing or constraint handling) and new SBSE techniques to reflect recent developments in SBSE.[3] In Table 1 the new terms that our queries considered are underlined. Further details on the meaning of these terms can be consulted in artificial intelligence, metaheuristics, and evolutionary computation books (e.g. [36,3,4,37–39]).

To perform our search we proceeded in four stages that were consecutively carried out. These search stages included:

- *Specialized SBSE repositories*. We performed searches in the two existing specialized repositories: Search Based Software Engineering Repository,[4] and the Bibliography on Genetic Programming.[5]
- *Publishing companies and general search engines*. We utilized the search engines ScienceDirect, IEEExplore, ACM Digital Libray, SpringerLink, and Google Scholar. These engines cover the main publication venues of the specialized journals, conferences, and workshops in both SPLs and SBSE as well as other publication outlets such as technical reports and dissertations.
- *Snowballing readings*. *Snowballing* refers to analyzing the reference list or citations of identified papers to further search other sources [15,40]. In addition, we followed the recent guidelines advocated by Wohlin et al. that suggest considering what other publications the identified papers are referenced by [40]. We manually performed this stage following the citation links provided by the publishing companies and Google Scholar.
- *SPLC14 keynote readings*. We also analyzed a recent survey on SPL and SBSE as a part of a keynote presentation at the Software Product Line Conference in 2014 [41]. We searched for any sources our searches might have missed. Details of this survey and a comparison with our work are presented in Section 6.

The queries we performed took all the combinations of one term from the SPL list and one or more terms of the SBSE terms

[2] Alternative term spellings or hyphenation are not shown in the table and were found not to be relevant for our searches.

[3] These techniques were proposed while preparing the early draft of this paper (see [35]) by our colleagues Ferrer, Chicano and Alba based on their extensive experience in SBSE.

[4] http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/repository.html.

[5] http://liinwww.ira.uka.de/bibliography/Ai/genetic.programming.html.

Table 1
Summary of SPL and SBSE search terms. Acronyms appear after their defining term.

SPL terms: application engineering, commonality, core asset, domain analysis, domain engineering, feature analysis, feature based, feature diagram, feature model, feature modeling, feature oriented, highly-configurable system, process family, product family, product line, product line engineering, software family, software product family, software product line, software reuse, SPL, variability, variability analysis, variability management, variability modeling, variability-intensive system, variant, variation, variation point
SBSE terms: ant colony optimization, ACO, artificial immune systems, AIS, bee colony, constraint handling, estimation distribution algorithm, EDA, evolutionary algorithm, evolutionary programming, evolutionary strategy, fire fly, genetic algorithm, GA, genetic programming, GP, grammatical evolution, GE, greedy randomized adaptive search procedure, GRASP, greedy search, harmony search, hill climbing, imperial competitive algorithm, integer programming, iterative local search, ILS, local search, memetic algorithm, metaheuristic, multiobjective optimization, multi-objective optimization, MOEA, mutation testing, optimization, optimization algorithm, particle swarm optimization, PSO, path relinking, scattered search, search based, simulated annealing, tabu search, variable neighborhood search

depending on the querying functionality of each search engine or repositories. The searches included the title, abstract, and keywords of the papers, and when supported by the search engine also their contents. For example, the following is a fragment of a query used in the IEEExplore engine[6]:

```
("product line") AND ("genetic algorithm" OR "GA"
  OR "genetic programming" OR "GP" OR "hill
  climbing" OR "simulated annealing")
```

In addition, we trimmed our search to include only publications from 2001 onwards because the publication of a seminal paper by Harman and Jones (see [19]), that appeared that year, coined the term SBSE and it is generally regarded as the start of this discipline. It should be noted though that techniques such as genetic algorithms had been sporadically used before in software engineering research; however, SBSE provided a conceptual framework under which to group this and other optimization techniques when applied to software engineering problems [19,1].

2.3. Screening of papers for inclusion and exclusion

We looked for the search terms in the title, abstract and keywords and whenever necessary at the introduction or at other places of the paper. The sole criteria for inclusion in our mapping study was that a clear application of SBSE techniques to SPLs was described. We should also make some remarks regarding the criteria for exclusion. First, there is certainly a large body of work on product lines and search-based techniques but in other domains such as civil engineering, marketing or operations research. Second, there are some incipient works on applying to SPL problems techniques from artificial intelligence such as those based on machine learning or automated planning that do not fall directly into the techniques considered by SBSE. Third, programming paradigms such as constraint programming have also been applied to SPL problems; however, these as well do not fall directly into the techniques employed by SBSE research. Hence, articles that fell into these three cases were not included. The decision on whether or not to include a paper was most of the times straightforward, meaning that at least one SBSE term was found and a clear connection to SPLs was easily identified.

2.4. Keywording using abstracts—classification scheme

We classified our articles into five dimensions aligned with the five research questions that our mapping study addresses. We deviate from the standard classification procedure whereby the classification schemes follow from the abstract keywords because our driving goal is slightly different. In our case, we want to bring to the attention of researchers and practitioners of SPL and SBSE communities the synergies of both disciplines. Hence, we decided on using a framework and terminology that are already familiar within these communities.

We should also remark that for the classifications based on life cycle stage (Section 2.4.1), SBSE techniques (Section 2.4.2), and types of case studies (Section 2.4.4) a primary source can be classified in more than one category. For the classifications of type of analysis (Section 2.4.3) and publication fora (Section 2.4.5) a primary source is classified in only one category. We reiterate and illustrate this fact in Section 3 where the results of our mapping study are summarized.

2.4.1. SPL life cycle stage classification

For this classification we used Pohl et al.'s (see [6]) SPL engineering framework shown in Fig. 2. This is a common and well-known framework within SPL research and has recently been used to illustrate and highlight some of the open questions and challenges in the field of SPLs [42]. This framework defines four sub-processes for each of the two main SPL activities, *Domain Engineering (DE)* and *Application Engineering (AE)* defined next.

Definition 1. Domain Engineering is the process of software product line engineering in which the commonality and the variability of the product line are defined and realized [6].

Definition 2. Application Engineering is the process of software product line engineering in which the applications of the product line are built by reusing domain artefacts and exploiting the product line variability [6].

We regard each sub-process of DE and AE as a life cycle stage.[7] In addition to the eight stages of this framework, we considered two more classification categories: one to cover all maintenance and evolution issues of SPLs, and one to contain publications that have tooling support as a main contribution. In summary, our classification terms are as follows and we will refer to them henceforth by their shorthand names in parenthesis[8]:

- *Domain Requirements Engineering (DRE)* is the sub-process of DE where the common and variable requirements of the product line are defined, documented in reusable requirements artifacts, and continuously managed.

[6] The search queries had to be broken down into smaller queries (as shown in the example) because of the search limitations of some search engines. We made sure however that we considered all possible combinations of one SPL term with one SBSE term.

[7] Domain Engineering and Application Engineering are the two common activities in all the SPL approaches. The clear distinction between their goals as stated in their definitions is the underlying reason behind considering their corresponding stages as separated classification terms.

[8] Though ME and TOOL are strictly speaking not a *life cycle stage*, for sake of description simplicity we overload this term to collectively refer to all our categories in this classification dimension.

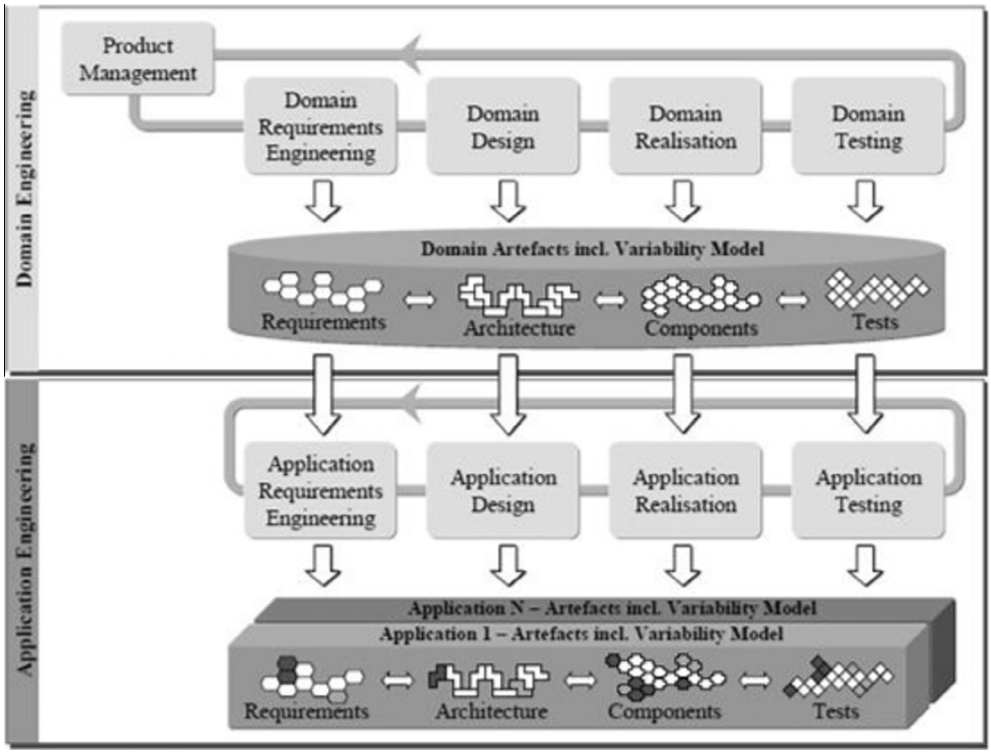


Fig. 2. Pohl et al.'s SPL framework, Fig. 2-1 from [6].

- *Domain Design (DD)* is the sub-process of DE where a reference architecture for the entire software product line is developed.
- *Domain Realization (DR)* is the sub-process of DE where the set of reusable components and interfaces of the product line is developed.
- *Domain Testing (DT)* is the sub-process of DE where evidence of defects in domain artifacts is uncovered and where reusable test artifacts for application testing are created.
- *Application Requirements Engineering (ARE)* is the sub-process of AE dealing with the elicitation of stakeholder requirements, the creation of the application requirements specification, and the management of application requirements.
- *Application Design (AD)* is the sub-process of AE where the reference architecture is specialized into the application architecture.
- *Application Realization (AR)* is the sub-process of AE where a single application is realized according to the application architecture by reusing domain realization artifacts.
- *Application Testing (AT)* is the sub-process of AE where domain test artifacts are reused to uncover evidence of defects in an application.
- *Maintenance and Evolution (ME)* refers to the maintenance and evolution of all the artifacts developed across the entire life cycle of SPLs. Reverse engineering artifacts or bug fixing are examples of activities that fall in this category.
- *Tool support (TOOL)* used to classify the publications that have tooling support as one of their main contributions (e.g. tool papers or demos).

2.4.2. SBSE techniques classification

We considered each of the SBSE search terms shown in Fig. 1 as one category, except of course acronyms that were aggregated together with their corresponding term. This decision simplified the classification effort. In addition, three more categories were added: one to contain deterministic searches,[9] and two to distinguish between evolutionary and exact multi-objective algorithms.

It should be pointed out that it is customary in SBSE articles to compare against base line search techniques. For our classification, we report the main SBSE technique(s) put forward by each paper when it(they) is(are) clearly identified in the paper and omit those techniques that are used only for base line comparisons (e.g. random search). If one category may subsume another we select the most specific one for classification.[10] Notice as well that search terms *mutation testing* and *constraint handling* are neither metaheuristic nor operations research algorithms, nonetheless they are considered relevant types of techniques within the SBSE community.

2.4.3. Type of statistical analysis classification

As we mentioned before, search-based techniques commonly rely on randomness, so adequate statistical analysis is necessary for the results to be useful and meaningful. Hence, we classified publications into four categories depending on the type of analysis performed:

- *Undefined* whenever we were not able to clearly discern what type of analysis was performed because the authors employ domain-specific terminology and models whose statistical foundations (if any) we were not able to ascertain.[11]
- *None* when there was clearly no analysis presented.
- *Basic* when some basic statistical measures were used (i.e. median, average, standard deviation).

[9] This category was required to classify primary sources that used algorithms such as Breadth-First Search. These sources resulted primarily from using the term "search based" in our queries.

[10] For example, if a paper that uses an algorithm such as SPEA2 it is classified as multi-objective evolutionary algorithm but not as a genetic algorithm.

[11] For example, the cost models of S8.

- *Statistical Tests* when any of the standard statistical analysis tests was used, for example as proposed in the guidelines by Arcuri and Briand [20].

2.4.4. Type of case studies classification

We classified publications according to the number of case studies, the types of artifacts, and the provenance of the artifacts. We should point out that for this classification we considered the artifacts that contained or expressed the variability of the SPLs, e.g. feature models or UML class diagrams.

