2016 IEEE Working Conference on Software Visualization

Visualization for Software Product Lines: A Systematic Mapping Study

Roberto E. Lopez-Herrejon
Dept. Software Engineering and IT
École de technologie supérieure, Canada
Email: roberto.lopez@etsmtl.ca

Sheny Illescas
Software System Engineering
Johannes Kepler University Linz
Austria
Email: k1257276@students.jku.at

Alexander Egyed
Software System Engineering
Johannes Kepler University Linz
Austria
Email: alexander.egyed@jku.at

***Abstract*—Software Product Lines (SPLs) are families of related systems whose members are distinguished by the set of features they provide. Over two 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. Typical SPLs involve large number of features which 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 at different stages of the life cycle of SPLs. In this paper we present a systematic mapping study on this subject. Our research questions aim to gather information regarding the techniques that have been applied, at what stages, how they were implemented, and the publication fora employed. Our goal is to identify common trends, gaps, and opportunities for further research and application.**

I. INTRODUCTION

Software Product Lines (SPLs) are families of related systems whose members are distinguished by the set of features they provide [3], [30]. *Variability* is the capacity of software artifacts to vary and its effective management and realization lie at the core of successful SPL development [37]. *Feature models* are tree-like structures that establish the relations between features and have become the de facto standard for modelling variability [17], [4]. Over the last decade, extensive research and practice both in academia and industry attest to the substantial benefits of applying SPL practices [30], [39], [16]. Among the benefits are better customization, improved software reuse, and faster time to market.

Typical SPLs have a large number of features that are combined in complex feature relations yielding a large number of individual software systems that must be effectively and efficiently designed, implemented and managed. Precisely this fact is what makes SPL-related problems suitable for the application of visualization techniques. This application has been explored by several researchers and has produced a number of publications on the subject. This is precisely what prompted us to perform a *systematic mapping study* to provide an overview of the research at the intersection of these two fields [18], [29], [5].

In contrast with a *systematic literature review* whose goal is primarily to identify best practice [18], [5], [42], [20], our general goal was to identify the quantity and the type of research and results available, and thus highlight possible open research problems and opportunities, for both visualization and SPL communities. More concretely we wanted to identify at what stages of the SPL development life cycle have visualization techniques been used, which ones, and what tools they use for their implementation. And finally, which are the fora where the research work was published.

Our study found 32 publications that revealed an ongoing interest in applying visualization techniques to SPL problems. Salient among the findings is the pre-eminent use of visualization techniques for capturing and designing SPLs, as well for providing support during their configuration. We hope that this mapping study not only serves to highlight the main research topics at the intersection of visualization and SPLs but that it also serves to encourage researchers to pursue work at the intersection of both areas.

The paper is structured as follows. Section II provides the basic background on SPLS. Section III presents the process we followed for our systematic mapping study. It details the research questions addressed, how the search was performed, the classification scheme used, and how the data was extracted and analysed. Section IV presents the results we obtained for each research question. Section V presents our analysis of the results found along with open questions and avenues worth of further investigation. Section VI concisely describes the existing review studies and surveys of SPLs and visualization. Section VII summarizes the conclusions of our study and future work.

II. SOFTWARE PRODUCT LINES OVERVIEW

As mentioned before, Software Product Lines (SPLs) are families of related systems whose members are distinguished by the set of features they provide [3], [30]. There is an extensive body of research that attests to the benefits of SPL practices and that has proposed multiple approaches, methods, and techniques for SPL development (e.g. [30], [14], [13], [6]). In this section we present the basic concepts of SPLs to set up the context of the mapping study.

A. Feature Models

Recall that a core concept in SPLs is *variability* which refers to the capacity of software artifacts to vary [37]. The

978-1-5090-3850-3/16 $31.00 © 2016 IEEE
DOI 10.1109/VISSOFT.2016.11

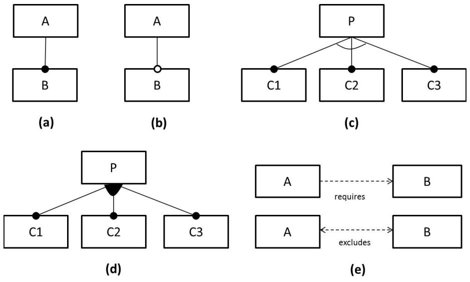


Fig. 1. Feature models graphical notation.

software products that constitute a SPL are characterized by the different combinations of features they have. These combinations are captured in *variability models* for which there are different alternatives [7]; however, feature models have become a de facto standard [17]. 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 (e.g. feature `B` in Figure 1(a)), and *optional* which may or may not be part of a program whenever its parent feature is selected (e.g. feature `B` in Figure 1(b)). 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 1(c) 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 1(d) shows that if feature P is selected, one of more of features `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)* [4]. The typical examples of this kind of relations are: *i) requires* relation whereby if a feature `A` is selected a feature `B` must also be selected, and *ii) 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 1(e).

