Empir Software Eng (2017) 22:2972–3016
DOI 10.1007/s10664-017-9499-z

# Reengineering legacy applications into software product lines: a systematic mapping

**Wesley K. G. Assunção[1,2] · Roberto E. Lopez-Herrejon[3] · Lukas Linsbauer[4] · Silvia R. Vergilio[1] · Alexander Egyed[4]**

Published online: 8 February 2017


**Abstract** Software Product Lines (SPLs) are families of systems that share common assets allowing a disciplined reuse. Rarely SPLs start from scratch, instead they usually start from a set of existing systems that undergo a reengineering process. Many approaches to conduct the reengineering process have been proposed and documented in research literature. This scenario is a clear testament to the interest in this research area. We conducted a systematic mapping study to provide an overview of the current research on reengineering of existing systems to SPLs, identify the community activity in regarding of venues and frequency of publications in this field, and point out trends and open issues that could serve as references for future research. This study identified 119 relevant publications. These

Communicated by: Per Runeson

✉ Wesley K. G. Assunção
wesleyk@inf.ufpr.br

Roberto E. Lopez-Herrejon
roberto.lopez@etsmtl.ca

Lukas Linsbauer
lukas.linsbauer@jku.at

Silvia R. Vergilio
silvia@inf.ufpr.br

Alexander Egyed
alexander.egyed@jku.at

[1] DInf, Federal University of Paraná (UFPR), CP: 19081, CEP: 81.531-980, Curitiba, Brazil

[2] COTSI, Federal University of Technology - Paraná (UTFPR), Cristo Rei Street, 19. CEP: 85.902-490, Toledo, Brazil

[3] Department of Software Engineering and IT, École de Technologie Supérieure, (ÉTS), Notre-Dame Street Ouest. 1100, H3C 1K3, Montreal, Canada

[4] ISSE, Johannes Kepler University Linz (JKU), Altenbergerstr. 69, 4040 Linz, Austria

primary sources were classified in six different dimensions related to reengineering phases, strategies applied, types of systems used in the evaluation, input artefacts, output artefacts, and tool support. The analysis of the results points out the existence of a consolidate community on this topic and a wide range of strategies to deal with different phases and tasks of the reengineering process, besides the availability of some tools. We identify some open issues and areas for future research such as the implementation of automation and tool support, the use of different sources of information, need for improvements in the feature management, the definition of ways to combine different strategies and methods, lack of sophisticated refactoring, need for new metrics and measures and more robust empirical evaluation. Reengineering of existing systems into SPLs is an active research topic with real benefits in practice. This mapping study motivates new research in this field as well as the adoption of systematic reuse in software companies.

**Keywords** Systematic reuse · Legacy systems · Evolution · Reengineering · Product family

## 1 Introduction

The premise of software reuse strategies is to use existing artefacts to build new software, aiming to reduce time-to-market, improving productivity and producing high quality software (Krueger 1992). According to Riva and Del Rosso, reuse is generally employed through an ad hoc strategy, called "clone-and-own methodology" (Riva and Del Rosso 2003). When customers request for additional features, existing systems are cloned and adapted to fulfill the new requirements. A main disadvantage of this methodology is the simultaneous maintenance of a typically large number of individual product variants (Faust and Verhoef 2003). In this scenario, as the number of features increases so does the complexity of their development and maintenance, hence a systematic reuse approach is necessary. A way to tackle this problem is the adoption of a *Software Product Line (SPL)* approach (Linden et al. 2007; Pohl and Böckle 2005).

An SPL is a set of systems that share common features, and are designed for a specific domain (Clements and Northrop 2001; Linden et al. 2007). The main advantage of an SPL is the systematic reuse of the common infrastructure – artefacts and assets – which is shared to create different product variants. *Software Product Line Engineering (SPLE)* is the discipline of developing and managing SPLs. SPLE identifies two types of assets: the *common assets*, reused in all products, and the *variable assets*, related to those features that are provided only by some products. In addition, SPLE deals with the effective management of SPLs throughout their entire life cycle. A way to undertake SPLE is by using existing product variants as the basis to create SPL assets, known as *extractive approach* (Krueger 2002).

Clone-and-own methodology was identified by a recent study as the most common reuse scenario in practice (Dubinsky et al. 2013). In this context, the extractive approach is the more common way to systematize the software reuse with SPLs and encompasses the reengineering of existing systems, leading to a systematic reuse and easier maintenance, because the systems are not maintained individually but instead as a group considering both common and variable assets. Besides these technical benefits, the reengineering of existing systems into an SPL allows companies to preserve their investment and aggregate knowledge obtained during the development of individual systems. Because of these

reasons, the extractive approach has attracted interest from companies, with many systems in production, and researchers (e.g. Lozano (2011)).

This increased interest prompted us to perform a *systematic mapping study*, an evidence-based method used to build a classification scheme and structure of a field of interest (Petersen et al. 2008, 2015). The goal of this study is threefold: (1) to provide an overview of the current research regarding the reengineering process of existing systems into SPLs, (2) identify the activity of the research community regarding the venues and frequency of publications in this field, and (3) point out research trends and gaps to direct future research on the subject.

This paper is an extension of our previous work (Assunção and Vergilio 2014). In our previous work we presented results of a systematic mapping on feature location and migration of existing systems to SPLs. We observed a consolidate research community in both areas and a strong relation between feature location and the reengineering process to migrate systems to SPLs. Despite extensive research, some phases on the reengineering process to undertake the migration task have not been fully investigated. Furthermore, we noticed the existence of a wide number of techniques adopted for the reengineering process and feature location. In this paper we make the following extensions to our previous work:

- *Inclusion of more primary sources*: we extended the range of dates, including two years of additional content, and the inclusion criteria, leading to 56 new pieces of work;
- *An enriched background*: more details about clone-and-own, software product lines, and the reengineering process are provided to support the readers that are new to the field;
- *Inclusion of new research questions and an extended classification schema*: eight research question are added and a classification schema with new dimensions is considered;
- *More in depth analysis*: the results are presented and analysed in more depth and detail. Such analysis takes into account the publication venues, the relation of the input/output artefacts with the strategies, and work on the intersection of different phases and strategies;
- *Detailed description of research avenues*: the main limitations of the existing approaches are pointed out. The goal is to synthesize evidence that suggests important implications for practice, and to identify challenges and areas for improvement.

The paper is organized as follows. Section 2 reviews the concepts of clone-and-own, software product lines, and the reengineering of systems to software product lines. Section 3 presents the methodology adopted in our mapping study. Section 4 describes the outcomes of the mapping process and discusses important findings and research opportunities that were identified. The threats to validity are presented in Section 5 and the related work in Section 6. Section 7 summarizes our conclusions.

## 2 Background

This section briefly reviews terminology and main concepts used throughout the paper.

### 2.1 Clone-and-Own

The most common scenario of reuse in practice is ad hoc techniques. For instance, when there exists demand for a new product that has some similar functionalities to an existing

product, usually developers fork the new product from other already existing software and then adapt it to fit the new requirements. These ad hoc practices are collectively called *Clone-and-Own (C&O)* (Dubinsky et al. 2013). Besides offering a simple and efficient way to reuse software artefact, products developed following C&O practices have their own and independent development life cycle. C&O approaches may work fine with small number of products, depending on products complexity, the development organization and its software engineering practices. However, in most situations, adding new systems is no longer doable either because of managerial, economical or technical reasons. For instance, the maintenance of many independent products leads to multiple problems like inefficient feature update or bug fixing, duplicate functionality, redundant and inadequate testing, etc (Dubinsky et al. 2013).

*Variability* is the capacity of software artefacts to vary. Besides the problems regarding maintenance of duplicated software artefacts, when we have to deal with a set of system variants another problem raises, known as *variability management*. The management of variability in a scenario with multiple system variants face several issues, i.e, extracting variability from technical artefacts, tool support, design decisions management and enforcement, testing of artefacts with variability, domain design, etc (Chen and Babar 2010; Metzger and Pohl 2014). The software product line approach is the premier alternative to cope effectively with the problems that emerge with the C&O practices, as discussed in next section.

## 2.2 Software Product Lines

*Software Product Line (SPL)* is "*a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission*" (Clements and Northrop 2001). Over the last two decades extensive research and practice have been done in the field of SPLs (Heradio et al. 2016). The benefits provided by SPL practices are better customization, improved software reuse, and faster time to market. The basis of the approach is that the products are built using a core asset base instead of being developed one by one from scratch (Heradio et al. 2016). Members of an SPL are distinguished by the set of features they provide (Pohl and Böckle 2005). A *feature* is "*a prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems*" (Kang et al. 1990). Features are the building blocks of the products of an SPL.

*Software Product Line Engineering (SPLE)* is the discipline responsible to develop SPLs. SPLE exploits the *commonality* (i.e., a property shared by all products of an SPL) and the *variability* (i.e., the capacity of different applications of the product line to vary). An effective management and realization of variability is at the core of successful SPL development (Svahnberg et al. 2005). Krueger reported three ways used by companies as start point to SPLE (Krueger 2002):

- The *proactive approach*: first engineers perform a complete domain analysis, to have a full scope of products, then they develop reusable domain artefacts, and finally they use these artefacts to application engineering;
- The *reactive approach*: engineers incrementally grow their family of products applying both domain and application engineering every time a new product is developed;
- The *extractive approach*: engineers use existing custom software systems by extracting the common and varying artefacts, migrating them to an SPL.

The extractive approach is the most common way to adopt SPLs in companies with many software system variants in production (Krueger 2002). Besides, providing the benefits of

systematic reuse promoted by SPLs, the investment and knowledge necessary to develop the existing system are held.

### 2.3 Reengineering of Systems

According to Chikofsky and Cross, *reengineering* is "*the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form*" (Chikofsky and Cross J.HI 1990). Sometimes this term is confused with the term reverse engineering. However, the same authors define *reverse engineering* as "*the process of analyzing a subject system to identify the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction.*" We can see that reverse engineering is concerned with understanding the systems, on the other hand, reengineering is concerned with restructuring/refactoring the systems. In this sense, reverse engineering is a prerequisite to the reengineering process, since we need to understand the subject system that we have to transform (Demeyer et al. 2009).

In the context of our work, the reengineering has focus on transforming a set of existing systems into an SPL. This set of existing systems can be reached by the use of ad hoc strategies of reuse, as mentioned in Section 2.1, or from a set of legacy systems. Reengineering of software systems into SPLs was the focus of two papers found in literature (Laguna and Crespo 2013; Fenske et al. 2013). These papers present a coarse-grained overview of the reengineering activity, answering questions about existing approaches, techniques, open challenges, and suggesting a taxonomy for existing approaches. From this point of view, our mapping study also has as goal the identification of approaches and techniques used for the reengineering process, but with focus on fine-grained details.

SPLE proposes activities to manage features, create variability models, and use variability mechanisms. However, taking into account the reengineering process, there are questions regarding the flow of reengineering phases, artefacts commonly used, tool support, etc., that remain unanswered. Recalling the goal of this paper, in this mapping we aim at exploring the literature to understand the reengineering process, and answer a set of research questions that remain open.

## 3 Systematic Mapping Process

Systematic mappings are studies designed to provide an overview of a research field and find research opportunities. After the search and selection of the relevant literature, the studies are classified and counted regarding categories of interest in the field (Petersen et al. 2008, 2015).

The study was carried out according to the mapping process proposed by Petersen et al. (2008, 2015), which includes the main activities: definition of research questions, conduction of the search and screening of papers, classification scheme, and data extraction and mapping. Each activity is described next.

### 3.1 Research Questions

As mentioned in the introduction, the goal of our systematic mapping is threefold: (1) to provide an overview of the current research on the field, (2) identify the publication venues and frequency, and (3) point out research trend and gaps for future work. Furthermore,

motivated by questions unanswered in related work, such as the flow of the reengineering process phases, input/output artefacts, and case studies used in evaluations; we formulated eight research questions grouped in three categories, as follows:

1. **Current research on reengineering systems into SPLs:**
   - *RQ1*: *What are the common phases of the reengineering process?* The goal of this question is to identify the common phases applied in the reengineering process. Furthermore, we can analyse the number of works devoted to each phase, and then to identify possible needs not addressed in the field;
   - *RQ2*: *What are the strategies used in the reengineering process?* In our context, a strategy is the application of a technique or method to obtain an SPL from existing systems. This question helps us to catalogue the strategies, techniques and methods, currently employed in the reengineering process;
   - *RQ3*: *What are the artefacts used as input and output?* A wide range of artefacts are produced along the software development process. In this question, we analyse what are the artefacts used for each strategy, considering both inputs and outputs;
   - *RQ4*: *What is the provenance of the systems used for the evaluation of the proposed approaches?* Our goal for this question is to obtain information regarding the provenance of the case studies used in the evaluations, for instance, academic or industrial systems. Based on this information, we can gauge at the maturity of the approaches in a specific scenario;
   - *RQ5*: *What are the tools available that support the reengineering process?* This question aims at cataloging the specific tools proposed and used to support the reengineering process. A list of existing tools can be used as reference for practitioners who need support for the reengineering process;
2. **Publications venues and frequency:**
   - *RQ6*: *Where has work been published?* There are few conferences and workshops devoted specifically to SPLs, but research on this subject can be published in different venues of Computer Science and Software Engineering. We want to know the most used fora to identify where the specialized community on this research topic has been publishing;
   - *RQ7*: *How have publication frequencies changed?* This question aims at analysing the evolution of the number of published papers on this research topic over the years. This information can help us to assess how relevant and active this topic is in the Software Engineering community.
3. **Trends and research opportunities:**
   - *RQ8*: *What are the research gaps and trends in the field of reengineering of systems variants into SPL?* This question aims at analyzing the limitation of existing approaches and identifying research directions for future work to motivate new research on this topic.

### 3.2 Conducting Search and Screening of Papers

In order to answer our research questions, the first step was the selection of relevant studies from the literature. To conduct the search of studies we must define a set of search terms. To reach a good set of terms we used a test-set of known papers that ought to be found. Using

**Table 1** Search Terms

| |
|---|
| **Feature Location Terms** |
| ”feature location”, “concept location”, “concern location”, “feature mining”, “feature identification”, “feature mapping” |
| **Reengineering Terms** |
| reengineering, refactoring, reconstruction, migration, migrating, evolution, legacy, restructuring, “re-engineering”, “re-structuring” |
| **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” |

this reference set, we tried different combinations of keywords. In this way, we reached three groups.

The first group for terms regarding the location of features, which are the building blocks of SPLs. Feature location is the initial and crucial task to identify the functional parts of existing systems to be re-engineered into SPLS. The second group relates to the notion of reengineering of systems. For the second group we use terms similar to those presented in a related work (Laguna and Crespo 2013). The third group relates to common SPL terminology. For the last group, we employed SPL terms collected from twelve mapping studies on several aspects of SPLs that we used in a recent survey on Search-Based Software Engineering for SPLs (Lopez-Herrejon et al. 2015). The set of terms used to perform the search is presented in Table 1.[1]

The search query composed by these terms is as follows:[2]

*(“feature location” OR “concept location” OR “concern location” OR “feature mining” OR “feature identification” OR “feature mapping”) AND (reengineering OR refactoring OR reconstruction OR migration OR migrating OR evolution OR legacy OR restructuring OR “re-engineering” OR “re-structuring”) AND (”application engineering” OR ”commonality” OR “core asset” OR “domain analysis” OR “domain engineering” OR “feature analysis” OR “feature based” OR “feature diagram” OR “feature model” OR “feature modeling” OR “feature oriented” OR “highly-configurable system” OR “process family” OR “product family” OR “product line” OR “product line engineering” OR “software family” OR “software product family” OR “software product line” OR “software reuse” OR “SPL” OR “variability” OR ”variability analysis” OR “variability management” OR”variability modeling” OR “variability-intensive system” OR ”variant” OR “variation” OR “variation point”)*

[1] Alternative term spellings or upper/lower case are not shown in the table and were found not to be relevant for our searches.

[2] Some databases have a limit of character for the search string. In these cases the search query used was: `("feature location" OR "concept location" OR "concern location" OR "feature mining") AND (reengineering OR refactoring OR reconstruction OR migration OR migrating ) AND ("product line" OR "product-line" OR "product family" OR "program family")`

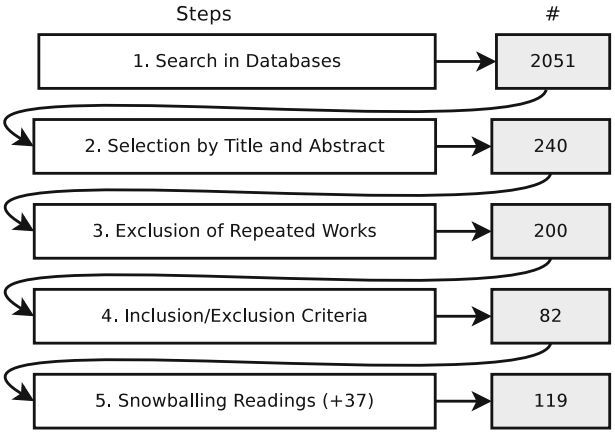



**Fig. 1** Steps of the search and selection of the relevant papers

The search and selection of the relevant primary sources were conducted in five steps, shown in Fig. 1. In the first step we performed the search for relevant publications by using the search string presented above. The search started in 2014 when we prepared our previous work (Assunção and Vergilio 2014), then we performed the search again starting on January 4th 2016 and ended on February 12th 2016. We used seven academic databases, shown in Table 2. In the table the last column presents the number of studies found in each database. At the end of this step, we obtained a set of 2051 papers. In the second step, the title of the works was read to select the relevant ones. When the title was not enough to clarify the relevance of the paper, then the abstract was read. After reviewing the title and abstract of the 2051 papers, we selected 240 studies.

In the third step 40 duplicate publications found in different databases were discarded (see Fig. 1), remaining 200. In Step 4, we read the full length papers and the inclusion/exclusion criteria presented in Table 3 were applied. We designed a set of inclusion and exclusion criteria to select only studies that fit the goals of our study. In summary we selected only papers peer reviewed, in English, available online, and with focus on reengineering multiple systems to SPLs. Considering these criteria, a final set with 82 papers was obtained.

In the last step (Step 5) we performed snowballing readings. In the snowballing reading, citations and the reference list of found papers are used to identify other relevant studies

**Table 2** Academic databases used in the mapping

| Database | URL | # |
|---|---|---|
| Science Direct | http://www.sciencedirect.com | 1080 |
| Scopus | http://www.scopus.com | 379 |
| Web of Science | http://www.isiknowledge.com | 7 |
| IEEE Xplore | http://ieeexplore.ieee.org | 3 |
| ACM Digital Library | http://dl.acm.org | 10 |
| Springer | http://www.springerlink.com | 333 |
| Google Scholar | http://scholar.google.com | 239 |

**Table 3** Inclusion and exclusion criteria

| |
|---|
| **Inclusion criteria** |
| – Text in English; |
| – Publications in journals, conferences, workshops, abstracts, tutorials, short papers, tool demonstration, entire thesis, book chapter, and technical reports; |
| – Available in an electronic format: eps, DOC, HTML, etc; |
| – With focus on reengineering of SPLs starting from multiple systems. |
| **Exclusion criteria** |
| – Position papers, doctoral symposium; |
| – Mapping studies, surveys, state-of-art and literature review; |
| – Papers not available online; |
| – Without focus on reengineering and SPL; |
| – Starting the reengineering process with a single system. |

(Wohlin 2014). We performed backward snowballing, that encompassed the use of the reference list of the 82 papers obtained in the previous step. We went through the reference lists looking for new relevant studies. For each paper identified for possible inclusion, we read the paper and applied the inclusion and exclusion criteria. Then, we identified 37 new relevant studies in this step. After this, the final set was composed of 119 papers.

The search and screening of papers discussed above were performed by the first and second authors. After composing the final set of primary sources the fourth author reviewed the included papers, hence assessed by an individual person.

### 3.3 Classification Scheme and Data Extraction

A classification scheme was used to guide the data extraction from relevant studies. Considering the goals and research questions of our mapping, there is no standard set of categories regarding the reengineering process known in literature. Considering this case we created our own classification scheme iteratively during the reading of the studies. The four steps to create the classification scheme were: (1) first we determined six dimensions related to the research questions about the reengineering process (RQ1 to RQ5), (2) then we read the primary sources, collected and documented all relevant concepts or terms taking into account the dimensions, (3) next the sets of concepts and terms from different papers were combined together, for that we analyzed which parts of the identified information were similar or common in different studies, and finally (4) one category for each similar/common item was created in the correlated dimension. This process was performed initially by the first author for our previous work (Assunção and Vergilio 2014) and refined after the search for new papers by the first three authors.

The dimensions defined about the reengineering process are: addressed reengineering phase (RQ1), type of technique or method applied (RQ2), artefact used as input for the process (RQ3), artefact generated as output (RQ3), type of systems used in the evaluation (RQ4), and existence of tool support (RQ5). The dimensions and categories, and a brief description of them are presented in Table 4. Each paper can belong to more than one category. Further details about the categories of each dimension are provided in the results.

Besides the classification scheme, we also captured complementary data regarding the reengineering process: what technique/method/algorithm was applied in each strategy

**Table 4** Classification Scheme

| Dimensions | Categories | Description |
|---|---|---|
| Reengineering Phase | Detection | Relevant information is extracted from the input artefacts, e.g. source code, to understand the existing structure, data flow, relationships, existing features, etc. |
| | Analysis | The information discovered is used to infer, design and organize new partitions that cluster the functional features. |
| | Transformation | When transformations are performed on the considered artefacts (such as source code) aiming at enabling the systematic reuse. |
| Strategy | Expert-driven | Strategy based on the expertise of specialists: software engineers, software architects, developers, stakeholders, etc. |
| | Static analysis | Static analysis relies on following or analysing structural information of static artefacts, in other words, without their execution (Wichmann et al. 1995). |
| | Dynamic analysis | When tools are used to collect and analyse information about the artefact's execution, in general considering a low-level of abstraction, such as source code (Cornelissen et al. 2009). |
| | Information Retrieval | This strategy leverages the fact that identifiers and comments represent domain knowledge. Commonly this strategy considers the textual similarity (Manning et al. 2008). |
| | Search-based | This strategy applies algorithms from the optimization field, such as Genetic Algorithms (Harman et al. 2009). |
| Input artefacts | Domain information | An example of this category is high level description of systems in specific domain and domain analysis. |
| | Requirements | Documents containing feature descriptions, customer requests, test sets generated, implementation and operation aspects, etc. |
| | Design models | Design artefacts include models such as class diagrams, state machines, or entity-relationship database model. |
| | Source code | Corresponds to the system implementation in a programming language. |
| Output artefacts | Features discovered | Features identified or mined from artefacts where they are not well-modularized or spread in multiple implementation units. |
| | Features mapped | Traceability links between known features and artefacts related with them, for instance, from requirements to source code. |
| | Reports | Reports with information such as the variability among the systems, impact on the reengineering to SPLs, and potential reuse in legacy system variants. |
| | Source code refactored | Source code refactored is an output provided to allow a better organization of the features with the SPLE. |
| Type of Systems | Industrial/Open source | Industrial or Open source systems are real case studies, developed by open source communities or by private companies. These systems vary from small to large systems. |
| | Academic/Illustrative | Academic or Illustrative systems, a.k.a. toy systems. Are generally small systems presented in text books or used to illustrate how approaches work. |
| | None | When the approaches do not use any system for their evaluation. |
| Tool support | Use of tool | When the study points the use of a tool to support the reengineering process. |
| | None | When the approaches do not use any tool to support the reengineering process. |

category (RQ2), name of the systems used in the evaluation (RQ4), and name of the tools (RQ5). Furthermore, to answer other research questions the following data was also collected: paper title and publication venue (RQ6), publication year (RQ7), and approaches limitations, gaps, and future work mentioned (RQ8). In the next sections, the results and analysis of this classification are presented.

To assess the quality of the classification we proceed as described next. A subset of six papers from the the primary sources were selected and classified individually by the first three authors. Then the authors had a meeting to discuss the classifications. After agreeing regarding the classification, the first author classified the remaining papers. The final set of classifications was reviewed by the second and fourth authors.

## 4 Results

In this section, we describe the results of the data extraction phase. In Section 4.8 we present our analysis, identified trends, and research opportunities.

### 4.1 RQ1 - Phases of the Reengineering Process

In the context of obtaining SPLs from existing systems, there is not an established or widely accepted set of phases to conduct the reengineering process. Because of this, we inferred the phases using the data from the primary sources we identified. In most cases the phases of proposed approaches are not clearly described, so we inferred them also by considering the type of input and output artefacts. In summary, we observed that the main tasks that the approaches do are: (1) to identify the features existing in a set of systems or map features to their implementation, (2) to analyse the available artefacts and information to propose a possible SPL representation, and (3) to perform the modification in the artefacts to obtain the SPL. These phases are respectively called *detection*, *analysis*, and *transformation*, recalling the terminology proposed by Anwikar et al. (2012).

Based on the above mentioned, we created an overview of the reverse engineering process with focus on SPLs, which is presented in Fig. 2. Depicted on the left part of the figure we have the systems developed following C&O practices. The solid line represents

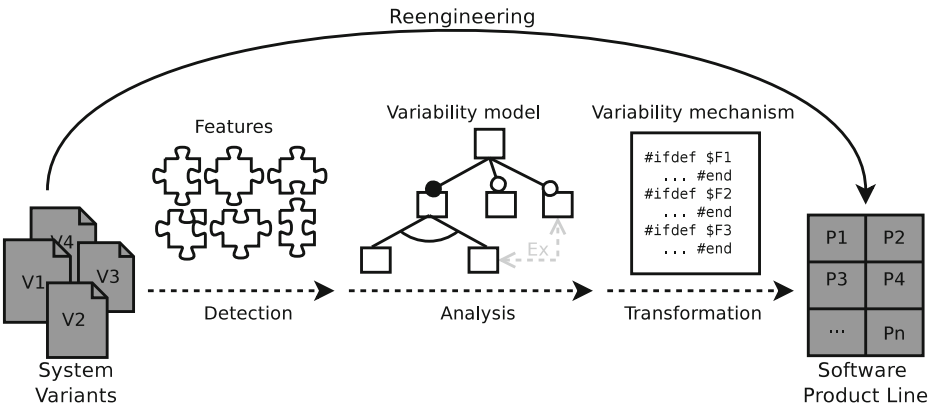


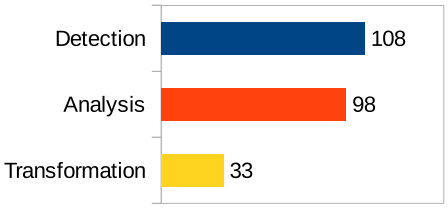

**Fig. 2** Software reengineering process

the entire process of reengineering, however, it is usually composed by some phases, shown by dashed lines.

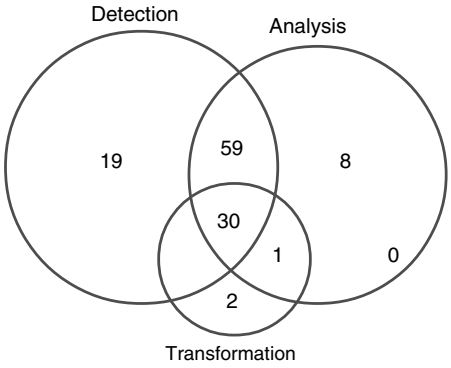
The generic phases of the reengineering process of existing systems into SPLs are: (1) *Detection phase* is the begining of the process, devoted to detecting the variability and commonality among existing products. Such as show in Fig. 2, the variabilities and commonalities are represented in terms of features. Common support in this phase is given by *feature location* techniques, which aim at locating the artifacts responsible for implementing the system functionalities. The management of the features and the mapping/traceability to the software artefacts that implement them are tasks of the SPLE domain engineering process (Dit et al. 2013; Rubin and Chechik 2013). (2) *Analysis phase* involves the organization of discovered variability and commonality. This step is devoted to creating the variability model, shown in the middle of Fig. 2, to express the valid combinations of features of an SPL. The feature model is the most popular form of variability model. Feature models are tree-like structures used to establish the existing relations between features (Kang et al. 1990). (3) *Transformation phase* is the last step of the process. Here artefacts that implement the features and the variability model are used to create the SPL, using a variability mechanism. For instance, the simplest mechanism is based on `#ifdef` statements made to the artefacts that are pre-processed, shown in the right of Fig. 2, following the desired feature selection, to create different products (Bachmann and Clements 2005). The reengineering can also be done considering design models (Wagner 2014).

Taking into account the phases of the reenginering process, Fig. 3a presents the distribution of studies among detection, analysis and transformation. We can observe that detection and analysis phases have received roughly the same attention. A possible reason for that is the existing relationship between them. When the detection is performed the analysis is a natural continuation. To extract information from artefacts without discovering helpful information for their reuse seems to make no sense. Consider that performing only detection without analysis would make sense for certain maintenance tasks (e.g. bug fixing), however, maintenance issues are outside of the scope of our mapping study. The transformation phase, which allows the actual systematic reuse of the artefacts, has not been extensively investigated.

Many proposed approaches have focus on more than one phase. Figure 3b presents the number of papers addressing each phase and the papers in more than one phase. As




**a** Primary sources per phase




**b** Intersection among the phases

**Fig. 3** Reengineering phases

mentioned before, most papers involve detection and analysis (59), followed by papers that cover the three phases (30). Those papers with focus in only one phase are mainly in detection (19) and few ones in analysis (8) and transformation (2). Not surprisingly, we did not find papers with focus on detection and transformation only, because it does not make sense to perform these phases without adequate analysis. Table 5 presents the reference of studies according to the phases and their intersections.

Figure 4 presents a bubble chart with the number of the papers in each phase by year. Again, we can observe a correlation between detection and analysis. In spite of the lower

**Table 5** Publications per phase

| Phase | # | References |
|---|---|---|
| Detection | 19 | (Lohar et al. 2013; Falessi et al. 2010; Maia et al. 2008; Li et al. 2005; Stuikys and Valincius 2011; Knodel et al. 2005; Al-msie'deen et al. 2013; Heidenreich et al. 2008; Ferrari et al. 2013; Eisenbarth et al. 2001; Rubin and Chechik 2012b; Ziadi et al. 2012; Rubin et al. 2012; Eyal-Salman et al. 2013c; Van Der Storm 2007; Shao et al. 2013; Linsbauer et al. 2014; Niu et al. 2014; Guzman and Maalej 2014) |
| Analysis | 8 | (Linsbauer et al. 2013; Ryssel et al. 2011; Segura et al. 2012; Stoermer and O'Brien 2001; Linsbauer et al. 2014; Lopez-Herrejon et al. 2015; Eriksson et al. 2005; Eyal-Salman et al. 2014) |
| Transformation | 2 | (Romero et al. 2013; Mohamed et al. 2014) |
| Detect. + Analy. | 59 | (Schulze et al. 2013; Passos et al. 2013; Seidl et al. 2012; She et al. 2011; Koziolek et al. 2013; Anwikar et al. 2012; Eyal-Salman et al. 2012; Davril et al. 2013; Merschen et al. 2011; Xue et al. 2012; Damaševičius et al. 2012; Xue et al. 2010; AL-Msie'deen et al. 2012; Kelly et al. 2011; Nunes et al. 2013; Yang et al. 2009; Eyal Salman et al. 2013; Ziadi et al. 2014; Valinčius et al. 2013; Ali et al. 2011; Eyal-Salman et al. 2013b; Olszak and Jørgensen 2012; She et al. 2014; Peng et al. 2013; Gamez and Fuentes 2013; Sampath 2013; Alves et al. 2007; Li et al. 2007; Frenzel et al. 2007; Polzer et al. 2012; AL-Msie'deen et al. 2013; Kulesza et al. 2007; Almeida et al. 2006; Noor et al. 2008; Bécan 2013; Trifu 2010; She 2013; Acher et al. 2013; Haslinger et al. 2011; Eyal-Salman et al. 2013a; Martinez et al. 2014; Nöbauer et al. 2014a; Klatt et al. 2014; Acher et al. 2011; Nöbauer et al. 2014b; Gharsellaoui et al. 2015; Maazoun et al. 2014a; Mefteh et al. 2014; Maâzoun et al. 2014b; Abbasi et al. 2014; Alves et al. 2008; Weston et al. 2009; Chen et al. 2005; Acher et al. 2012; Hariri et al. 2013; Mu et al. 2009; Yu et al. 2013; Bagheri et al. 2012; Boutkova and Houdek 2011) |
| Analy. + Transf. | 1 | (Kolb et al. 2006) |
| Detect. + Analy. + Transf. | 30 | (Santos et al. 2013; Nunes et al. 2012; Rubin et al. 2013; Araújo et al. 2013; Knodel et al. 2005; Ramos and Penteado 2008; Duszynski et al. 2011; Klatt et al. 2013; Lago and Vliet 2004; Gamez and Fuentes 2011; Xue 2012; de Oliveira et al. 2012; Otsuka et al. 2011; Bayer et al. 2004; Bécan et al. 2013; Rubin and Chechik 2012a; Kang et al. 2005; Rubin and Chechik 2010; Losavio et al. 2013; Zhang et al. 2011; Breivold et al. 2008; Nie et al. 2012; Martinez et al. 2015; Rubin 2014; Fischer et al. 2015; Tang and Leung 2015; Fischer et al. 2014; Rubin et al. 2015; Faust and Verhoef 2003; Kumaki et al. 2012) |

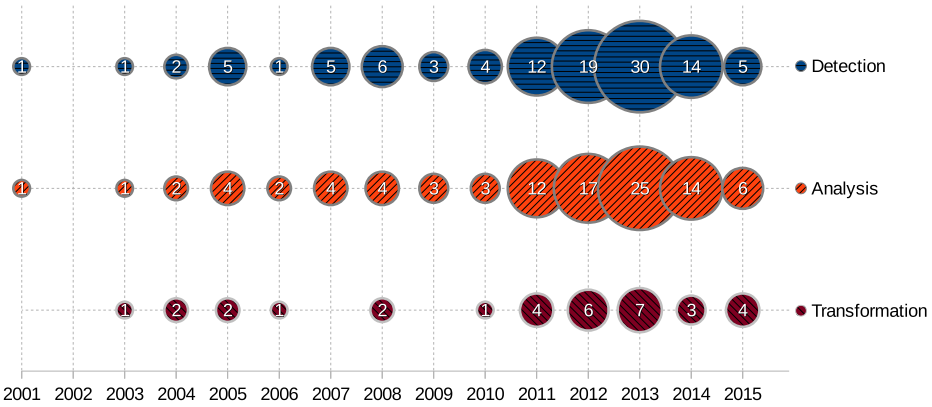


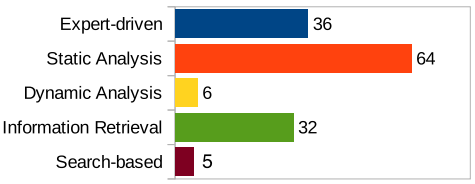

**Fig. 4** Phases addressed by the studies per year

number of papers tackling transformation, we notice an interest in this phase over the years, mainly since 2011.

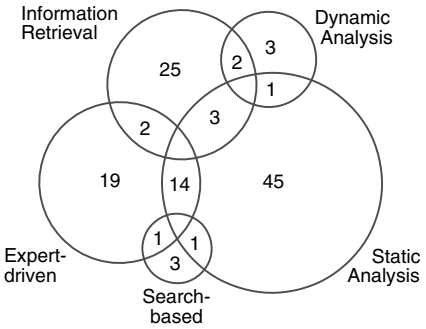
## 4.2 RQ2 - Strategies to Perform the Reengineering

According to our schema presented in Section 3, the strategies used in the reengineering were grouped in five categories. Figure 5a shows the number of papers in each strategy category. Static Analysis is the category with the largest number of papers. Expert-driven is the second most common strategy. Almost with the same attention appears Information Retrieval. Its preference is due to the ability to deal with documents/artefacts at a high level of abstraction, e.g. requirements in natural language.

We also observed that some studies do not use only one type of strategy, but instead use a combination of different strategies. This combination is sometimes named "hybrid" in literature (Rubin and Chechik 2013). For an overview about these combinations, Fig. 5b presents the number of studies in the intersections of different categories of strategies. Static analysis is the strategy most combined with other strategies, there are works using this strategy combined with all the other four strategies. Expert-driven and Information Retrieval are combined with other three strategies. Dynamic analysis and Search-based have studies




**a** Primary sources per strategy




**b** Intersection among the strategies

**Fig. 5** Reengineering strategies

combined with only two other categories. The most frequent combination of strategies is Static analysis and Expert-driven, with 14 studies. Almost half of the primary sources (14 out of 36) that apply Expert-driven are combined with almost one fourth of studies (14 out of 64) that apply Static Analysis. Table 6 shows the references for the studies considering the intersections.

The classification of primary sources in each category and the techniques/methods found during the mapping are presented in Table 7. A technique or method is the concrete application of an algorithm, tool, or approach. In the fourth, fifth, and sixth columns of the table we present the percentage of works that deal with each phase. For example, in the category Expert-driven 32 out of 36 studies deal with the detection phase, 31 with analysis, and 17 with transformation.

The reengineering process conducted by experts (Expert-driven strategy) is the strategy that has the most number of works in the three phases of the reengineering process and has the largest number or works in transformation phase. Expertise seems to be adequate to perform the entire reengineering process. In the category of Static analysis we can observe that some techniques/methods also have a good number of studies in the three phases, for instance Heuristics and Overlaps; however, it does not happen for all techniques/methods of the strategy. Static analysis has the largest number of different techniques/methods, mainly dealing with detection and analysis. The strategy Dynamic analysis has only two methods that also have focus on the first two phases of the reengineering process. Information Retrieval has a good coverage of studies on the detection phase. This can be justified because of its ability to deal with large amount of data to discovery information. On the other hand the Search-based strategy deals mainly with analysis phase. We observed that search-based techniques have been used to create variability models that best represent existing systems, a complex activity.

Figure 6 shows the number of publications per year and type of strategy. The first strategies applied in 2001 were Expert-driven, Dynamic analysis and Information Retrieval. In 2004 the first work on Static Analysis appears, and recently in 2011, on a Search-based strategy. Until 2013 the number of research papers grew, addressing the strategies Expert-driven, Static Analysis, and Information Retrieval, but in the last two years the number of studies on these strategies decreased. The number of studies addressing the strategies Dynamic analysis and Search-based remains constant.

### 4.3 RQ3 - Input and Output Artefacts

In this section we summarize the results of the artefacts used. This summary is based on the type of artefact provided as input and produced as output by each paper, shown in Table 8. Regarding the input artefacts, Source code is the most common input artefact with 73 primary sources. Java, C, C++, and C# are the programming languages generally used. Requirements were the second most common with 55 primary sources. Examples of requirement artefacts are specifications, feature descriptions, customer requests, test suites, and documentation. Design models with 30 primary sources cover artefacts such as: class diagrams, state machines, and entity-relationship database models. Nine papers use Domain information, such as products description, user comments, documentation of systems in specific domain, and domain analysis.

Different types of artefacts are produced. We grouped them into four categories. Features mapped, the most common output with 34 studies, and Features discovered (27 primary sources) are in general outputs of the detection and analysis phases. When the features are known and well defined it is only necessary to obtain the mapping of features to the

**Table 6** Publications per strategy

| Strategy | # | References |
|---|---|---|
| Expert-driven | 19 | (Schulze et al. 2013; Santos et al. 2013; Passos et al. 2013; Koziolek et al. 2013; Rubin et al. 2013; Knodel et al. 2005; Ramos and Penteado 2008; Stuikys and Valincius 2011; Knodel et al. 2005; Almeida et al. 2006; Heidenreich et al. 2008; Lago and Vliet 2004; de Oliveira et al. 2012; Otsuka et al. 2011; Zhang et al. 2011; Stoermer and O'Brien 2001; Eriksson et al. 2005; Abbasi et al. 2014; Faust and Verhoef 2003) |
| Static Analysis | 45 | (Alves et al. 2007; Kulesza et al. 2007; Kelly et al. 2011; Valinčius et al. 2013; Chen et al. 2005; Yu et al. 2013; Hariri et al. 2013; Rubin et al. 2012; Nöbauer et al. 2014a; Klatt et al. 2014; Linsbauer et al. 2014; Xue et al. 2010; She et al. 2014; Trifu 2010; She 2013; Van Der Storm 2007; Losavio et al. 2013; Nie et al. 2012; Damaševičius et al. 2012; Nunes et al. 2012; She et al. 2011; Nunes et al. 2013; Bécan et al. 2013; Rubin and Chechik 2012a; Tang and Leung 2015; Bayer et al. 2004; Kang et al. 2005; Araújo et al. 2013; Romero et al. 2013; Seidl et al. 2012; Merschen et al. 2011; Gamez and Fuentes 2013; Polzer et al. 2012; Gamez and Fuentes 2011; Linsbauer et al. 2013; Duszynski et al. 2011; Martinez et al. 2014; Li et al. 2005; Acher et al. 2012; Frenzel et al. 2007; Mu et al. 2009; Haslinger et al. 2011; Rubin and Chechik 2012b; 2010; Peng et al. 2013) |
| Dynamic Analysis | 3 | (Anwikar et al. 2012; Maia et al. 2008; Olszak and Jørgensen 2012) |
| Information Retrieval | 25 | (Ryssel et al. 2011; Sampath 2013; AL-Msie'deen et al. 2013; Xue et al. 2012; AL-Msie'deen et al. 2012; Eyal-Salman et al. 2013b; Xue 2012; Gharsellaoui et al. 2015; Maazoun et al. 2014a; Maâzoun et al. 2014b; Eyal-Salman et al. 2013a, c; Mefteh et al. 2014; Ziadi et al. 2012; Niu et al. 2014; Eyal-Salman et al. 2012; Eyal Salman et al. 2013; Shao et al. 2013; Davril et al. 2013; Falessi et al. 2010; Ferrari et al. 2013; Guzman and Maalej 2014; Li et al. 2007; Kumaki et al. 2012; Ziadi et al. 2014) |
| Search-based | 3 | (Segura et al. 2012; Lopez-Herrejon et al. 2015; Linsbauer et al. 2014) |
| Exp. + Stat. | 14 | (Weston et al. 2009; Bécan 2013; Nöbauer et al. 2014b; Breivold et al. 2008; Martinez et al. 2015; Fischer et al. 2015; 2014; Mohamed et al. 2014; Noor et al. 2008; Acher et al. 2013; Kolb et al. 2006; Acher et al. 2011; Rubin 2014; Rubin et al. 2015) |
| Exp. + IR | 2 | (Boutkova and Houdek 2011; Bagheri et al. 2012) |
| Exp. + SB | 1 | (Lohar et al. 2013) |
| Stat. + Dyn | 1 | (Klatt et al. 2013) |
| Stat. + IR | 3 | (Eyal-Salman et al. 2014; Alves et al. 2008; Al-msie'deen et al. 2013) |
| Stat. + SB | 1 | (Ali et al. 2011) |
| Dyn. + IR | 2 | (Yang et al. 2009; Eisenbarth et al. 2001) |

elements, commonly in source code. When these features are disorganized or spread across many code units, it is necessary to discover the features and its elements. 13 primary sources has as output Reports, which are often generated with information such as the variability among the systems, impact on the reengineering to SPLs, and potential reuse in legacy

**Table 7** Strategies and methods used

| Strategy | Technique/Method | # | # per Phase | | | References |
|---|---|---|---|---|---|---|
| | | | Det. | An. | Tr. | |
| Expert-driven | Expertise | 36 | 32 | 31 | 17 | (Schulze et al. 2013; Lohar et al. 2013; Santos et al. 2013; Passos et al. 2013; Koziolek et al. 2013; Rubin et al. 2013; Knodel et al. 2005; Ramos and Penteado 2008; Stuikys and Valincius 2011; Knodel et al. 2005; Almeida et al. 2006; Noor et al. 2008; Heidenreich et al. 2008; Bécan 2013; Lago and Vliet 2004; de Oliveira et al. 2012; Otsuka et al. 2011; Acher et al. 2013; Zhang et al. 2011; Stoermer and O'Brien 2001; Breivold et al. 2008; Kolb et al. 2006; Acher et al. 2011; Nöbauer et al. 2014b; Martinez et al. 2015; Rubin 2014; Eriksson et al. 2005; Fischer et al. 2015; Mohamed et al. 2014; Abbasi et al. 2014; Fischer et al. 2014; Rubin et al. 2015; Faust and Verhoef 2003; Weston et al. 2009; Bagheri et al. 2012; Boutkova and Houdek 2011) |
| Static Analysis | Clustering | 13 | 12 | 12 | 0 | (Weston et al. 2009; Kelly et al. 2011; Valinčius et al. 2013; Chen et al. 2005; Bécan 2013; Nöbauer et al. 2014b; Rubin et al. 2012; Damaševičius et al. 2012; Eyal-Salman et al. 2014; Alves et al. 2008; Hariri et al. 2013; Yu et al. 2013) |
| | Graph-based | 12 | 12 | 11 | 3 | (Chen et al. 2005; Klatt et al. 2014; Xue et al. 2010; Klatt et al. 2013; She et al. 2014; Trifu 2010; She 2013; Van Der Storm 2007; Losavio et al. 2013; Nie et al. 2012; Damaševičius et al. 2012; Peng et al. 2013) |
| | Heuristics | 11 | 11 | 11 | 7 | (Bécan 2013; Nöbauer et al. 2014b; Nunes et al. 2012; She et al. 2011; Nunes et al. 2013; Bécan et al. 2013; Rubin and Chechik 2012a; Tang and Leung 2015; Bayer et al. 2004; Kang et al. 2005; Araújo et al. 2013) |
| | Overlaps | 11 | 10 | 9 | 6 | (Martinez et al. 2015; Fischer et al. 2015; 2014; Linsbauer et al. 2013; Duszynski et al. 2011; Martinez et al. 2014; Linsbauer et al. 2014; Haslinger et al. 2011; Rubin 2014; Rubin et al. 2015; Rubin and Chechik 2012b) |

**Table 7** (continued)

| Strategy | Technique/Method | # | # per Phase | | | References |
|---|---|---|---|---|---|---|
| | | | Det. | An. | Tr. | |
| | Structural Similarity | 10 | 9 | 8 | 4 | (Noor et al. 2008; Acher et al. 2013; Kolb et al. 2006; Acher et al. 2011; Al-msie'deen et al. 2013; Rubin and Chechik 2010; Peng et al. 2013; Rubin 2014; Rubin et al. 2015; Rubin et al. 2012) |
| | Model Transformation | 8 | 7 | 7 | 3 | (Romero et al. 2013; Kulesza et al. 2007; Araújo et al. 2013; Seidl et al. 2012; Merschen et al. 2011; Gamez and Fuentes 2013; Polzer et al. 2012; Gamez and Fuentes 2011) |
| | Dependency Analysis | 5 | 5 | 4 | 1 | (Breivold et al. 2008; Ali et al. 2011; Nöbauer et al. 2014a; Klatt et al. 2014; Linsbauer et al. 2014) |
| | Rule-based | 3 | 2 | 2 | 1 | (Hariri et al. 2013; Mohamed et al. 2014; Mu et al. 2009) |
| | Aspect Programming | 2 | 2 | 2 | 0 | (Alves et al. 2007; Kulesza et al. 2007) |
| | Data Flow Analysis | 2 | 2 | 2 | 2 | (Bayer et al. 2004; Kang et al. 2005) |
| | Program slicing | 1 | 1 | 0 | 0 | (Li et al. 2005) |
| | Propositional logic | 1 | 1 | 1 | 0 | (Acher et al. 2012) |
| | Reflexion Method | 1 | 1 | 1 | 0 | (Frenzel et al. 2007) |
| Dynamic Analysis | Execution Tracing | 5 | 5 | 3 | 1 | (Klatt et al. 2013; Anwikar et al. 2012; Maia et al. 2008; Olszak and Jørgensen 2012; Eisenbarth et al. 2001) |
| | Data Access Semantics | 1 | 1 | 1 | 0 | (Yang et al. 2009) |
| Information Retrieval | Formal Concept Analysis | 17 | 15 | 14 | 1 | (Eisenbarth et al. 2001; Yang et al. 2009; Eyal-Salman et al. 2014; Ryssel et al. 2011; Sampath 2013; AL-Msie'deen et al. 2013; Al-msie'deen et al. 2013; Xue et al. 2012; AL-Msie'deen et al. 2012; Eyal-Salman et al. 2013b; Xue 2012; Gharsellaoui et al. 2015; Maazoun et al. 2014a; Maâzoun et al. 2014b; ; Eyal-Salman et al. 2013a, c Mefteh et al. 2014) |

**Table 7** (continued)

| Strategy | Technique/Method | # | # per Phase | | | References |
|---|---|---|---|---|---|---|
| | | | Det. | An. | Tr. | |
| | Latent Semantic Indexing | 14 | 14 | 11 | 1 | (Al-msie'deen et al. 2013; Xue et al. 2012; AL-Msie'deen et al. 2012; Eyal-Salman et al. 2013b; Xue 2012; Gharsellaoui et al. 2015; Maazoun et al. 2014a; Maâzoun et al. 2014b; Eyal-Salman et al. 2013a, c; Eyal-Salman et al. 2012; Eyal Salman et al. 2013; Shao et al. 2013; Alves et al. 2008) |
| | Other Natural Language Processing techniques | 7 | 7 | 3 | 0 | (Mefteh et al. 2014; Boutkova and Houdek 2011; Falessi et al. 2010; Ferrari et al. 2013; Guzman and Maalej 2014; Bagheri et al. 2012; Niu et al. 2014) |
| | Vector Space Model | 4 | 4 | 3 | 1 | (Eyal-Salman et al. 2013a, c; Kumaki et al. 2012; Alves et al. 2008) |
| | Word Frequency | 2 | 2 | 1 | 0 | (Ziadi et al. 2012; Ziadi et al. 2014) |
| | Data Mining | 2 | 2 | 1 | 0 | (Davril et al. 2013; Guzman and Maalej 2014) |
| | Ontology | 2 | 2 | 2 | 0 | (Bagheri et al. 2012; Li et al. 2007) |
| Search-based | Genetic Algorithm | 3 | 1 | 2 | 0 | (Lohar et al. 2013; Segura et al. 2012; Lopez-Herrejon et al. 2015) |
| | Genetic Programming | 1 | 0 | 1 | 0 | (Linsbauer et al. 2014) |
| | Non-dominated Genetic Algorithm II | 1 | 1 | 1 | 0 | (Ali et al. 2011) |
| | Hill climbing | 1 | 0 | 1 | 0 | (Lopez-Herrejon et al. 2015) |
| | Random search | 1 | 0 | 1 | 0 | (Lopez-Herrejon et al. 2015) |

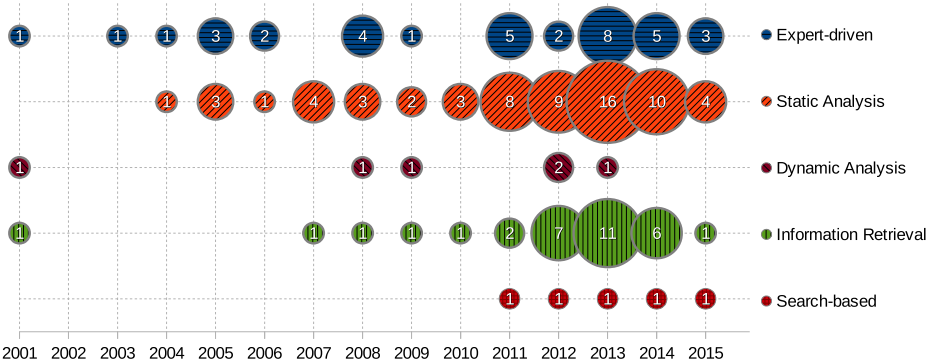



**Fig. 6** Strategies addressed by the studies per year

system variants. Source code refactored, the second more generated output with 31 studies, is a common output of the transformation phase. After generating a feature-to-code traceability, the source-code elements associated to a feature can be: clustered into a Java package (in case of object-oriented programming, e.g. Olszak and Jørgensen (2012)), migrated to an aspect (in aspect-oriented programming, e.g. Alves et al. (2007)), or reformulated as components assets (e.g. Knodel et al. (2005)).

To analyse the relationship between strategies and inputs/outputs artefacts we use Fig. 7, which shows a bubble chart that maps the studies considering both dimensions. Expert-driven, Static analysis and Information Retrieval are strategies that have primary sources that use the four categories of input and output. In these strategies the most common input is the Source code (26, 38, and 20, respectively), and the most common output is the Source code refactored (15) for Expert-drive and Features discovered for Static analysis (30) and Information Retrieval (17). Dynamic analysis does not use Domain information as input and it also does not have primary sources that generate Reports. Finally, the Search-based strategy has primary sources that use as input Requirements, Design models and Source code, and generate only Features discovered and Reports.

Studies of all strategies have as input Requirements, Design models and Source code, but only studies based on Expert-driven, Static Analysis and Information Retrieval strategies use Domain information. For all strategies there are studies producing as output Features discovered and Source code refactored. About other outputs, only the Search-based strategy does not generate Features mapped. Moreover, Dynamic Analysis and Search-based strategies do not generate Reports, what is expected, since the studies with these strategies do not use Domain artifacts as input.

### 4.4 RQ4 - Systems Used for the Evaluation

We also analysed if the approaches proposed were evaluated and what kind of systems were used. Our results are summarized in Fig. 8. Most primary sources (57) use industrial/open source systems in the evaluations of their studies; 44 studies use academic/illustrative systems. 12 studies use both types of systems. In these cases, academic/illustrative systems are generally used in a controlled experiment and industrial/open source systems for a better evaluation. None type of evaluation was found in six of the papers. The systems used vary in the domain and size. Table 9 presents the main systems used in the evaluations.

### 4.5 RQ5 - Tool Support for the Reengineering Process

We present a summary of the tools proposed in the collected papers and used for evaluation. An total of 19 tools for the reengineering support were found. We only consider tools that are specific for the reengineering process. A brief description of the tools, and corresponding work references are presented in Table 10. The first tool appeared in 2007 (Alves et al. 2007)

**Table 8** Categories of inputs and outputs artefacts used

| Category | # | References |
|---|---|---|
| Input | | |
| Domain information | 9 | (Bagheri et al. 2012; Almeida et al. 2006; Davril et al. 2013; Hariri et al. 2013; Acher et al. 2012; Yu et al. 2013; Ferrari et al. 2013; Mefteh et al. 2014; Guzman and Maalej 2014) |
| Requirements | 55 | (Almeida et al.2006; Koziolek et al. 2013; Knodel et al. 2005; Ramos and Penteado 2008; Noor et al. 2008; Lago and Vliet 2004; Kolb et al. 2006; Maia et al. 2008; Losavio et al. 2013; Eyal-Salman et al. 2014; Stuikys and Valincius 2011; Linsbauer et al. 2013; Eyal Salman et al. 2013; Eyal-Salman et al. 2013b; Xue 2012; Eisenbarth et al. 2001; Eyal-Salman et al. 2013c; Van Der Storm 2007; Eyal-Salman et al. 2013a; She 2013; She et al. 2011; Haslinger et al. 2011; Linsbauer et al. 2014; Lopez-Herrejon et al. 2015; Nunes et al. 2012, 2013; Ryssel et al. 2011; Ali et al. 2011; Bécan 2013; Bécan et al. 2013; Rubin et al. 2013; Araújo et al. 2013; Falessi et al. 2010; Valinčius et al. 2013; Li et al. 2007; Trifu 2010; Shao et al. 2013; Nöbauer et al. 2014a; Martinez et al. 2015; Rubin 2014; Faust and Verhoef 2003; Kumaki et al. 2012; Niu et al. 2014; Alves et al. 2008; Weston et al. 2009; Chen et al. 2005; Mu et al. 2009; Boutkova and Houdek 2011; Merschen et al. 2011; Polzer et al. 2012; Rubin et al. 2015; Stoermer and O'Brien 2001; Santos et al. 2013; Mefteh et al. 2014; Eriksson et al. 2005) |
| Design models | 30 | (Faust and Verhoef 2003; Acher et al. 2013; 2011; Xue 2012; Kumaki et al. 2012; Lago and Vliet 2004; Knodel et al. 2005; Yang et al. 2009; Romero et al. 2013; Kulesza et al. 2007; Passos et al. 2013; Seidl et al. 2012; Eyal-Salman et al. 2012; Peng et al. 2013; Mohamed et al. 2014; Gamez and Fuentes 2011; Heidenreich et al. 2008; Segura et al. 2012; Li et al. 2005; Rubin and Chechik 2010; Knodel et al. 2005; Li et al. 2007; Martinez et al. 2015; Rubin 2014; Rubin et al. 2015; Schulze et al. 2013; Xue et al. 2010; Martinez et al. 2014; Bécan 2013; Nie et al. 2012) |
| Source code | 73 | (Almeida et al. 2006; Faust and Verhoef 2003; Li et al. 2007; Eisenbarth et al. 2001; Nöbauer et al. 2014b; Lohar et al. 2013; Gharsellaoui et al. 2015; Stuikys and Valincius 2011; de Oliveira et al. 2012; Van Der Storm 2007; Damaševičius et al. 2012; Olszak and Jørgensen 2012; Kang et al. 2005; Sampath 2013; Tang and Leung 2015; Breivold et al. 2008; Alves et al. 2007; Seidl et al. 2012; Mohamed et al. 2014; Noor et al. 2008; Linsbauer et al. 2013; Xue et al. 2012; Klatt et al. 2013; Frenzel et al. 2007; AL-Msie'deen et al. 2013; Otsuka et al. 2011; Zhang et al. 2011; Maazoun et al. 2014a; Fischer et al. 2014; Eyal Salman et al. 2013; Ziadi et al. 2012; Bayer et al. 2004; She et al. 2014; Gamez and Fuentes 2013; Xue 2012; Kelly et al. 2011; Bécan 2013; Rubin and Chechik 2012a; Rubin et al. 2012; Lago and Vliet 2004; Passos et al. 2013; Eyal-Salman et al. 2012; Peng et al. 2013; Knodel et al. 2005; Koziolek et al. 2013; Ramos and Penteado 2008; Eyal-Salman et al. 2013a, b, c; Nunes et al. 2012; Rubin et al. 2013; Valinčius et al. 2013; Shao et al. 2013; Nöbauer et al. 2014a; Polzer et al. 2012; Santos et al. 2013; Al-msie'deen et al. 2013; Rubin and Chechik 2012b; Klatt et al. 2014; Trifu 2010; Anwikar et al. 2012; Martinez et al. 2015; Rubin 2014; Rubin et al. 2015; Eyal-Salman et al. 2014; Maâzoun et al. 2014b; Duszynski et al. 2011; AL-Msie'deen et al. 2012; Kolb et al. 2006; Ziadi et al. 2014; Fischer et al. 2015; Linsbauer et al. 2014; Abbasi et al. 2014) |

**Table 8** (continued)

| Category | # | References |
|---|---|---|
| Output | | |
| Features mapped | 34 | (Rubin et al. 2015; Fischer et al. 2014, 2015; Eyal-Salman et al. 2014; Martinez et al. 2015; Hariri et al. 2013; Rubin 2014; Boutkova and Houdek 2011; Eisenbarth et al. 2001; Romero et al. 2013; Schulze et al. 2013; Eyal-Salman et al. 2013a, b; Heidenreich et al. 2008; Lago and Vliet 2004; Van Der Storm 2007; Eyal-Salman et al. 2013c; Polzer et al. 2012; Anwikar et al. 2012; Linsbauer et al. 2014; Kulesza et al. 2007; Nunes et al. 2013; Seidl et al. 2012; Trifu 2010; Eriksson et al. 2005; Shao et al. 2013; Eyal-Salman et al. 2012; Merschen et al. 2011; Stuikys and Valincius 2011; Linsbauer et al. 2013; Passos et al. 2013; Peng et al. 2013; Ziadi et al. 2014; Eyal Salman et al. 2013) |
| Features discovered | 27 | (Frenzel et al. 2007; Xue et al. 2012; Maia et al. 2008; Almeida et al. 2006; Rubin and Chechik 2010; Ferrari et al. 2013; Falessi et al. 2010; Martinez et al. 2015; Eriksson et al. 2005; Damaševičius et al. 2012; Sampath 2013; AL-Msie'deen et al. 2013; Maazoun et al. 2014a; She et al. 2014; Bécan 2013; Valinčius et al. 2013; Maâzoun et al. 2014b; Abbasi et al. 2014; Acher et al. 2013; 2011; Kumaki et al. 2012; Yang et al. 2009; Li et al. 2005; Xue et al. 2010; She 2013; Linsbauer et al. 2014; Lopez-Herrejon et al. 2015) |
| Reports | 13 | (Merschen et al. 2011; Bagheri et al. 2012; Rubin 2014; Zhang et al. 2011; Guzman and Maalej 2014; Yu et al. 2013; Niu et al. 2014; Mu et al. 2009; Knodel et al. 2005; Koziolek et al. 2013; Noor et al. 2008; Stoermer and O'Brien 2001; Martinez et al. 2015) |
| Source code refactored | 31 | (Santos et al. 2013; Alves et al. 2007; Knodel et al. 2005; Tang and Leung 2015; Kolb et al. 2006; Ali et al. 2011; Olszak and Jørgensen 2012; Nunes et al. 2012; Martinez et al. 2015; Fischer et al. 2014; 2015; Rubin 2014; de Oliveira et al. 2012; Klatt et al. 2013; Rubin and Chechik 2012a; Rubin et al. 2015; Faust and Verhoef 2003; Mohamed et al. 2014; Kumaki et al. 2012; Breivold et al. 2008; Bayer et al. 2004; Losavio et al. 2013; Gamez and Fuentes 2011; Rubin et al. 2013; Kang et al. 2005; Otsuka et al. 2011; Xue 2012; Ramos and Penteado 2008; Maazoun et al. 2014a; Gharsellaoui et al. 2015; Kelly et al. 2011) |

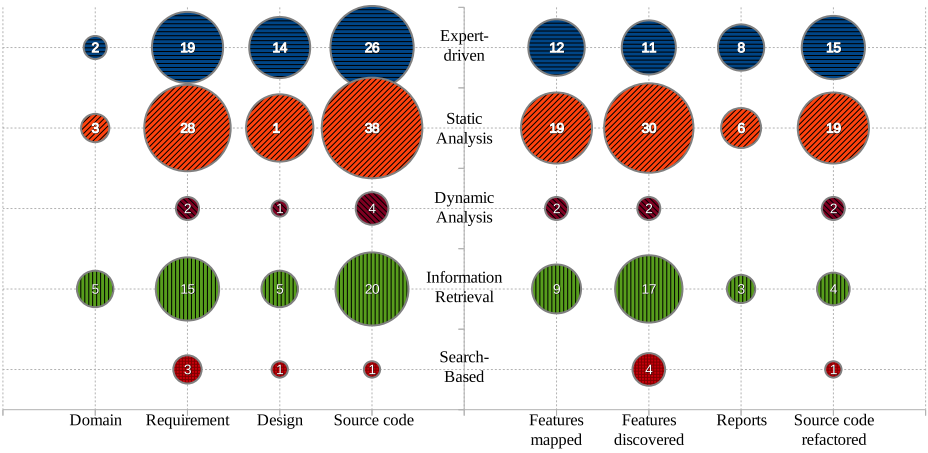



**Fig. 7** Input and output used by the strategies

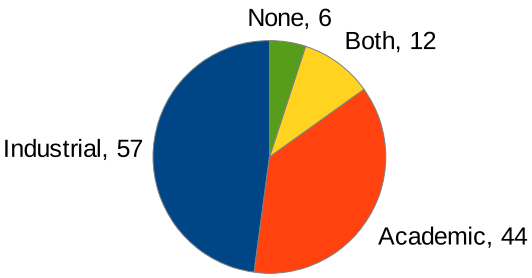



**Fig. 8** Number of studies per type of system used in the evaluation

and since then, papers that present tools has become more common. This was expected, because of the increasing number of papers on the subject.

Table 11 presents the phases, strategies, input and output per tool. We can observe that most tools have focus on phases of detection and analysis, usually applying the Static analysis strategy. For Dynamic analysis and Search-based strategies, we found only one tool. Eight tools cover the three phases of the reengineering process. Regarding the input and output, the majority of tools use source code as input, followed by requirements and design artifacts. Considering that source code is the most common artifact, it is expected that refactoring of source code is the most common output. The mapping and/or discovering of features in system variants are also well covered by the tools. There are only two tools that generate reports to support SPL adoption.

### 4.6 RQ6 - Type and Fora of the Publications

First, we analyse the fora of the publications. The number of studies in each category is presented in Fig. 9. We can see that most papers were published in peer reviewed venues: journals, conferences, and workshops (∼92 % = 110 papers). This indicates that the area is being disseminated through a wide range of scientific outlets.

Conferences are the most frequent publication venue, followed by journals and workshops. The papers in these three types are distributed among 63 distinct venues: 17 journals, 38 conferences, and 8 workshops. Figure 10 shows the main fora with the most number of publications. From the 63 publication venues only 18 have two or more publications, and are accounting for a total of 65 papers, or 55 %. International Software Product Line Conference (SPLC) and Working Conference on Reverse Engineering (WCRE) are specific conferences and the preferred ones. Together they published 24 papers (20 %) of the primary sources. Information and Software Technology, Journal of Systems and Software, and Software: Practice and Experience are the journals with the largest number of publications. All of them are general Software Engineering publication venues. The main workshops are International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS), International Workshop on Model-driven Approaches in Software Product Line Engineering (MAPLE) and International Workshop on Reverse Variability Engineering (REVE).

The distribution of the number of publications does not correlate with the number of distinct venues. For example, 52 (∼67 %) conference publications appear in 12 (∼32 %) distinct venues, while all the 20 journal papers appear in 17 distinct venues. This is maybe due to the small number of conferences devoted to SPLs and maintenance/evolution, which are

**Table 9** Systems used in evaluations

Industrial/Open source

ArgoUML (Eyal-Salman et al. 2013a; Linsbauer et al. 2014; Klatt et al. 2014; Ziadi et al. 2012; Al-msie'deen et al. 2013; Linsbauer et al. 2013; Eyal Salman et al. 2013; Eyal-Salman et al. 2013b; AL-Msie'deen et al. 2013); Eyal-Salman et al. 2013c, 2014; Fischer et al. 2014), Linux kernel (She et al. 2011; Xue et al. 2012; Peng et al. 2013; Xue 2012; She 2013), Softpedia Repository (Hariri et al. 2013; Yu et al. 2013; Guzman and Maalej 2014; Bagheri et al. 2012) Eclipse Project and Plugins (Knodel et al. 2005; Bayer et al. 2004; Martinez et al. 2015), Berkeley DB (Linsbauer et al. 2014; Xue 2012; Nie et al. 2012), Smart Home (Araújo et al. 2013; Alves et al. 2008; Weston et al. 2009), Automarker systems (Niu et al. 2014; Boutkova and Houdek 2011), JHotDraw (Trifu 2010; Olszak and Jørgensen 2012), Prevayler (Linsbauer et al. 2014; Tang and Leung 2015), FraSCAti (Acher et al. 2013; 2011), Microsoft Dynamics AX (Nöbauer et al. 2014b; 2014a), Defense domain systems (Rubin et al. 2015; Eriksson et al. 2005), Gantt Project (Noor et al. 2008), BlueJ (Olszak and Jørgensen 2012), Apache Web Server (Linsbauer et al. 2014), CCHIT Health (Lohar et al. 2013), CM-1 NASA (Lohar et al. 2013), Communications-Based Train Control (CBTC) (Ferrari et al. 2013), Curl (Linsbauer et al. 2014), DesktopSearcher (Linsbauer et al. 2014), E-Clinic (Lohar et al. 2013), eCos kernel (She et al. 2011), FreeBSD (She et al. 2011), Fujitsu Kyushu Network Technologies (Otsuka et al. 2011), Health watcher (Al-msie'deen et al. 2013), I-Trust (Lohar et al. 2013), Alcatel-Lucent IXM-PF (Zhang et al. 2011), Image Memory Handler (IMH) (Kolb et al. 2006), Java Buffer Library (Damaševičius et al. 2012), Jforum (Yang et al. 2009), JGossip (Yang et al. 2009), Labor Market Monitoring Software Product Line (LMMSPL) (Shao et al. 2013), LLVM Compiler (Linsbauer et al. 2014), MVNForum (Yang et al. 2009), Pooka Email Client (Ali et al. 2011), Power control and protection system (Breivold et al. 2008), Printworks (Li et al. 2005), QlikView (Nöbauer et al. 2014b), SELEX Sistemi Integrati Systems (Falessi et al. 2010), SIP Communicator (Ali et al. 2011), WebStore (Santos et al. 2013), Wget (Linsbauer et al. 2014), x264 Library (Linsbauer et al. 2014), Traffic management systems (Niu et al. 2014), Collaborative Software Suite (CoSS) (Hariri et al. 2013), Sudoku (Tang and Leung 2015), Web Product configurators (Abbasi et al. 2014), KePlast platform (Linsbauer et al. 2014), GameOfLife (Fischer et al. 2014), Electric motor controller (Rubin et al. 2015), Global trading and settlement system (GTSS) (Faust and Verhoef 2003)

Academic/Illustrative

MobileMedia (Al-msie'deen et al. 2013; Eyal-Salman et al. 2013b; Linsbauer et al. 2013; Eyal-Salman et al. 2012, 2014; Mefteh et al. 2014; Tang and Leung 2015), Mobile Phone (Araújo et al. 2013; Nie et al. 2012; Li et al. 2007; Maazoun et al. 2014a; Gharsellaoui et al. 2015), SPLOT Feature Models (She et al. 2014; Bécan 2013; Bécan et al. 2013; Haslinger et al. 2011; Acher et al. 2012; Lopez-Herrejon et al. 2015), Graph Product Line (GPL) (Linsbauer et al. 2014; Ziadi et al. 2012), Video On Demand (Linsbauer et al. 2013; Fischer et al. 2014), Wingsoft Financial Management System (Xue et al. 2010; Xue 2012), Banking System (Martinez et al. 2014; Maâzoun et al. 2014b), ZipMe (Linsbauer et al. 2014; Fischer et al. 2014), Washing Machine (Rubin and Chechik 2010; Rubin 2014), Airbag (Schulze et al. 2013), DirectBank (Peng et al. 2013), Home Automation (Nie et al. 2012), Home Service Robot (Kang et al. 2005), Insurance Polocy (Nie et al. 2012), J2ME Games Product Line (Kulesza et al. 2007), LinkedList (Linsbauer et al. 2014), PKJab (Linsbauer et al. 2014), Project Factory (Noor et al. 2008), SensorNetwork (Linsbauer et al. 2014), Graphical Editor (Maia et al. 2008), Text Editing System SPL (AL-Msie'deen et al. 2012), Vending Machine (Martinez et al. 2014), Microwave Oven (Rubin 2014), Xfig System (Eisenbarth et al. 2001), Fame-DBMS (Linsbauer et al. 2014), Jbook (Santos et al. 2013), Notepad SPL (Ziadi et al. 2014), Software Design Robot (Kumaki et al. 2012), Library management systems (Chen et al. 2005), Wiki engines (Acher et al. 2012), Suppliers offering systems (Acher et al. 2012), ModelAnalyzer (Fischer et al. 2014), Draw Product Line (Fischer et al. 2014), PCM Dataset (Bécan et al. 2013)

the preferred ones in this category (conferences). On the other hand, the journals with related publications are with general purpose on Software Engineering and Computer Science. In this category, there is a great number of possibilities (journals) that could be chosen.

### 4.7 RQ7 - Number and Frequency of Publications

The evolution on the number of publications along the years is depicted in Fig. 11. The first papers appeared in 2001 and there was an increase in the number of publications in 2005

**Table 10** Tools used in the reengineering process

| Name | Reference | Description |
|---|---|---|
| Variability to Aspect tool | (Alves et al. 2007) | Aims at extracting variations from existing products by isolating such variations into aspects. |
| FeatureMapper | (Heidenreich et al. 2008; Seidl et al. 2012) | A tool that allows defining mappings of features to model elements, specifying feature realisations. |
| CoDEx Tool | (Trifu 2010) | Creates and maintains direct traceability links between functional concerns and their respective implementations in code. |
| ThreeVaMar | (Rubin and Chechik 2010) | This algorithm accepts as input a model with duplications that represent systems and produces the model of a product line. |
| Feature Model Extraction | (Haslinger et al. 2011) | An algorithm reverse engineers a basic feature model from the feature sets, which describes the features each system provides. |
| RecFeat | (Nunes et al. 2012) | A history-sensitive heuristic for recovering features in code of degenerate program families. |
| ETHOM | (Segura et al. 2012) | Uses an evolutionary algorithm for the automated generation of feature models. |
| Clone-Differentiator Tool | (Xue 2012) | Automatically characterizes clones returned by a clone detector by differentiating Program Dependence Graphs (PDGs) of clones. It is able to provide a precise characterization of semantic differences of clones. |
| MapHist Tool | (Nunes et al. 2013) | MapHist tool applies heuristics to explore the evolution history of the family members in order to expand feature mappings in evolving program families. |
| SPLevo tools | (Klatt et al. 2013) | SPLevo is a software development tool supporting the consolidation of customized product copies into a Software Product Line. |
| Theme/SPL | (Araújo et al. 2013) | A tool to enhance feature modelling with traceability and improved support for cross-cutting concerns. |
| BUT4Reuse | (Martinez et al. 2014; Martinez et al. 2015) | This tool provides technologies for leveraging commonality and variability of software artefacts. |
| ExtractorPL | (Ziadi et al. 2014) | A language-independent approach which provides a quick automatic front-end to refactor a set of systems into an SPL. |

**Table 10** (continued)

| Name | Reference | Description |
|---|---|---|
| ECCO Tool | (Fischer et al. 2014; 2015) | This tool automatically locate reusable parts in existing systems and compose a new system from a selection of desired features. |
| Model Driven SaaS | (Mohamed et al. 2014) | This eclipse plugin executes the QVT transformations that were defined based on the evolution rules. |
| AUFM Suite | (Bagheri et al. 2012) | In-house Eclipse-based plug-in for feature modeling. |
| JfeTkit | (Tang and Leung 2015) | JFeTkit (Java Feature Mining Toolkit) extracts featured code from the software legacy. |
| FMr-T | (Maâzoun et al. 2014b) | FMr-T (Feature Model recovery Tool) is a feature model extraction tool that identifies code variability. |
| ArborCraft | (Weston et al. 2009) | This tool suite automatically processes natural-language requirements documents into a candidate feature model. |

and 2007. We observe a "boom" between 2011 and 2013, when 54.6 % of the papers were published.

## 4.8 RQ8 - Trends and Research Opportunities

During the analysis of the primary sources we identified some research gaps and limitations. In this section, we report the research opportunities and trends uncovered by our mapping study.

### *4.8.1 Automation and Tool Support*

We observed that further studies should envisage the implementation of tools to automate the entire process of reengineering of existing variants to an SPL. In many papers the authors expose only an intention to provide a tool support to their methods. The first reason to provide tool support to the reengineering process is to reduce the manual effort (Stoermer and O'Brien 2001; Yang et al. 2009; Mefteh et al. 2014; Abbasi et al. 2014). Moreover, an automated process can improve the overall quality of the reengineering process, since this process is a labour-intensive task and error-prone (Stoermer and O'Brien 2001). In this sense, authors argue for the necessity of providing tool support, such as Sampath (2013), Passos et al. (2013), Zhang et al. (2011), de Oliveira et al. (2012), and Acher et al. (2012), enabling an easier and better application of their approaches.

Despite the need to automate the entire reengineering process, authors point out the missing tool support for specific tasks. For instance, for detection phase, Santos et al. point out as future research the use of test-based feature location to automate the mapping of features to source code (Santos et al. 2013). As another example, now for the analysis phase, Xue et al. (2010) mention the possible use of tools to automate reconciliation of inconsistent

**Table 11** Tools per Phase and Strategy

| Tool | Phases | | | Strategies | | | | | Input | | | | Output | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Detec. | Analy. | Trans. | Exp. | Stat. | Dyn. | IR | SB | Dom. | Requi. | Design | Code | Mapp. | Disc. | Repo. | Refac. |
| Variability to Aspect tool | ✓ | ✓ | | | ✓ | | | | | | | ✓ | | | | ✓ |
| FeatureMapper | ✓ | ✓ | | ✓ | ✓ | | | | | | ✓ | ✓ | ✓ | | | |
| CoDEx Tool | ✓ | ✓ | | | ✓ | | | | | ✓ | | ✓ | ✓ | | | |
| ThreeVaMar | ✓ | ✓ | ✓ | | ✓ | | | | | | ✓ | | | ✓ | | |
| Feature Model Extraction | ✓ | ✓ | | | ✓ | | | | | | ✓ | | | ✓ | | |
| RecFeat | ✓ | ✓ | ✓ | | ✓ | | | | | ✓ | | ✓ | | | | ✓ |
| ETHOM | | ✓ | | | | | | ✓ | | | ✓ | | | ✓ | | |
| Clone-Differentiator Tool | ✓ | ✓ | ✓ | | | | ✓ | | | ✓ | ✓ | ✓ | | | | ✓ |
| MapHist Tool | ✓ | ✓ | | | ✓ | | | | | ✓ | | | ✓ | | | |
| SPLevo tools | ✓ | ✓ | ✓ | | ✓ | ✓ | | | | | | ✓ | | | | ✓ |
| Theme/SPL | ✓ | ✓ | ✓ | | ✓ | | | | | ✓ | | | | ✓ | | |
| BUT4Reuse | ✓ | ✓ | ✓ | ✓ | ✓ | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ExtractorPL | ✓ | ✓ | | | | | ✓ | | | | | ✓ | ✓ | | | |
| ECCO Tool | ✓ | ✓ | ✓ | ✓ | ✓ | | | | | | | ✓ | ✓ | | | ✓ |
| Model Driven SaaS | | | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | | | | ✓ |
| AUFM Suite | ✓ | ✓ | | ✓ | | | ✓ | | ✓ | | | | | | ✓ | |
| JfeTkit | ✓ | ✓ | ✓ | | ✓ | | | | | | | ✓ | | | | ✓ |
| FMr-T | ✓ | ✓ | | | | | ✓ | | | | | ✓ | | ✓ | | |
| ArborCraft | ✓ | ✓ | | ✓ | ✓ | | | | | ✓ | | | | ✓ | | |

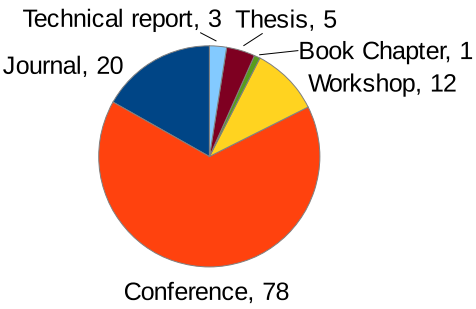



**Fig. 9** Percentage of publications per type

feature models and Li et al. (2005) argue that there is still no tool for feature aggregation and abstraction. Regarding the transformation phase, Olszak and Jørgensen point out the labour-intensive task of manually annotating feature entry points (Olszak and Jørgensen 2012). She et al. describe as challenge the automation of feature location and dependency mining when the focus of reengineering is large-scale systems (She et al. 2011).

For those papers that provide a tool, the authors recommend possible improvements. Bécan (2013) and Merschen et al. (2011) point out that it is necessary to improve their tools. According to the authors, the usability has impact on the effort saved. With respect to the motivation to use their tools, Merschen et al. (2011), Damaševičius et al. (2012), Ferrari et al. (2013), and Acher et al. (2013) mention their intention on integrating their tools into standard frameworks and environments to make them useful for developers and engineers. So, besides the importance of existence of tools to support the reengineering, their usability and integration into popular development frameworks should be considered. Martinez et al. present a tool support for the reengineering process, however they argue for extend their tool to deal with different types of artefacts (Martinez et al. 2015).

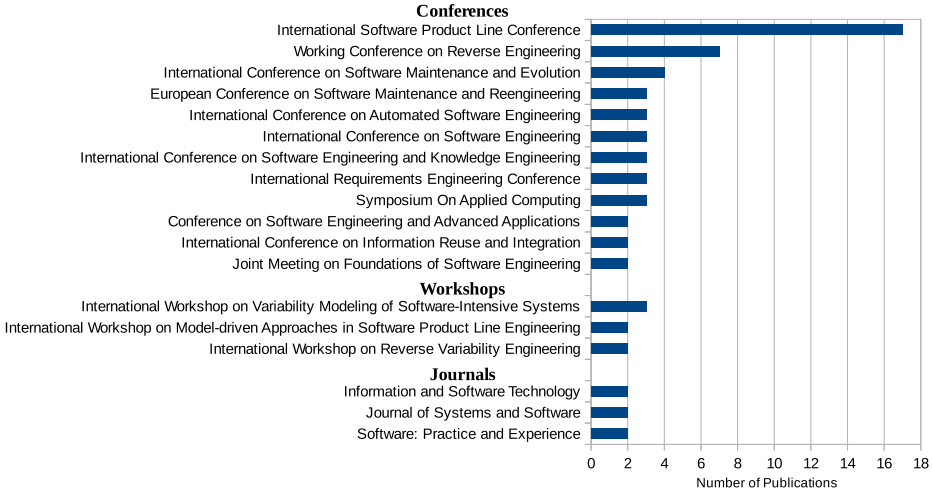



**Fig. 10** Conferences, workshops and journals with most publications

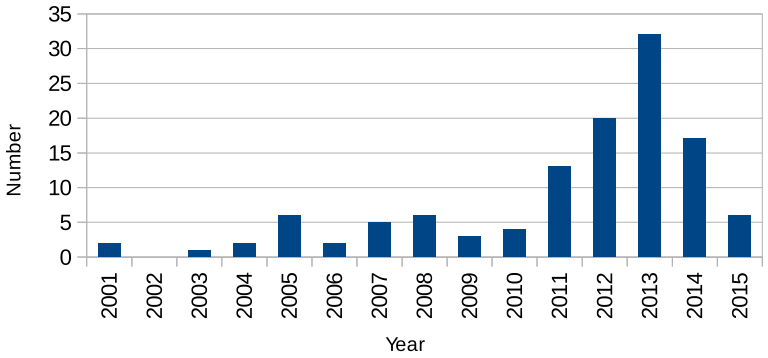



**Fig. 11** Number of publications per year

In summary, given the increasing interest in SPLs, the implementation of tools to support the phases of the entire process is fundamental to the practice and use in the industry.

### *4.8.2 Exploiting Multiple Sources of Information for Reengineering*

Another research direction observed is exploiting different information sources during the reengineering process. A research opportunity presented by Knodel et al. is using test cases, commonly available in most projects, in conjunction with other sources to determine features (Knodel et al. 2005). Trifu argues for the extraction of direct flow relations from sources other than the source code (Trifu 2010). Kelly et al. suggest exploring source code comments and documentation to enrich their approach for concept mining (Kelly et al. 2011). Eyal-Salman et al. indicate as future work the use of relationships between source code elements to improve the traceability and feature identification (Eyal Salman et al. 2013). Duszynski et al. (2011) and Peng et al. (2013) mentioned as research direction the use of design knowledge such as architecture models to allow the reengineering at a high abstraction level. Bécan et al. do not point out what specific source should be explored, but recommend that all artefacts that may be present in software projects can be used (Bécan et al. 2013). In the same way, Yu et al. envisage the use of multi-grained resources, such as code bases, historical code changes, mailing lists, bug databases, software descriptions, user evaluations, etc Yu et al. (2013). To have a benefit in using different sources of information, Kulesza et al. say that links between the different artefacts should be constantly managed (Kulesza et al. 2007). This enables the use of different sources in conjunction, such as proposed by Almeida et al. (2006), that recommend as future work the use of both domain analysis and domain design in the software evolution. In the same way She suggests the combination of bottom-up vs. top-down synthesis using artefacts at different levels of abstraction to cover different points of view (She 2013).

### *4.8.3 Feature Management*

Feature management is an important task in the reengineering process, responsible for providing the variability among the features that compose the product variants.

An identified trend is the automated recovery of feature dependencies and interactions considering aspects such as non-functional characteristic of systems (Li et al. 2005; Ali et al. 2011; Bagheri et al. 2012). These aspects may help to refine the feature mappings

and improve the resulting SPL (Li et al. 2007). Many studies generate feature models as output but, in general, constraints, such as one feature requires or excludes another feature, are not considered (Haslinger et al. 2011; Eyal-Salman et al. 2012; Damaševičius et al. 2012; AL-Msie'deen et al. 2012; Al-msie'deen et al. 2013; Ferrari et al. 2013; Xue 2012; Ziadi et al. 2014; She et al. 2014). Automatic recovery of constraints is an open issue to be addressed in new studies. Some authors pointed out the research opportunity related with the support to introduce, move, or delete features on a migrated SPL (Polzer et al. 2012; de Oliveira et al. 2012). In fact it is not directly related with the reengineering process, but if considered during the reengineering, the future evolution and maintenance may be easier. Another research direction is the reengineering of partial variability, where a subset of features with variability are considered more important and hence given priority in the reengineering process, i.e. they will be migrated first, ahead of other lower priority features (Romero et al. 2013; Losavio et al. 2013).

#### *4.8.4 Hybrid Approaches*

Hybrid approaches can improve the results when compared with the application of only one type of strategy. For example, new approaches could consider the combination of different strategies. Dynamic analysis can be combined with static analysis such as recommended in the papers (Eisenbarth et al. 2001; Frenzel et al. 2007) or static analysis combined with information retrieval (Romero et al. 2013). Some future directions are related to the use of incremental and interactive approaches including the expert engineers in the automatic process. It will be useful in situations with unsound or incomplete input to provide the required information to enable the automated process (Haslinger et al. 2011; Davril et al. 2013).

Another research direction is the combination of techniques to better explore the artefacts used in the reengineering. For example the combination of Formal Concept Analysis and Latent Semantic Indexing to further explore the requirements artefacts (Eyal-Salman et al. 2012; AL-Msie'deen et al. 2013). Furthermore, some authors expose the opportunity of using additional techniques (Rubin and Chechik 2012b; Anwikar et al. 2012; Acher et al. 2012). Recently, studies applying search-based algorithms have appeared. The Search-based strategy has been little explored in the area of SPLs (Lopez-Herrejon et al. 2015) and has the potential to exploit hybrid approaches (Harman et al. 2014). Based on this, Lopez-Herrejon et al. point out the use of search-based algorithms to address many variability management challenges (Lopez-Herrejon et al. 2015).

#### *4.8.5 New Measures and Metrics*

Measures and metrics are fundamental to support the reengineering process. However, we observe that more specific factors should be considered during the reengineering tasks. Some research opportunities are presented next.

Some authors indicated as way to improve the results of the reengineering the use of new similarity measures. Rubin et al. point out the lack of alternative methods for calculating graph similarity to deal with model variants (Rubin and Chechik 2012a). Nöbauer et al. mentioned the need of sophisticated similarity calculation method to identify commonalities in existing products (Nöbauer et al. 2014b). Eyal-Salman et al. identified the opportunity of new research on the combination of lexical similarity with structural similarity to achieve better results on the detection phase (Eyal-Samal et al. 2013a, b, c); Niu et al. mention the necessity of novel ways to compute requirements similarity (Niu et al. 2014).

Studies with sophisticated metrics to validate the costs and benefits of the reengineering process should be carried out (Rubin et al. 2012). Research in this direction, mainly in real scenarios, can help to evaluate the effort saved (Bécan et al. 2013). Besides, it is important a more rigorous measurement and reporting for quantifying business benefits (Rubin 2014). For example, to assess how is the evolution of the system after the reengineering (Bécan 2013). In general, the reengineering will provide benefits for the existing systems, but how about the cost of adding new systems or new specific features?

Other possibilities for further investigation are regarding the existing relationships among artefacts, mainly source code (e.g., method call, class inheritance, etc.) (Klatt et al. 2014; Eyal Salman et al. 2013). This requires studies on semantic similarity measure among elements (Nie et al. 2012). The combination of measures and metrics can minimize the influence of the input to reach good results in the reengineering of systems in different domains (Falessi et al. 2010).

#### *4.8.6 More Robust Empirical Evaluation*

The great majority of the proposed approaches need better evaluation. In some cases a better evaluation is required because only academic and illustrative systems were used to introduce the approaches, however, they acknowledge the importance of using real case studies (Heidenreich et al. 2008; Van Der Storm 2007; Anwikar et al. 2012; Rubin and Chechik 2012b; Araújo et al. 2013; Bécan 2013). It is also common the authors evaluate their approaches with product variants of existing SPLs, an example is the use of ArgoUML-SPL. Using existing SPLs in the evaluation makes possible to compare the reengineered SPL with the original one. But they mention, as future work, empirical evaluation considering industrial partners (Knodel et al. 2005; Noor et al. 2008; Knodel et al. 2005; Klatt et al. 2013; Abbasi et al. 2014). Evaluation of the approaches in different domains and with complex case studies are mentioned as important future work (Ramos and Penteado 2008; Ziadi et al. 2012; Acher et al. 2013; Shao et al. 2013; Nöbauer et al. 2014a). Other authors just mention the need of further evaluation (Knodel et al. 2005; Alves et al. 2007; Maia et al. 2008; Breivold et al. 2008; Trifu 2010; Ali et al. 2011; Acher et al. 2011; Seidl et al. 2012; Segura et al. 2012; Linsbauer et al. 2013; Santos et al. 2013; Passos et al. 2013; Davril et al. 2013; Linsbauer et al. 2014; Chen et al. 2005; Mefteh et al. 2014; Acher et al. 2012; Hariri et al. 2013; Guzman and Maalej 2014; Eriksson et al. 2005; Mohamed et al. 2014; Kumaki et al. 2012; 2012). Besides further studies related with the better evaluation, another common issue is the lack of a framework for comparing reengineering approaches (Yang et al. 2009; Xue et al. 2010; Rubin and Chechik 2010; Segura et al. 2012; Nunes et al. 2012; AL-Msie'deen et al. 2012; Lohar et al. 2013).

#### *4.8.7 Other Issues and Challenges*

In the following we describe some trends and opportunities mentioned in few papers that can be a starting point to new studies.

New refactoring techniques should be proposed. In the results we observed a lack of studies on the transformation phase. In two papers Rubin et al. point out the need of sophisticated techniques for refactoring of models variants to generate an SPL (Rubin and Chechik 2010; 2012a). Maâzoun et al. cited as future work the use of semantics in the refactoring of SPLs (Maâzoun et al. 2014b). Moreover, together with these techniques, it is important to devise testing tasks to check the impact on the quality of SPL refactorings (Kolb et al. 2006).

Some authors point out the importance of creating general guidelines covering the recent advances of the field. Otsuka et al. (2011), Nunes et al. (2012), and Ziadi et al. (2012) argue about the creation of guidelines to formalize the tasks of their proposed approaches. In a similar way, other authors believe that the guidelines can lead to more automated support (Stoermer and O'Brien 2001; Ramos and Penteado 2008; Martinez et al. 2015). Kang et al. point out the need for guidelines for evaluating product line assets (Kang et al. 2005).

She cites the importance of future studies to cope with unsound or incomplete input (She 2013). Both Rubin et al. (2013) and Bayer et al. (2004) presented operators to support the reengineering of existing systems. An open research area is extending and refining this catalogue of operators as well as dealing with incomplete or uncertain input information.

Nunes et al. (2013) and Trifu (2010) purpose new research to better explore techniques of seeding for mining of features in the source code. Polzer et al. point out the importance of proposing approaches of general purpose to support the reengineering of system variants in different domains (Polzer et al. 2012). Heidenreich et al. argue about new studies to create more elaborated visualization tools to support feature mapping (Heidenreich et al. 2008). Visualization is also pointed as a trend by Hariri et al. (2013) and by Guzman and Maalej (2014). A challenge exposed by Schulze et al. is reuse of regulations of functional safety besides the implementation artefacts (Schulze et al. 2013). Finally, the reengineering seems to be a good context to deep analysis of the cost-benefit of the systematic reuse, i.e. it is an opportunity for the application of Value-Based Software Engineering (Zhang et al. 2011).

## 5 Threats to Validity

The validity threats we faced are related to the systematic mapping process. A first threat is concerned with the research questions. To minimize this kind of threat, we had several discussions about the questions and goals of our search. We argue the research questions reflect the goals of our work.

A second threat is about the terms used in the search queries. To address this threat we composed three groups of terms that best represent our goals. Most of the terms used were extracted from related works (Laguna and Crespo 2013; Lopez-Herrejon et al. 2015).

A third threat is concerned with the databases used. We perform the search for primary sources on eight databases. The databases selected are well known and include the most relevant ones, also we considered more databases than the related systematic mapping (Laguna and Crespo 2013).

A fourth threat to validity is our classification scheme. We created a classification scheme to enable answering our research questions. The steps to compose the presented classification scheme were: (i) first we determined six dimensions related to the research questions, (ii) then we collected and documented all relevant information from the primary sources taking into account the dimensions and research questions, (iii) then we analysed what of the identified information are similar or common in different studies, and finally (iv) one category for each similar/common item was created in the correlated dimension. Other researchers may possibly obtain another scheme.

A fifth threat to validity concerns the data extraction using the classification scheme. During the creation of the classification scheme, the data extracted from the primary sources was documented in a text document. This document was used to extract the information and when necessary the studies were reread to clarify some doubt about the right category for the paper. Also, we had many meetings and discussions about the extraction of the data.

The last threat is related to a possible incorrect identification of research gaps and limitations of existing studies. To reduce this threat we collected each mentioned limitation and described future work for each primary source. After collecting this information from all studies, we analyzed the data in order to identify common gaps and limitations.

## 6 Related Work

The related study most similar to ours is the work of Laguna and Crespo (2013). They performed a systematic mapping study on SPL evolution, but they also consider the refactoring of existing SPLs, which is not our focus. In contrast from Laguna and Crespo's mapping study, our study has the following differences:

- *Addition of a broader set of search terms* : to conduct the search we considered a broader set of terms related to reengineering and SPLs, furthermore we include a new set of terms related to feature location (see Section 3.2);
- *Inclusion of a different set of research questions*: our systematic mapping has seven research questions to shed more light in the reengineering process. Laguna and Crespo presented four research questions covering coarse-grained aspects of the reengineering process and SPL refactoring, on the other hand we have focus on fine-grained aspects such as phases and techniques/methods. Besides of mapping the works regarding the approach, techniques and challenges of the reengineering process, as done by Laguna and Crespo, we also mapped the common phases, the artefacts considered as input and produced as output during the obtaining of the SPLs;
- *Different analysis of case studies used for evaluation*: we collect in the primary sources and presented the case studies used to validate the proposed approaches;
- *Publication venues and evolution*: our work also presents analysis about the common publication venues preferred by researchers and evolution of the publications along the years;
- *Extended classification scheme*: we adopt a different classification scheme using a more fine-grained categorization regarding the reengineering phases, techniques and methods, and the type of input and output artefacts;
- *Updated primary sources*: Laguna and Crespo's mapping study considered papers published until 2011, on the other hand our mapping includes papers published until 2015.

Fenske et al. construct a detailed taxonomy of SPL reengineering, giving different names to distinct activities and showing their relationships (Fenske et al. 2013). In summary, they considered migration, refactoring and mapping in the reengineering process. In contrast, in this paper we considered the reengineering phases and strategies. Furthermore, their effort was only concentrated on the classification of a corpus of available work, presenting what studies exist in SPLs reengineering without providing further details or analysis of them. The scope of their study is smaller than ours. They considered only three dimensions regarding the purpose of the reengineering, the technique used, and the number of software systems used as input for the process. They included works that also perform the reengineering from a single system to an SPL, which is not our focus.

Lozano presents a survey of approaches to detect variability concepts in source code (Lozano 2011). Her focus is specifically the detection of variabilities. One of the conclusions pointed by Lozano is that some techniques, e.g. to address architectural degradation, are limited to address SPLs from single products. This conclusion corroborates our motivation

in performing this mapping study to map works starting from a set of products instead of a single product. Tiarks et al. present a state-of-art of clone detection with focus on migration (Tiarks et al. 2011). They performed an assessment of clone detection in source code. In contrast with those studies, we focus our study on the entire reengineering process, not only presenting the scenario of detection phase.

Two systematic literature reviews have as focus the requirements engineering for software product lines. Alves et al. have used SPLE adoption strategies to classify the papers (Alves et al. 2010). From the results we observed that 39 % of the papers employed an extractive approach, on the other hand 57 % of the papers adopt a proactive approach. The authors suggest to researchers and practitioners focus more on extractive strategy, as well as reactive. The work of Bakar et al. presents approaches that only extract features from natural language requirements for reuse in SPLE (Bakar et al. 2015). This study is related to ours, however, it reports only approaches that use requirements as input. Differently, our study also considers the other artefacts used in the reengineering.

Koziolek et al. reported an exploratory case study on experiences and lessons learned from domain analysis in four large-scale cases on more than 20 industrial software systems (Koziolek et al. 2015). After the domain analysis only one case study resulted in an SPL. For the other cases stakeholders decided to use smaller integration scenarios, mainly due to high migration costs in the industrial domain and because of the business flexibility. According to the authors, business flexibility is often an argument against an SPL approach once some stakeholders were afraid of tightly collaborating with other business units. This study shows that the reengineering is not the best choice in such conditions. However, the authors point that more studies are necessary to further support their findings.

Harman et al. present a survey with directions for future work on Search-Based Software Engineering (SBSE) to SPLs (Harman et al. 2014). Regarding the reengineering process, the authors present some studies on the reverse engineer of feature models from a set of instances applying SBSE techniques. Another claim made by the authors is that the recent advances in genetic improvement might be exploited by SPL researchers and practitioners. We also have presented an overview and roadmap on the use of genetic improvement to SPLs (Lopez-Herrejon et al. 2015). Some connections are drawn between recent and ongoing research on reverse engineering SPLs and their evolution with the GISMOE approach.

Galster et al. performed a literature review about variability in software systems (Galster et al. 2014). As result the authors proposed an empirically grounded classification of the dimensions of variability. They also attest lack of evidence for the validity of existing approaches. Chen and Babar present a systematic review on the field of variability management (Chen and Babar 2011). The authors pointed out that there is a lack of robust evaluation for the approaches. Nevertheless, they also show that most of the studies report positive effects of the proposed variability management approaches. About variability management, Metzger and Pohl presented achievements and challenges of the field (Metzger and Pohl 2014). The authors identified some research challenges that existed for quite some time are still opened. For instance, product line quality assurance techniques, scoping, domain design, application requirements engineering, as well as application design and realization. Despite of the three works have done an extensive survey of variability and variability management literature, none of them discussed about reengineering of existing system.

The study of Heradio et al. reported a bibliometric analysis of research on software product lines (Heradio et al. 2016). The goal of their paper is to cover the entire field of SPLs. They identify the most influential publications, the most researched topics, and how the interest in SPL topics has evolved along the time. They conclude that software

architecture was the initial motor of research in SPLs, and that feature modeling has been the most important topic for the last fifteen years. Differently of this bibliometric analysis, we present here a systematic mapping with focus in the sub-field of reengineering.

## 7 Concluding Remarks

In this paper we describe the results of a systematic mapping study on the reengineering of existing systems to an SPL. We observed that studies in this field are presented in a wide range of venues such as conferences, workshops, journals, technical reports, PhD and master theses, and book chapters. The increase in the number of publications until 2013 points out a great interest in this topic. Despite of a decrease in the number of publications in 2014-2015, the topic of reenginering still has attention of the research community. Different strategies based on existing methods present in Software Engineering are used to support the reengineering process. Static analysis, Expert-driven, and Information Retrieval are the most common strategies applied. Dynamic analysis has few works along the years and the Search-based strategy has appeared recently. Artefacts of almost all software engineering process are considered as input and output. Source code and Requirements are the most common inputs. Feature discovery and Source code refactored are the most common outputs. To evaluate the proposed approaches, several systems are used. In the category of industrial systems we collected 50 different systems. ArgoUML and Linux Kernel are the most common. In the category of academic systems, we collected 32 systems used in evaluation. MobileMedia is the most common. Regarding tools supporting the reengineering process, we identified 15 tools in the primary sources.

During the classification and reading of the studies, we could observe the existence of research opportunities and trends. So, eight major areas for future research are presented, namely: automation and tool support, exploiting multiple sources of information for reengineering, feature management, hybrid approaches, refactoring techniques, need of use guidelines, new measures and metrics, and more robust empirical evaluation.

We hope that this mapping not only motivate new research on this topic, but also encourages software companies to consider the implementation of the systematic reuse of their products. C&O is a common practice in industry, so companies have available the resources needed for the reengineering. From the findings presented in this mapping study companies can be aware of the reengineering process to obtain an SPL, the available tools, artifacts commonly used and created, as well as, approaches proposed to perform the reengineering. As a result of the systematic reuse, companies are able to maintain existing products and evolve their portfolio of products by reusing existing artifacts in an easier way.

**Acknowledgments** This work was supported by the Brazilian Agencies CAPES: 007126/2014-00 and CNPq: 453678/2014-9 and 305358/2012-0, and Austrian Science Fund (FWF): P 25289-N15.

## References

Alves V, Niu N, Alves C, Valença G. (2010) Requirements engineering for software product lines: A systematic literature review. Inf Softw Technol 52(8):806–820. doi:10.1016/j.infsof.2010.03.014

Assunção WKG, Vergilio SR (2014) Feature location for software product line migration: A mapping study 18th Software Product Line Conference - 2nd International Workshop on REverse Variability Engineering (REVE), pp 1–8. doi:10.1145/2647908.2655967

Bachmann F, Clements P (2005) Variability in software product lines. Tech. Rep. CMU/SEI-2005-TR-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA

Bakar NH, Kasirun ZM, Salleh N (2015) Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. J Syst Softw 106:132–149. doi:10.1016/j.jss.2015.05.006

Chen L, Babar MA (2010) 14th International Conference Software Product Lines: Going Beyond (SPLC 2010), chap. Variability Management in Software Product Lines: An Investigation of Contemporary Industrial Challenges. Springer Berlin Heidelberg, Berlin, pp 166–180. doi:10.1007/978-3-642-15579-6_12

Chen L, Babar MA (2011) A systematic review of evaluation of variability management approaches in software product lines. Inf Softw Technol 53(4):344–362. doi:10.1016/j.infsof.2010.12.006

Chikofsky E, Cross J.HI (1990) Reverse engineering and design recovery: a taxonomy. IEEE Softw 7(1):13–17. doi:10.1109/52.43044

Clements P, Northrop L (2001) Software Product Lines: Practices and Patterns. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA

Cornelissen B, Zaidman A, van Deursen A, Moonen L, Koschke R (2009) A systematic survey of program comprehension through dynamic analysis. IEEE Trans Softw Eng 35(5):684–702. doi:10.1109/TSE.2009.28

Demeyer S, Ducasse S, Nierstrasz O (2009) Object-oriented reengineering patterns. Square Bracket associates, Switzerland. Version of 2009-09-28

Dit B, Revelle M, Gethers M, Poshyvanyk D (2013) Feature location in source code: a taxonomy and survey. Journal of Software: Evolution and Process 25(1):53–95. doi:10.1002/smr.567

Dubinsky Y, Rubin J, Berger T, Duszynski S, Becker M, Czarnecki K (2013) An exploratory study of cloning in industrial software product lines 17th European Conference on Software Maintenance and Reengineering (CSMR), pp 25–34. doi:10.1109/CSMR.2013.13

Faust D, Verhoef C (2003) Software product line migration and deployment. Software: Practice and Experience 33(10):933–955

Fenske W, Thüm T, Saake G (2013) A taxonomy of software product line reengineering 8th International Workshop on Variability Modelling of Software-Intensive Systems, VaMoS 2014, pp 1–8. ACM, New York, NY, USA. doi:10.1145/2556624.2556643

Galster M, Weyns D, Tofan D, Michalik B, Avgeriou P (2014) Variability in software systems - systematic literature review. IEEE Trans Softw Eng 40(3):282–306. doi:10.1109/TSE.2013.56

Harman M, Jia Y, Krinke J, Langdon WB, Petke J, Zhang Y (2014) Search based software engineering for software product line engineering: A survey and directions for future work 18th International Software Product Line Conference - Volume 1, SPLC 2014, pp 5–18. ACM, New York, NY, USA. doi:10.1145/2648511.2648513

Harman M, Mansouri SA, Zhang Y (2009) Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Tech. Rep. Technical Report TR-09-03, Department of Computer Science, King's College London

Heradio R, Perez-Morago H, Fernandez-Amoros D, Cabrerizo FJ, Herrera-Viedma E (2016) A bibliometric analysis of 20 years of research on software product lines. Inf Softw Technol 72:1–15. doi:10.1016/j.infsof.2015.11.004

Kang K, Cohen S, Hess J, Novak W, Peterson A (1990) Feature-Oriented Domain Analysis (FODA) Feasibility Study. Tech. Rep. CMU/SEI-90-TR-21, SEI, CMU

Koziolek H, Goldschmidt T, Gooijer T, Domis D, Sehestedt S, Gamer T, Aleksy M (2015) Assessing software product line potential: an exploratory industrial case study. doi:10.1007/s10664-014-9358-0

Krueger CW (1992) Software reuse. ACM Comput Surv (CSUR) 24(2):131–183. doi:10.1145/130844.130856

Krueger CW (2002) Easing the transition to software mass customization Software Product-Family Engineering, pp 282–293. Springer

Laguna MA, Crespo Y (2013) A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring. Sci Comput Program 78(8):1010–1034. doi:10.1016/j.scico.2012.05.003

Linden FJVD, Schmid K, Rommes E (2007) Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer-Verlag New York, Inc., Secaucus, NJ, USA

Lopez-Herrejon R, Linsbauer L, Assunção W. K, Fischer S, Vergilio SR, Egyed A (2015) Genetic improvement for software product lines: An overview and a roadmap 2015 Annual Conference on Genetic and

Evolutionary Computation, Genetic Improvement 2015 Workshop, GECCO, pp 823–830. ACM, New York, NY, USA. doi:10.1145/2739482.2768422
Lopez-Herrejon R, Linsbauer L, Egyed A (2015) A systematic mapping study of search-based software engineering for software product lines. Inf Softw Technol 61(0):33–51. doi:10.1016/j.infsof.2015.01.008
Lozano A (2011) An overview of techniques for detecting software variability concepts in source code Workshops - Advances in Conceptual Modeling: Recent Developments and New Directions, LNCS, vol. 6999, pp 141–150. Springer Berlin Heidelberg. doi:10.1007/978-3-642-24574-9_19
Manning CD, Raghavan P, Schütze H., et al. (2008) Introduction to information retrieval, vol 1, Cambridge University Press
Metzger A, Pohl K (2014) Software product line engineering and variability management: Achievements and challenges Future of Software Engineering, FOSE 2014, pp 70–84. ACM, New York, NY, USA. doi:10.1145/2593882.2593888
Petersen K, Feldt R, Mujtaba S, Mattsson M (2008) Systematic mapping studies in software engineering. British Computer Society, Swinton, UK, pp 68–77
Petersen K, Vakkalanka S, Kuzniarz L (2015) Guidelines for conducting systematic mapping studies in software engineering: An update. Inf Softw Technol 64:1–18. doi:10.1016/j.infsof.2015.03.007
Pohl K, Böckle G. (2005) Linden, F.J.v.d.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag New York, Inc., Secaucus, NJ, USA
Riva C, Del Rosso C (2003) Experiences with software product family evolution Sixth International Workshop on Principles of Software Evolution (IWPSE), pp 161–169. doi:10.1109/IWPSE.2003.1231223
Rubin J, Chechik M (2013) A survey of feature location techniques. In: Reinhartz-Berger I., Sturm A., Clark T., Cohen S., Bettin J. (eds) Domain Engineering, pp 29–58. Springer Berlin Heidelberg. doi:10.1007/978-3-642-36654-3_2
Svahnberg M, van Gurp J, Bosch J (2005) A taxonomy of variability realization techniques: Research articles. Software - Practice and Experience 35(8):705–754. doi:10.1002/spe.v35:8
Tiarks R, Koschke R, Falke R (2011) An extended assessment of type-3 clones as detected by state-of-the-art tools. Softw Qual J 19(2):295–331. doi:10.1007/s11219-010-9115-6
Wagner C (2014) Model-Driven Software Migration: A Methodology Reengineering, Recovery and Modernization of Legacy Systems, Springer Vieweg
Wichmann BA, Canning AA, Clutterbuck DL, Winsborrow LA, Ward NJ, Marsh DWR (1995) Industrial perspective on static analysis. Softw Eng J 10(2):69–75
Wohlin C (2014) Guidelines for snowballing in systematic literature studies and a replication in software engineering 18th International Conference on Evaluation and Assessment in Software Engineering, EASE'14, pp 38:1–38:10. ACM, New York, NY, USA. doi:10.1145/2601248.2601268

## Primary Sources

Abbasi E, Acher M, Heymans P, Cleve A (2014) Reverse engineering web configurators IEEE Conference on software maintenance, reengineering and reverse engineering (CSMR-WCRE), pp 264–273. doi:10.1109/CSMR-WCRE.2014.6747178
Acher M, Cleve A, Collet P, Merle P, Duchien L, Lahire P (2011) Reverse engineering architectural feature models. In: Crnkovic I, Gruhn V, Book M (eds) Software Architecture, Lecture Notes in Computer Science, vol. 6903, pp 220–235. Springer Berlin Heidelberg. doi:10.1007/978-3-642-23798-0_25
Acher M, Cleve A, Collet P, Merle P, Duchien L, Lahire P (2013) Extraction and evolution of architectural variability models in plugin-based systems. doi:10.1007/s10270-013-0364-2
Acher M, Cleve A, Perrouin G, Heymans P, Vanbeneden C, Collet P, Lahire P (2012) On extracting feature models from product descriptions. In: 6th international workshop on variability modeling of software-intensive systems, vamos. ACM, New York, pp 45–54. doi:10.1145/2110147.2110153
AL-Msie'deen R, Seriai A, Huchard M, Urtado C, Vauttier S, Salman H (2013) Feature location in a collection of software product variants using formal concept analysis. In: Favaro J, Morisio M (eds) Safe and Secure Software Reuse, Lecture Notes in Computer Science, vol. 7925, pp 302–307. Springer Berlin Heidelberg. doi:10.1007/978-3-642-38977-1_22
Al-msie'deen R, Seriai A, Huchard M, Urtado C, Vauttier S (2013) Mining features from the object-oriented source code of software variants by combining lexical and structural similarity. In: IEEE 14th international conference on information reuse and integration (IRI), pp 586–593. doi:10.1109/IRI.2013.6642522

AL-Msie'deen R, Seriai A, Huchard M, Urtado C, Vauttier S, Salman H (2012) An approach to recover feature models from object-oriented source code. Journé,e Lignes de Produits pp 15–26

Ali N, Wu W, Antoniol G, Di Penta M, Guéhéneuc Y, Hayes J (2011) Moms: Multi-objective miniaturization of software. In: 27th IEEE international conference on software maintenance (ICSM), pp 153–162

Almeida E, Mascena J, Cavalcanti A, Alvaro A, Garcia V, Lemos Meira S, Lucrédio D (2006) The domain analysis concept revisited: A practical approach. In: Morisio M (ed) Reuse of Off-the-Shelf Components, Lecture Notes in Computer Science, vol. 4039, pp. 43–57. Springer Berlin Heidelberg. doi:10.1007/11763864_4

Alves V, Matos Pedro J, Cole L, Vasconcelos A, Borba P, Ramalho G (2007) Extracting and evolving code in product lines with aspect-oriented programming. In: Rashid A, Aksit M (eds) Transactions on Aspect-Oriented Software Development IV, Lecture Notes in Computer Science, vol. 4640, pp 117–142 Springer Berlin Heidelberg. doi:10.1007/978-3-540-77042-8_5

Alves V, Schwanninger C, Barbosa L, Rashid A, Sawyer P, Rayson P, Pohl C, Rummler A (2008) An exploratory study of information retrieval techniques in domain analysis. In: 12th international software product line conference, SPLC, pp 67–76. doi:10.1109/SPLC.2008.18

Anwikar V, Naik R, Contractor A, Makkapati H (2012) Domain-driven technique for functionality identification in source code. SIGSOFT Softw Eng Notes 37(3):1–8. doi:10.1145/180921.2180923

Araújo JA, Goulão M, Moreira A, Simão I, Amaral V, Baniassad E (2013) Advanced modularity for building SPL feature models: a model-driven approach. In: 28th symposium on applied computing, SAC. ACM, New York, 1246–1253. doi:10.1145/2480362.2480596

Bagheri E, Ensan F, Gasevic D (2012) Decision support for the software product line domain engineering lifecycle. Int Conf Autom Softw Eng 19(3):335–377. doi:10.1007/s10515-011-0099-7

Bayer J, Forster T, Ganesan D, Girard JF, John I, Knodel J, Kolb R, Muthig D (2004) Definition of reference architectures based on existing systems. Tech. Rep. Report No. 034.04/E, Fraunhofer IESE-report No 034.04/E

Bécan G. (2013) Reverse engineering feature models in the real. Tech. Rep. dumas-00855005 Centre National de la Recherche Scientifique

Bécan G, Acher M, Baudry B, Ben Nasr S (2013) Breathing ontological knowledge into feature model management. Tech. Rep. RT-0441, INRIA - Institut National des Sciences Appliqué,es - Université de Rennes 1

Boutkova E, Houdek F (2011) Semi-automatic identification of features in requirement specifications. In: 19th IEEE international requirements engineering conference (RE), pp 313–318. doi:10.1109/RE.2011.6051627

Breivold H, Larsson S, Land R (2008) Migrating Industrial tab9 towards Software Product Lines: Experiences and Observations through Case Studies. In: 34th euromicro conference on software engineering and advanced applications (SEAA), pp 232–239. doi:10.1109/SEAA.2008.13

Chen K, Zhang W, Zhao H, Mei H (2005) An approach to constructing feature models based on requirements clustering. In: 13th IEEE international requirements engineering conference (RE), pp 31–40. doi:10.1109/RE.2005.9

Damaševičius R, Paškevičius P, Karčiauskas E, Marcinkevičius R (2012) Automatic extraction of features and generation of feature models from java programs. Inform Technol Control 41(4): 376–384

Davril JM, Delfosse E, Hariri N, Acher M, Cleland-Huang J, Heymans P (2013) Feature model extraction from large collections of informal product descriptions. In: 9th joint meeting on foundations of software engineering, ESEC/FSE. ACM, New York, pp 290–300. doi:10.1145/2491411.2491455

Duszynski S, Knodel J, Becker M (2011) Analyzing the source code of multiple software variants for reuse potential. In: 18th working conference on reverse engineering (WCRE), pp 303–307. doi:10.1109/WCRE.2011.44

Eisenbarth T, Koschke R, Simon D (2001) Derivation of feature component maps by means of concept analysis. In: European conference on software maintenance and reengineering (CSMR), pp 176–179. doi:10.1109/.2001.914982

Eriksson M, Morast H, Börstler J, Borg K (2005) The pluss toolkit?: Extending telelogic doors and ibm-rational rose to support product line use case modeling. In: 20th IEEE/ACM international conference on automated software engineering, ASE. ACM, New York, pp 300–304. doi:10.1145/1101908.1101955

Eyal Salman H, Djamel Seriai A, Dony C, Al-Msie'Deen R (2013) Identifying traceability links between product variants and their features. In: 1st international workshop on reverse variability engineering (REVE). Genova, Italie, pp 17–22

Eyal-Salman H, Seriai A, Dony C (2014) Feature location in a collection of product variants: Combining information retrieval and hierarchical clustering. In: 26th international conference on software engineering and knowledge engineering (SEKE)

Eyal-Salman H, Seriai A, Dony C (2013a) Feature-to-code traceability in a collection of product variants using formal concept analysis and information retrieval. In: 39th euromicro conference on software engineering and advanced applications (SEAA), pp 1–8

Eyal-Salman H, Seriai A, Dony C (2013b) Feature-to-code traceability in a collection of software variants: Combining formal concept analysis and information retrieval. In: IEEE 14th international conference on information reuse and integration (IRI), pp 209–216. doi:10.1109/IRI.2013.6642474

Eyal-Salman H, Seriai A, Dony C (2013c) Feature-to-code Traceability in Legacy Software Variants. In: 39th euromicro conference on software engineering and advanced applications (SEAA), pp 57–61. doi:10.1109/SEAA.2013.65

Eyal-Salman H, Seriai A, Dony C, Al-msie'deen R (2012) Recovering traceability links between feature models and source code of product variants. In: VARIability for you workshop: Variability modeling made useful for everyone, VARY. ACM, New York, pp 21–25. doi:10.1145/2425415.2425420

Falessi D, Cantone G, Canfora G (2010) A comprehensive characterization of NLP techniques for identifying equivalent requirements. In: 2010 ACM-IEEE International symposium on empirical software engineering and measurement, ESEM. ACM, New York, pp 1–10. doi:10.1145/1852786.1852810

Faust D, Verhoef C (2003) Software product line migration and deployment. Softw: Pract Experience 33(10):933–955

Ferrari A, Spagnolo GO, Dell'Orletta F (2013) Mining commonalities and variabilities from natural language documents. In: 17th international software product line conference, SPLC. ACM, New York, pp 116–120. doi:10.1145/2491627.2491634

Fischer S, Linsbauer L, Lopez-Herrejon R, Egyed A (2015) The ECCO tool: Extraction and composition for clone-and-own. In: IEEE/ACM 37th IEEE international conference on software engineering (ICSE), vol. 2, pp 665–668. doi:10.1109/ICSE.2015.218

Fischer S, Linsbauer L, Lopez-Herrejon R, Egyed A (2014) Enhancing clone-and-own with systematic reuse for developing software variants. In: IEEE 30th international conference on software maintenance and evolution, ICSME. IEEE Computer Society, Washington, DC, pp 391–400. doi:10.1109/ICSME.2014.61

Frenzel P, Koschke R, Breu A, Angstmann K (2007) Extending the reflexion method for consolidating software variants into product lines. In: 14th working conference on reverse engineering (WCRE), pp 160–169. doi:10.1109/WCRE.2007.28

Gamez N, Fuentes L (2011) Software product line evolution with cardinality-based feature models. In: Schmid K (ed) Top Productivity through Software Reuse, Lecture Notes in Computer Science, vol. 6727, pp 102–118. Springer Berlin Heidelberg. doi:10.1007/978-3-642-21347-2_9

Gamez N, Fuentes L (2013) Architectural evolution of famiware using cardinality-based feature models. Inf Softw Technol 55(3):563–580. doi:10.1016/j.infsof.2012.06.012. Special Issue on Software Reuse and Product Lines

Gharsellaoui H, Maazoun J, Bouassida N, Ben Ahmed S, Ben-Abdallah H (2015) A real-time scheduling of reconfigurable os tasks with a bottom-up spl design approach. In: 2015 International conference on evaluation of novel approaches to software engineering (ENASE), pp 169176

Guzman E, Maalej W (2014) How do users like this feature? a fine grained sentiment analysis of app reviews. In: 22nd IEEE international requirements engineering conference (RE), pp 153–162. doi:10.1109/RE.2014.6912257

Hariri N, Castro-Herrera C, Mirakhorli M, Cleland-Huang J, Mobasher B (2013) Supporting domain analysis through mining and recommending features from online product listings. IEEE Trans Softw Eng 39(12):1736–1752. doi:10.1109/TSE.2013.39

Haslinger E, Lopez-Herrejon R, Egyed A (2011) Reverse engineering feature models from programs' feature sets. In: 18th working conference on reverse engineering (WCRE), pp 308–312. doi:10.1109/WCRE.2011.45

Heidenreich F, Kopcsek J, Wende C (2008) Featuremapper: Mapping features to models. Companion of the 30th international conference on software engineering, ICSE companion. ACM, New York, pp 943–944. doi:10.1145/1370175.1370199

Kang K, Kim M, Lee J, Kim B (2005) Feature-oriented re-engineering of legacy systems into product line assets - a case study. In: Obbink H, Pohl K (eds) Software Product Lines, Lecture Notes in Computer Science, vol. 3714, pp 45–56. Springer Berlin Heidelberg. doi:10.1007/11554844_6

Kelly M, Alexander J, Adams B, Hassan A (2011) Recovering a balanced overview of topics in a software domain. In: 11th IEEE international working conference on source code analysis and manipulation (SCAM), pp 135–144. doi:10.1109/SCAM.2011.23x

Klatt B, Krogmann K, Seidl C (2014) Program dependency analysis for consolidating customized product copies. In: 30th IEEE international conference on software maintenance and evolution (ICSME), pp 496–500

Klatt B, Küster M, Krogmann K (2013) A graph-based analysis concept to derive a variation point design from product copies. In: International workshop on reverse variability engineering (REVE), pp 1–8

Knodel J, Forster T, Girard JF (2005) Comparing design alternatives from field-tested systems to support product line architecture design. In: Ninth european conference on software maintenance and reengineering (CSMR), pp 344–353. doi:10.1109/CSMR.2005.18

Knodel J, John I, Ganesan D, Pinzger M, Usero F, Arciniegas J, Riva C (2005) Asset recovery and their incorporation into product lines. In: 12th working conference on reverse engineering (WCRE), pp 1–10. doi:10.1109/WCRE.2005.8

Kolb R, Muthig D, Patzke T, Yamauchi K (2006) Refactoring a legacy component for reuse in a software product line: A case study. J Softw Maint Evol Res Pract 18(2):109–132. doi:10.1002/smr.v18:2

Koziolek H, Goldschmidt T, de Gooijer T, Domis D, Sehestedt S (2013) Experiences from identifying software reuse opportunities by domain analysis. In: 17th international software product line conference, SPLC. ACM, New York, pp 208–217. doi:10.1145/2491627.2491641

Kulesza U, Alves V, Garcia A, Neto A, Cirilo E, Lucena C, Borba P (2007) Mapping features to aspects: A model-based generative approach. In: Moreira A, Grundy J (eds) Early Aspects: Current Challenges and Future Directions, Lecture Notes in Computer Science, vol. 4765, pp 155–174. Springer Berlin Heidelberg. doi:10.1007/978-3-540-76811-1_9

Kumaki K, Tsuchiya R, Washizaki H, Fukazawa Y (2012) Supporting commonality and variability analysis of requirements and structural models. In: 16th international software product line conference - volume 2, SPLC. ACM, New York, pp 115–118. doi:10.1145/2364412.2364431

Lago P, Vliet H (2004) Observations from the recovery of a software product family. In: Nord R (ed) Software Product Lines, Lecture Notes in Computer Science, vol. 3154, pp 214–227. Springer Berlin Heidelberg. doi:10.1007/978-3-540-28630-1_13

Li S, Chen F, Liang Z, Yang H (2005) Using feature-oriented analysis to recover legacy software design for software evolution. In: International conference on software engineering and knowledge engineering (SEKE), pp 336–341

Li Y, Yin J, Shi D, Li Y, Dong J (2007) Software product line oriented feature map. In: Shi Y, Albada G, Dongarra J, Sloot P (eds) International Conference on Computational Science (ICCS), Lecture Notes in Computer Science, vol. 4488, pp 1115–1122. Springer Berlin Heidelberg. doi:10.1007/978-3-540-72586-2_156

Linsbauer L, Angerer F, Grünbacher P, Lettner D, Prähofer H, Lopez-Herrejon R, Egyed A (2014) Recovering feature-to-code mappings in mixed-variability software systems. In: IEEE 30th international conference on software maintenance and evolution, ICSME

Linsbauer L, Lopez-Herrejon ER, Egyed A (2013) Recovering traceability between features and code in product variants. In: 17th international software product line conference, SPLC. ACM, New York, pp 131–140. doi:10.1145/2491627.2491630

Linsbauer L, Lopez-Herrejon R, Egyed A (2014) Feature model synthesis with genetic programming. In: Le Goues C, Yoo S (eds) 6th Symposium on Search-Based Software Engineering (SSBSE), Lecture Notes in Computer Science, vol. 8636, pp 153–167. Springer International Publishing. doi:10.1007/978-3-319-09940-8_11

Lohar S, Amornborvornwong S, Zisman A, Cleland-Huang J (2013) Improving trace accuracy through data-driven configuration and composition of tracing features. In: 9th joint meeting on foundations of software engineering, ESEC/FSE. ACM, New York, pp 378–388. doi:10.1145/2491411.2491432

Lopez-Herrejon R, Linsbauer L, Galindo JA, Parejo J, Benavides D, Segura S, Egyed A (2015) An assessment of search-based techniques for reverse engineering feature models. J Syst Softw 103:353–369. doi:10.1016/j.jss.2014.10.037

Losavio F, Ordaz O, Levy N, Baiotto A (2013) Graph modelling of a refactoring process for product line architecture design. In: 39th latin american computing conference (CLEI), pp 1–12. doi:10.1109/CLEI.2013.6670632

Maazoun J, Bouassida N, Ben-Abdallah H (2014) A bottom up spl design method. In: 2nd international conference on model-driven engineering and software development (MODELSWARD), pp 309–316

Maâzoun J, Bouassida N, Ben-Abdallah H (2014) Feature model recovery from product variants based on a cloning technique. In: 26th international conference on software engineering and knowledge engineering (SEKE)

Maia MDA, Sobreira V, Paixão KR, Amo S, Silva IR (2008) Using a sequence alignment algorithm to identify specific and common code from execution traces. In: 4th international workshop on program comprehension through dynamic analysis (PCODA), pp 6–10

Martinez J, Ziadi T, Bissyandé TF, Klein J, Le Traon Y (2015) Bottom-up adoption of software product lines: a generic and extensible approach. In: 19th international conference on software product line, SPLC ACM, New York, pp 101–110. doi:10.1145/2791060.2791086

Martinez J, Ziadi T, Klein J, le Traon Y (2014) Identifying and visualising commonality and variability in model variants. In: Cabot J, Rubin J (eds) Modelling Foundations and Applications, Lecture Notes in Computer Science, vol. 8569, pp 117–131. Springer International Publishing. doi:10.1007/978-3-319-09195-2_8

Mefteh M, Bouassida N, Ben-Abdallah H (2014) Feature model extraction from documented UML use case diagrams. Ada User J 35(2):107–116

Merschen D, Polzer A, Botterweck G, Kowalewski S (2011) Experiences of applying model-based analysis to support the development of automotive software product lines. In: 5th international workshop on variability modelling of software-intensive systems, vamos. ACM, New York, pp 141–150. doi:10.1145/1944892.1944910

Mohamed F, Abu-Matar M, Mizouni R, Al-Qutayri M, Al Mahmoud Z (2014) Saas dynamic evolution based on model-driven software product lines. In: 6th international conference on cloud computing technology and science (cloudcom), pp 292–299. doi:10.1109/CloudCom.2014.131

Mu Y, Wang Y, Guo J (2009) Extracting software functional requirements from free text documents. In: International conference on information and multimedia technology (ICIMT), pp 194–198. doi:10.1109/ICIMT.2009.47

Nie K, Zhang L, Geng Z (2012) Product line variability modeling based on model difference and merge. In: IEEE 36th annual computer software and applications conference workshops (COMPSACW), pp 509–513. doi:10.1109/COMPSACW.2012.95

Niu N, Savolainen J, Niu Z, Jin M, Cheng JR (2014) A systems approach to product line requirements reuse. IEEE Syst J 8(3):827–836. doi:10.1109/JSYST.2013.2260092

Nöbauer M, Seyff N, Groher I (2014) Inferring variability from customized standard software products. In: 18th international software product line booktitle - volume 1, SPLC. ACM, New York, pp 284–293. doi:10.1145/2648511.2648544

Nöbauer M, Seyff N, Groher I (2014) Similarity analysis within product line scoping: An evaluation of a semi-automatic approach. In: Jarke M, Mylopoulos J, Quix C, Rolland C, Manolopoulos Y, Mouratidis H, Horkoff J(eds) Advanced Information Systems Engineering, Lecture Notes in Computer Science, vol. 8484, pp 165–179. Springer International Publishing. doi:10.1007/978-3-319-07881-6_12

Noor M, Grünbacher P, Hoyer C (2008) A collaborative method for reuse potential assessment in reengineering-based product line adoption. In: Meyer B, Nawrocki J, Walter B (eds) Balancing Agility and Formalism in Software Engineering, Lecture Notes in Computer Science, vol. 5082, pp 69–83. Springer Berlin Heidelberg. doi:10.1007/978-3-540-85279-7_6

Nunes C, Garcia A, Lucena C, Lee J (2012) History-sensitive heuristics for recovery of features in code of evolving program families. In: 16th international software product line conference - volume 1, SPLC. ACM, New York, pp 136–145. doi:10.1145/2362536.2362556

Nunes C, Garcia A, Lucena C, Lee J (2013) Heuristic expansion of feature mappings in evolving program families. Software: Practice and Experience pp 1–35. doi:10.1002/spe.2200

de Oliveira AL, Ferrari FC, Braga RT, Penteado RA, de Camargo VV (2012) Restructuring frameworks towards framework product lines. In: Latin american workshop on aspect-oriented software development (LA-WASP), pp 1–2

Olszak A, Jørgensen BN (2012) Remodularizing java programs for improved locality of feature implementations in source code. Sci Comput Program 77(3):131–151. doi:10.1016/j.scico.2010.10.007

Otsuka J, Kawarabata K, Iwasaki T, Uchiba M, Nakanishi T, Hisazumi K (2011) Small inexpensive core asset construction for large gainful product line development: Developing a communication system firmware product line. In: 15th international software product line conference, volume 2, SPLC. ACM, New York, pp 1–5. doi:10.1145/2019136.2019159

Passos L, Czarnecki K, Apel S, Wàsowski A, Kästner C, Guo J (2013) Feature-oriented software evolution. In: 7th international workshop on variability modelling of software-intensive systems, vamos. ACM, New York, pp 1–8. doi:10.1145/2430502.2430526

Peng X, Xing Z, Tan X, Yu Y, Zhao W (2013) Improving feature location using structural similarity and iterative graph mapping. J Syst Softw 86(3):664–676. doi:10.1016/j.jss.2012.10.270

Polzer A, Merschen D, Botterweck G, Pleuss A, Thomas J, Hedenetz B, Kowalewski S (2012) Managing complexity and variability of a model-based embedded software product line. Innov Syst Softw Eng 8(1):35–49. doi:10.1007/s11334-011-0174-z

Ramos MA, Penteado RA (2008) Embedded software revitalization through component mining and software product line techniques. J Universal Comput Sci 14(8):1207–1227

Romero D, Urli S, Quinton C, Blay-Fornarino M, Collet P, Duchien L, Mosser S (2013) SPLEMMA: A generic framework for controlled-evolution of software product lines. In: 5th international workshop on model-driven approaches in software product line engineering; 4th workshop on scalable modeling techniques for software product lines, MAPLE/SCALE. New York, pp 59–66. doi:10.1145/2499777.2500709

Rubin J (2014) Cloned product variants: From ad-hoc to well-managed software reuse. Ph.D. thesis, University of Toronto Graduate Department of Computer Science

Rubin J, Chechik M (2010) From products to product lines using model matching and refactoring. In: 2nd international workshop on model-driven approaches in software product line engineering (MAPLE), collocated with the 14th international software product line conference, pp 1–8

Rubin J, Chechik M (2012) Combining related products into product lines. In: 15th International Conference on Fundamental Approaches to Software Engineering, FASE, pp 285–300. Springer-Verlag, Berlin, Heidelberg. doi:10.1007/978-3-642-28872-2_20

Rubin J, Chechik M (2012) Locating distinguishing features using diff sets. In: 27th IEEE/ACM international conference on automated software engineering, ASE. ACM, New York, pp 242–245. doi:10.1145/2351676.2351712

Rubin J, Czarnecki K, Chechik M (2013) Managing cloned variants: a framework and experience. In: 17th international software product line conference, SPLC. ACM, New York, pp 101–110. doi:10.1145/2491627.2491644

Rubin J, Czarnecki K, Chechik M (2015) Cloned product variants: from ad-hoc to managed software product lines. Int J Softw Tools Technol Trans 17(5):627–646. doi:10.1007/s10009-014-0347-9

Rubin J, Kirshin A, Botterweck G, Chechik M (2012) Managing forked product variants. In: 16th international software product line conference - volume 1, SPLC. ACM, New York, pp 156–160 doi:10.1145/2362536.2362558

Ryssel U, Ploennigs J, Kabitzsch K (2011) Extraction of feature models from formal contexts. In: 15th international software product line conference - volume 2, SPLC. ACM, New York, pp 1–8. doi:10.1145/2019136.2019141

Sampath P (2013) An elementary theory of product-line variations. Formal Aspects of Computing pp 1–33. doi:10.1007/s00165-013-0276-5

Santos A, Gaia F, Figueiredo E, Neto PS, Araújo JA (2013) Test-based SPL extraction: An exploratory study. In: 28th symposium on applied computing, SAC. ACM, New York, pp 1031–1036. doi:10.1145/2480362.2480559

Schulze M, Mauersberger J, Beuche D (2013) Functional safety and variability: Can it be brought together? In: 17th international software product line conference, SPLC. ACM, New York, pp 236–243. doi:10.1145/2491627.2491654

Segura S, Parejo J, Hierons RM, Benavides D, Ruiz-Cortés A, De andalucía EDLJ (2012) ETHOM: An evolutionary algorithm for optimized feature models generation. Tech. Rep. ISA-2012-TR-01, Applied Software Engineering Research Group, Department of Computing Languages and tab9 University of Sevilla

Seidl C, Heidenreich F, Aßmann U (2012) Co-evolution of models and feature mapping in software product lines. In: 16th international software product line conference - volume 1, SPLC. ACM, New York, pp 76–85. doi:10.1145/2362536.2362550

Shao J, Wu W, Geng P (2013) An improved approach to the recovery of traceability links between requirement documents and source codes based on latent semantic indexing. In: Murgante B, Misra S, Carlini M, Torre C, Nguyen HQ, Taniar D, Apduhan B, Gervasi O (eds) Computational Science and Its Applications (ICCSA), Lecture Notes in Computer Science, vol. 7975, pp 547–557. Springer Berlin Heidelberg. doi:10.1007/978-3-642-39640-3_40

She S (2013) Feature model synthesis. Ph.D. thesis, University of Waterloo Electrical and Computer Engineering Department

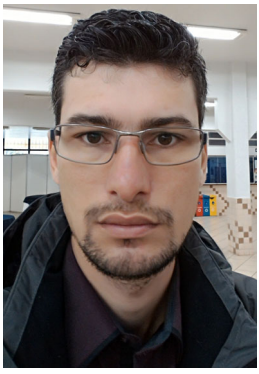
She S, Lotufo R, Berger T, Wąsowski A, Czarnecki K (2011) Reverse engineering feature models. 33rd international conference on software engineering, ICSE. ACM, New York, pp 461–470. doi:10.1145/1985793.1985856

She S, Ryssel U, Andersen N, Wąsowski A, Czarnecki K (2014) Efficient synthesis of feature models. Inform Softw Technol 0(0):1–22. doi:10.1016/j.infsof.2014.01.012

Stoermer C, O'Brien L (2001) MAP - Mining architectures for product line evaluations. In: Working IEEE/IFIP conference on software architecture (WICSA), pp 35–44. doi:10.1109/WICSA.2001.948405

Stuikys V, Valincius K (2011) A domain understanding through context-based feature modelling: a research framework. In: 17th international conference on information and software technologies (ICIST), pp 141–148

Tang Y, Leung H (2015) Top-down feature mining framework for software product line. In: 17th international conference on enterprise information systems, pp 71–81. doi:10.5220/0005370300710081
Trifu M (2010) Tool-supported identification of functional concerns in object-oriented code. Ph.D. thesis, Karlsruhe Institute of Technology
Valinčius K, Štuikys V, Damaševičius R (2013) Understanding of e-commerce is through feature models and their metrics to support re-modularization. International Journal on Computer Science and Information Systems (IADIS) 8(1):47–65
Van Der Storm T (2007) Generic feature-based software composition. In: 6th international conference on software composition (SC), SC'07. Springer-Verlag, Berlin, Heidelberg, pp 66–80
Weston N, Chitchyan R, Rashid A (2009) A framework for constructing semantically composable feature models from natural language requirements. In: 13th international software product line conference, SPLC. Carnegie Mellon University, Pittsburgh, pp 211–220
Xue Y (2012) Reengineering legacy software products into software product line. Ph.D. thesis, National University of Singapore Department of Computer Science
Xue Y, Xing Z, Jarzabek S (2010) Understanding feature evolution in a family of product variants. In: 17th working conference on reverse engineering (WCRE), pp 109–118
Xue Y, Xing Z, Jarzabek S (2012) Feature location in a collection of product variants. In: 19th working conference on reverse engineering (WCRE), pp 145–154. doi:10.1109/WCRE.2012.24
Yang Y, Peng X, Zhao W (2009) Domain feature model recovery from multiple applications using data access semantics and formal concept analysis. In: 16th working conference on reverse engineering (WCRE), pp 215–224. doi:10.1109/WCRE.2009.15
Yu Y, Wang H, Yin G, Liu B (2013) Mining and recommending software features across multiple web repositories. ACM, New York, doi:10.1145/2532443.2532453
Zhang G, Shen L, Peng X, Xing Z, Zhao W (2011) Incremental and iterative reengineering towards software product line: an industrial case study. In: 27th IEEE international conference on software maintenance (ICSM), pp 418–427. doi:10.1109/ICSM.2011.6080809
Ziadi T, Frias L, da Silva M, Ziane M (2012) Feature identification from the source code of product variants. In: 16th european conference on software maintenance and reengineering (CSMR), pp 417–422. doi:10.1109/CSMR.2012.52
Ziadi T, Henard C, Papadakis M, Ziane M, Le Traon Y (2014) Towards a language-independent approach for reverse-engineering of software product lines. In: 29th Symposium On Applied Computing (SAC) pp 1064–1071



**Wesley K. G. Assunção** received a bachelor's degree in Information Systems from Faculdade Sul Brasil in 2006 and the MSc degree in 2012 from Federal University of Paraná (UFPR), Brazil. He is currently PhD candidate at the Post-graduation Program in Informatics of Federal University of Paraná (UFPR), being a member of Research Group on Software Engineering - GRES. His areas of interest are: Software Testing, Software Product Lines, Search Based Software Engineering and Multi-Objective Evolutionary Algorithms.

**Roberto E. Lopez-Herrejon** is an Associate Professor at the Department of Software Engineering and Information Technology of the École de Technologie Supérieure of the University of Quebec in Montreal, Canada. Prior he was a senior postdoctoral researcher at the Johannes Kepler University in Linz, Austria. He was an Austrian Science Fund (FWF) Lise Meitner Fellow (2012–2014) at the same institution. From 2008 to 2014 he was an External Lecturer at the Software Engineering Masters Programme of the University of Oxford, England. From 2010 to 2012 he held an FP7 Intra-European Marie Curie Fellowship sponsored by the European Commission. He obtained his Ph.D. from the University of Texas at Austin in 2006, funded in part by a Fulbright Fellowship sponsored by the U.S. State Department. From 2005 to 2008, he was a Career Development Fellow at the Software Engineering Centre of the University of Oxford sponsored by Higher Education Founding Council of England (HEFCE). His main expertise is in software customization, software product lines, and search based software engineering.

**Lukas Linsbauer** is currently a PhD student at the Institute for Software Systems Engineering at the Johannes Kepler University (JKU) in Linz, Austria under the supervision of Prof. Alexander Egyed and Dr. Roberto Erick Lopez-Herrejon. He received his master's degree in computer science from the JKU after only four years of study for each of which he received a merit scholarship. His research interests are in traceability, software product lines, variability modeling and management, and highly variable and configurable systems.

**Silvia R. Vergilio** received the MS (1991) and DS (1997) degrees from University of Campinas, UNICAMP, Brazil. She is currently at the Computer Science Department at the Federal University of Paraná, Brazil, where she has been a faculty member since 1993. She has been involved in several projects and her research interests are in the areas of Software Engineering, such as: software testing, software architecture, software metrics, and search-based software engineering.

**Alexander Egyed** heads the Institute for Software Systems Engineering (ISSE) at the Johannes Kepler University, Austria. He received his Doctorate from the University of Southern California, USA and previously worked many years in industry before joining academia. Dr. Egyed was recognized among the Top 10 scholars in software engineering and his work has received numerous awards.