For artifact provenance we defined the following categories:

- *SPLOT*[12] which is a repository for feature models widely used within the SPL research community.
- *Random* when the artifacts are generated randomly.
- *Open source* when artifacts come from projects developed as open source, e.g. Linux kernel [43].
- *Academic* when the artifacts come from academia, either from research papers or projects mainly carried out at research or university institutions.
- *Industrial* when the artifacts belong to actual industrial cases.
- *None* when no artifacts are used for the evaluation, for instance in TOOL papers.

2.4.5. Type of publication fora classification

The classification of publication fora is straightforward because we used the name of the journal, conference, workshop or book where the publication appeared. We included two additional categories for technical reports and student dissertations.

2.5. Data extraction and mapping study

For gathering the data we proceeded with the following steps which gave us the confidence that our data was consistently classified:

1. We created a guideline document defining each of the classification terms and an Excel spreadsheet to collect the classification information. The spreadsheet contained the following data fields: *(i)* SPL life cycle stage, *(ii)* types of artifacts employed by SBSE technique, *(iii)* rationale for the categorization if any, *(iv)* SBSE techniques employed, *(v)* analysis performed, *(vi)* number of case studies evaluated, *(vii)* type of artifacts used in case studies, *(viii)* provenance of the case studies, and *(ix)* a general field for any remarks. We drew a distinction between fields *(ii)* and *(vii)* because in some cases papers only present a running example or do not perform a proper evaluation involving multiple case studies, multiple executions, etc.
2. We formed two groups to carry out the classification task independently.
3. We held a meeting to discuss and pilot the classification terms. For this meeting each group had independently collected, in the data spreadsheets, the information of some selected primary sources. The results obtained were compared and contrasted, and all the discrepancies were analyzed and clarified. We also relied on our previous experience with an earlier version of our work (see [35]) to calibrate our classification.
4. The two teams performed the classification of all primary sources independently.
5. We held a second meeting where the classification for every single paper for each criterion was discussed until a consensus was reached.

[12] http://www.splot-research.org/.

The effort to gather the data varied significantly between papers. For some papers it was a simple task to find all the classification data required because their overall structure, headings, and terms adhered to standard convention. For some other papers, more effort was necessary to dig out this information as they did not have, for instance, a clearly labeled evaluation section. In such cases, it was required to read almost the entire source.

3. Results

In this section we present and describe the results obtained in our systematic mapping study. As mentioned in Sections 2.2 and 2.3, we first performed search queries in specialized repositories and search engines. These queries were performed between 31st July 2014 and 14th August 2014 and produced a total of 2630 hits. As a second step we sieved the articles based on the title, abstract and keywords, resulting in 103 papers. We should remark that the main reason for this large difference (i.e. $2630 - 103 = 2527$) comes from the fact that there is an extensive body of research at the intersection of search-based techniques and product lines *in domains such as manufacturing or management but not for SPLs*. Another reason for the large difference is that many search string terms such as "variability" or even the acronym "SPL" are also used in other domains with a different meaning. For the third step, we looked into the introduction and other relevant parts of the papers. This resulted in the exclusion of 35 papers. Some of the reasons for exclusion were that the focus of those papers was not on SPLs but on standard one-off software systems (e.g. [44]), that they employed techniques such as machine learning that are not considered within SBSE (e.g. [45]), or focused only on a project description (e.g. [46]). For the fourth step we performed snowballing readings, resulting in 3 more articles. The selection of these readings followed the same screening process of the third step. A recent survey on SBSE and SPL was published in association with a keynote at the Software Product Line Conference in 2014 [41]. As a fifth step in our search process we took a detailed look at the papers referenced in this survey. We found 6 new sources. They either came from journal sources we did not consider in our searches or their connection with our search terms did not appear in the abstract or title but instead appeared at other more detailed parts of the papers (e.g. evaluation section). The five steps and their results are summarized in Fig. 3, and Appendix A lists the details of the 77 articles that form the primary sources of our mapping study presented in the order they were found.

The first interesting result of our mapping study is the growth in number of publications as shown in Fig. 4. From 2007 to 2010 we observed a low number of publications, followed by a sharp increase since 2011. For instance, in 2013 the number of publications almost doubled those from 2012. For 2014, we expect this growing trend to continue and to be sustained as the publications for the fall conferences, journals, technical reports, etc. become available.

In the following subsections we shall present the results obtained for each of our research questions, and in Section 4 we provide a more detailed analysis and discussion of our findings.

3.1. Results RQ1—SPL life cycle stages

Fig. 5 shows the number of publications classified along the SPL life cycle stages described in Section 2.4.1 and Table 2 lists the primary sources associated with each category. The first thing to notice is that several publications were categorized in two categories, e.g. S14 categorized in DT and TOOL.

The most frequent stage where SBSE techniques are used was DT (Domain Testing) with 28 publications. The majority of these

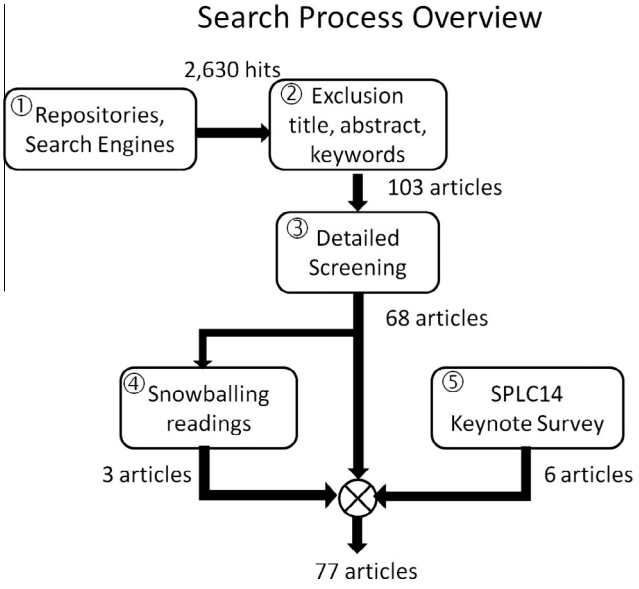


Fig. 3. Steps of search and primary sources selection.

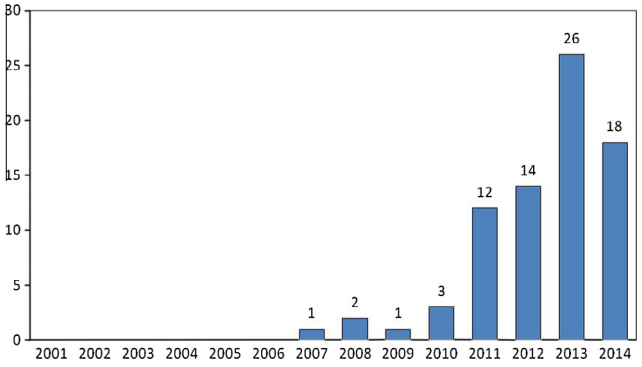


Fig. 4. Publications per year since 2001.

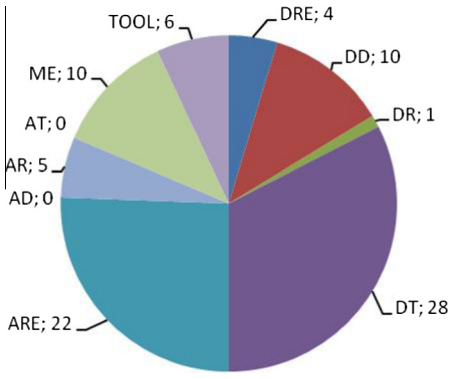


Fig. 5. SPL life cycle stages and publications.

publications focus on computing test suites that cover certain types (e.g. 2-wise or 3-wise) of feature combinations that are derived from feature models. The second most frequent stage was ARE (Application Requirements Engineering) with 22 publications. Many of these publications dealt with optimizing product configurations or derivations with different characteristics and attributes. For DD (Domain Design) and ME (Maintenance and Evolution) we found 10 publications. In the case of DD the focus was mainly in deriving product line architectures, whereas for ME it was on applications for reverse-engineering and for fixing inconsistencies, for instance in feature models. In the TOOL category the main focus was on the analysis and generation of feature models and visualization. In the category of AR (Application Realization) four of its five publications also are categorized in ARE. This indicates a strong relation between both stages. We found 4 publications for DRE (Domain Requirements Engineering) where the main focus was on the correct definition of variability models. We found only 1 publication for DR (Domain Realization) which was also categorized in DD and whose focus was on the building and selection of components for SPLs. It is also important to highlight that for stages AD (Application Design) and AT (Application Testing) no publications were found. In Section 4 we provide a more detailed analysis and discussion of these findings.

3.2. Results RQ2—SBSE techniques used

Table 3 shows the SBSE techniques found by our mapping study, the acronyms we use to refer to them, and the primary sources for each technique. Notice again that a publication can employ more than one SBSE technique, for instance S59 uses GA, EA, and AVM.

Fig. 6a summarizes the number of publications for each SBSE technique. The first thing to notice is the large and diverse set of SBSE techniques used, making a total of 15. Not surprisingly GA (Genetic Algorithm) and MOEA (Multi-Objective Evolutionary Algorithms) came first with 19 publications each. We believe this is because GA is among the most basic evolutionary algorithms and because of the nature of many SPL problems where usually multi-objectives are considered which makes them well-suited for MOEA. The third most frequent technique was GRE (GREedy algorithms) with 13 publications, follwed by SA (Simulated Annealing), EMOA (Exact Multi-Objective Algorithm), DS (Deterministic Search), Evolutionary Algorithm (EA), and Integer Programming (IP). The remaining references were spread out among the remaining 7 techniques.

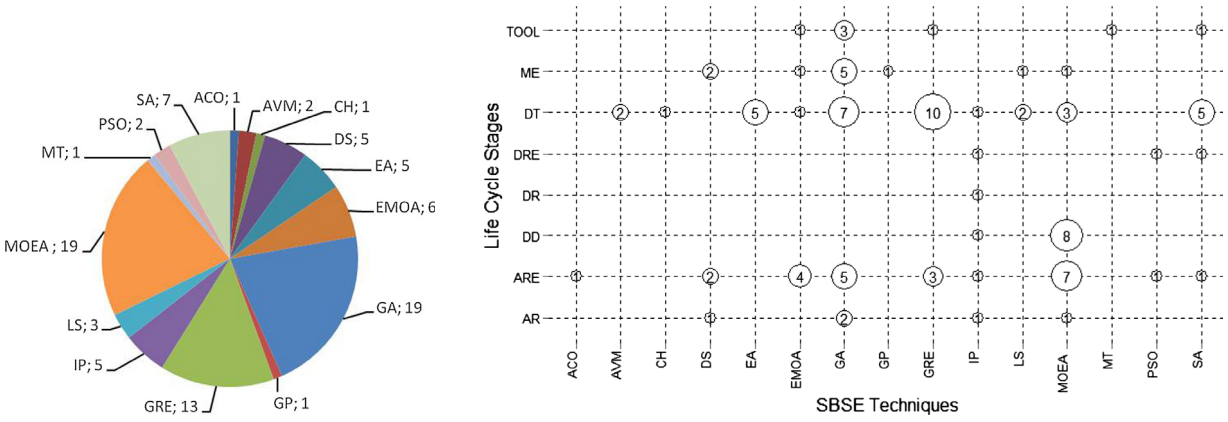
Fig. 6b shows a plot of SBSE techniques and their use broken down by SPL life cycle stages. The first thing to remark is the difference among totals per stage. For instance notice the discrepancy on GRE. In Fig. 6a it has 13 publications associated to it, whereas in Fig. 6b it has 14. The difference comes from the fact that a publication can be associated to more than one life cycle stage. In this concrete example, publication S69 is categorized in DT and TOOL (see Table 2). Fig. 6b corroborates the fact that DT is the stage where SBSE techniques have been used the most. This figure also shows some revealing information. For example, MOEA has so far been primarily used in stages such as DD and ARE and its application remains mostly untapped for DT. Along the same lines, GP (Genetic Programming), CH (Constraint Handling), and Ant Colony

Table 2
Primary sources and life cycle stages.

Stage	Primary sources identifiers
DRE	S8, S22, S38, S77
DD	S2, S5, S26, S28, S44, S48, S50, S60, S66, S67
DR	S50
DT	S1, S9, S11, S12, S14, S15, S17, S18, S23, S24, S27, S35, S37, S39, S41, S42, S47, S53, S54, S58, S59, S61, S62, S63, S64, S65, S69, S76
ARE	S4, S6, S20, S21, S25, S28, S29, S30, S32, S33, S34, S43, S45, S46, S51, S56, S68, S70, S72, S73, S74, S75
AD	None
AR	S21, S33, S51, S55, S70
AT	None
ME	S3, S7, S10, S16, S19, S31, S36, S49, S57, S71
TOOL	S13, S14, S25, S40, S52, S69

Table 3
Primary sources and search based techniques.

Technique	Acronym	Primary sources identifiers
Ant colony optimization	ACO	S72
Alternative variable method	AVM	S59, S61
Constraint handling	CH	S35
Deterministic search	DS	S7, S19, S28, S33, S38
Evolutionary algorithm (1 + 1)	EA	S41, S58, S59, S61, S64
Exact multi-objective algorithm	EMOA	S16, S25, S27, S45, S56, S68
Genetic algorithm	GA	S3, S6, S10, S13, S14, S17, S20, S21, S36, S37, S40, S43, S47, S57, S59, S61, S65, S70, S71
Genetic programming	GP	S57
Greedy algorithm	GRE	S9, S12, S15, S23, S24, S34, S42, S47, S54, S62, S69, S74, S75
Integer programming	IP	S8, S50, S53, S55, S73
Local search	LS	S39, S49, S76
Multi-objective evolutionary algorithm	MOEA	S1, S2, S4, S5, S18, S26, S29, S30, S31, S32, S44, S45, S48, S51, S60, S63, S66, S67, S68
Mutation testing	MT	S52
Particle swarm optimization	PSO	S22, S46
Simulated annealing	SA	S11, S12, S20, S47, S62, S69, S77



(a) SBSE Techniques and Publications

(b) References per Life Cycle Stage and SBSE Technique

Fig. 6. SBSE techniques summary.

Optimization (ACO) have been used only in one publication each. These findings point to potential open areas for further research about which we elaborate more in Section 4.

3.3. Results RQ3—type of comparative analysis

Table 4 shows the primary sources categorized for each type of analysis, while Fig. 7a summarizes their distribution. The *Basic* type of analysis was the most frequent with 32 publications, followed by *Statistical Tests* with 20, *None* with 18, and *Undefined* with 7. Fig. 7b shows the break down of type of analysis per SPL life cycle stage. As before, we should point out the difference among totals per type of analysis. For example, *Undefined* in Fig. 7a has a value of 7 while in Fig. 7b it has a value of 8. In this particular example, paper S28 appears in two stages, DD and ARE (see Table 2), therefore the extra count.

Fig. 7b clearly shows that DT has the majority of the *Statistical Tests* analyses found and that the publications at this life cycle stage at least have a *Basic* type of analysis. Also, this figure shows that most TOOL publications do not provide any form of analysis, which is also expected. In contrast, for stage ARE, the second most common stage (see Fig. 5), the analysis is predominately of *Basic* type.

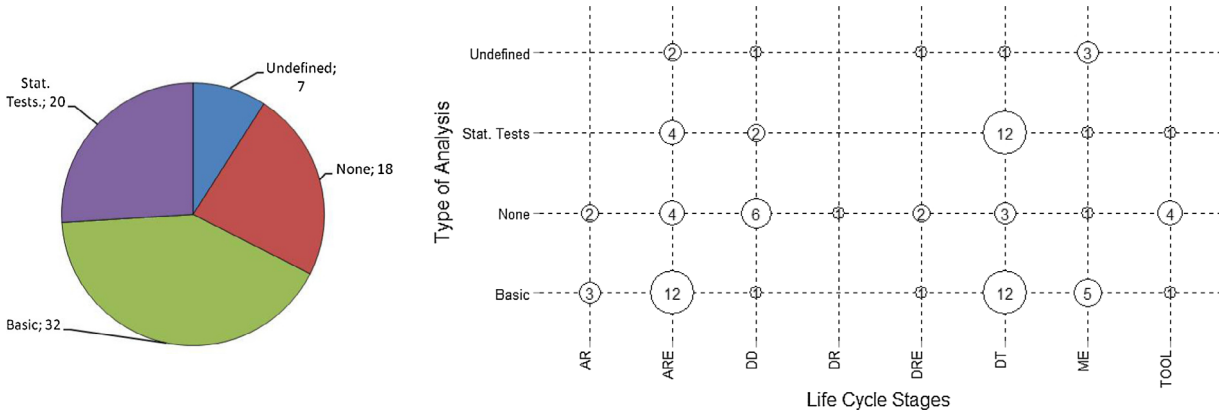
As mentioned before, because of the randomness involved in most of the SBSE techniques, using the adequate statistical analysis is of utmost importance for the results obtained to be reliable and meaningful. Our findings help to raise awareness of the need to employ adequate statistical analysis so that it could be properly addressed by the researchers and practitioners working at the intersection of SPLs and SBSE. We elaborate further on this issue in Section 4.

3.4. Results RQ4—evaluation case studies

Table 5 shows the primary sources categorized for each provenance type and Fig. 8a depicts their number. Academic case studies

Table 4
Primary sources and type of analysis.

Analysis	Primary sources identifiers
Undefined	S8, S9, S10, S28, S29, S31, S71
None	S2, S5, S14, S22, S24, S25, S26, S33, S40, S44, S46, S48, S49, S50, S55, S69, S73, S77
Basic	S3, S4, S6, S7, S12, S15, S16, S19, S20, S21, S23, S27, S32, S34, S35, S36, S37, S38, S41, S51, S52, S53, S54, S56, S58, S62, S67, S68, S70, S72, S74, S76
Stat. tests	S1, S11, S13, S17, S18, S30, S39, S42, S43, S45, S47, S57, S59, S60, S61, S63, S64, S65, S66, S75

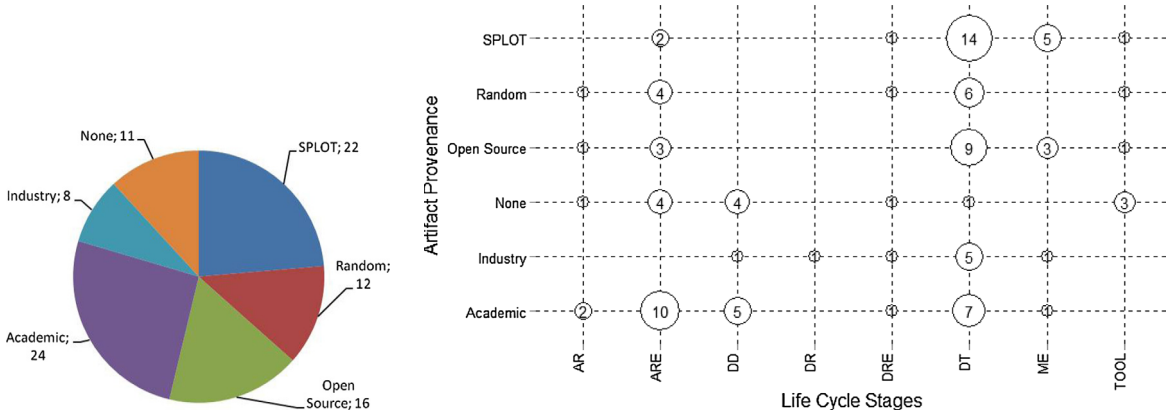


(a) Types of Analysis Used (b) Type of Analysis References per Life Cycle Stages

Fig. 7. Type of analysis summary.

Table 5
Primary sources and case study provenance.

Provenance	Primary sources identifiers
SPLOT	S1, S3, S4, S7, S8, S15, S16, S18, S27, S31 S32, S35, S36, S37, S39, S41, S42, S54, S58, S64, S65, S69
Random	S6, S11, S13, S33, S34, S38, S41, S53, S54, S59, S61, S72
Open source	S9, S10, S11, S12, S19, S23, S24, S30, S39, S41, S51, S52, S64, S65, S71, S74
Academic	S2, S17, S20, S21, S23, S29, S45, S47, S48, S56, S57, S60, S62, S63, S64, S66, S67, S68, S70, S73, S74, S75, S76, S77
Industry	S1, S8, S24, S49, S50, S59, S61, S76
None	S5, S14, S22, S25, S26, S28, S40, S43, S44, S46, S55



(a) Artifacts Provenance (b) Type of Provenance References per Life Cycle Stages

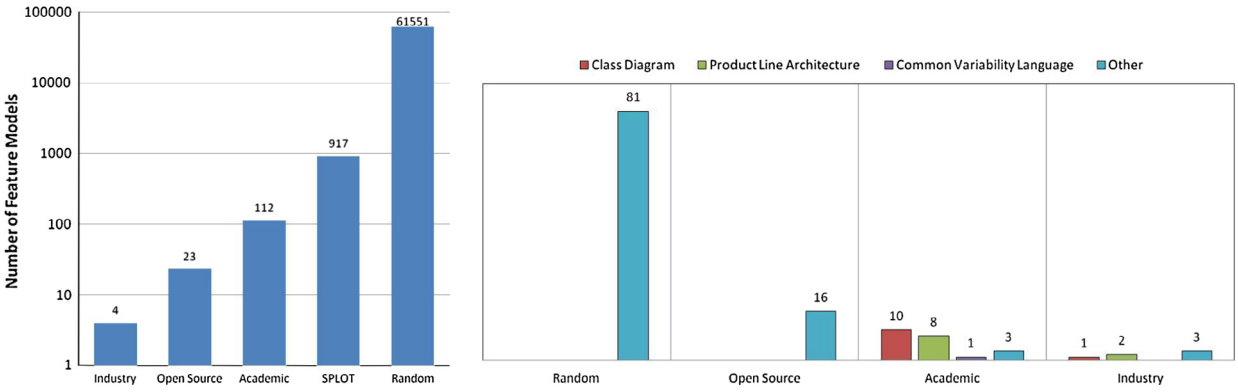
Fig. 8. Evaluation artifacts summary.

were found in 24 publications, followed closely by SPLOT with 22. Open Source case studies were with 16 publications, followed by Random with 12, and None with 11. Last place was occupied by Industry case studies with 8 publications. These numbers indicate a predominance of feature models artifacts as well as a strong need of more case studies stemming from industry, which has been partially addressed through random generation or usually smaller but yet illustrative academic examples.

Fig. 8b shows a plot of artifact provenance broken down by SPL life cycle stages. Again, it is important to remark the difference in the values with the previous figure. For example, for Industry this figure shows 9 references whereas Fig. 8a shows 8. As before, this difference is because a publication can be associated with more than one stage. In our example, paper [S50] is assigned to DD and DR (see Table 2). Fig. 8b highlights that stage DT has the largest number of publications with case studies than any other stage, being SPLOT and Open Source case studies the most frequent ones. In contrast, stage DR has only one publication, namely [S50].

Our mapping study revealed 45 publications which used feature models, 22 publications which used other types of artifacts, and 11 publications with no case studies. Notice here that [S75] had both feature models and other types of artifacts. Considering the dominance of feature models, we performed further analysis on their provenance which is summarized in Fig. 9a. In total 62,607 feature models were used as case studies, of which 61,551 were randomly generated (a single publication [S74] generated 50,000), followed by 917 SPLOT feature models.[13] Far behind were Academic feature models with 112, Open Source with 23, and Industry with 4 feature models. Another interesting finding was the big range of number of

[13] Several SPLOT models were repeatedly used in multiple primary sources.



(a) Feature Model Provenance (b) Non-Feature Model Provenance

Fig. 9. Case studies provenance summary.

Table 6
Publication fora.

Acronym	Primary sources identifiers	Publication name
	Conference publications	
ACSC	S20	Australasian computer science conference
ASE	S30	International conference on automated software engineering
CAISE	S37	Conference on advanced information systems engineering
CEC	S29, S63	IEEE congress on evolutionary computation
COMPSAC	S66	International computers, software & applications conference
ECSA	S70	European conference on software architecture
GECCO	S1, S65	Genetic and evolutionary computation conference
HASE	S35	International IEEE symposium on high-assurance systems engineering
ICSE	S4, S19	International conference on software engineering
ICSM	S27, S51	International conference on software maintenance
ICST	S39, S53, S54	International conference on software testing, verification and validation
ISSTA	S12	International symposium on software testing and analysis
JCC	S44	Chilean computing conference
MODELS	S24	Model driven engineering languages and systems
PIC	S34	International conference on progress in informatics and computing
RE	S36	International requirements engineering conference
SAC	S55	Symposium on applied computing
SSBSE	S2, S3, S57, S58, S59, S60	International symposium search-based software engineering
SCC	S43, S46	International conference on services computing
SEAMS	S21	International symposium on software engineering for adaptive and self-managing systems
SEDM	S28	International conference on software engineering and data mining
SPLC	S17, S18, S22, S23, S25, S61, S62, S67, S68, S74, S75, S76, S77	International conference software product lines
	Journal publications	
IJITDM	S8	International journal of information technology and decision making
ESE	S11	Empirical software engineering
ESWA	S13	Expert systems with applications
IETS	S52	The institution of engineering and technology (IET) software
JSJU	S72	Journal of Shanghai Jiaotong University
JSS	S6, S49, S71	Journal of systems and software
Omega	S50	Omega
PAIS	S73	Practical applications of intelligent systems
PCS	S48	Procedia computer science
TSE	S9, S64	IEEE transactions on software engineering
	Miscellaneous publications	
BC	S38	Book chapter
CMSBSE	S26, S31, S32, S33	International workshop on combining modeling and search-based software engineering
CoRR	S41, S42, S47	Computing research repository
DS@ICSE	S5	Doctoral symposium ICSE
ICSTW	S69	Workshop international conference on software testing, verification and validation
MScThesis	S45	MSC thesis
NFPinDSML	S56	Int. workshop on nonfunctional system properties in domain specific modeling languages
SP@SSBSE	S7	Short paper—fast abstract SSBSE
TOOL@SPLC	S14	Tool track paper at SPLC
TR	S10	Technical report
Vamos	S15, S16, S40	International workshop on variability modeling of software-intensive systems

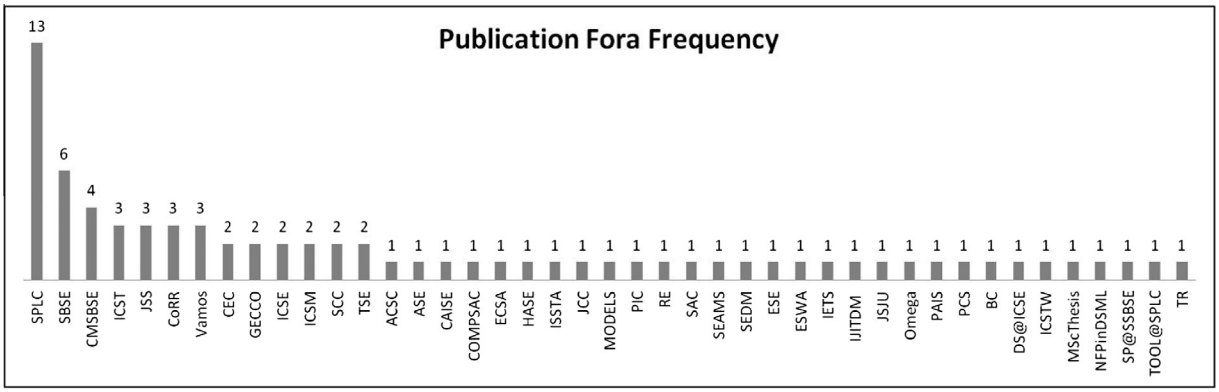


Fig. 10. Publications fora summary.

case studies used. We found as low as 2 case studies, e.g. [S4] which used 2 feature models from SPLOT, and as high as over 50,000 random feature models in article [S74].

We also performed further analysis on the other types of artifacts, besides feature models, used as case studies which is summarized in Fig. 9b. Salient among them were UML class diagrams (CD), Product Line Architecture (PLA) models mostly based on components, and Common Variability Language[14] (CVL) which is an ongoing OMG proposal alternative and complementary to feature models. There were also several ad hoc artifacts used by some approaches, for instance to represent different cost models (labeled as "other" in the figure). In this figure, this latter category accounted for all the Random and Open Source case studies which were predominant over Academic and Industry studies for artifacts CD, PLA and CVL.

3.5. Results RQ5—publication fora

We divided the publications in three groups: conferences, journals, and miscellaneous. The last group to contain book chapters, workshop papers, technical reports, and dissertations. This division was made to reflect the length (short vs large), nature (peer-reviewed or not) and in some cases maturity of the sources analyzed (workshops vs journals). Our study found 22 conference fora, 10 journal fora, and 11 miscellaneous. Table 6 shows the primary sources categorized along these three groups, and Fig. 10 depicts the number of primary sources for each publication forum.

It should not come as a surprise that the most frequent venues for publications are the most prominent conferences for SPL and SBSE communities, respectively SPLC with 13 publications and SSBSE with 6. These were followed with 4 publications by a specialized workshop (CMSBSE) that aims at combining modeling and SBSE research. Vamos (a specialized workshop on SPL) and ICST (a specialized conference in software testing) followed with 3 publications each. Among the journal venues, JSS and TSE were the most frequent ones with 3 and 2 publications respectively. The remaining publications were more or less evenly spread out.

We should remark that out of the 77 primary sources found, 46 correspond to conference fora, 13 journal publications, and 18 miscellaneous publications. We believe these numbers denote on one hand a clear and increasing interest among several communities in the research at the intersection of SBSE and SPLs, but on the other hand they also highlight that this research area is still very young (e.g. very few journal publications and many workshop publications). In the next section we provide an overview of the work identified by our study and sketch some potential avenues for further research.

[14] http://www.omgwiki.org/variability/doku.php.

4. Analysis and discussion

In this section we analyze the findings revealed by our systematic mapping study. For each one, we present an overview of some of the relevant primary sources followed by a concise discussion on open questions and potential areas for further research.

4.1. Predominance of SBSE for SPL testing

Software testing is the most prevalent application realm of SBSE techniques [47,1]. We believe this fact in addition to the recent interest in SPL testing (e.g. [25,24]) is the underlying reason why we found the highest number of publications in this area.

Recall from Section 2.4.1 that in SPL development the testing activities are divided in Domain Testing (DT) and Application Testing (AT). However, our mapping study found 28 publications for DT but none for AT. The natural question is thus: Why is that the case? Though it is not possible to satisfactorily answer this question with the information gathered in our mapping study, we would like to put forward two issues that might potentially serve as starting points for further research to properly address it. The first is that the main distinction between Domain Engineering and Application Engineering lies at their focus; the former focuses on the entire software family while the latter focuses on individual system applications (i.e. members of the software family) [6]. Consequently, the activities at the Domain Engineering level can in principle benefit the most from the application of SBSE techniques because they typically need to cope with, and hence search, properties of a large number of individual system applications. The second is an observation made by Metzer and Pohl, who recently pointed out [42], that in AT the research focus has mostly been on deriving tests from reusable artifacts or minimizing the retesting of parts already tested for other applications, in a similar way to regression testing. Again, we reiterate that these issues might serve as starting points on the quest to shed light on the reason behind our finding.

The majority of testing publications found focus on *Combinatorial Interaction Testing (CIT)* whose goal is to compute test suites called *covering arrays* such that their products contain *all* possible combinations of t selected and unselected features based on the domain constraints expressed by the SPL's feature model. Each of these valid combinations is called a *t-wise set* [S15]. The most typical case was *pairwise covering arrays* whose value of t is 2. These arrays must include products that have for any two features `A` and `B` in the feature model the valid combinations from: both features selected, both features not selected, `A` selected and not selected `B`, and `A` not selected and `B` selected.

We now provide a short overview of some of the publications found by our study. Based on Fig. 6b it is clear that greedy

algorithms are the most common SBSE technique used for DT testing (e.g. [S9,S12,S23]). In second place is GA with 7 references. An example in this category is the work by Ensan et al. that uses an objective function based on cyclometric complexity metric adapted to feature models [S37]. In third place there was a tie between SA and EA with 5 publications. An example using SA to compute covering arrays is the work by Garvin et al., whose tool named CASA performs three nested search strategies aiming at iteratively reducing the sizes of the test suites [S11]. An example of EA is the work by Henard et al. that uses a similarity metric as fitness function and shows how to overcome scalability issues for generating covering arrays beyond pairwise [S41,S64].

Our systematic mapping study also helped unveil interesting research threads described next for which we sketch open issues and avenues worthy of further investigation.

Test suite prioritization. Prioritization determines the order of the products in a test suite according to some criteria. In this sense, the work by Al-Hajjaji et al. propose a similarity-based prioritization approach [S62], while Sánchez et al. compare five SPL-specific prioritization criteria and analyze their effect in detecting faults in order to provide faster feedback and reduce debugging efforts [S54]. In the SPL domain, the notion of prioritization also considers how the test suites are actually computed based on the feature combinations expressed by feature models. For instance, Johansen et al. propose a greedy algorithm that adds weights to products to guide the computation of the t-wise sets [S24]. An alternative parallel evolutionary algorithm was proposed by Lopez-Herrejon et al. for this scheme that can produce smaller test suites [S65].

Test suite prioritization has an extensive existing body of research for single software systems (for an account see for example [48]) which has not been extensively explored within the context of SPLs. Among the salient open issues are: adequate and adapted coverage criteria for SPLs, combination with clustering techniques, and cost values stemming for example from non-functional properties.

Multi-objective optimization. Quite often SPL problems require the optimization of multiple and sometimes contradicting objectives. The typical examples for SPL test suites are the minimization of their sizes and the maximization of their t-wise coverage.

The work by Wang et al. presents an approach to minimize test suites using weights in the fitness function [S1], that is, it uses a *scalarizing function* that transforms a multi-objective problem to a single-objective one [49]. A similar approach was taken by Henard et al. who also flatten many objectives using a scalarizing function [S19]. We should point out, however, that there is an extensive body of work on the downsides of scalarization in multi-objective optimization (e.g. [50]). Among the shortcomings are the fact that weights may show a preference of one objective over the other and, most importantly, the impossibility of reaching some parts of the Pareto front when dealing with convex fronts.

In contrast, Lopez-Herrejon et al. propose an exact algorithm that computes the true Pareto front of feature models using SAT solvers that presents scalability issues for larger feature models [S27], and also make a comparison of four classical multi-objective evolutionary algorithms (i.e. NSGA-II, PAES, MOCell, and SPEA2) for the computation of pairwise testing and analyzed three different seeding strategies for the initial population [S63].

There is again a wealth of research in multi-objective optimization that remains largely untapped, for an overview see for example [37,38]. Among the possibilities for further investigation are: employing other multi-objective evolutionary algorithms, including more optimization objectives that consider information such as control-flow or non-functional properties, and extending and adapting for the realm of SPL testing the definitions and interpretations of standard quality indications such as hypervolume [51] or generational distance [52] (e.g. what are hypervolume values are acceptable for a concrete SPL testing task?, could they be inferred from the feature models?, how could they be meaningfully interpreted by the software engineer who needs to make a decision based on them?).

Exploiting more SPL knowledge. Because of the typically large number of individual systems of a SPL, any information that could be exploited to reduce the search effort is worth of consideration. For example, Haslinger et al. leverage information from feature models to speed up the computation of covering arrays by eliminating redundant t-sets [S15,S42]. Some other examples are the work by Xu et al. that exploits static analysis techniques for achieving coverage more effectively [S17], and the work by Lopez-Herrejon et al. that studies seeding strategies [S63]. In this area, there is a recent surge of software analysis approaches specially developed for SPLs, surveyed in [53], that could also be leveraged for this purpose.

Need of community-wide testing benchmarks. Apart from randomly generated artifacts, our systematic mapping study found that feature models from the SPLOT repository were the most common type of artifacts and were predominantly used for testing. In total, we accounted for 917 SPLOT feature models. The SPLOT repository contained 538 feature models at the time of writing.[15] Hence several SPLOT feature models have been repeatedly used across some of the papers we found. However, the selection of which feature models to analyze in each paper seemed to be arbitrary, at worst, or partially-justified, at best. Despite incipient attempts to compare between testing approaches [54], there is a strong need for a community-wide testing benchmark that could effectively help to fairly analyze all the different proposals. A first step towards such a benchmark is advocated in [S47].

4.2. SBSE for product configuration

Recall that Application Requirements Engineering (ARE) deals with the creation and management of the requirements of the individual systems that are part of a SPL [6]. One of the most prominent tasks in ARE is *product configuration* whereby engineers or users select the requirements or features desired for their systems based on multiple and sometimes conflicting preferences that come from the distinct stakeholders involved throughout the entire development process. Product configuration naturally lends itself to the application of SBSE techniques because of the vast number of combinations SPL requirements can typically have. Hence ARE is one of the stages with some of the first applications of SBSE techniques to SPLs. For instance, Guo et al. employ a genetic algorithm that considers resource constraints such as cost or memory [S6], and Shi et al. use a greedy search to find optimal feature selections [S34].

Pascual et al. also employ a genetic algorithm but apply it to the runtime configuration and adaptation of mobile applications [S21,S70]. A similar goal is pursued by Sanchez et al. but instead they use feature model metrics to optimize the configurations [S33]. Considering that usually many objectives should be simultaneously optimized, there has been significant work on using multi-objective approaches. Among them, Cruz et al. employ the multi-objective algorithm NSGA-II to create and manage product portfolios based on customer satisfaction and costs [S29]. Sayyad et al. perform a more exhaustive application and analysis of multi-objective evolutionary algorithms for configuration tasks [S4,S30,S32]. Similarly, the work of Olaechea et al. proposes an exact method to compute Pareto fronts showing their capability to handle small and medium size problems and provide guidelines for choosing either exact or evolutionary approaches [S56,S45,S68].

[15] Consulted on August 21th, 2014.

Configuration is also performed in more specialized artifacts such as business process families [S43], in other development paradigms such as SPLs based on service orientation [S46], or to *miniaturize* software so that it fits into different resource-constrained hardware platforms [S51].

In contrast with ARE, the Application Realization (AR) stage focuses on the actual realization of those individual systems based on the configured or selected requirements [6]. Thus, it is not surprising that almost all the primary resources categorized into the AR stage were also categorized into ARE (see Table 2). We believe that this is because once a product is configured, a natural following step is to realize it using assets developed during the Domain Engineering activities. The only exception we found was S55 that deals with the realization of SPLs with service orientation.

The most pressing need in this area is the availability of more extensive and robust tool support, as our mapping study found only one such tool in the work by Murashkin et al. [S25]. Requirements for adequate tool support for configuration in SPLs have been collected by Rabiser et al. [22].

4.3. Need to improve empirical evidence robustness

Our systematic mapping study also revealed an undesirable finding, the lack of thorough empirical analysis with an emphasis on using adequate statistical tests. This is specially critical because many of the SBSE techniques rely on randomness. Another example is the use of a small number of feature models, as low as one or two, from which general conclusions cannot confidently be drawn. Except for the publications whose main purpose was either presenting a tool or for short papers that describe on-going work, this should not be an acceptable practice. This lack of proper empirical analysis has also been observed across multiple areas of Software Engineering and SBSE and guidelines have been proposed to address this issue (see [20,16]).

Fig. 11 shows the type of analysis by year. Relevant in this figure is the number of articles classified under the category of Statistical Tests. We should point out that the first guideline paper for SBSE techniques authored by Arcuri and Briand appeared in mid 2011 [55]. Out of the 20 papers under this category, 12 refer to either this guideline or its journal version [20], 2 use other guidelines, and 6 do not make reference to any concrete guideline article. This finding shows the positive impact that guidelines can have, for instance, for improving the quality of some aspects of the empirical analysis. Another striking find was the low number of industrial and open source case studies. Increasing their number can only help to strengthen the empirical robustness of this area of research.

4.4. Need of better tooling support

Adoption of new ideas and techniques can either be helped or hampered by the underlying tool support. Our systematic mapping study found only six publications whose main contribution was providing tool support. Work by Segura and colleagues presents a tool for the generation of hard feature models to test feature model analysis tools [S13,S40,S52]. Henard et al. present a tool called PLEDGE that provides a product line editor and a generator of t-wise covering arrays [S14]. A similar tool, CITLab is presented by Calvagna et al. whose main goal is integrating multiple CIT approaches for SPLs [S69]. Murashkin et al. present a tool for the visualization and exploration of variants in a multi-dimensional space [S25].

We need to say that several publications do make their source code and related artifacts openly available. However, for our study we only considered those that, by being presented as tool papers (e.g. in tool tracks at conferences), were considered, by their own authors, to be mature enough to be used by a wider community. A detailed comparison of tool availability is outside the scope of our mapping study, but nonetheless a worthy item for future work.

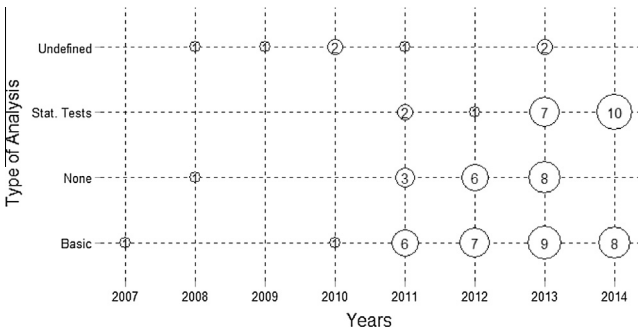


Fig. 11. Type of analysis articles per year.

4.5. SBSE for SPL maintenance and evolution

SPLs, as any software system, need to be maintained and evolved in order to keep up with market, societal, and technological changes. SPL maintenance and evolution pose an interesting and unique set of challenges, an overview of them is presented by Laguna and Crespo [28].

Lopez-Herrejon et al. proposed a genetic algorithm to reverse engineer feature models from sets of valid feature combinations [S3]. This work has been further extended to use genetic programming [S57]. The work by Yi et al. has a similar goal while focusing on mining binary cross-tree constraints [S36]. Lopez-Herrejon and Egyed used a deterministic search to assess the complexity of fixing model inconsistencies [S7,S16]. A similar goal is presented in the work by Henard et al. whose approach aims to test and fix feature models [S19]. Ullah and colleagues present COPE+, an approach for creating product portfolios from single products by taking into account the customer preferences [S10,S71]. Karimpour and Ruhe propose a bi-objective approach for adding and selecting new features in existing SPLs [S31]. Del Rosso presents an approach for tuning performance based on product family architectures particularly in the domain of embedded systems [S49].

Salient among the open questions in this area is the study of reverse engineering tasks that simultaneously involve several artifacts. For example, reverse engineering feature models and architectural models such that the set of combinations denoted by the feature models can actually be realized by the architectural components. Application of co-evolutionary (see [56]) approaches might prove useful to address this question.

Another interesting avenue is exploring further into the wealth of results in the area of the Next Release Problem[16] (e.g. [58,59]) in combination with recent work on SPL economics (e.g. [60,61]), which may help to address the most common development scenario of SPLs whereby multiple existing variants that must be consolidated into a SPL infrastructure to effectively address their maintenance problems [62].

4.6. SBSE for SPL domain design

Recall that Domain Design (DD) is the sub-process of Domain Engineering where a Product Line Architecture (PLA) is developed [6]. Few works rely on specialized models for this task. For example, integrated decision models [S28], combinations of models such

[16] This problem consist of determining which features—in the general sense of software development and not in relation to SPLs—should be added to the next software release of existing systems based on user requirements while meeting resource constraints [57].

as Petri Nets and process models [S48], and cost-based models [S50]. However, most of the work has focused on UML-based class and component diagrams. The work by Colanzi and colleagues has focused on obtaining Product Line Architectures using novel cohesion measures tailored to SPLs [S2,S5]. They also performed a study of different SBSE representations used for software architectures identifying their drawbacks, and proposed a representation based on a PLA metamodel for which they define a set of search operators [S26]. Their group has also applied design patterns as mutation operators [S44,S60], and a specialized feature-driven crossover operator [S66].

Some of the open issues in this area are: *(i)* studying the advantages and disadvantages of genetic programming for the representation of PLA metamodels (e.g. along the lines suggested in [S57]), *(ii)* using SPL specific refactorings beyond the GoF patterns (e.g. [63,64]) and recent advances in search-based refactoring (e.g. [65]), *(iii)* exploring and adapting results from software module clustering (e.g. [66,67]), and *(iv)* performing empirical user studies to compare against manually produced PLAs (e.g. [68]).

4.7. Genetic improvement for SPL

Genetic Programming (GP) is a form of evolutionary computation that employs tree-based representation of computer programs whose fitness is determined on how well the encoded programs solve a computational problem [69]. Despite the extensive research on GP, our study only found one article that applies GP for SPLs [S57]. Within SBSE, GP has been turned into a technique referred to as *Genetic Improvement (GI)* whose aim is improving existing programs rather than evolving them from scratch as proposed by GP [41]. GI has been successfully used, for instance, to specialize boolean satisfiability (SAT) programs written in C++ for concrete tasks such as combinatorial interaction testing [70], repair broken functionality [71], or add new functionality [72].

In their keynote paper, Harman and colleagues advocate the potential use of GI for SPL development [41]. For instance, to generate a Pareto surface of programs that exhibit the same functionality but with different quality attributes such as performance. Another possibility they propose is actually growing new functionality, hence creating a new branch or variant product of a SPL. Crystallizing these visions would require extensive research that exploits current results of variability mining [73] and SPL evolution [28], for instance in migrating product variants to SPLs [74,75]; studying the different approaches employed for this task can shed light on how they could be automated by means of GI.

5. Threats to validity

We faced similar validity threats as any other systematic mapping study. A threat to validity is the selection of the search queries. We addressed this threat with a carefully chosen selection of search terms based on previous studies on the core research areas of our systematic mapping study, SPLs and SBSE. The SPL search terms were collected and aggregated from 12 systematic mapping studies and literature reviews. The SBSE search terms were based on three studies which we extensively complemented with terms from recent developments. The selection of SBSE terms may appear to lack other classical operations research algorithms that in principle could be applicable to SPLs. Similarly, the selection of generic search terms such as "search based" relies on our subjective experience with the SBSE literature. However, we would argue that our familiarity with the publication fora and the consistency between our results and Harman et al.'s survey (see [41] and Section 6) give us confidence that we did not miss, incorrectly classify or mistakenly include primary sources.

A second threat to validity comes from how the search for primary sources was carried out. We searched two specialized community repositories and employed five standard bibliography search engines. We consider that these sources cover all possible publicly available publication venues within both SPL and SBSE communities.

A third threat to validity is the selection of criteria for inclusion and exclusion. In order to get a better and more comprehensive picture of the state of the research at the intersection of SPL and SBSE we included all types of publications, i.e. from non-peer-reviewed to journal publications, provided that they clearly apply at least one SBSE technique to a SPL problem.

A fourth threat to validity is our classification scheme. We addressed this threat for the life cycle stages classification by selecting stage terms and definitions based on a well-known framework within the SPL community. We added new terms to our classification to include maintenance, evolution, and tooling issues not explicitly considered as a stage by this framework. To classify the level of analysis we defined our terms based on the guidelines proposed by Arcuri and Briand [20]. For the classification of case studies we considered the most common sources of both provenance and artifact types used within the SPL community.

A fifth threat to validity refers to the way the data was extracted for creating the mapping study. We took the following steps to address this threat. First, we created a guideline document defining each of the classification terms and an spreadsheet to collect the classification information. Second, we formed two groups for the classification task. Third, we held a meeting to discuss and pilot the classification terms prior to performing the classification. At this meeting, the criteria were tested and calibrated using some actual papers that had been collected. Fourth, the two teams performed the classification independently. Fifth, we held a second meeting where the classification for every single paper for each criterion was discussed until a consensus was reached. These steps gave us the confidence that our data was consistently classified.

In addition, we further performed many manual and automated checks and verifications to revalidate the consistency and accuracy of our data while we were aggregating the information and preparing the graphs and figures presented in this work.

6. Related work

In this section we briefly summarize the surveys and studies carried out in either SPLs or in SBSE.

SBSE surveys. The survey by Harman et al. presents a general overview of SBSE techniques and the areas where it has been employed [1]. Freitas et al. performed a bibliometric analysis of SBSE [47]. Their goal was to identify trends in the number of publications, the publication fora, the authorship and collaborations among members of the SBSE community. In contrast with our work, they have a different focus, namely general software engineering and not SPLs. Ali et al. performed a systematic review of empirical investigation of search-based test case generation techniques [34]. Similar to ours their review assessed how the found techniques were empirically evaluated, but in contrast their work focused exclusively on testing and for non-SPL software systems. Similarly, the work by Afzal et al. performed a systematic study on testing of non-functional search based testing [33], that is, for properties such as safety, usuability, or quality of service.

Mark Harman and his colleagues carried out a survey on SBSE and SPL for a keynote talk at the Software Product Line Conference 2014. In sharp contrast with our work, their goal was to provide a bird's-eye view at the intersection of both fields and not to perform a detailed systematic mapping study as in our case. Hence their work does not provide a detailed classification or analysis along

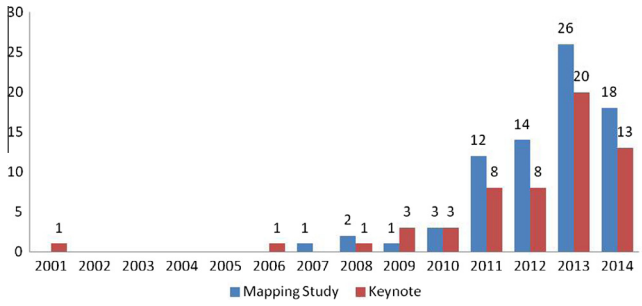


Fig. 12. Publications per year comparison.

research questions as our study does. Fig. 12 contrasts the number of publications found by their survey in comparison with our study. Overall their results are consistent with our findings and indicate an increasing interest in the area.[17]

The results presented in this paper are a substantial extension of an earlier draft [35]. The earlier version helped us pilot the research questions, fine tune the search terms for our queries, and provided insight into the classification scheme. The results between both works are completely consistent, for instance the first 42 primary sources were already identified in the earlier draft.

SPL surveys. There has been many recent systematic mapping studies and systematic literature reviews in this area. Here we summarize the studied areas.