B. Software Product Line Engineering Framework

As described before, there are many approaches for SPL development (e.g. [30], [14], [13], [6]). For our study, we selected the SPL engineering framework proposed by Pohl et al.'s [30], shown in Figure 2. This framework is well-known within SPL research community and has been used to highlight not only some of the open questions and challenges in the field of SPLs [25], but we also have used it for other systematic mapping studies [22], [23]. This framework defines two main SPL activities as follows [30]:

Definition 1: **Domain Engineering (DE)** is the process of software product line engineering in which the commonality and the variability of the product line are defined and realised.

Definition 2: **Application Engineering (AD)** is the process of software product line engineering in which the applications of the product line are built by reusing domain artefacts and exploiting the product line variability.

Each of the two main activities is divided in four sub-processes defined as [30]:

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

We regard each sub-process of DE and AE as a *life cycle stage* of SPLs and we use these terms for the classification of the visualization techniques as described in Section III-D. We made this decision because DE and AE are the two common activities in all the SPL approaches (hence applicable to any SPL approach) and because they have a clear distinction between their goals as stated in their definitions. In addition to the eight stages of this framework, we considered one more stage to cover all maintenance and evolution issues of SPLs 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.

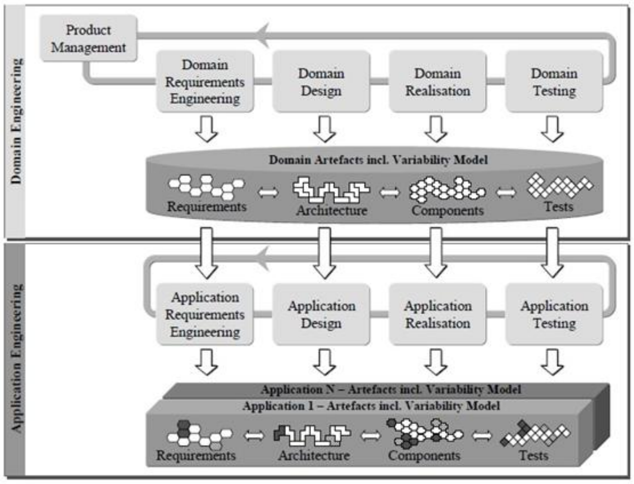


Fig. 2. Pohl et al.'s SPL Framework, Figure 2-1 from [30].

III. SYSTEMATIC MAPPING STUDY

Evidence-Based Software Engineering (EBSE) has as its driving goal *"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"* [19]. One of the most common approaches advocated by EBSE are systematic mapping studies that aim to provide an overview of the results available within an area by categorizing them along criteria such as type, forum, frequency, etc. [29].

In this paper we perform our systematic mapping study following the protocol proposed by Petersen et al. [29], 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 IV we present the results obtained, and in Section V their analysis.

A. Definition of Research Questions

Recall that the main goal underlying our work is to provide an overview of research that applies visualization techniques to tackle SPL problems. 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 SPLs. Our mapping study then focuses on the following research questions:

- ***RQ1. In what stages of the SPL life cycle 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 employed throughout the entire life cycle of SPLs [30] as explained in Section II-B.
- ***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 SPL problems and analyse 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 identify the technical support that has been exploited in SPL problems.
- ***RQ4. What are the publication fora used?***
  Rationale: Software visualization and SPL 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.

B. Conduct Search for Primary Sources.

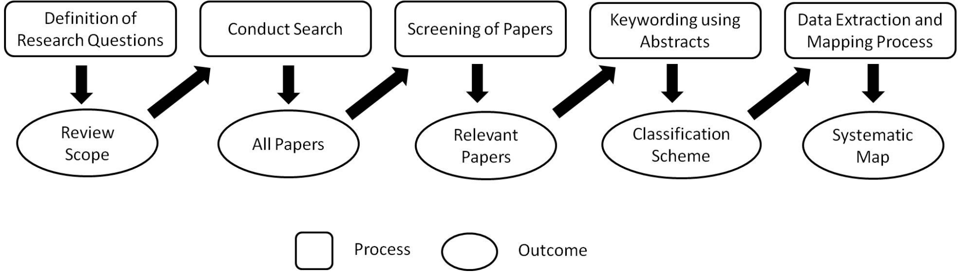
In this step of the systematic mapping the strings of terms to be used for the search are defined. Because of work focuses on the intersection of two fields, SPLs and visualization, we used to sets of terms, one for each field. Table I shows the list of all search terms we used[1]. Regarding the SPL terms, they come from our two previous mapping studies [22], [23] and are based on the terms collected from twelve systematic mapping and literature review studies in SPLs [2], [32], [12], [8], [11], [9], [15], [21], [26], [24], [10], [13]. It is important to remark that none of these mapping studies are related to software visualization as will be detailed in Section VI. For gathering the terms for visualization we selected them from seven surveys and studies in the area of visualization [27], [36], [31], [34], [28], [35], [1]. Again, none of these works relate to SPLs.

We carried out the search process in two stages. First, we used the search engines of publishing companies and orga-

---

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

Fig. 3. Systematic Mapping Study Process [29].



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

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

TABLE I
Summary of SPL and visualization search terms.

nizations, namely ScienceDirect, IEEEXplore, ACM Digital Libray, and SpringerLink. These are the common publishing outlets that contain journals, conferences, and workshops in both SPLs and software visualization. At the second stage, we performed so-called *snowballing readings* which refer to those papers which are either cited or cite the papers obtained in the first search stage [5], [41]. We manually performed the second stage 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 SPL 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[2]:

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

C. 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 criteria for inclusion in our mapping study was that a clear application of visualization techniques to SPL problems was described.

The criteria to exclude papers in our study was: *i)* papers which did not apply any visualization techniques to SPLs[3], *ii)* papers not written in English, *iii)* vision or position papers that had no implementation to back them up, *iv)* graduate or undergraduate dissertations and thesis, and *v)* non peer-reviewed documents such as technical reports.

The decision on whether or not to include a paper was most of the times straightforward, in other words, that at least one visualization term was found and a clear connection to SPLs could be easily drawn. While performing the searches we found out for a couple of approaches, that there were papers that presented fundamentally the same approach (e.g. 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 new problem domain.

D. Keywording using Abstracts — Classification Scheme

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

1) SPL life cycle stage classification: For this classification dimension we used the eight stages derived from the Pohl et al.'s framework [30], plus the maintenance and evolution (ME) stage as described in Section II-B. We deviate from the standard classification procedure whereby the classification schemes follow from the abstract keywords because our driving goal is to bring to the attention of researchers and practitioners of SPL and software visualization communities the research opportunities at the intersection of both disciplines. Hence, we decided on using a framework and terminology that is already familiar within the SPL community and readily accessible for software visualization researchers. We should remark that for this dimension a primary source can be classified in more than one category.

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 [38], [40]. We should also remark

---

[2] 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 SPL terms.

[3] We should mention that we included papers that apply visualization techniques even though the application was not their main focus or contribution.

that for this dimension a primary source can also be classified in more than one category.

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.

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.

E. Data Extraction and Mapping Study

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

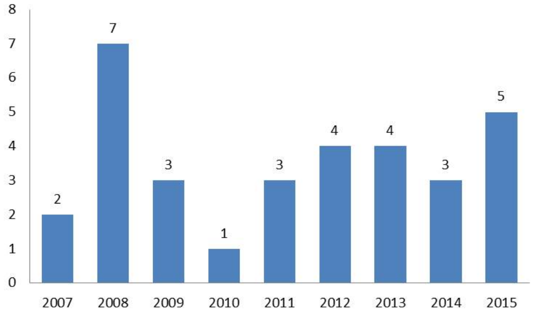
1) We created a guideline document defining each of the classification terms and an Excel spreadsheet to collect the classification information. The spreadsheet contained the following data fields: *i)* SPL life cycle stage, *ii)* rationale for the categorization, *iii)* visualization techniques employed, *iv)* rationale for the classification, *v)* visualization tools employed, *vi)* rationale for the classification, and *vii)* a general field for any remarks.
2) We formed two groups to carry out 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 five selected primary sources. Any discrepancies were discussed and analyzed to homogenize the classification criteria.
4) The two teams performed the classification of all primary sources independently.
5) We held a second meeting where the classification for every single paper for each criterion was discussed until a consensus was reached.

The effort to gather the data varied between papers but for the majority it was a simple task to find all the classification information required. The most time-consuming part was in some cases finding out the implementation tools employed.

IV. RESULTS

The search using the four search engines mentioned above yielded 391 hits for potential papers to consider as primary sources. We performed a more detailed reading of the title, abstract and keywords to gauge at the relevance of the papers found. As result we obtained 32 relevant papers. The most common reason for exclusion was that those papers did not apply visualization techniques to SPLs, for example some simply mention visualization as part of future work but provide no actual application. The exclusion of each paper was double-checked to make sure we did not eliminate any relevant primary source.

Fig. 4. Publications per year



We did snowballing on those papers which produced 9 new relevant papers. Then we performed a more exhaustive screening on the 41 papers which also considered more detailed points of the exclusion criteria and reading several sections of the papers. This detailed screening eliminated 6 papers, from the original 32, and 3 snowballing papers. At the end, our mapping study considers 32 primary sources, listed in the Appendix in the order they were found, which are shown sorted in a histogram by publication year in Figure 4. This figure shows a spike in the number of publications in 2008, and since then a constant and increasing 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 V.

A. RQ1. SPL life cycle stages

Table II shows the use of visualization techniques per life cycle which are summarized in Figure 5. They show that visualization techniques have been used pre-eminently for the Domain Engineering activities of SPLs, that is, those that involve the entire product line as explained in Section II. More concretely, our study found for the requirements engineering (DRE) and the design (DD) stages 20 and 21 references respectively. We should point out that it is at these stages when feature models are defined and commonly the requirements of the features are additionally reified as attributes of the feature models. This fact we believe explains this first finding.

For the stages where the requirements for each product are captured (ARE) and analyzed (AD), we respectively found 15 and 13 primary sources. Here again there is a common reliance on feature models to guide the product configuration process – where the engineer selects the desired combinations of features, analyzing different trade-offs. As before, we believe that the use of feature models at this stage also explains this finding.

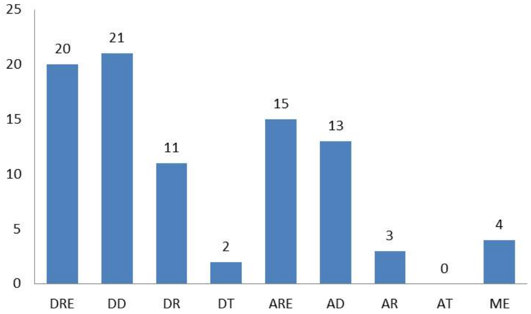
For the stage where the product line is realized at the domain level (DR) we found 11 primary sources. At this stage, our study found that colors are used to describe what software artifacfts or their pieces belong to particular features. For instance, in [S28] the authors use colors to annotate UML-based models and in [S29] the authors follow the same idea but applied to source code.

TABLE II

PRIMARY SOURCES AND LIFE CYCLE STAGE

Stage	Primary Sources Identifiers
DRE	S3, S5, S9, S13, S14, S15, S17, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S32
DD	S3, S5, S9, S10, S13, S14, S15, S17, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S32
DR	S4, S7, S9, S10, S20, S21, S23, S28, S29, S30, S31
DT	S6, S11
ARE	S2, S6, S8, S9, S12, S13, S15, S16, S17, S18, S20, S24, S26, S28, S32
AD	S2, S9, S12, S13, S15, S16, S17, S18, S20, S24, S26, S28, S32
AR	S9, S20, S28
AT	None
ME	S1, S6, S14, S31

Fig. 5. Publications per stage



Our study found 4 primary sources for the maintenance and evolution stage (ME). The authors of [S1] use a tree to depict the evolution of products across time, while the authors of [S31] use bars to depict the evolution of features also across time. The authors of [S6] trace the evolution of bugs across product evolution. Support for maintenance tasks concerning the feature models is addressed in [S14].

Our study found three primary sources for the artifact realization (AR) stage. All of them use different notions of fragments of models (e.g. features, concerns, and deltas) which are configured, analyzed and composed. For this stage they primarily rely on colors to distinguish the fragments which are visualized as models.

We found that despite the extensive research on SPL testing, described in Section VI, only two primary sources exploit visualization techniques for this stage. In [S11] authors employ basic techniques such as tree maps and bubble charts to depict covering arrays. In [S6] authors visualize bug evolution to help with SPL testing tasks. Our study found no visualization techniques used for testing the AT stage.

B. RQ2. Visualization techniques

Table III summarizes the use of visualization techniques. Considering the number of primary sources that visualize different aspects of feature models, it is not a surprise that the most frequent technique was visualization of trees with 11 papers. This was followed by general graphs with 6 papers, and the more specialized form of graphs that constitute concept lattices. We describe next some of the visualization techniques found.

TABLE III

VISUALIZATION TECHNIQUES

Technique	Primary Sources Identifiers
Trees	S1, S2, S3, S4, S6, S9, S16, S17, S18, S24, S32
Graphs (nodes and edges)	S7, S12, S20, S21, S26, S30
Concept lattices	S13, S25
Bar diagrams	S19, S22
Colored code / model elements	S28, S29
Feature histograms	S4
Tables/ Matrices	S5
Bubble chart	S8
Levelized structure map	S10
Bubble map	S11
Heat map	S11
Tree map	S11
Grid	S11
Feature blueprints	S14
Feature relation graphs	S15
Component model annotations	S23
Flow maps	S26
3D cone trees	S27
Feature survival chart	S31
3D color spheres	S32

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 [S14]. Feature relation graphs depict features and their relations as colored concentric circles whose color, width and size depends on the properties of the relations [S15]. 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 [S31].

C. RQ3. Visualization tools

Table IV summarizes our findings for visualization tools. It is noticeable that 15 primary sources did not provide clear description of the tools or APIs they used for the implementation. By judging from the screenshots provided we speculate that the majority relied on the basic graphics API provided by the Java SDK. The second most common tool was the Eclipse Modeling Framework (EMF)[4] in combination with Graphical Editing Framework (GEF)[5] which provide a solid framework infrastructure for creating, among other things, domain specific languages for which visual representations could be devised. The third place was Prefuse[6], a visualization toolkit that has been superseded by D3.js[7]. Graphviz[8] is a software visualization tool specialized on graphs. CCVisu is a visual clustering tool [S7]. Google charts provides support for visualizing data in websites[9]. ConExp is a tool for for-

---

[4] https://eclipse.org/modeling/emf/

[5] https://eclipse.org/gef/

[6] http://prefuse.org/

[7] https://d3js.org/

[8] http://www.graphviz.org/

[9] https://developers.google.com/chart/

TABLE IV
VISUALIZATION TOOLS

Tool	Primary Sources Identifiers
Adhoc	S2, S3, S4, S5, S6, S10, S17, S19, S21, S22, S23, S25, S27, S31, S32
Eclipse EMF-GEF	S9, S12, S16, S18, S20, S28, S29, S30
Prefuse	S24, S26
Graphviz	S1
CCVisu	S7
Google charts	S8
D3.js	S11
ConExp	S13
Moose	S14
Processing 2.0	S15

mal concept analysis [S13]. Moose is a software analysis platform [S14]. Processing[10] is a software sketchbook and language for visual arts.

D. RQ4. Publication fora

Table V summarizes the publication fora sorted by type of publication (e.g. conference, journal, or workshop) and their frequency. It should not come as a surprise that the two leading conferences in SPLs and software visualization are the most frequent publication outlets with 6 and 3 publications respectively. These two conferences are followed by ICSE, SEAA, and WCRE with two 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 is on visualization. From the journal publications [S25] and [S26] have visualization as the main focus of the article, whereas in [S24] it is a secondary concern. From the workshop publications, the most frequent venue was VISPLE, a specialized workshop that at intersection of SPL and visualization that ran for three occasions as an associated workshop at SPLC conference.

V. ANALYSIS

In this section we analyze the core findings revealed by our systematic mapping study. We shortly discuss open questions and potential areas for further research.

A. Pre-eminence of visualization of feature models

Our study revealed that feature models were the most common artifact visualized. Consequently those life cycle stages that commonly use feature models were the most frequently found by out study. Namely, domain requirements engineering (DRE) and design (DD), and application requirements engineering (ARE) and design (DD). 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 life cycle stages has been barely employed. For instance, from the extensive ongoing work on SPL testing, see Section VI, only two approaches have relied on any form of visualization technique. We argue these life stages are worthy avenues for further research and application of visualization techniques.

B. 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 use either 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.

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 (e.g.[S29]) or models (e.g. [S28]). However, relying on colors presents an inherent scalability problem for the common cases where SPLs have hundreds, if not thousands, of features. Our study also revealed that even though features are modeled and implemented with different artifacts throughout the SPL life cycle, and hence represent multivariate data, none of the primary sources exploits this fact for visualization purposes. We argue that handling scalability and multivariate data visualization are two open challenges with a high impact potential in the SPL community.

VI. RELATED WORK

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

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

- SPL adoption [12]. This work identified four adoption strategies and 23 barriers that can hinder SPL adoption in industrial projects.
- Agile methods [9]. This study found that most of the applications of agile methods follow XP or Scrum and identified SPL practices that can be exploited by Agile techniques.
- Requirements engineering [2]. This study found that the application of requirements engineering techniques for SPL was still not mature and advocate that more empirical studies should be performed to improve the rigor, credibility, and validity of the proposed approaches.
- Service orientation [26], [24]. 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.
- SPL testing [11], [8], [10], [22]. These studies provide a taxonomy and classify over more than a hundred sources along several dimensions. Among their findings is the pre-eminence of combinatorial approaches for selecting representative products to test and that there is still a great lack of empirical industrial applications.
- SPL evolution [21]. They made an assessment of the maturity level of techniques to migrate individual systems or groups of software variants into SPLs.

---

[10] https://processing.org/

TABLE V
PUBLICATION FORA

Acronym	Primary Sources Identifiers	Publication Name
	Conference Publications	
SPLC	S1, S3, S6, S8, S13, S16	International Conference Software Product Lines
VISSOFT	S11, S14, S15	IEEE Working Conference on Software Visualization
ICSE	S4, S7	International Conference on Software Engineering
SEAA	S12, S21	Euromicro Conference on Software Engineering and Advanced Applications
WCRE	S19, S23	Working Conference on Reverse Engineering
SoftVis	S2	ACM Symposium on Software Visualization
FSE	S5	Foundations of Software Engineering
Modularity	S9	International Conference on Modularity
IC3	S10	International Conference on Contemporary Computing
ASE	S20	International Conference on Automated Software Engineering
COMPSAC	S18	International Conference on Computer Software and Applications
RE	S31	IEEE International Requirements Engineering Conference
ISVC	S32	International Symposium Advances in Visual Computing
	Journal Publications	
IST	S24	Information and Software Technology
Proceedia	S25	Procedia Technology
ISTTT	S26	International Journal on Software Tools for Technology Transfer
	Workshop Publications	
VISPLE	S27, S28, S29, S30	Workshop on Visualisation in Software Product Line Engineering
REV	S17	International Workshop on Requirements Engineering Visualization
PLEASE	S22	International Workshop on Product Line Approaches in Software Engineering

- Variability management [6], [13]. Among their collective findings are that a large majority of the reported approaches have not been sufficiently evaluated using scientifically rigorous methods (e.g. following [42]) and that software quality attributes have not received much attention.
- Product configuration support [32], [15]. These studies performed a combination of questionnaire and tool survey to identify the requirements for tools to support configuration.
- Search-Based Software Engineering (SBSE) [23]. This study analyzed what and how search-based techniques – including metaheuristic search based optimization techniques and classical operations research techniques – have been employed for SPLs problems. The study found the pre-eminence of metaheuristic approaches, e.g. genetic algorithms, applied to SPL testing.

Visualization surveys. Schots et al. performed an extensive review of visualization for software reuse [35]. They found four of the primary sources that our study identified even though SPLs are a form of systematic software reuse.

Seriai et al. performed a systematic mapping study on the validation of visualization tools [36]. Their main finding was that despite the increasing research and application of visualization techniques their evaluation lacks rigour. Novais et al. carried out a systematic mapping study in software evolution visualization [27]. 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 a systematic mapping study of visualization tools and techniques for software comprehension [31]. Their study corroborates the lack of robust empirical evaluation, and found that most approaches use bi-dimensional visualizations and do not address user interactions.

Abuzaid and Scott performed a systematic literature review 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 [1]. Paredes et al. carried out a systematic mapping study of the use of information visualization for software development following Agile approaches [28]. They found visualization used for designing, developing, communication and keeping track of progress.

VII. CONCLUSIONS AND FUTURE WORK

In this paper we presented a systematic mapping on the use of visualization for SPL development which found 32 primary sources. Our study revealed the pre-eminent use of visualization techniques for SPL development activities that involve feature models, a de facto standard for describing the combinations of features in the products of a SPL. We also found that most primary sources rely on basic visualization techniques and tools, e.g. ad hoc or based in Eclipse tools, that barely exploit the wealth of techniques available in the software and information visualization communities. We hope our work can entice researchers in SPL and visualization communities to pursue further work in the subject. As part of our future work, we want to take a closer look on the interaction capabilities on the identified approaches and analyze the empirical foundations on which they rely upon.

VIII. ACKNOWLEDGEMENTS

This research was funded by the Austrian Science Fund (FWF) projects P25289-N15 and P25513-N15.

REFERENCES

[1] D. Abuzaid and S. Titang. The visualization of software quality metrics. a systematic literature review. Bachelor thesis. University of Gothenburg. Chalmers University of Technology, 2014.

[2] V. Alves, N. Niu, C. F. Alves, and G. Valença. Requirements engineering for software product lines: A systematic literature review. *Information & Software Technology*, 52(8):806–820, 2010.

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

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

[5] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham. Using Mapping Studies in Software Engineering. In *Proceedings of PPIG 2008*, pages 195–204. Lancaster University, 2008.

[6] L. Chen and M. A. Babar. A systematic review of evaluation of variability management approaches in software product lines. *Inform. & Software Tech.*, 53(4):344–362, 2011.

[7] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wasowski. Cool features and tough decisions: a comparison of variability modeling approaches. In U. W. Eisenecker, S. Apel, and S. Gnesi, editors, *VaMoS*, pages 173–182. ACM, 2012.

[8] P. A. da Mota Silveira Neto, I. do Carmo Machado, J. D. McGregor, E. S. de Almeida, and S. R. de Lemos Meira. A systematic mapping study of software product lines testing. *Information & Software Technology*, 53(5):407–423, 2011.

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

[10] I. do Carmo Machado, J. D. McGregor, Y. C. Cavalcanti, and E. S. de Almeida. On strategies for testing software product lines: A systematic literature review. *Information and Software Technology*, 56(10):1183 – 1199, 2014.

[11] E. Engström and P. Runeson. Software product line testing - a systematic mapping study. *Inform. & Software Tech.*, 53(1):2–13, 2011.

[12] J. Ferreira Bastos, P. Anselmo da Mota Silveira Neto, E. Santana de Almeida, and S. Romero de Lemos Meira. Adopting software product lines: A systematic mapping study. In *Evaluation Assessment in Software Engineering (EASE 2011), 15th Annual Conference on*, pages 11–20, April 2011.

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

[14] R. Heradio, H. Perez-Morago, D. Fernández-Amorós, F. J. Cabrerizo, and E. Herrera-Viedma. A bibliometric analysis of 20 years of research on software product lines. *Information & Software Technology*, 72:1–15, 2016.

[15] G. Holl, P. Grünbacher, and R. Rabiser. A systematic review and an expert survey on capabilities supporting multi product lines. *Inform. & Software Tech.*, 54(8):828–852, 2012.

[16] T. Käkölä and J. C. Dueñas, editors. *Software Product Lines - Research Issues in Engineering and Management*. Springer, 2006.

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

[18] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. version 2.3. EBSE Technical Report EBSE-2007-01, Software Engineering Group, School of Computer Science and Mathematics, Keele University, UK and Department of Computer Science, University of Durham, UK, 2007.

[19] B. Kitchenham, T. Dybaa, and M. Jorgensen. Evidence-based software engineering. In *ICSE*, pages 273–281. IEEE CS Press, 2004.

[20] B. A. Kitchenham, D. Budgen, and O. P. Brereton. Using mapping studies as the basis for further research - a participant-observer case study. *Information & Software Technology*, 53(6):638–651, 2011.

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

[22] R. E. Lopez-Herrejon, S. Fischer, R. Ramler, and A. Egyed. 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, Graz, Austria, April 13-17, 2015*, pages 1–10. IEEE Computer Society, 2015.

[23] R. E. Lopez-Herrejon, L. Linsbauer, and A. Egyed. A systematic mapping study of search-based software engineering for software product lines. *Journal of Information and Software Technology*, 2015.

[24] S. Mahdavi-Hezavehi, M. Galster, and P. Avgeriou. Variability in quality attributes of service-based software systems: A systematic literature review. *Information & Software Technology*, 55(2):320–343, 2013.

[25] A. Metzger and K. Pohl. Software product line engineering and variability management: achievements and challenges. In J. D. Herbsleb and M. B. Dwyer, editors, *FOSE*, pages 70–84. ACM, 2014.

[26] B. Mohabbati, M. Asadi, D. Gasevic, M. Hatala, and H. A. Müller. Combining service-orientation and software product line engineering: A systematic mapping study. *Information & Software Technology*, 55(11):1845–1859, 2013.

[27] R. L. Novais, A. Torres, T. S. Mendes, M. G. Mendonça, and N. Zazworka. Software evolution visualization: A systematic mapping study. *Information & Software Technology*, 55(11):1860–1883, 2013.

[28] J. Paredes, C. Anslow, and F. Maurer. Information visualization for agile software development. In Sahraoui et al. [33], pages 157–166.

[29] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *EASE*, pages 68–77. British Computer Society, 2008.

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

[31] M. P. Prado, A. M. R. Vincenzi, F. A. A. de M. N. Soares, F. Cesar, G. P. de Paula, H. A. D. do Nascimento, J. C. Silva, J. L. de Oliveira, L. C. Lima, and T. Fernandes. Characterization of techniques and tools of visualization applied to software comprehension: A systematic mapping. In *International Conference on Software Engineering Advances (ICSEA)*, 2013.

[32] R. Rabiser, P. Grünbacher, and D. Dhungana. Requirements for product derivation support: Results from a systematic literature review and an expert survey. *Inform. & Software Tech.*, 52(3):324–346, 2010.

[33] H. A. Sahraoui, A. Zaidman, and B. Sharif, editors. *Second IEEE Working Conference on Software Visualization, VISSOFT 2014, Victoria, BC, Canada, September 29-30, 2014*. IEEE Computer Society, 2014.

[34] M. Schots. On the use of visualization for supporting software reuse. In P. Jalote, L. C. Briand, and A. van der Hoek, editors, *36th International Conference on Software Engineering, ICSE '14, Companion Proceedings, Hyderabad, India, May 31 - June 07, 2014*, pages 694–697. ACM, 2014.

[35] M. Schots, R. Vasconcelos, and C. Werner. A quasi-systematic review on software visualization approaches for software reuse. Technical report, Federal University of Rio de Janeiro, 2014.

[36] A. Seriai, O. Benomar, B. Cerat, and H. A. Sahraoui. Validation of software visualization tools: A systematic mapping study. In Sahraoui et al. [33], pages 60–69.

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

[38] A. Telea. *Data visualization - principles and practice*. A K Peters, second edition, 20015.

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

[40] M. O. Ward, G. G. Grinstein, and D. A. Keim. *Interactive Data Visualization - Foundations, Techniques, and Applications*. A K Peters, 2010.

[41] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In M. J. Shepperd, T. Hall, and I. Myrtveit, editors, *EASE*, page 38. ACM, 2014.

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

APPENDIX

PRIMARY SOURCES LIST

[S1] Tetsuya Kanda, Takashi Ishio, and Katsuro Inoue. Extraction of product evolution tree from source code of product variants. In *Proceedings of the 17th International Software Product Line Conference*, SPLC '13, pages 141–150, New York, NY, USA, 2013. ACM.

[S2] Daren Nestor, Steffen Thiel, Goetz Botterweck, Ciarán Cawley, and Patrick Healy. Applying visualisation techniques in software product lines. In *Proceedings of the 4th ACM Symposium on Software Visualization*, SoftVis '08, pages 175–184, New York, NY, USA, 2008. ACM.

[S3] Alessio Ferrari, Giorgio O. Spagnolo, Stefania Gnesi, and Felice Dell'Orletta. 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, pages 402–410, New York, NY, USA, 2015. ACM.

[S4] Michael Stengel, Mathias Frisch, Sven Apel, Janet Feigenspan, Christian Kästner, and Raimund Dachselt. View infinity: A zoomable interface for feature-oriented software development. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 1031–1033, New York, NY, USA, 2011. ACM.

[S5] Sana Ben Nasr, Guillaume Bécan, Mathieu Acher, João Bosco Ferreira Filho, Benoit Baudry, Nicolas Sannier, and Jean-Marc Davril. 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, pages 982–985, New York, NY, USA, 2015. ACM.

[S6] Thiago Henrique Burgos de Oliveira, Martin Becker, and Elisa Yumi Nakagawa. 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, pages 181–185, New York, NY, USA, 2012. ACM.

[S7] Sven Apel and Dirk Beyer. Feature cohesion in software product lines: An exploratory study. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 421–430, New York, NY, USA, 2011. ACM.

[S8] Alexandr Murashkin, MichałAntkiewicz, Derek Rayside, and Krzysztof Czarnecki. Visualization and exploration of optimal variants in product line engineering. In *Proceedings of the 17th International Software Product Line Conference*, SPLC '13, pages 111–115, New York, NY, USA, 2013. ACM.

[S9] Nishanth Thimmegowda and Jörg Kienzle. