Received: 6 January 2017 | Revised: 23 May 2017 | Accepted: 25 August 2017

DOI: 10.1002/smr.1912

WILEY Journal of Software: Evolution and Process

RESEARCH ARTICLE - EMPIRICAL

A systematic mapping study of information visualization for software product line engineering

Roberto Erick Lopez-Herrejon[1] | **Sheny Illescas**[2] | **Alexander Egyed**[2]

[1]École de technologie supérieure, Université du Québec, Montreal, Canada
[2]Johannes Kepler University Linz, ,Linz, Austria

Correspondence
Roberto Erick Lopez-Herrejon, Department of Software Engineering and IT, École de technologie supérieure, Montreal, Canada.
Email: roberto.lopez@etsmtl.ca

Funding information
Austrian Science Fund (FWF), Grant/Award Number: P25289-N15 and P25513-N15; NSERC, Grant/Award Number: RGPIN-2017-05421

Abstract

Software product lines (SPLs) are families of related systems whose members are distinguished by the set of features they provide. Over 2 decades of research and practice can attest to the substantial benefits of applying SPL practices such as better customization, improved software reuse, and faster time to market. Software product line engineering (SPLE) refers to the paradigm of developing SPLs. Typical SPLE efforts involve a large number of features that are combined to form also large numbers of products, implemented using multiple and different types of software artifacts. Because of the sheer amount of information and its complexity, visualization techniques have been used for different SPLE activities. In this paper, we present an extended systematic mapping study on this subject. Our research questions aim to gather information regarding the techniques that have been applied, at what SPLE activities, how they were implemented, the publication fora used, the methods of empirical evaluation, and the provenance of the evaluation examples. Our driving goal is to identify common trends, gaps, and opportunities for further research and application.

KEYWORDS

software product lines, software product line engineering, systematic mapping study, visualization

1 | INTRODUCTION

Software product lines (SPLs) are families of related systems whose members are distinguished by the set of features they provide.[1,2] *Variability* is the capacity of software artifacts to vary, and its effective management and realization lie at the core of successful SPL development.[3] *Feature models* are tree-like structures that establish the relations between features and have become the de facto standard for modeling variability.[4,5] Over the last decades, extensive research and practice both in academia and industry attest to the substantial benefits of applying SPL practices.[2,6,7] Among the benefits are better customization, improved software reuse, and faster time to market.

Software product line engineering (SPLE) refers to the paradigm of developing SPLs. Typical SPLE efforts involve a large number of features that are combined in complex feature relations yielding a large number of individual software systems that must be effectively and efficiently designed, implemented and managed. Precisely, this fact is what makes SPLE problems suitable for the application of visualization techniques. This application has been explored by several researchers and has produced a number of publications on the subject.

This is precisely what originally prompted us to perform a *systematic mapping study* (SMS) to provide an overview of the research at the intersection of these 2 fields,[8-10] in contrast with a *systematic literature review* (SLR) whose goal is primarily to identify best practice. [8,10-12]

This paper extends our earlier mapping study,[13] whose general goal was to identify the quantity and the type of research and results available (eg, techniques and tools used) and thus highlight possible open research problems and opportunities, for both visualization and SPLE communities. Our focus for that paper was on identifying the activities of SPLE where visualization techniques have been used, which techniques, the tools used for their implementation, and the fora where the research work has been published.

We extend that paper by adding 2 new research questions (ie, RQ5 and RQ6) that concern the type of empirical study and the provenance of the examples used for the empirical evaluation. Furthermore, we used a new indexing search engine and re-executed all our search queries to update and enhanced our previous findings. We identified 37 primary sources, 5 new ones in comparison with the earlier work, and provide a more detailed description of the results and their analysis. We corroborated our earlier findings of the preeminent use of visualization techniques for SPLE activities of capturing and designing SPLs and their configuration. Additionally, we found that case studies of academic examples are the main form of

empirical evaluation, but there were also a few experiments. We hope that this mapping study not only serves to highlight the main research topics at the intersection of visualization and SPLE but that it also serves to encourage researchers to pursue work at the intersection of both areas.

The paper is structured as follows. Section 2 provides the basic background on SPLs and visualization terminology. Section 3 presents the process we followed for our SMS. It details the research questions addressed, how the search was performed, the classification scheme used, and how the data was extracted and analyzed. Section 4 presents the results we obtained for each research question. Section 5 presents our analysis of the results found. Section 6 describes the research avenues that our study identified for further investigation. Section 7 summarizes the threats to validity of our mapping study and how they were addressed. Section 8 concisely presents the existing review studies and surveys of SPLs and visualization. Section 9 summarizes the conclusions of our study.

2 | SOFTWARE PRODUCT LINES AND VISUALIZATION BACKGROUND

In this section, we present the basic concepts and terminology of SPLE and SPLs and provide the working definitions of visualization and its subfields of information visualization and software visualization.

2.1 | Software product lines overview

As mentioned before, SPLs are families of related systems whose members are distinguished by the set of features they provide.[1,2] Software product line engineering refers to the paradigm of developing SPLs. There is an extensive body of research that attests to the benefits of SPL practices and that has proposed multiple SPLE approaches, methods, and techniques (eg, [2,14-16]).

2.1.1 | Feature models

Recall that a core concept in SPLs is *variability*, which refers to the capacity of software artifacts to vary.[3] The software products that constitute an SPL are characterized by the different combinations of features they have. These combinations are captured in *variability models* for which there are different alternatives[17]; however, feature models have become a de facto standard.[4] In this type of model, features are depicted as labelled boxes and their relationships as lines, collectively forming a tree-like structure. The typical graphical notation for feature models is shown in Figure 1.

A feature can be classified as *mandatory*, which is selected whenever its parent feature is also selected (eg, feature `B` in Figure 1A), and *optional*, which may or may not be part of a program whenever its parent feature is selected (eg, feature `B` in Figure 1B). Features can also be grouped into *alternative groups* and *or groups*. In alternative groups, if the parent feature of the group is selected, exactly one feature from the group must be selected. For example, Figure 1C illustrates that if feature `P` is selected, then one of the group features `C1`, `C2`, or `C3` must be selected. In *or* groups if the parent feature of the group is selected, then one or more features from the group can be selected. For example, Figure 1D shows that if feature `P` is selected, one or more features among `C1`, `C2`, or `C3` must be selected. In addition to hierarchical parent-child relations, features can also relate across different branches of the feature model with *cross-tree constraints (CTCs)*.[5] The typical examples of this kind of relations are as follows: (1) *requires* relation whereby if a feature `A` is selected a feature `B` must also be selected and (2) *excludes* relation whereby if a feature `A` is selected then feature `B` must not be selected and vice versa. In a feature model, these latter relations are commonly depicted with dotted single-arrow lines and dotted double-arrow lines, respectively, see Figure 1E.

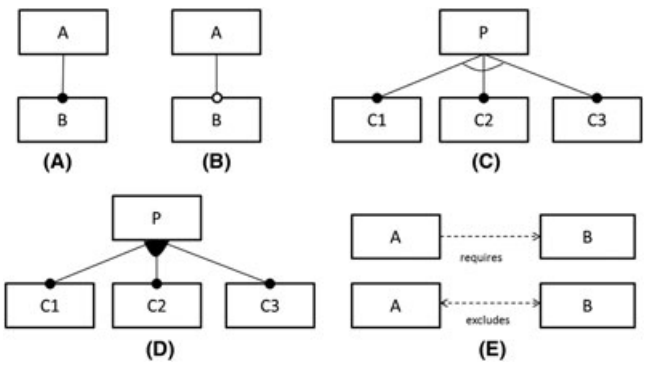

FIGURE 1 Feature models graphical notation

2.1.2 | Software product line engineering framework

As described before, there are many SPLE approaches (eg, previous studies[2,14-16]). For our study, we selected the SPLE framework proposed by Pohl et al,[2] shown in Figure 2. This framework is well known within the SPLE research community and has been used to highlight some of the open questions and challenges in the field of SPLE.[18] Additionally, we have used this framework in other SMSs on SPLs for other domains different from visualization.[19,20] This framework defines 2 main SPLE activities as follows [2]:

Definition 1. Domain engineering (DE) is the process of SPLE in which the commonality and the variability of the product line are defined and realized.

Definition 2. Application engineering (AE) is the process of SPLE in which the applications of the product line are built by reusing domain artifacts and exploiting the product line variability.

Each of the 2 main activities is divided in 4 subactivities[2]:

- *Domain requirements engineering (DRE)* is the subactivity of DE where the common and variable requirements of the product line are defined, documented in reusable requirements artifacts, and continuously managed.
- *Domain design (DD)* is the subactivity of DE where a reference architecture for the entire SPL is developed.
- *Domain realisation (DR)* is the subactivity of DE where the set of reusable components and interfaces of the product line is developed.
- *Domain testing (DT)* is the subactivity of DE where evidence of defects in domain artifacts is uncovered and where reusable test artifacts for application testing are created.
- *Application requirements engineering (ARE)* is the subactivity of AE dealing with the elicitation of stakeholder requirements, the creation of the application requirements specification, and the management of application requirements.
- *Application design (AD)* is the subactivity of AE where the reference architecture is specialized into the application architecture.
- *Application realisation (AR)* is the subactivity of AE where a single application is realized according to the application architecture by reusing DR artifacts.
- *Application testing (AT)* is the subactivity of AE where domain test artifacts are reused to uncover evidence of defects in an application.

For the sake of simplicity and brevity, henceforth, we refer to each subactivity of the DE and AE described above as an *activity* of an SPL life cycle. We use these terms for the classification of the visualization techniques as described in Section 3.4. We made this decision because DE and AE are the 2 common activities in all SPLE approaches and because they have a clear distinction between their goals as stated in their definitions. In addition to the 8 activities of this framework, we considered one more activity to cover all maintenance and evolution issues of SPLE, which we defined as follows:

- *Maintenance and evolution (ME)* refers to the maintenance and evolution of all the artifacts developed across the entire life cycle of SPLs. Reverse engineering artifacts or bug fixing are examples of activities that fall in this category.

2.2 | Visualization terminology

There are multiple definitions for visualization and its subfields. In this section, we provide our working definitions that we will rely on throughout the paper.

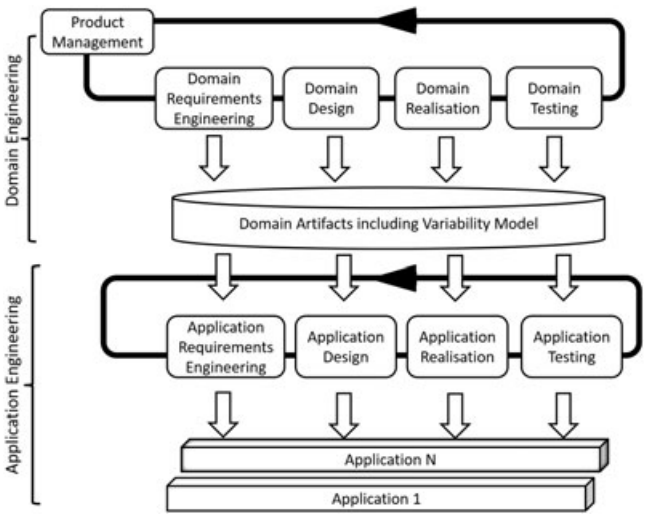

FIGURE 2 Software product line engineering framework of Pohl et al, Figure 2-1 from the same study[2]

Definition 3. Visualization is the visual representation of a domain space using graphics, images, animated sequences, and sound augmentation to present the data, structure, and dynamic behavior of large, complex data sets that represent systems events, processes, objects, and concepts (William et al[21]).

Definition 4. Information visualization is visualization applied to abstract quantities and relations to get insight in the data (Chi[22]). The most common examples of visualization techniques are trees, maps, and graphs.[23]

Definition 5. Software visualization is the art and science of generating visual representations of various aspects of software and its development process. The goal of software visualization is to help to comprehend software systems and to improve the productivity of the software development process (Diehl[24]).

Based on these definitions, the term visualization has a broader scope, whereas information visualization and software visualization are subfields of the former.[23]

3 | SYSTEMATIC MAPPING STUDY

The main goal of *evidence-based software engineering (EBSE)* is "to provide the means by which current best evidence from research can be integrated with practical experience and human values in the decision making process regarding the development and maintenance of software."[25] There are several types of empirical studies that fall under the umbrella of EBSE.

One of the most common approaches advocated by EBSE is *SMSs* that search a broad field of expertise to get an overview of the state of art or state of practice on a topic.[11] Systematic mapping studies (SMSs) aim to provide an overview of the results available within an area by categorizing them along criteria such as type, forum, and frequency.[9,11,26] SMSs are used when the research questions for the literature review are broader or the field of study is less explored.[11] Closely related to SMSs are *SLRs* whose main role is establishing whether particular techniques or practices work better than other and under what conditions.[26] Some examples are identifying the benefits of using tools in a particular context or the state of adoption in industry of a particular tool or development approach. [26]

We opted to use an SMS because our goal was to provide an overview of the state-of-the-art research at the intersection of 2 fields. We performed our study based on the protocol proposed by Petersen et al,[9] whose main stages we present in Figure 3. Next, we describe each of the processes and how they were performed for our mapping study. In Section 4, we present the results obtained, and in Section 5 their analysis.

3.1 | Definition of research questions

The main goal of our work is to provide an overview of research that applies visualization techniques in SPLE activities. Therefore, our driving motivation is to gather and summarize evidence of research that lies at the intersection of the research fields of software visualization and SPLE. Our mapping study then focuses on the following research questions:

- ***RQ1. In what activities of SPLE have visualization techniques been used?*** *Rationale:* Visualization techniques have been applied at many stages of software development, so our interest is finding out where they have been used throughout the entire life cycle of SPLs[2] as explained in Section 2.1.2.
- ***RQ2. What visualization techniques have been used?*** *Rationale:* There are a large number of visualization techniques available in literature. Our goal here is cataloguing their use for SPLE activities and analyze if there are common trends in their application.
- ***RQ3. What visualization tools have been used?*** *Rationale:* There exist many tools, libraries, APIs, etc that support multiple visualization techniques for different platforms. The goal of this question is to catalogue those that have been used in SPLE activities.
- ***RQ4. What are the publication fora used?*** *Rationale:* Software visualization and SPLE research appears in many conferences, journals, workshops, etc in a large array of research communities. Hence, identifying the publication fora may be beneficial for researchers wanting to keep up to date on development on the subject as well as to seek collaborations or to publish the results of their research.

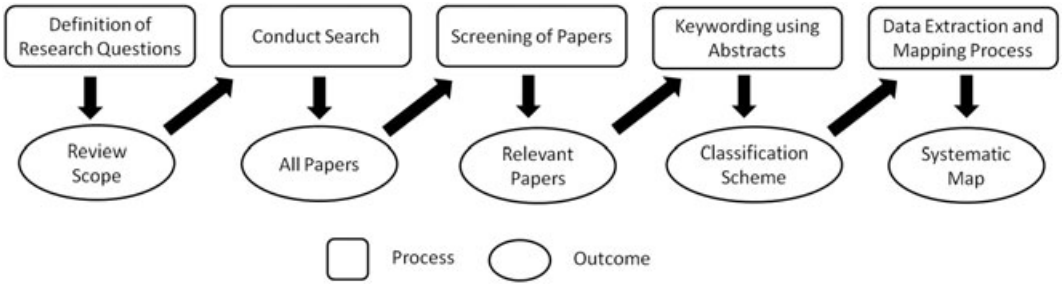

FIGURE 3 Systematic mapping study process[9]

TABLE 1 Summary of software product line engineering (SPLE) and visualization search terms

SPLE 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
Visualization terms: visual, visualization, visualizing, information visualization, software visualization

- ***RQ5. What forms of empirical evaluation have been used?*** *Rationale:* Here, we categorize the primary sources according to the type of empirical studies used for their evaluation. The goal of this question is to assess the empirical strength put forward to support the use of a visualization approach for SPLE activities.
- ***RQ6. What is the provenance of evaluation examples used?*** *Rationale:* Here, our focus is on cataloguing the number and provenance of the examples used for the empirical evaluation.

3.2 | Conduct search for primary sources

In this step of the systematic mapping, the terms to be used for the search are defined. Because our study focuses on the intersection of 2 fields, SPLE and visualization, we used 2 sets of terms, one for each field. Table 1 shows the list of all search terms we used.* Regarding the SPLE terms, they come from our 2 previous mapping studies[19,20] and are based on the terms collected from 12 systematic mapping and literature review studies in SPLs. [15,27-37] It is important to remark that none of these mapping studies are related to software visualization as will be detailed in Section 8. For gathering the terms for visualization, we selected them from 7 surveys and studies in the area of visualization.[38-44] Again, none of these works relate to SPLE.

We conducted the search process in 2 stages. First, we used the search engines of publishing companies and organizations we have used in our previous conference paper[13]: ScienceDirect, IEEExplore, ACM Digital Libray, and SpringerLink. We additionally used the Web of Science search engine of Thomson Reuters. These search engines are capable of indexing the common publishing outlets that contain journals, conferences, and workshops in both SPLE and software visualization. At the second stage, we performed so-called *snowballing readings*, which refer to those papers that are either cited or cite the papers obtained in the first search stage.[10,45] We performed the second stage manually following the citation links provided by the publishing companies and also with Google Scholar.

The queries we performed took all the combinations of one term from the visualization list and one or more terms of the SPLE terms depending on the querying functionality of each search engine. The searches considered the title, abstract, and keywords of the papers, and when supported by the search engine also their contents. As an example, consider the following a query fragment used in the IEEExplore engine†:

```
("visualization") AND ("software product line OR "feature model" OR "variability management"
OR"product line engineering")
```

3.3 | Screening of papers for inclusion and exclusion

During the screening process, we looked for the search terms in the title, abstract, and keywords and whenever necessary at the introduction or at other places of the paper. The sole criterion for inclusion of a primary source in our mapping study was a description, anywhere on the text, of an application of a visualization technique to an SPLE activity.

The criteria to exclude papers in our study were: (1) papers that did not apply any visualization techniques to SPLE activities‡, (2) papers not written in English, (3) vision or position papers that had no implementation to back them up, (4) graduate or undergraduate dissertations and thesis, and (5) non–peer-reviewed documents such as technical reports.

The decision on whether or not to include a paper was in most cases straightforward, in other words, that at least one visualization term was found and a clear connection to an SPLE activity could be drawn. While performing the searches, we found out for a couple of approaches that there were papers that presented fundamentally the same approach (eg, firstly published as a part of research paper and secondly published as a tool paper). For such cases, we included the paper that was published first and excluded the other related ones. However, we kept those subsequent papers whenever they contributed new material for the approach in question, for example, an application to a different problem.

*Alternative term spellings or hyphenation are not shown in the table and were found not to be relevant for our searches.
†The search queries had to be broken down into smaller queries (as shown in the example) because of the search limitations of some search engines. We made sure however that we considered all possible combinations in the cartesian product of the visualization and SPLE terms.
‡We should mention that we included papers that apply visualization techniques even though the application was not their main focus or contribution.

3.4 | Keywording using abstracts—classification scheme

We classified our articles into 6 dimensions aligned with each research question that our systematic mapping study addresses.

3.4.1 | Software product line engineering activity classification

For this classification dimension, we used the 8 activities derived from the framework of Pohl et al,[2] plus the ME activity as described in Section 2.1.2. We decided to use this framework because it is already familiar within the SPLE community and it is readily accessible for software visualization researchers. This approach is an alternative to the standard classification procedure of SMSs whereby the classification schemes follow from key words identified in the primary sources. We should remark that for this dimension, a primary source can be classified in more than one category.

3.4.2 | Visualization techniques classification

For this classification, we considered each different visualization technique found in our primary sources as a category following standard terminology from the field such as trees, graphs, and bubble or heat maps. [23,46] We should also remark that for this dimension, a primary source can also be classified in more than one category.

3.4.3 | Visualization tools classification

For this classification, we considered each different tool, library, framework, or special-purpose language mentioned in the primary sources. We included an extra category *ad hoc* for those cases where there is no explicit mention of the implementation details and the tool support could not be traced through the paper references or authors' websites.

3.4.4 | Type of publication fora classification

The classification of publication fora is straightforward because we used the name of the journal, conference, or workshop where the publication appeared.

3.4.5 | Empirical evaluation

Here, we categorize the primary sources according to the type of empirical study used for their evaluation. For our study, we consider the following categories[11]:

- *Case study* is an empirical inquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified.[47]
- *Experiment* (or controlled experiment) is an empirical inquiry that manipulates one factor or variable of the studied setting. Based on randomization, different treatments are applied to or by different subjects, while keeping other variables constant, and measuring the effects on outcome variables. In human-oriented experiments, humans apply different treatments to objects, while in technology-oriented experiments, different technical treatments are applied to different objects.
- *None* when no evaluation is provided in the primary source.

3.4.6 | Provenance of evaluation examples

We classified publications according to the number of case studies, the types of artifacts, and the provenance of the artifacts. We should point out that for this classification, we considered the artifacts that contained or expressed the variability of the SPL, eg, feature models or UML class diagrams. For artifact provenance, we defined the following categories:

- *Random* when the examples are generated randomly.
- *Open source* when examples come from projects developed as open source, eg, Linux kernel.[48]
- *Academic* when the examples come from academia, either from research papers or projects mainly conducted at research or university institutions.
- *Industrial* when the examples belong to actual industrial cases.
- *None* when no examples are used for the evaluation.

3.5 | Data extraction and mapping study

We used the same process to gather the data for our mapping study that we used in our earlier conference paper[13] and extended it to obtain the information for our 2 new research questions RQ5 and RQ6. This process consists of the following steps:

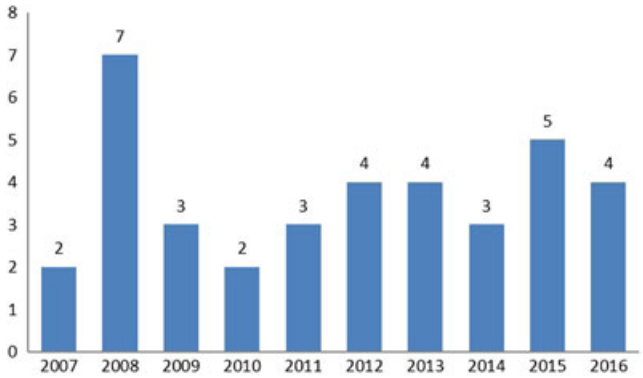

FIGURE 4 Publications per year

1. We created a guideline document to define each of the classification terms and an Excel spreadsheet to collect the classification information. The spreadsheet contained the following data fields: (1) SPLE activity, (2) visualization techniques used, (3) visualization tools used, (4) empirical evaluation support, (5) provenance of evaluation examples, and (6) a general field for any remarks. Additionally, for each classification, there was a spreadsheet cell to provide rationale to justify classification selected.
2. We formed 2 groups, both with background knowledge on SPLE and visualization, to perform the classification task independently.
3. We held a meeting to pilot the classification terms. In this meeting, each group presented its classification of a group of 5 selected primary sources. Any discrepancies were discussed and analyzed to homogenize the classification criteria.
4. The 2 teams performed the classification of all primary sources independently.
5. We held a second meeting where the classification for every single paper for each criterion was discussed until a consensus was reached.

We revised the collected data from our earlier conference paper[13] and extended it with the information of the new questions. In general, the effort to gather the data varied between papers but for the large majority, it was a simple task to find all the classification information required. The most time-consuming parts were in some cases determining the implementation tools used and the provenance of the evaluation examples.

4 | RESULTS

In our previous conference paper,[13] we identified 32 primary sources using 4 search engines. For this new extension, we re-executed the search queries in those 4 engines and performed the queries anew in Web of Science, from November 22 to December 12, 2016. In total, we obtained 792 hits. We performed a more detail reading of the title, abstract, and keywords to gauge the relevance of the papers found. As a result, we obtained 44 relevant papers. The most common reason for exclusion was that those papers did not apply visualization techniques to SPLE activities, for example, some simply mention visualization as part of future work but provide no actual application, or they referred to product lines but in other domains and not in software. The exclusion of each paper was double-checked to make sure we did not eliminate any relevant primary source.

We performed snowballing on the relevant 44 papers but did not identify any more relevant papers. After close examination, we eliminated 7 papers yielding a total of 37 primary sources for our mapping study listed in Appendix. We identified the 32 primary sources of our previous work[13] and 5 new primary sources of which 4 were recent publications and one was picked up by the new search engine we used for our current work. Figure 4 shows a histogram with the publications by year. This figure shows a spike in the number of publications in 2008, and since then a constant interest in the topic.

In the following subsections, we present the results obtained for each of our research questions, while their collective analysis is presented in Section 5. Further details are available at the Mendeley group called Information Visualization for Variable Software Systems.[§]

4.1 | RQ1. software product line engineering activities

Table 2 shows the use of visualization techniques per SPLE activity, which are summarized in Figure 5. They show that the DE activities of requirements engineering (DRE) and design (DD), with 24 and 26 primary sources, respectively, are the most preeminent activities where visualization techniques have been used. Do notice, as explained in Section 2, that these 2 activities are when feature models are defined and the requirements of the features are frequently reified as attributes of the feature models.[5] We believe that this known fact helps to explain this first finding.

Our study found respectively 16 and 14 primary sources for the activities where the requirements for each product are captured (ARE) and analyzed (AD). In these activities, software engineers commonly rely on feature models to guide the product configuration process, whereby they select the desired combinations of features and analyze different trade-offs. We argue that the use of feature models at these activities explains this finding.

[§]https://www.mendeley.com/groups/11683371/infovisvar/

TABLE 2 Primary sources and SPLE activity

Stage	Primary Sources Identifiers
DRE	64,65,70,71,73,74,76,77,78,79,81,82,85,87,88,89,92,94,95,96,97,98,99,100
DD	64,65,66,70,71,73,74,76,77,78,79,81,82,85,87,88,89,90,92,94,95,96,97,98,99,100
DR	64,65,66,68,75,83,89,90,95,96,97,99
DT	69,72
ARE	64,69,71,73,76,78,79,80,84,86,89,91,92,93,95,100
AD	64,71,73,76,78,79,80,86,89,91,92,93,95,100
AR	64,71,89,95
AT	None
ME	67,68,69,70,71

DRE, domain requirements engineering;
DD, domain design; DR, domain realization;
DR, domain realization;
DT, domain testing; DR, domain realization;
ARE, application requirements engineering;
AD, application design;
AR, application realization;
AT, application testing;
ME, maintenance and evolution.

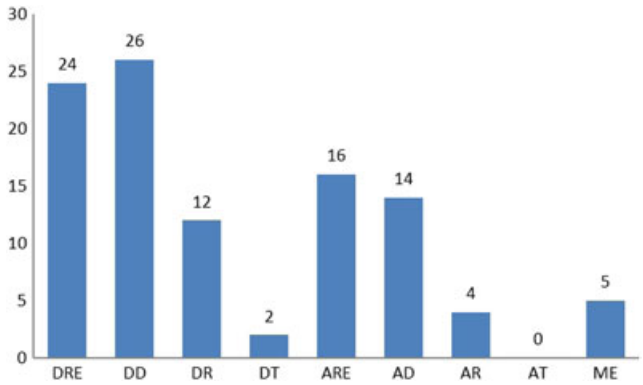

FIGURE 5 DRE, domain requirements engineering; DD, domain design; DR, domain realization; DT, domain testing; ARE, application requirements engineering; AD, application design; AR, application realization; AT, application testing; ME, maintenance and evolution

For the activity where the product line is realized at the domain level (DR), we found 12 primary sources. Here, our study found that colors are used to describe what software artifacts or their pieces belong to particular features. For instance, Heidenreich et al use colors to annotate UML-based models,[64] Kästner et al follow the same idea but applied to source code,[65] while Illescas et al use colors to distinguish features and their interactions.[66]

Our study found 5 primary sources for the ME activity. Kanda et al use a tree to depict the evolution of products across time,[67] while Wnuk et al use bars to depict the evolution of features also across time.[68] De Oliveira et al trace the evolution of bugs across product evolution.[69] Urli et al support for maintenance tasks concerning the feature models.[70] Anquetil et al focus on extracting and managing traceability links in models for evolution tasks.[71]

Our study found 4 primary sources for the AR stage. All of them use different notions of fragments of models (eg, features, concerns, and deltas), which are configured, analyzed, and composed. For this activity, these works primarily rely on colors to distinguish the fragments that are visualized as models.

We found that despite the extensive research on testing SPLs, as described in Section 8, only 2 primary sources exploit visualization techniques for testing. Lopez-Herrejon et al use basic techniques such as tree maps and bubble charts to depict covering arrays.[72] De Oliveira et al visualize bug evolution to help with SPL testing tasks.[69] Our study found no visualization techniques used for testing the AT stage.

4.2 | RQ2. Visualization techniques

Table 3 summarizes the use of visualization techniques. Trees were the most used technique with 15 primary sources. We argue that this can be explained given the number of primary sources that visualize different aspects of feature models, which are essentially trees. The second most used visualization technique was graphs with 9 primary sources. Concept lattices, a more specialized form of graphs, were used in 2 primary sources.

TABLE 3 Visualization techniques

Technique	Primary Sources Identifiers
Trees	67,69,71,74,79,82,83,85,86,87,89,91,92,93,100
Graphs (nodes and edges)	66,71,75,78,80,82,95,96,99
Concept lattices	76,77
Bar diagrams	81,94
Colored code/model elements	64,65
Feature histograms	83
Tables/matrices	88
Bubble chart	84
Levelized structure map	90
Bubble map	72
Heat map	72
Tree map	72
Grid	72
Feature blueprints	70
Feature relation graphs	73
Component model annotations	97
Flow maps	78
3D cone trees	98
Feature survival chart	68
3D color spheres	100
Logical gates	74
Nested colored circles	66

TABLE 4 Visualization tools

Tool	Primary Sources Identifiers
Adhoc	68,69,71,74,77,81,82,83,86,87,88,90,92,94,96,97,98,100
Eclipse EMF-GEF	64,65,80,89,91,93,95,99
Prefuse	78,79,85
D3.js	66,72
Graphviz	67
CCVisu	75
Google charts	84
ConExp	76
Moose	70
Processing 2.0	73

Bar diagrams and using color to distinguish elements in source code and models had also 2 primary sources. There were also 17 other visualization techniques with just a single primary source. Next, we describe some of the visualization techniques found by our study.

Feature blueprints are feature models where the size of the feature box depends on the number of internal and external constraints found in a feature and use colors to distinguish optional from mandatory features. [70] Feature relation graphs depict features and their relations as colored concentric circles whose color, width, and size depend on the properties of the relations.[73] Feature survival charts display one bar for each feature and use colors to describe its evolution, for example, when a feature is in the scope of a SPL and when it was deprecated.[68] Other examples are the use of logical gates for depicting feature dependencies[74] or nested circles to depict the hierarchical nesting of features and their components. [66]

4.3 | RQ3. Visualization tools

Table 4 summarizes our findings for visualization tools. The first noticeable finding was that 18 primary sources did not provide a clear description of the tools or APIs they used for the implementation. We speculate, based on the screenshots in the articles, that the majority relied on the basic graphics API provided by the Java SDK.[49]

The second most common tool was the Eclipse Modeling Framework[50] used with the Graphical Editing Framework[51] to jointly provide a solid development framework infrastructure for creating, among other things, domain-specific languages for which visual representations could be derived.

TABLE 5 Publication fora

Acronym	Primary Sources Identifiers	Publication name
	Conferences	
SPLC	67,69,76,82,84,87,91	International Conference on Software Product Lines
VISSOFT	66,70,72,73	IEEE Working Conference on Software Visualization
ICSE	75,83	International Conference on Software Engineering
SEAA	80,96	Euromicro Conference on Software Engineering and Advanced Applications
WCRE	94,97	Working Conference on Reverse Engineering
SoftVis	86	ACM Symposium on Software Visualization
FSE	88	Foundations of Software Engineering
Modularity	89	International Conference on Modularity
IC3	90	International Conference on Contemporary Computing
ASE	95	International Conference on Automated Software Engineering
COMPSAC	93	International Conference on Computer Software and Applications
RE	68	IEEE International Requirements Engineering Conference
ISVC	100	International Symposium Advances in Visual Computing
WICSA	74	IEEE/IFIP Working Conference on Software Architecture
SLE	85	International Conference on Software Language Engineering
	Journals	
IST	79	Information and Software Technology
Proceedia	77	Procedia Technology
ISTTT	78	International Journal on Software Tools for Technology Transfer
SoSym	71	Software and System Modeling
	Workshops	
VISPLE	64,65,98,99	Workshop on Visualization in Software Product Line Engineering
REV	92	International Workshop on Requirements Engineering Visualization
PLEASE	81	International Workshop on Product Line Approaches in Software Engineering

The third place was Prefuse,[52] a visualization toolkit that has been superseded by D3.js,[53] found in fourth place with 2 primary sources. Graphviz is a software visualization tool specialized on graphs.[54] CCVisu is a visual clustering tool.[75] Google charts provides support for visualizing data in websites.[55] ConExp is a tool for formal concept analysis.[76] Moose is a software analysis platform.[70] Processing is a software sketchbook and language for visual arts.[56]

4.4 | RQ4. Publication fora

Table 5 summarizes the publication fora sorted by type of publication (eg, conference, journal, or workshop) and their frequency. Not surprisingly, the leading conferences in SPLE (SPLC) and software visualization (VISSOFT) are the most frequent publication outlets with 7 and 4 publications, respectively. These 2 conferences are followed by ICSE, SEAA, and WCRE with 2 publications each. The remaining conferences are in the general area of software engineering with the exception of ISVC and SoftVis (now merged into VISSOFT) whose focus were on visualization. From the journal publications, Huysegoms et al[77] and Pleuss and Botterweck[78] have visualization as the main focus of the article, whereas in Asadi et al[79] and Anquetil et al,[71] it is a secondary concern. Regarding workshop publications, the most frequent venue was VISPLE, a specialized workshop at the intersection of SPL and visualization that ran for 3 occasions associated to SPLC conferences.

4.5 | RQ5. Forms of empirical evaluation

Table 6 summarizes the empirical support results. By far, the most frequent form of empirical support was case studies with 28 primary sources. A distant second place was category *none* with 5 primary sources that did not perform any evaluation. The last place was taken by *experiment* for which we identified 4 primary sources.

4.6 | RQ6. Provenance of evaluation examples

Table 7 summarizes the frequency of provenance sources. Our mapping study found that academic examples were the most frequent provenance source with 16 primary sources that used them, closely followed by industrial examples with 15 primary sources. Examples from open source

TABLE 6 Empirical support

Technique	Frequency	Primary Sources Identifiers
Case Study	28	64,65,66,67,68,70,71,72,73,74,75,76,77,78,80,81,86,88,90,91,92,93,94,95,96,97,98,100
None	5	69,82,87,89,99
Experiment	4	79,83,84,85

TABLE 7 Provenance of examples

Provenance	Frequency	Primary Sources Identifiers
Academic	16	64,65,66,72,74,75,78,83,84,85,89,90,92,95,96,98
Industrial	15	65,68,70,73,76,77,78,80,86,88,91,93,94,97,100
Open source	5	67,78,80,81,82
None	4	69,71,87,99
Random	2	78,79

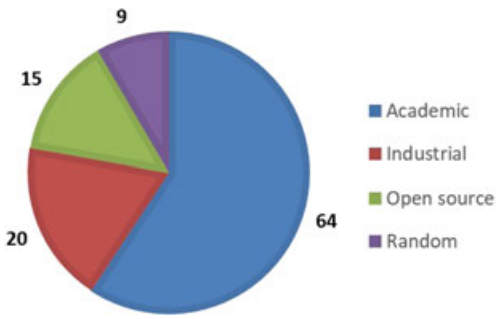

FIGURE 6 Provenance of examples

projects were used in 5 primary sources, and randomly generated examples were used in 2 primary sources. There were also 4 primary sources that did not use any examples.

Our study also found that almost all primary sources use a single type of provenance; however, there were some salient exceptions. Primary source Rabiser et al[80] used industrial and open source examples, primary source Kästner et al[65] used academic and industrial examples, and primary source Pleuss and Botterweck [78] used examples of all 4 types of provenance.

Figure 6 shows the number of examples per provenance type. Our study found that academic examples were the most prevalent with 64, followed by industrial examples with 20, open source examples with 15, and 9 randomly generated. We should also point out that 26 primary sources used a single example for their evaluation, 2 primary sources used 2 examples each, and 3 primary sources used 4 examples each. For example sizes of 3, 6, 7, and 8, there were respectively one primary source for each. Lastly, primary source Apel and Beyer[75] used 40 academic examples.

5 | ANALYSIS

In this section, we analyze the core findings revealed by our systematic mapping study.

5.1 | Preeminence of visualization of feature models

Our study revealed that feature models were the most common artifact visualized. Consequently, those activities that commonly use feature models were the most frequently found by our study, namely, DRE and DD, and ARE and AD. Because of the same reason, the most common technique used for visualization were trees and graphs (including concept lattices). Our study also highlighted that visualization for some activities has been barely used. For instance, from the extensive ongoing work on SPL testing, see Section 8, only 2 works have relied on any form of visualization technique. A similar situation is for the activity of AR, where, for instance, there is an extensive work on features and their interactions at source level and beyond,[57] for which visualization techniques have barely been explored. [66]

5.2 | Use of basic tools and techniques

An important finding of our study was that most approaches do not tap on the wealth of tooling and visualization techniques that are currently available. Instead, they either use ad hoc techniques or are based on development frameworks of ecosystems like Eclipse. Though useful and accessible entry points, they are not geared for information visualization and lack, for instance, features like more complex interactions or layout possibilities. Thus, it remains an open question, worthy of further research, identifying any underlying reasons why researchers did not employ more advanced visualization techniques or tools. Also, a common trend we found was the use of colors to distinguish the artifacts that belong to each feature, for instance, coloring the background of pieces of source code (eg, Kästner et al[65]) or models (eg, Heidenreich et al [64]).

5.3 | Empirical support and replication

Our study found that the large majority of primary sources used case studies as their form of empirical evaluation. However, such case studies were commonly in the form of feature models that are available within the SPLE community, meaning primarily academic case studies that have been extensively used for multiple purposes yet they do not constitute a formal benchmark agreed upon for comparison purposes. In other words, the context surrounding the case studies is diluted, if not lost, in the primary sources that focus on visualization.

Among the 15 works that used industrial examples for their evaluation, 3 of them used more than one example; namely, primary sources Rabiser et al,[80] Urli et al,[70] and Wnuk et al[68] used respectively 3, 2, and 3 examples. It should be remarked the fact that the number of works that use industrial examples is shortly behind the number of works that use academic examples is a good indication of both the importance of visualization techniques in industrial settings and the attention that the research community is putting on the issue. Clearly, the more compelling the empirical evidence in industrial settings, the easier the adoption of visualization techniques for SPLE would be. There were 5 works that used open source examples. Primary source Kanda et al[67] used 6, Duszynski and Becker[81] used 4, Pleuss and Botterweck[78] used 3, while Rabiser et al[80] and Santos et al[82] used 1 each. Open source examples stem mostly from systems domains, eg, Linux kernel,[58] and represent just a handful number.

Our mapping study found 4 primary sources that use experiments for their evaluation. Stengel et al devised an experiment involving 7 students using academic examples of SPLE for assessing the zooming capability of a development tool.[83] Murashkin et al performed an experiment with 3 individuals using an academic example for evaluating a tool for multidimensional feature configuration.[84] Asadi et al conducted an experiment on 3 independent variables of feature model characteristics and their impact on running time on their proposed approach for feature configuration.[79] They use 8 randomly generated examples and did not involve human subjects. Jaksic et al performed an experiment with 16 persons divided in 2 groups for the evaluation of the usability of a tool for designing feature models using an academic example.[85] These works highlight the incipient use of experiments and their limited scope at the intersection of SPLE and information visualization. Evidently, more experiments with larger scopes are duly needed. We must finally highlight recent work by Asadi et al who performed a thorough empirical evaluation of several approaches for feature configuration. [59]

6 | RESEARCH AVENUES

Our mapping study revealed several avenues worthy of further research, some of which are part of our own future work. In this section, we succinctly describe them.

Scalable use of colors for larger and more complex case studies. The primary sources that rely on colors to distinguish features in their visualizations use small case studies with only a handful of features. In practice, SPLs are usually at least an order of magnitude larger in terms of number of features. In addition, the assignment of colors is in most cases fixed and arbitrary that can lead to usability problems. We argue that exploring other coloring schemes like using customizable color palettes that consider zoom-in and zoom-out interaction capabilities (ie, similar to map visualization) could help address scalability issues for more complex case studies.

Multiview visualizations and navigation. Software product lines, like most modern software systems, use a vast number of different artifact types beyond source code. Typical examples are model diagrams, test cases, documentation, build files, etc. All these artifacts must be orchestrated and kept consistent. However, our study found that most of the visualizations focus on only one artifact, the most common being feature models. Except for a few cases, navigating from one artifact to another is not supported, which enormously hinders the understanding of the relation among artifacts that is important in the development of SPLs. For example, understanding how features and their relations are actually realized at source code can help, for instance, during testing and maintenance tasks. Existing feature to code traceability tools could be used[57]; for an early example, see Illescas et al.[66] In addition, another avenue for research is the use of multivariate data visualization techniques (eg, 2 books[23,46]), which could be helpful to understand relations among artifacts, and specially for handling complex relations among artifacts, for example, by reducing their dimensionality. It should also be mentioned that all the visualizations use the same modalities of interaction, that is, input from keyboard and mouse. It would be interesting to explore other modalities, for example, touch given the growing availability of touch screens and displays.

Integration of visualization in SPL methods and their tool chain. Most of the approaches found by our study are stand alone tools for visualizing artifacts used by SPLs. However, very few were designed and conceived to be integrated into methods for SPL development or their supporting tools. We argue this is a crucial step for adoption of visualization technologies by the SPL community at large. This implies making these technologies

accessible for integration into common IDE environments such as Eclipse or IntelliJ, such that developers and researchers could seamlessly add visualization into their development practices and workflows.

Improving the empirical evidence for visualization in SPLs. First and foremost, our study identified a lack of empirical evidence to support the visualization techniques proposed in the primary sources. There are several actions that can be taken to address this lack. First, to perform a comparative study of tools that visualize feature models. For such study, we could use feature models from the Software Produt Line Online Tools (SPLOT). Located at the following URL: [¶] repository that contains almost 900 examples, mostly academic. The important aspects here will be the scalability, interaction capabilities, and usability of the visualizations. Second, to expand the application of visualization techniques to SPLE activities that are currently understudied. In particular for domain testing, where there is an extensive ongoing research in the SPL community that has proposed different algorithms and measures of test coverage (eg, combinatorial interaction testing[19]) that are amenable to visualization and could help software engineers in making trade-off decisions. Third, to explore open source repositories such as Git to obtain further examples of SPLs from other domains whose artifacts could be effectively visualized. Fourth, to instrument tools that allow gathering information on the effort of using visualization of SPLs, for example, by analyzing eye gaze and response time. [60]

7 | THREATS TO VALIDITY

We encountered validity threats common to any other SMS. The first threat to validity is the selection of the search terms. We addressed this threat with a carefully chosen selection of terms based on previous studies on SPLs and information visualization. The SPL search terms were collected and aggregated from 12 SMSs and literature reviews. For the information visualization terms, we used 7 survey studies. We argue that this choice of selection process prevents any bias between the 2 fields of our study.

A second threat to validity comes from how the search for primary sources was conducted. To address this threat, we used 5 standard bibliography search engines. We devised the search queries for each engine, making sure we cover all combinations of both fields, and systematically collected the hits retrieved for further analysis. All the process and information was cross-checked by the two teams who did the analysis, which gave us the confidence that we did not omit any relevant primary source.

A third threat to validity is the selection of criteria for inclusion and exclusion. In our case, the sole criterion was an application of an information visualization technique in an SPLE activity. We excluded sources that were not peer-reviewed or those without any implementation provided (eg, position or vision papers). We argue that these filters ensure that the selected primary sources are a clear reflection of the state of the art and practice at the intersection of the 2 fields of our mapping study.

A fourth threat to validity is our classification scheme. For the field of SPLs, we addressed this threat by selecting classification terms from the activities of a well known framework within the SPL community. To these activities, we added a new classification term for maintenance and evolution that we defined according to standard terminology in SPL and software engineering. We use this alternative approach for classification because it eliminates errors in understanding and classification of primary sources.

A fifth threat to validity concerns the way the data was extracted for creating the mapping study. We followed an approach that is a well-accepted standard procedure to address this thread. As explained before, we took the following steps to tackle this thread. First, we created a guideline to document the classification terms and set up a spreadsheet to collect the classification information. Second, we divided by institution the authors in 2 groups for the classification task. Third, we held a virtual meeting to discuss and pilot the classification terms with a small sample of the identified primary sources. At this meeting, the classification scheme was calibrated and homogenized. Fourth, the 2 teams performed the classification independently. Fifth, we held a second virtual meeting during which we discussed for every primary source all the classification information until consensus was reached. In addition, throughout the entire process, we performed several manual and automated checks to verify the consistency and accuracy of the data while being collected and during the preparation of the graphs and figures.

8 | RELATED WORK

In this section, we briefly summarize the surveys and studies conducted either in SPLE or in information visualization.

8.1 | SPL surveys

There has been several recent SMSs and SLRs in SPLs. Regarding SPL adoption, study of Bastos et al identified 4 adoption strategies and 23 barriers that can hinder SPL adoption in industrial projects.[29] The study of Silva et al found that most of the applications of agile methods follow XP or Scrum and identified SPL practices that can be exploited by Agile techniques.[32] An SLR on requirements engineering and SPL was performed by Alves et al.[27] Their work indicates that the application of requirements engineering techniques for SPL was still not mature as most of the case studies used were indeed toy examples. Thus, they advocate that more empirical studies should be performed to address this serious limitation and hence improve the rigor, credibility, and validity of the proposed approaches.

[¶] http://splot-research.org/

Two mapping studies have been performed on service orientation.[35,36] Among their findings is that there are still many research avenues to purse and that most of the work is on performance and availability whereas other quality attributes are mostly disregarded and are not in industrial settings.

Four studies on SPL testing have been published.[19,30,31,37] These studies provide a taxonomy and classify over more than a hundred sources along several dimensions. Among their findings is the preeminence of combinatorial approaches for selecting representative products to test and that there is still a great lack of empirical industrial applications.

In the area of SPL evolution, Laguna et al made an assessment of the maturity level of techniques to migrate individual systems or groups of software variants into SPLs.[34] Recent work by Assunção et al has further extended and updated the work of Laguna et al to provide a deeper analysis of techniques, tools, and case studies for migration to SPLs.[61] Also, recently, Montalvillo et al performed a mapping study whose focus is on evolution of SPL as consequence of change in requirements. [62]

There have been 2 studies on variability management in general.[15,16] Among their collective findings are that a large majority of the reported approaches have not been sufficiently evaluated using scientifically rigorous methods (eg, following Wohlin et al[11]) and that software quality attributes have not received much attention.

Rabiser et al performed a systematic review of requirements for supporting product configuration,[28] whereas Holl et al conducted a systematic review of the capabilities to support multiproduct lines. [33]

Our previous study on search-based software engineering analyzed what and how search-based techniques, including metaheuristic search-based optimization, have been used for SPLs problems.[20] The study found the preeminence of metaheuristic approaches, eg, genetic algorithms, applied to SPL testing.

8.2 | Visualization surveys

Schots et al performed an extensive review of visualization for software reuse.[43] They found 4 of the primary sources that our study identified even though SPLs are a form of systematic software reuse.

Seriai et al performed an SMS on the validation of visualization tools.[39] Their main finding was that despite the increasing research and application of visualization techniques, their evaluation lacks rigor. Novais et al performed an SMS in software evolution visualization.[38] Similarly to Seriai et al, they found a lack of empirical studies that, for instance, validate the usefulness of the proposed techniques. Prado et al performed an SMS of visualization tools and techniques for software comprehension.[40] Their study corroborates the lack of robust empirical evaluation and found that most approaches use bidimensional visualizations and do not address user interactions.

Abuzaid and Scott performed an SLR on visualization of software quality metrics, which describe the different techniques used to facilitate comprehension of common metrics like McCabe's complexiy or lines of code.[44] Paredes et al performed an SMS of the use of information visualization for software development following Agile approaches.[42] They found visualization used for designing, developing, communication, and keeping track of progress.

9 | CONCLUSIONS

In this paper, we presented an SMS on visualization techniques for SPLE. This study identified 37 primary sources and corroborated the preeminent use of visualization techniques for SPLE activities that involve feature models, a de facto standard for describing the combinations of features in the products of a SPL. It also confirmed the fact that most primary sources rely on basic visualization techniques and tools, eg, ad hoc or based in Eclipse tools, that barely exploit the wealth of techniques available in the software and information visualization communities.

We also analyzed the types of empirical evaluation used and the provenance of the evaluation examples. We found that most primary sources use case studies of academic provenance. There is, however, a good number of industrial and open-source examples used. We also identified a few instances that use experiments to evaluate their proposed approaches. We sketch several avenues for further research. As part of our future work, we want to take a closer look on the interaction capabilities on the identified approaches, perform a comparative study of the tools that visualize feature models with special focus on scalability, and apply visualization techniques for SPL testing. We hope our work can entice researchers in SPLE and visualization communities to pursue further work in the subject.

ACKNOWLEDGMENTS

This research was funded by the Austrian Science Fund (FWF) projects P25289-N15 and P25513-N15 and NSERC grant RGPIN-2017-05421.

ORCID

Roberto Erick Lopez-Herrejon http://orcid.org/0000-0002-7067-8269

REFERENCES

1. Batory DS, Sarvela JN, Rauschmayer A. Scaling step-wise refinement. *IEEE Trans Software Eng*. 2004;30(6):355-371.
2. Pohl K, Bockle G, van der Linden FJ. *Software Product Line Engineering: Foundations, Principles and Techniques*. Heilderberg, Germany: Springer; 2005:1-467.
3. Svahnberg M, van Gurp J, Bosch J. A taxonomy of variability realization techniques. *Softw Pract Exper*. 2005;35(8):705-754.
4. Kang K, Cohen S, Hess J, Novak W, Peterson A. Feature-oriented domain analysis (FODA) feasibility study. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University; 1990.
5. Benavides D, Segura S, Cortés AR. Automated analysis of feature models 20 years later: a literature review. *Inf Syst*. 2010;35(6):615-636.
6. van d. Linden FJ, Schmid K, Rommes E. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Heilderberg, Germany: Springer; 2007:1-333.
7. Käkölä Timo, Dueñas JC, eds. *Software Product Lines—Research Issues in Engineering and Management*. Heidelberg, Germany: Springer; 2006:1-635.
8. Kitchenham B, Charters S. Guidelines for performing systematic literature reviews in software engineering. version 2.3. EBSE Technical Report EBSE-2007-01, Software Engineering Group, School of Computer Science and Mathematics, Keele University, UK and Department of Computer Science, University of Durham, UK; 2007. http://cdn.elsevier.com/promis_misc/525444systematicreviewsguide.pdf.
9. Petersen K, Feldt R, Mujtaba S, Mattsson M. Systematic mapping studies in software engineering. In: EASE British Computer Society; 2008; Bari, Italy. 68-77. http://dl.acm.org/citation.cfm?id=2227115.2227123.
10. Budgen D, Turner M, Brereton P, Kitchenham B. Using mapping studies in software engineering. In: Proceedings of PPIG 2008; Lancaster University; 2008; Lancaster, UK. 195-204.
11. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B. *Experimentation in Software Engineering*. Heilderberg, Germany: Springer; 2012:1-236.
12. Kitchenham BA, Budgen D, Brereton OP. Using mapping studies as the basis for further research—a participant-observer case study. *Inf Software Technol*. 2011;53(6):638-651.
13. Lopez-Herrejon RE, Illescas S, Egyed A. Visualization for software product lines: a systematic mapping study. In: Sharif B, Parnin C, Fabry J, eds. *2016 IEEE Working Conference on Software Visualization, VISSOFT 2016; October 3-4, 2016*. Raleigh, NC, USA: IEEE Computer Society; 2016:26-35.
14. Heradio R, Perez-Morago H, Fernández-Amorós D, Cabrerizo FJ, Herrera-Viedma E. A bibliometric analysis of 20 years of research on software product lines. *Inf Software Technol*. 2016;72:1-15.
15. Galster M, Weyns D, Tofan D, Michalik B, Avgeriou P. Variability in software systems—a systematic literature review. *IEEE Trans Software Eng*. 2014;40(3):282-306.
16. Chen L, Babar MA. A systematic review of evaluation of variability management approaches in software product lines. *Inf Software Tech*. 2011;53(4):344-362.
17. Czarnecki K, Grünbacher P, Rabiser R, Schmid K, Wasowski A. Cool features and tough decisions: a comparison of variability modeling approaches. In: Eisenecker UW, Apel S, Gnesi S, eds. *VaMoS*. Leipzig, Germany: ACM; 2012:173-182.
18. Metzger A, Pohl K. Software product line engineering and variability management: achievements and challenges. In: Herbsleb JD, Dwyer MB, eds. *FOSE*. Hyderabad, India: ACM; 2014:70-84.
19. Lopez-Herrejon RE, Fischer S, Ramler R, Egyed A. A first systematic mapping study on combinatorial interaction testing for software product lines. In: Eighth IEEE International Conference on Software Testing, Verification and Validation, ICST 2015 Workshops, April 13-17, 2015 IEEE Computer Society; 2015; GRAZ, Austria. 1-10.
20. Lopez-Herrejon RE, Linsbauer L, Egyed A. A systematic mapping study of search-based software engineering for software product lines. *J Inf Software Technol*. May 2015;61:33-51. https://doi.org/10.1016/j.infsof.2015.01.008.
21. Williams JG. Visualization. *Annu Rev Inf Sci Technol (ARIST)*. 1995;30:161-207. https://www.learntechlib.org/p/80656.
22. Chi E. *A Framework for Visualizing Information*, Human–Computer Interaction Series. Heilderberg, Germany: Springer; 2002:1-147.
23. Telea A. *Data Visualization—Principles and Practice*. 2nd ed. Boca Raton, Florida, US: A K Peters; 2015:1-605.
24. Diehl S. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Heilderberg, Germany: Springer; 2007:1-187.
25. Kitchenham B, Dybaa T, Jorgensen M. Evidence-based software engineering. In: ICSE IEEE CS Press; 2004; Edinburgh, Scotland, UK. 273-281.
26. Kitchenham BA, Budgen D, Brereton P. *Evidence-Based Software Engineering and Systematic Reviews*. Boca Raton, Florida, USA: CRC Press; 2016: 1-399.
27. Alves V, Niu N, Alves CF, Valença G. Requirements engineering for software product lines: a systematic literature review. *Inf Software Technol*. 2010;52(8):806-820.
28. Rabiser R, Grünbacher P, Dhungana D. Requirements for product derivation support: results from a systematic literature review and an expert survey. *Inf Software Tech*. 2010;52(3):324-346.
29. Ferreira Bastos J, Anselmo da Mota Silveira Neto P, Santana de Almeida E, Romero de Lemos Meira S. Adopting software product lines: a systematic mapping study. In: 15th Annual Conference on Evaluation Assessment in Software Engineering (EASE 2011); 2011; Durham, UK. 11-20.
30. da Mota Silveira Neto PA, do Carmo Machado I, McGregor JD, de Almeida ES, de Lemos Meira SR. A systematic mapping study of software product lines testing. *Inf Software Technol*. 2011;53(5):407-423.
31. Engström E, Runeson P. Software product line testing—a systematic mapping study. *Inf Software Tech*. 2011;53(1):2-13.
32. da Silva IF, da Mota Silveira Neto PA, O'Leary P, de Almeida ES, de Lemos Meira SR. Agile software product lines: a systematic mapping study. *Softw Pract Exper*. 2011;41(8):899-920.
33. Holl G, Grünbacher P, Rabiser R. A systematic review and an expert survey on capabilities supporting multi product lines. *Inf Software Tech*. 2012;54(8):828-852.
34. Laguna MA, Crespo Y. A systematic mapping study on software product line evolution: from legacy system reengineering to product line refactoring. *Sci Comput Program*. 2013;78(8):1010-1034.
35. Mohabbati B, Asadi M, Gasevic D, Hatala M, Müller HA. Combining service-orientation and software product line engineering: a systematic mapping study. *Inf Software Technol*. 2013;55(11):1845-1859.

36. Mahdavi-Hezavehi S, Galster M, Avgeriou P. Variability in quality attributes of service-based software systems: A systematic literature review. *Inf Software Technol*. 2013;55(2):320-343.
37. do Carmo Machado I, McGregor JD, Cavalcanti YC, de Almeida ES. On strategies for testing software product lines: a systematic literature review. *Inf Software Technol*. 2014;56(10):1183-1199. http://www.sciencedirect.com/science/article/pii/S0950584914000834.
38. Novais RL, Torres A, Mendes TS, Mendonça MG, Zazworka N. Software evolution visualization: a systematic mapping study. *Inf Software Technol*. 2013;55(11):1860-1883. https://doi.org/10.1016/j.infsof.2013.05.008.
39. Seriai A, Benomar O, Cerat B, Sahraoui HA. Validation of software visualization tools: a systematic mapping study. In: Sahraoui et al,[63] ed. *Second IEEE Working Conference on Software Visualization, VISSOFT 2014, September 29-30, 2014*. Victoria, BC, Canada: IEEE Computer Society; 2014:60-69, https://doi.org/10.1109/VISSOFT.2014.19.
40. Prado MP, Vincenzi AMR, de M. N. Soares FAA, Cesar F, de Paula GP, do Nascimento HAD, Silva JC, de Oliveira JL, Lima LC, Fernandes T. Characterization of techniques and tools of visualization applied to software comprehension: a systematic mapping. In: International Conference on Software Engineering Advances (ICSEA); 2013; Venice, Italy. 297-303.
41. Schots M. On the use of visualization for supporting software reuse. In: Jalote P, Briand LC, van der Hoek A, eds. *36th International Conference on Software Engineering, ICSE '14, Companion Proceedings, May 31 - June 07, 2014*. Hyderabad, India: ACM; 2014:694-697, https://doi.org/10.1145/2591062.2591095.
42. Paredes J, Anslow C, Maurer F. Information visualization for agile software development. In: Sahraoui HA, Zaidman A, Sharif B, eds. *Second IEEE Working Conference on Software Visualization, VISSOFT 2014, September 29-30, 2014*. Victoria, BC, Canada: IEEE Computer Society; 2014:157-166, https://doi.org/10.1109/VISSOFT.2014.32.
43. Schots M, Vasconcelos R, Werner C. A quasi-systematic review on software visualization approaches for software reuse. Technical Report, Federal University of Rio de Janeiro; 2014.
44. Abuzaid D, Titang S. The visualization of software quality metrics. A systematic literature review. Bachelor thesis. University of Gothenburg. Chalmers University of Technology; 2014.
45. Wohlin C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Shepperd MJ, Hall T, Myrtveit I, eds. *EASE*. London, England, United Kingdom: ACM; 2014:38.
46. Ward MO, Grinstein GG, Keim DA. *Interactive Data Visualization—Foundations, Techniques, and Applications*. Natick, MA, USA: A K Peters; 2010:1-496. http://www.akpeters.com/product.asp?ProdCode=4735.
47. Runeson P, Höst M, Rainer A, Regnell B. *Case Study Research in Software Engineering—Guidelines and Examples*. Hoboken, NJ, USA: Wiley; 2012:1-237. http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118104358.html.
48. Berger T, She S, Lotufo R, Wasowski A, Czarnecki K. Variability modeling in the real: a perspective from the operating systems domain. In: Pecheur C, Andrews J, Nitto ED, eds. *ASE*. Antwerp, Belgium: ACM; 2010:73-82.
49. Java SDK. http://www.oracle.com/technetwork/java/index.html. Accessed on September 9, 2017.
50. Eclipse Modeling Framework (EMF). https://eclipse.org/modeling/emf/. Accessed on September 9, 2017.
51. Graphical Editing Framework (GEF). https://eclipse.org/gef/. Accessed on September 9, 2017.
52. Prefuse. http://prefuse.org/. Accessed on September 9, 2017.
53. D3.js. https://d3js.org/. Accessed on September 9, 2017.
54. Graphviz. http://www.graphviz.org/. Accessed on September 9, 2017.
55. Google charts. https://developers.google.com/chart/. Accessed on September 9, 2017.
56. Processing. https://processing.org/. Accessed on September 9, 2017.
57. Fischer S, Linsbauer L, Lopez-Herrejon RE, Egyed A. A source level empirical study of features and their interactions in variable software. In: 16th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2016, October 2-3, 2016 IEEE Computer Society; 2016; Raleigh, NC, USA. 197-206. https://doi.org/10.1109/SCAM.2016.16.
58. Berger T, She S, Lotufo R, Wasowski A, Czarnecki K. A study of variability models and languages in the systems software domain. *IEEE Trans Software Eng*. 2013;39(12):1611-1640. https://doi.org/10.1109/TSE.2013.34.
59. Asadi M, Soltani S, Gasevic D, Hatala M. The effects of visualization and interaction techniques on feature model configuration. *Empirical Software Eng*. 2016;21(4):1706-1743. https://doi.org/10.1007/s10664-014-9353-5.
60. Sharif B, Maletic JI. iTrace: overcoming the limitations of short code examples in eye tracking experiments. In: 2016 IEEE International Conference on Software Maintenance and Evolution, ICSME 2016, October 2-7, 2016 IEEE Computer Society; 2016; Raleigh, NC, USA. 647.
61. Assunção WKG, Lopez-Herrejon RE, Linsbauer L, Vergilio SR, Egyed A. Reengineering legacy applications into software product lines: a systematic mapping. *Empirical Software Eng*. accepted for publication. 2017;22(6):2972-3016.
62. Montalvillo L, Díaz O. Requirement-driven evolution in software product lines: a systematic mapping study. *J Syst Software*. 2016;122:110-143. https://doi.org/10.1016/j.jss.2016.08.053.
63. Sahraoui HA, Zaidman Andy, Sharif Bonita, eds. *Second IEEE Working Conference on Software Visualization, VISSOFT 2014, Victoria, BC, CANADA, September 29-30, 2014*: IEEE Computer Society; 2014. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6970369.

APPENDIX A: PRIMARY SOURCES LIST

64. Heidenreich Florian, Savga Ilie, Wende Christian. On controlled visualisations in software product line engineering. In: Software Product Lines, 12th International Conference, SPLC 2008, September 8-12, 2008, Proceedings. Second Volume (Workshops); 2008; Limerick, Ireland. 335-341.
65. Kästner Christian, Trujillo Salvador, Apel Sven. Visualizing software product line variabilities in source code. In: Software Product Lines, 12th International Conference, SPLC 2008, September 8-12, 2008, Proceedings. Second Volume (Workshops); 2008; Limerick, Ireland. 303-312.
66. Illescas Sheny, Lopez-Herrejon RE, Egyed Alexander. Towards visualization of feature interactions in software product lines. In: 2016 IEEE Working Conference on Software Visualization, VISSOFT 2016, October 3-4, 2016; 2016; Raleigh, NC, USA. 46-50.

67. Kanda Tetsuya, Ishio Takashi, Inoue Katsuro. Extraction of product evolution tree from source code of product variants. *Proceedings of the 17th International Software Product Line Conference*, SPLC '13. New York, NY, USA: ACM; 2013:141-150. http://doi.acm.org/10.1145/2491627.2491637.
68. Wnuk K., Regnell B., Karlsson L. What happened to our features? Visualization and understanding of scope change dynamics in a large-scale industrial setting. In: 17th IEEE International Requirements Engineering Conference RE '09; 2009; Atlanta, Georgia, USA. 89-98.
69. de Oliveira THB, Becker Martin, Nakagawa EY. Supporting the analysis of bug prevalence in software product lines with product genealogy. In: *Proceedings of the 16th International Software Product Line Conference - Volume 1*, SPLC '12. New York, NY, USA: ACM; 2012:181-185.
70. Urli S., Bergel A., Blay-Fornarino M., Collet P., Mosser S. A visual support for decomposing complex feature models. In: 2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT); 2015; Bremen, Germany. 76-85.
71. Anquetil Nicolas, Kulesza U, Mitschke Ralf, Moreira Ana, Royer J-C, Rummler Andreas, Sousa A. A model-driven traceability framework for software product lines. *Software Syst Model*. 2010;9(4):427-451. https://doi.org/10.1007/s10270-009-0120-9.
72. Lopez-Herrejon RE, Egyed A. Towards interactive visualization support for pairwise testing software product lines. In: 2013 First IEEE Working Conference on Software Visualization (VISSOFT); 2013; Eindhoven, Netherlands. 1-4.
73. Martinez J., Ziadi T., Mazo R., Bissyandé TF, Klein J., Traon YL. Feature relations graphs: a visualisation paradigm for feature constraints in software product lines. In: 2014 Second IEEE Working Conference on Software Visualization (VISSOFT); 2014; Victoria, BC, Canada. 50-59.
74. Garba Muhammad, Noureddine Adel, Bashroush Rabih. MUSA: a scalable multi-touch and multi-perspective variability management tool. In: 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016, April 5-8, 2016; 2016; Venice, Italy. 299-302.
75. Apel Sven, Beyer Dirk. Feature cohesion in software product lines: an exploratory study. *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11. New York, NY, USA: ACM; 2011:421-430, https://doi.org/10.1145/1985793.1985851.
76. Loesch F., Ploedereder E. Optimization of variability in software product lines. In: 2007. SPLC 2007. 11th International Software Product Line Conference; 2007; Kyoto, Japan. 151-162.
77. Huysegoms Tom, Snoeck Monique, Dedene Guido, Goderis Antoon, Stumpe Frank. Visualizing variability management in requirements engineering through formal concept analysis. *Procedia Technol*. 2013;9:189-199. http://www.sciencedirect.com/science/article/pii/S2212017313001758, {CENTERIS} 2013 - Conference on {ENTERprise} Information Systems / ProjMAN 2013 - International Conference on Project MANagement/ {HCIST} 2013 - International Conference on Health and Social Care Information Systems and Technologies.
78. Pleuss Andreas, Botterweck Goetz. Visualization of variability and configuration options. *STTT*. 2012;14(5):497-510. https://doi.org/10.1007/s10009-012-0252-z.
79. Asadi Mohsen, Soltani Samaneh, Gasevic Dragan, Hatala Marek, Bagheri Ebrahim. Toward automated feature model configuration with optimizing non-functional requirements. *Inf Software Technol*. 2014;56(9):1144-1165. http://www.sciencedirect.com/science/article/pii/S0950584914000640, Special Sections from Asia-Pacific Software Engineering Conference (APSEC), 2012 and Software Product Line conference (SPLC), 2012.
80. Rabiser R., Dhungana D., Heider W., Grünbacher P. Flexibility and end-user support in model-based product line tools. In: 2009. SEAA '09. 35th Euromicro Conference on Software Engineering and Advanced Applications; 2009; Patras, Greece. 508-511.
81. Duszynski S., Becker M. Recovering variability information from the source code of similar software products. In: 2012 3rd International Workshop on Product Line Approaches in Software Engineering (PLEASE); 2012; Zurich, Switzerland. 37-40.
82. Santos AR, do Carmo Machado Ivan, de Almeida S. RiPLE-HC: visual support for features scattering and interactions. In: Proceedings of the 20th International Systems and Software Product Line Conference, SPLC 2016, September 16-23, 2016; 2016; Beijing, China. 320-323.
83. Stengel Michael, Frisch Mathias, Apel Sven, Feigenspan Janet, Kästner Christian, Dachselt Raimund. View infinity: a zoomable interface for feature-oriented software development. In: *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11. New York, NY, USA: ACM; 2011:1031-1033. https://doi.org/10.1145/1985793.1985987.
84. Murashkin Alexandr, Antkiewicz M, Rayside Derek, Czarnecki Krzysztof. Visualization and exploration of optimal variants in product line engineering. In: *Proceedings of the 17th International Software Product Line Conference*, SPLC '13. New York, NY, USA: ACM; 2013:111-115. https://doi.org/10.1145/2491627.2491647.
85. Jaksic Aleksandar, France RobertB., Collet Philippe, Ghosh Sudipto. Evaluating the usability of a visual feature modeling notation. In: Software Language Engineering - 7th International Conference, SLE 2014, September 15-16, 2014. Proceedings; 2014; Västerås, Sweden. 122-140.
86. Nestor D, Thiel S, Botterweck G, Cawley C, Healy P. Applying visualisation techniques in software product lines. In: Proceedings of the 4th ACM Symposium on Software Visualization, SoftVis '08; 2008; ACM: New York, NY, USA:175-184.
87. Ferrari A, Spagnolo GO, Gnesi S, Dell'Orletta F. CMT and FDE: tools to bridge the gap between natural language documents and feature diagrams. In: Proceedings of the 19th International Conference on Software Product Line, SPLC '15; 2015; ACM: New York, NY, USA:402-410.
88. Ben Nasr S, B'ecan G, Acher M, Ferreira Filho JB, Baudry B, Sannier N, Davril JM. Matrixminer: a red pill to architect informal product descriptions in the matrix. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015; 2015; ACM: New York, NY, USA:982-985.
89. Thimmegowda N, Kienzle J. Visualization algorithms for feature models in concern-driven software development. In: Companion Proceedings of the 14th International Conference on Modularity, MODULARITY Companion 2015; 2015; ACM: New York, NY, USA:39-42.
90. Kaur M, Kumar P. Spotting the phenomenon of bad smells in MobileMedia product line architecture. In: 2014 Seventh International Conference on Contemporary Computing (IC3); 2014; Noida, India. 357-363.
91. Botterweck G, Thiel S, Nestor D, b Abid S, Cawley C. Visual tool support for configuring and understanding software product lines. In: Software Product Line Conference, 2008. SPLC '08. 12th International, 2008; Limerick, Ireland. 77-86.
92. Sellier D, Mannion M. Visualising product line requirement selection decision inter-dependencies. In: 2007. REV 2007. Second International Workshop on Requirements Engineering Visualization; 2007; New Delhi, India. 7-7.
93. Botterweck G, Thiel S, Cawley C, Nestor D, Preuner A. Visual configuration in automotive software product lines. In: 2008. COMPSAC '08. 32nd Annual IEEE International Computer Software and Applications; 2008; Turku, Finland. 1070-1075.
94. Duszynski S, Knodel J, Becker M. Analyzing the source code of multiple software variants for reuse potential. In: 2011 18th Working Conference on Reverse Engineering (WCRE); 2011; Koblenz, Germany. 303-307.

95. Pietsch C, Kehrer T, Kelter U, Reuling D, Ohrndorf M. SiPL—a delta-based modeling framework for software product line engineering. In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE); 2015; Lincoln, Nebraska, USA. 852-857.
96. de Medeiros TFL, de Almeida ES, de Lemos Meira SR. Codescoping: a source code based tool to software product lines scoping. In: 2012 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA); 2012; Cesme, Izmir, Turkey. 101-104.
97. Duszynski S, Knodel J, Naab M, Hein D, Schitter C. Variant comparison—a technique for visualizing software variants. In: 2008. WCRE '08. 15th Working Conference onReverse Engineering; 2008; Antwerp, Belgium. 229-233.
98. Trinidad P, Ruiz-Cortés A, Benavides D, Segura S. Three-dimensional feature diagrams visualization. In: 2nd SPLC Workshop on Visualisation in Software Product Line Engineering (ViSPLE), Irish Software Engineering Research Centre (Lero), Irish Software Engineering Research Centre (Lero): Limerick; 2008; Ireland:295-302. http://www.lero.ie/visple2008.
99. Heuer A, Lauenroth K, Müller M, Scheele J. Towards effective visual modeling of complex software product lines. In: Software Product Lines - 14th International Conference, SPLC 2010, September 13-17, 2010. Workshop Proceedings (Volume 2 : Workshops, Industrial Track, Doctoral Symposium, Demonstrations and Tools); 2010; Jeju Island, South Korea. 229-238. http://splc2010.postech.ac.kr/SPLC2010_second_volume.pdf.
100. Cawley C, Botterweck G, Healy P, Abid SB, Thiel S. A 3d visualisation to enhance cognition in software product line engineering. In: Bebis G, Boyle R, Parvin B, Koracin D, Kuno Y, Wang J, Pajarola R, Lindstrom P, Hinkenjann A, Encarnação ML, eds. *Advances in Visual Computing: 5th International Symposium, ISVC 2009, Las Vegas, NV, USA, November 30-December 2, 2009. Proceedings, Part II*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009:857-868. https://doi.org/10.1007/978-3-642-10520-382

How to cite this article: Lopez-Herrejon RE, Illescas S, Egyed A. A systematic mapping study of information visualization for software product line engineering. *J Softw Evol Proc*. 2017;e1912. https://doi.org/10.1002/smr.1912