Bastos et al. perfomed a mapping study SPL adoption [23]. Their work identified four adoption strategies and 23 barriers that can hinder SPL adoption in industrial projects.

Another mapping study has been done by da Silva on agile methods and SPL. et [26]. Their study found that most of the applications of agile methods follow XP or Scrum and identified SPL practices that can be exploited by Agile techniques.

A systematic literature review on requirements engineering and SPL was performed by Alves et al. [21]. Their work indicates that the application of requirements engineering techniques for SPL was still not mature as most of the case studies used were indeed toy examples. Thus they advocate that more empirical studies should be performed to address this serious limitation and hence improve the rigor, credibility, and validity of the proposed approaches.

Mohabbati et al. performed a mapping study on service orientation and SPLs. [29]. They conclude that despite the increasing interest in combining both areas there are still a lot of ground to explore. Mahdavi-Hezavehi et al. performed a more focused study on service orientation and variability in quality attributes [30]. They found that most of the current works focus on performance and availability whereas other attributes are mostly disregarded and are not in industrial settings.

Recently two systematic mapping studies in SPL testing have appeared [25,24], both of which attest that the application of SBSE techniques for SPL testing is an area ripe for research that needs to be further explored. The respective authors further compared the results against each other yielding interesting conclusions on the reliability of mapping studies [76]. A recent literature review categorized SPL testing strategies into two fundamental aspects: the selection of the products to test, and the actual testing of the products [31]. Despite the extensive work carried out on both of these aspects, the authors found that there is still a great lack of empirical industrial applications.

[17] Their survey does not always distinguish supporting references and works at the intersection of SPLs and SBSE, so we cannot accurately know which of their references were counted for each year. This explains the minor discrepancies in years 2001 and 2009.

Laguna et al. performed a systematic mapping study on SPL evolution [28], where they made an assessment of the maturity level of techniques to migrate individual systems or groups of software variants into SPLs.

Chen et al. performed a systematic review of variability management [77]. Their findings indicate that a large majority of the reported approaches have not been sufficiently evaluated using scientifically rigorous methods (e.g. empirical studies [16]), overall the available evidence is quite sparse and its quality quite low. Similarly, Galster et al. performed a thorough systematic literature review of variability in software systems [32]. Among their salient findings, they identified that software quality attributes have not received much attention and that testing is under-represented. In addition, they put forward a set of dimensions to unify and integrate the otherwise disperse research on variability in software systems.

Rabiser et al. performed a systematic review of requirements for supporting product configuration [22], whereas Holl et al. carried out a systematic review of the capabilities to support multi product lines [27]. In contrast with all these SPLs studies our work has a novel and distinct focus.

7. Conclusions and future work

In this paper we present the results of the first systematic mapping study on the application of SBSE techniques to SPL problems. Our study corroborates the increasing interest in applying this type of techniques as shown by the number of recent publications. The most common application is for testing at the Domain Engineering level, for example in computing test suites in Combinatorial Interaction Testing. The most common techniques used are genetic algorithms and multi-objective evolutionary algorithms. Our work identified a need to improve evaluations with a more adequate empirical methodology and adequate statistical analysis.

Our work also revealed research areas and possible opportunities. For example, developing adequate community-wide testing benchmarks, exploiting more SPL knowledge to reduce the search effort, the strong need to provide robust tooling support, etc. We also inadvertently found that there is a wealth of research literature on product line scoping and design in the area of manufacturing and marketing that relies on SBSE techniques. This begs the question if any of the research done in those areas can be applicable to SPLs. We hope that this mapping study not only serves to highlight the main research topics at the intersection of SBSE and SPLs but that it also serves to entice both researchers and practitioners to explore the great synergy potential that these two research areas can offer to the Software Engineering community at large.

Acknowledgements

This research is partially funded by the Austrian Science Fund (FWF) Projects P25289-N15, P25513-N15, and Lise Meitner Fellowship M1421-N15. We thank Javier Ferrer, Francisco Chicano, and Enrique Alba for their help with the earlier versions of this work.

Appendix A. Primary sources

A.1. References list

[S1] Shuai Wang, Shaukat Ali, and Arnaud Gotlieb. Minimizing test suites in software product lines using weight-based genetic algorithms. In GECCO, pages 1493–1500, 2013.

[S2] Thelma E. Colanzi and Silvia R. Vergilio. Applying search based optimization to software product line architectures: Lessons learned. In Proceedings of SSBSE, volume 7515 of LNCS, pages 259–266, 2012.

[S3] Roberto Erick Lopez-Herrejon, José A. Galindo, David Benavides, Sergio Segura, and Alexander Egyed. Reverse engineering feature models with evolutionary algorithms: An exploratory study. In Proceedings of SSBSE, volume 7515 of LNCS, pages 168–182, 2012.

[S4] Abdel Salam Sayyad, Tim Menzies, and Hany Ammar. On the value of user preferences in search-based software engineering: a case study in software product lines. In Proceedings of ICSE, pages 492–501, 2013.

[S5] Thelma E. Colanzi. Search based design of software product lines architectures. In ICSE, pages 1507–1510, 2012.

[S6] Jianmei Guo, Jules White, Guangxin Wang, Jian Li, and Yinglin Wang. A genetic algorithm for optimized feature selection with resource constraints in software product lines. Journal of Systems and Software, 84(12):2208–2221, 2011.

[S7] Roberto E. Lopez-Herrejon and Alexander Egyed. Searching the variability space to fix model inconsistencies: A preliminary assessment. In SSBSE 2011, 2011.

[S8] Zhiqiao Wu, Jiafu Tang, C.K. Kwong, and Ching-Yuen Chan. An optimization model for reuse scenario selection considering reliability and cost in software product line development. Intl. Jnl. of Inform. Tech. and Decision Making, 10(5):811–841, 2011.

[S9] Myra B. Cohen, Matthew B. Dwyer, and Jiangfan Shi. Constructing interaction test suites for highly-configurable systems in the presence of constraints: A greedy approach. IEEE Trans. Software Eng., 34(5):633–650, 2008.

[S10] Muhammad Irfan Ullah. Cope+: A method for design and evaluation of product variants. Technical Report SERG-2009-03, August 2009.

[S11] Brady J. Garvin, Myra B. Cohen, and Matthew B. Dwyer. Evaluating improvements to a meta-heuristic search for constrained interaction testing. Empirical Software Engineering, 16(1):61–102, 2011.

[S12] Myra B. Cohen, Matthew B. Dwyer, and Jiangfan Shi. Interaction testing of highly-configurable systems in the presence of constraints. In ISSTA, pages 129–139, 2007.

[S13] Sergio Segura, José Antonio Parejo, Robert M. Hierons, David Benavides, and Antonio Ruiz Cortés. Automated generation of computationally hard feature models using evolutionary algorithms. Expert Syst. Appl., 41(8):3975–3992, 2014.

[S14] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, and Yves Le Traon. Pledge: a product line editor and test generation tool. In SPLC Workshops, pages 126–129, 2013.

[S15] Evelyn Nicole Haslinger, Roberto E. Lopez-Herrejon, and Alexander Egyed. Using feature model knowledge to speed up the generation of covering arrays. In VaMoS, page 16, 2013.

[S16] Roberto E. Lopez-Herrejon and Alexander Egyed. Towards fixing inconsistencies in models with variability. In Proceedings of VaMoS, pages 93–100, 2012.

[S17] Zhihong Xu, Myra B. Cohen, Wayne Motycka, and Gregg Rothermel. Continuous test suite augmentation in software product lines. In Proceedings SPLC, pages 52–61, 2013.

[S18] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, and Yves Le Traon. Multi-objective test generation for software product lines. In Proceedings of SPLC, pages 62–71, 2013.

[S19] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, and Yves Le Traon. Towards automated testing and fixing of re-engineered feature models. In Proceedings of ICSE, pages 1245–1248, 2013.

[S20] Lei Tan, Yuqing Lin, Huilin Ye, and Guoheng Zhang. Improving product configuration in software product line engineering. In 36th Australasian Computer Science Conference, ACSC '13, pages 125–133, 2013.

[S21] Gustavo G. Pascual, Mónica Pinto, and Lidia Fuentes. Run-time adaptation of mobile applications using genetic algorithms. In SEAMS, pages 73–82, 2013.

[S22] Hadi Serajzadeh and Fereidoon Shams. The application of swarm intelligence in service-oriented product lines. In SPLC Workshops, page 12, 2011.

[S23] Martin Fagereng Johansen, Øystein Haugen, and Franck Fleurey. An algorithm for generating t-wise covering arrays from large feature models. In SPLC (1), pages 46–55, 2012.

[S24] Martin Fagereng Johansen, Øystein Haugen, Franck Fleurey, Anne Grete Eldegard, and Torbjørn Syversen. Generating better partial covering arrays by modeling weights on sub-product lines. In MoDELS, pages 269–284, 2012.

[S25] Alexandr Murashkin, Michal Antkiewicz, Derek Rayside, and Krzysztof Czarnecki. Visualization and exploration of optimal variants in product line engineering. In Proceedings of SPLC, pages 111–115, 2013.

[S26] Thelma E. Colanzi and Silvia R. Vergilio. Representation of software product line architectures for search-based design. In Proceedings of CMSBSE@ICSE, pages 28–33, 2013.

[S27] Roberto E. Lopez-Herrejon, Francisco Chicano, Javier Ferrer, Alexander Egyed, and Enrique Alba. Multi-objective optimal test suite computation for software product line pairwise testing. In ICSM, pages 404–407. IEEE, 2013.

[S28] Zhiqiao Wu, Jiafu Tang, and Xiaoqing Wang. Integrated design of production strategy and reuse scenario for product line development. In SEDM, pages 69–74, June 2010.

[S29] Jonathas Cruz, Pedro Santos Neto, Ricardo Britto, Ricardo Rabelo, Werney Ayala, Thiago Soares, and M. Mota. Toward a hybrid approach to generate software product line portfolios. In IEEE Congress on Evolutionary Computation, pages 2229–2236, 2013.

[S30] Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, and Hany Ammar. Scalable product line configuration: A straw to break the camel's back. In ASE, pages 465–474. IEEE, 2013.

[S31] Reza Karimpour and Günther Ruhe. Bi-criteria genetic search for adding new features into an existing product line. In Proceedings of CMSBSE@ICSE, pages 34–38, 2013.

[S32] Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, and Hany Ammar. Optimum feature selection in software product lines: Let your model and values guide your search. In Proceedings of CMSBSE@ICSE, pages 22–27, 2013.

[S33] Luis Emiliano Sanchez, Sabine Moisan, and Jean-Paul Rigault. Metrics on feature models to optimize configuration adaptation at run time. In Proceedings of CMSBSE@ICSE, pages 39–44, 2013.

[S34] Runyu Shi, Jianmei Guo, and Yinglin Wang. A preliminary experimental study on optimal feature selection for product derivation using knapsack approximation. In PIC, pages 665–669, 2010.

[S35] Linbin Yu, Feng Duan, Yu Lei, Raghu Kacker, and D. Richard Kuhn. Combinatorial test generation for software product lines using minimum invalid tuples. In HASE, pages 65–72. IEEE Computer Society, 2014.

[S36] Li Yi, Wei Zhang, Haiyan Zhao, Zhi Jin, and Hong Mei. Mining binary constraints in the construction of feature models. In RE, pages 141–150, 2012.

[S37] Faezeh Ensan, Ebrahim Bagheri, and Dragan Gasevic. Evolutionary search-based test generation for software product line feature models. In CAiSE, pages 613–628, 2012.

[S38] Wei Zhang, Haiyan Zhao, and Hong Mei. Binary-search based verification of feature models. In Top Productivity through Software Reuse, volume 6727 of LNCS, pages 4–19. 2011.
[S39] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, and Yves Le Traon. Assessing software product line testing via model-based mutation: An application to similarity testing. In ICST Workshops, pages 188–197, 2013.
[S40] Sergio Segura, José A. Galindo, David Benavides, José Antonio Parejo, and Antonio Ruiz Cortés. BeTTy: benchmarking and testing on the automated analysis of feature models. In Proceedings of VaMoS, pages 63–71, 2012.
[S41] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, Patrick Heymans, and Yves Le Traon. Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test suites for large software product lines. CoRR, abs/1211.5451, 2012.
[S42] Evelyn Nicole Haslinger, Roberto E. Lopez-Herrejon, and Alexander Egyed. Improving casa runtime performance by exploiting basic feature model analysis. CoRR, abs/1311.7313, 2013.
[S43] Ivana Ognjanovic, Bardia Mohabbati, Dragan Gasevic, Ebrahim Bagheri, and Marko Boskovic. A metaheuristic approach for the configuration of business process families. In Louise E. Moser, Manish Parashar, and Patrick C.K. Hung, editors, IEEE SCC, pages 25–32. IEEE, 2012.
[S44] Giovani Guizzo, Thelma Elita Colanzi, and Silvia Regina Vergilio. Applying design patterns in product line search-based design: Feasibility analysis and implementation aspects. In Proceedings of the Chilean Computing Conference (JCC '13), Temuco, Chile, 11–15 November 2013.
[S45] Rafael Ernesto Olaechea Velazco. Comparison of exact and approximate multi-objective optimization for software product lines. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, 2013.
[S46] Hongxia Zhang, Rongheng Lin, Hua Zou, Fangchun Yang, and Yao Zhao. The collaborative configuration of service-oriented product lines based on evolutionary approach. In IEEE SCC, pages 751–752. IEEE, 2013.
[S47] Roberto E. Lopez-Herrejon, Javier Ferrer, Francisco Chicano, Evelyn Nicole Haslinger, Alexander Egyed, and Enrique Alba. Towards a benchmark and a comparison framework for combinatorial interaction testing of software product lines. CoRR, abs/1401.5367, 2014.
[S48] Renzhong Wang and Cihan H. Dagli. Computational system architecture development using a holistic modeling approach. Procedia Computer Science, 12(0):13 – 20, 2012. Complex Adaptive Systems 2012.
[S49] Christian Del Rosso. Software performance tuning of software product family architectures: Two case studies in the real-time embedded systems domain. Journal of Systems and Software, 81(1):1–19, 2008.
[S50] Jiafu Tang, Wu Zhiqiao, C.K. Kwong, and Xinggang Luo. Integrated production strategy and reuse scenario: A cofaq model and case study of mail server system development. Omega, 41(3):536 – 552, 2013.
[S51] Nasir Ali, Wei Wu, Giuliano Antoniol, Massimiliano Di Penta, Yann-Gaël Guéhéneuc, and Jane Huffman Hayes. Moms: Multi-objective miniaturization of software. In ICSM, pages 153–162. IEEE, 2011.
[S52] Sergio Segura, David Benavides, and Antonio Ruiz Cortés. Functional testing of feature model analysis tools: a test suite. IET Software, 5(1):70–82, 2011.
[S53] Hauke Baller, Sascha Lity, Malte Lochau, and Ina Schaefer. Multi-objective test suite optimization for incremental product family testing. In ICST, pages 303–312, 2014.
[S54] Ana B. Sánchez, Sergio Segura, and Antonio Ruiz Cortés. A comparison of test case prioritization criteria for software product lines. In ICST, pages 41–50, 2014.
[S55] Bardia Mohabbati, Marek Hatala, Dragan Gasevic, Mohsen Asadi, and Marko Boskovic. Development and configuration of service-oriented systems families. In William C. Chu, W. Eric Wong, Mathew J. Palakal, and Chih-Cheng Hung, editors, SAC, pages 1606–1613. ACM, 2011.
[S56] Rafael Olaechea, Steven Stewart, Krzysztof Czarnecki, and Derek Rayside. Modeling and multi-objective optimization of quality attributes in variability-rich software. In Proceedings of the Fourth International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages, NFPinDSML '12, pages 2:1–2:6, New York, NY, USA, 2012. ACM.
[S57] Lukas Linsbauer, Roberto Erick Lopez-Herrejon, and Alexander Egyed. Feature model synthesis with genetic programming. In SSBSE, pages 153–167, 2014.
[S58] Christopher Henard, Mike Papadakis, and Yves Le Traon. Mutation-based generation of software product line test configurations. In SSBSE, pages 92–106, 2014.
[S59] Shuai Wang, Shaukat Ali, and Arnaud Gotlieb. Random-weighted search-based multi-objective optimization revisited. In SSBSE, pages 199–214, 2014.
[S60] Giovani Guizzo, Thelma Elita Colanzi, and Silvia Regina Vergilio. A pattern-driven mutation operator for search-based product line architecture design. In SSBSE, pages 77–91, 2014.
[S61] Shuai Wang, David Buchmann, Shaukat Ali, Arnaud Gotlieb, Dipesh Pradhan, and Marius Liaaen. Multi-objective test prioritization in software product line testing: An industrial case study. In SPLC, 2014.
[S62] Mustafa Al-Hajjaji, Thomas Thum, Jens Meinicke, Malte Lochau, and Gunter Saake. Similarity-based prioritization in software product-line testing. In SPLC 14, 2014.
[S63] Roberto E. Lopez-Herrejon, Javier Ferrer, J. Francisco Chicano, Alexander Egyed, and Enrique Alba. Comparative analysis of classical multi-objective evolutionary algorithms and seeding strategies for pairwise testing of software product lines. In CEC, 2014.
[S64] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, Patrick Heymans, and Yves Le Traon. Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for software product lines. IEEE Trans. Software Eng., 40(7):650–670, 2014.
[S65] Roberto Erick Lopez-Herrejon, Javier Ferrer, Francisco Chicano, Evelyn Nicole Haslinger, Alexander Egyed, and Enrique Alba. A parallel evolutionary algorithm for prioritized pairwise testing of software product lines. In Dirk V. Arnold, editor, GECCO, pages 1255–1262. ACM, 2014.
[S66] Thelma Elita Colanzi and Silvia Regina Vergilio. A feature-driven crossover operator for product line architecture design optimization. In COMPSAC 14, 2014.
[S67] Thelma Elita Colanzi, Silvia Regina Vergilio, Itana M.S. Gimenes, and Willian Nalepa Oizumi. A search-based approach for software product line design. In SPLC 14, 2014.
[S68] Rafael Olaechea, Derek Rayside, Jianmei Guo, and Krzysztof Czarnecki. Comparison of exact and approximate multi-objective optimization for software product lines. In SPLC 14, 2014.
[S69] Andrea Calvagna, Angelo Gargantini, and Paolo Vavassori. Combinatorial testing for feature models using citlab. In ICST Workshops, pages 338–347, 2013.
[S70] Gustavo G. Pascual, Mónica Pinto, and Lidia Fuentes. Run-time support to manage architectural variability specified

with CVL. In Khalil Drira, editor, ECSA, volume 7957 of Lecture Notes in Computer Science, pages 282–298. Springer, 2013.

[S71] Muhammad Irfan Ullah, Günther Ruhe, and Vahid Garousi. Decision support for moving from a single product to a product portfolio in evolving software systems. Journal of Systems and Software, 83(12):2496–2512, 2010.

[S72] Ying-lin Wang and Jin-wei Pang. Ant colony optimization for feature selection in software product lines. Journal of Shanghai Jiaotong University (Science), 19(1):50–58, 2014.

[S73] Jian Li, Xijuan Liu, Yinglin Wang, and Jianmei Guo. Formalizing feature selection problem in software product lines using 0–1 programming. In Yinglin Wang and Tianrui Li, editors, Practical Applications of Intelligent Systems, volume 124 of Advances in Intelligent and Soft Computing, pages 459–465. Springer Berlin Heidelberg, 2012.

[S74] Sheng Chen and Martin Erwig. Optimizing the product derivation process. In Software Product Lines - 15th International Conference, SPLC 2011, Munich, Germany, August 22–26, 2011, pages 35–44, 2011.

[S75] Alexander Nöhrer and Alexander Egyed. Optimizing user guidance during decision-making. In Software Product Lines - 15th International Conference, SPLC 2011, Munich, Germany, August 22–26, 2011, pages 25–34, 2011.

[S76] João Bosco Ferreira Filho, Olivier Barais, Mathieu Acher, Benoit Baudry, and Jérôme Le Noir. Generating counterexamples of model-based software product lines: an exploratory study. In Tomoji Kishi, Stan Jarzabek, and Stefania Gnesi, editors, 17th International Software Product Line Conference, SPLC 2013, Tokyo, Japan - August 26–30, 2013, pages 72–81. ACM, 2013.

[S77] Johannes Müller. Value-based portfolio optimization for software product lines. In Software Product Lines - 15th International Conference, SPLC 2011, Munich, Germany, August 22–26, 2011, pages 15–24, 2011.

References

[1] M. Harman, S.A. Mansouri, Y. Zhang, Search-based software engineering: trends, techniques and applications, ACM Comput. Surv. 45 (1) (2012) 11.

[2] J.A. Clark, J.J. Dolado, M. Harman, R.M. Hierons, B.F. Jones, M. Lumkin, B.S. Mitchell, S. Mancoridis, K. Rees, M. Roper, M.J. Shepperd, Formulating software engineering as a search problem, IEE Proc. – Softw. 150 (3) (2003) 161–175.

[3] A. Eiben, J. Smith, Introduction to Evolutionary Computing, Springer-Verlag, 2003.

[4] S. Luke, Essentials of Metaheuristics, Lulu, 2009 <http://cs.gmu.edu/∼sean/book/metaheuristics/>.

[5] D.S. Batory, J.N. Sarvela, A. Rauschmayer, Scaling step-wise refinement, IEEE Trans. Softw. Eng. 30 (6) (2004) 355–371.

[6] K. Pohl, G. Bockle, F.J. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques, Springer, 2005.

[7] M. Svahnberg, J. van Gurp, J. Bosch, A taxonomy of variability realization techniques, Softw. Pract. Exper. 35 (8) (2005) 705–754.

[8] K. Kang, S. Cohen, J. Hess, W. Novak, A. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study, Tech. Rep. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.

[9] D. Benavides, S. Segura, A.R. Cortés, Automated analysis of feature models 20 years later: a literature review, Inf. Syst. 35 (6) (2010) 615–636.

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

[11] T. Käkölä, J.C. Dueñas (Eds.), Software Product Lines – Research Issues in Engineering and Management, Springer, 2006.

[12] S. Yoo, M. Harman, S. Ur, GPGPU test suite minimisation: search based software engineering performance improvement using graphics cards, Emp. Softw. Eng. 18 (3) (2013) 550–593, http://dx.doi.org/10.1007/s10664-013-9247-y. http://dx.doi.org/10.1007/s10664-013-9247-y.

[13] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering. version 2.3., EBSE Technical Report EBSE-2007-01, Software Engineering Group, School of Computer Science and Mathematics, Keele University, UK and Department of Computer Science, University of Durham, UK, 2007 <http://cdn.elsevier.com/promis_misc/525444systematicreviewsguide.pdf>.

[14] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: EASE, British Computer Society, 2008, pp. 68–77 <http://dl.acm.org/citation.cfm?id=2227115.2227123>.

[15] D. Budgen, M. Turner, P. Brereton, B. Kitchenham, Using mapping studies in software engineering, in: Proceedings of PPIG 2008, Lancaster University, 2008, pp. 195–204.

[16] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, Experimentation in Software Engineering, Springer, 2012.

[17] B.A. Kitchenham, D. Budgen, O.P. Brereton, Using mapping studies as the basis for further research – a participant-observer case study, Inform. Softw. Technol. 53 (6) (2011) 638–651.

[18] B. Kitchenham, T. Dybaa, M. Jorgensen, Evidence-based software engineering, in: ICSE, IEEE CS Press, 2004, pp. 273–281.

[19] M. Harman, B.F. Jones, Search-based software engineering, Inform. Softw. Tech. 43 (14) (2001) 833–839.

[20] A. Arcuri, L.C. Briand, A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering, Softw. Test. Verif. Reliab. 24 (3) (2014) 219–250, http://dx.doi.org/10.1002/stvr.1486. http://dx.doi.org/10.1002/stvr.1486.

[21] V. Alves, N. Niu, C.F. Alves, G. Valença, Requirements engineering for software product lines: a systematic literature review, Inform. Softw. Technol. 52 (8) (2010) 806–820.

[22] R. Rabiser, P. Grünbacher, D. Dhungana, Requirements for product derivation support: results from a systematic literature review and an expert survey, Inform. Softw. Technol. 52 (3) (2010) 324–346.

[23] J. Ferreira Bastos, P. Anselmo da Mota Silveira Neto, E. Santana de Almeida, S. Romero de Lemos Meira, Adopting software product lines: a systematic mapping study, in: 15th Annual Conference on Evaluation Assessment in Software Engineering (EASE 2011), 2011, pp. 11–20, http://dx.doi.org/10.1049/ic.2011.0002.

[24] P.A. da Mota Silveira Neto, I. do Carmo Machado, J.D. McGregor, E.S. de Almeida, S.R. de Lemos Meira, A systematic mapping study of software product lines testing, Inform. Softw. Technol. 53 (5) (2011) 407–423.

[25] E. Engström, P. Runeson, Software product line testing – a systematic mapping study, Inform. Softw. Technol. 53 (1) (2011) 2–13.

[26] I.F. da Silva, P.A. da Mota Silveira Neto, P. O'Leary, E.S. de Almeida, S.R. de Lemos Meira, Agile software product lines: a systematic mapping study, Softw. Pract. Exper. 41 (8) (2011) 899–920.

[27] G. Holl, P. Grünbacher, R. Rabiser, A systematic review and an expert survey on capabilities supporting multi product lines, Inform. Softw. Technol. 54 (8) (2012) 828–852.

[28] M.A. Laguna, Y. Crespo, A systematic mapping study on software product line evolution: from legacy system reengineering to product line refactoring, Sci. Comput. Program. 78 (8) (2013) 1010–1034.

[29] B. Mohabbati, M. Asadi, D. Gasevic, M. Hatala, H.A. Müller, Combining service-orientation and software product line engineering: a systematic mapping study, Inform. Softw. Technol. 55 (11) (2013) 1845–1859.

[30] S. Mahdavi-Hezavehi, M. Galster, P. Avgeriou, Variability in quality attributes of service-based software systems: a systematic literature review, Inform. Softw. Technol. 55 (2) (2013) 320–343.

[31] I. do Carmo Machado, J.D. McGregor, Y.C. Cavalcanti, E.S. de Almeida, On strategies for testing software product lines: a systematic literature review, Inform. Softw. Technol. 56 (10) (2014) 1183–1199, http://dx.doi.org/10.1016/j.infsof.2014.04.002. <http://www.sciencedirect.com/science/article/pii/S0950584914000834>.

[32] M. Galster, D. Weyns, D. Tofan, B. Michalik, P. Avgeriou, Variability in software systems – a systematic literature review, IEEE Trans. Softw. Eng. 40 (3) (2014) 282–306.

[33] W. Afzal, R. Torkar, R. Feldt, A systematic mapping study on non-functional search-based software testing, in: Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering (SEKE'2008), San Francisco, CA, USA, July 1–3, 2008, Knowledge Systems Institute Graduate School, 2008, pp. 488–493.

[34] S. Ali, L.C. Briand, H. Hemmati, R.K. Panesar-Walawege, A systematic review of the application and empirical investigation of search-based test case generation, IEEE Trans. Soft. Eng. 36 (6) (2010) 742–762.

[35] R.E. Lopez-Herrejon, J. Ferrer, F. Chicano, L. Linsbauer, A. Egyed, E. Alba, A hitchhiker's guide to search-based software engineering for software product lines, CoRR abs/1406.2823.

[36] S.J. Russell, P. Norvig, Artificial Intelligence – A Modern Approach (3. internat. ed.), Pearson Education, 2010.

[37] C.C. Coello, G.B. Lamont, D.A. Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, Genetic and Evolutionary Computation, second ed., Springer, 2007.

[38] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, first ed., Wiley, 2001.

[39] G. Rozenberg, T. Bäck, J.N. Kok (Eds.), Handbook of Natural Computing, Springer, 2012.

[40] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: M.J. Shepperd, T. Hall, I. Myrtveit (Eds.), EASE, ACM, 2014, p. 38.

[41] M. Harman, Y. Jia, J. Krinke, W.B. Langdon, J. Petke, Y. Zhang, Search based software engineering for software product line engineering: a survey and directions for future work, in: S. Gnesi, A. Fantechi, P. Heymans, J. Rubin, K. Czarnecki (Eds.), 18th International Software Product Line Conference, SPLC '14, Florence, Italy, September 15–19, 2014, ACM, 2014, pp. 5–18. http://doi.

acm.org/10.1145/2648511.2648513, http://dx.doi.org/10.1145/2648511.2648513.
[42] A. Metzger, K. Pohl, Software product line engineering and variability management: achievements and challenges, in: J.D. Herbsleb, M.B. Dwyer (Eds.), FOSE, ACM, 2014, pp. 70–84.
[43] T. Berger, S. She, R. Lotufo, A. Wasowski, K. Czarnecki, Variability modeling in the real: a perspective from the operating systems domain, in: C. Pecheur, J. Andrews, E.D. Nitto (Eds.), ASE, ACM, 2010, pp. 73–82.
[44] W.K.G. Assunção, S.R. Vergilio, A multi-objective solution for retrieving class diagrams, in: BRACIS, IEEE, 2013, pp. 249–255.
[45] J. Guo, K. Czarnecki, S. Apel, N. Siegmund, A. Wasowski, Variability-aware performance prediction: a statistical learning approach, in: E. Denney, T. Bultan, A. Zeller (Eds.), ASE, IEEE, 2013, pp. 301–311.
[46] R.E. Lopez-Herrejon, A. Egyed, Sbse4vm: Search based software engineering for variability management, in: Cleve et al. [78], pp. 441–444 <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6498212>.
[47] F.G. de Freitas, J.T. de Souza, Ten years of search based software engineering: a bibliometric analysis, in: SSBSE, 2011, pp. 18–32.
[48] S. Yoo, M. Harman, Regression testing minimization, selection and prioritization: a survey, Softw. Test. Verif. Reliab. 22 (2) (2012) 67–120.
[49] E. Zitzler, Evolutionary multiobjective optimization, in: Rozenberg et al. [39], pp. 871–904.
[50] R. Marler, J. Arora, Survey of multi-objective optimization methods for engineering, Struct. Multidisc. Optim. 26 (6) (2004) 369–395, http://dx.doi.org/10.1007/s00158-003-0368-6. http://dx.doi.org/10.1007/s00158-003-0368-6.
[51] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE Trans. Evol. Comp. 3 (4) (1999) 257–271.
[52] D.A. Van Veldhuizen, Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations, Ph.D. thesis, Air Force Institute of Technology, USA, aAI9928483, 1999.
[53] T. Thüm, S. Apel, C. Kästner, I. Schaefer, G. Saake, A classification and survey of analysis strategies for software product lines, ACM Comput. Surv. 47 (1) (2014) 6.
[54] G. Perrouin, S. Oster, S. Sen, J. Klein, B. Baudry, Y.L. Traon, Pairwise testing for software product lines: comparison of two approaches, Softw. Qual. J. 20 (3–4) (2012) 605–643.
[55] A. Arcuri, L.C. Briand, A practical guide for using statistical tests to assess randomized algorithms in software engineering, in: R.N. Taylor, H. Gall, N. Medvidovic (Eds.), Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu, HI, USA, May 21–28, 2011, ACM, 2011, pp. 1–10. http://doi.acm.org/10.1145/1985793.1985795, http://dx.doi.org/10.1145/1985793.1985795..
[56] E. Popovici, A. Bucci, R.P. Wiegand, E.D. de Jong, Coevolutionary principles, in: Rozenberg et al. [39], pp. 987–1033, http://dx.doi.org/10.1007/978-3-540-92910-9_31.
[57] A.J. Bagnall, V.J. Rayward-Smith, I. Whittley, The next release problem, Inform. Softw. Technol. 43 (14) (2001) 883–890.
[58] Y. Zhang, M. Harman, S.A. Mansouri, The multi-objective next release problem, in: H. Lipson (Ed.), GECCO, ACM, 2007, pp. 1129–1137.
[59] M. Harman, J. Krinke, I. Medina-Bulo, F. Palomo-Lozano, J. Ren, S. Yoo, Exact scalable sensitivity analysis for the next release problem, ACM Trans. Softw. Eng. Methodol. 23 (2) (2014) 19.
[60] R. Heradio, D. Fernández-Amorós, J.A. Cerrada, I. Abad, A literature review on feature diagram product counting and its usage in software product line economic models, Int. J. Softw. Eng. Knowl. Eng. 23 (8) (2013) 1177.
[61] R. Heradio, D. Fernández-Amorós, L. Torre-Cubillo, A.P. García-Plaza, Improving the accuracy of coplimo to estimate the payoff of a software product line, Expert Syst. Appl. 39 (9) (2012) 7919–7928.
[62] Y. Dubinsky, J. Rubin, T. Berger, S. Duszynski, M. Becker, K. Czarnecki, An exploratory study of cloning in industrial software product lines, in: Cleve et al. [78], pp. 25–34, http://dx.doi.org/10.1109/CSMR.2013.13, http://dx.doi.org/10.1109/CSMR.2013.13.
[63] R.E. Lopez-Herrejon, L. Montalvillo-Mendizabal, A. Egyed, From requirements to features: an exploratory study of feature-oriented refactoring, in: E.S. de Almeida, T. Kishi, C. Schwanninger, I. John, K. Schmid (Eds.), SPLC, IEEE Computer Society, Los Alamitos, CA, USA, 2011, pp. 181–190. http://dx.doi.org/10.1109/SPLC.2011.52.
[64] S. Schulze, M. Lochau, S. Brunswig, Implementing refactorings for FOP: lessons learned and challenges ahead, in: A. Classen, N. Siegmund (Eds.), FOSD@GPCE, ACM, 2013, pp. 33–40.
[65] M.W. Mkaouer, M. Kessentini, S. Bechikh, M. Ó. Cinnéide, A robust multi-objective approach for software refactoring under uncertainty, in: Goues and Yoo [79], pp. 168–183.
[66] K. Praditwong, M. Harman, X. Yao, Software module clustering as a multi-objective search problem, IEEE Trans. Softw. Eng. 37 (2) (2011) 264–282.
[67] S.S. Islam, J. Krinke, D. Binkley, M. Harman, Coherent clusters in source code, J. Syst. Softw. 88 (2014) 1–24.
[68] O. Räihä, H. Kundi, K. Koskimies, E. Mäkinen, Synthesizing architecture from requirements: a genetic approach, in: P. Avgeriou, J. Grundy, J.G. Hall, P. Lago, I. Mistrík (Eds.), Relating Software Requirements and Architectures, Springer, 2011, pp. 307–331.
[69] R. Poli, W.B. Langdon, N.F. McPhee, A Field Guide to Genetic Programming, Lulu, 2008.
[70] J. Petke, M. Harman, W.B. Langdon, W. Weimer, Using genetic improvement and code transplants to specialise a C++ program to a problem class, in: M. Nicolau, K. Krawiec, M.I. Heywood, M. Castelli, P. García-Sánchez, J.J. Merelo, V.M.R. Santos, K. Sim (Eds.), Genetic Programming – 17th European Conference, EuroGP 2014, Granada, Spain, April 23–25, 2014, Revised Selected Papers, Lecture Notes in Computer Science, vol. 8599, Springer, 2014, pp. 137–149. http://dx.doi.org/10.1007/978-3-662-44303-3.
[71] C.L. Goues, S. Forrest, W. Weimer, Current challenges in automatic software repair, Softw. Qual. J. 21 (3) (2013) 421–443, http://dx.doi.org/10.1007/s11219-013-9208-0. http://dx.doi.org/10.1007/s11219-013-9208-0.
[72] M. Harman, Y. Jia, W.B. Langdon, Babel pidgin: SBSE can grow and graft entirely new functionality into a real world system, in: Goues and Yoo [79], pp. 247–252.
[73] C. Kästner, A. Dreiling, K. Ostermann, Variability mining: consistent semi-automatic detection of product-line features, IEEE Trans. Softw. Eng. 40 (1) (2014) 67–82, http://dx.doi.org/10.1109/TSE.2013.45. http://doi.ieeecomputersociety.org/10.1109/TSE.2013.45.
[74] J. Rubin, K. Czarnecki, M. Chechik, Managing cloned variants: a framework and experience, in: T. Kishi, S. Jarzabek, S. Gnesi (Eds.), 17th International Software Product Line Conference, SPLC 2013, Tokyo, Japan – August 26–30, 2013, ACM, 2013, pp. 101–110. http://doi.acm.org/10.1145/2491627.2491644, http://dx.doi.org/10.1145/2491627.2491644.
[75] S. Fischer, L. Linsbauer, R.E. Lopez-Herrejon, A. Egyed, Enhancing clone-and-own with systematic reuse for developing software variants, in: 30th International Conference on Software Maintenance and Evolution, 2014, pp. 391–400.
[76] C. Wohlin, P. Runeson, P.A. da Mota Silveira Neto, E. Engström, I. do Carmo Machado, E.S. de Almeida, On the reliability of mapping studies in software engineering, J. Syst. Softw. 86 (10) (2013) 2594–2610.
[77] L. Chen, M.A. Babar, A systematic review of evaluation of variability management approaches in software product lines, Inform. Softw. Tech. 53 (4) (2011) 344–362.
[78] A. Cleve, F. Ricca, M. Cerioli (Eds.), 17th European Conference on Software Maintenance and Reengineering, CSMR 2013, Genova, Italy, March 5–8, 2013, IEEE Computer Society, 2013. <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6498212>.
[79] C.L. Goues, S. Yoo (Eds.), Proceedings of the Search-Based Software Engineering – 6th International Symposium, SSBSE 2014, Fortaleza, Brazil, August 26–29, 2014, Lecture Notes in Computer Science, vol. 8636, Springer, 2014.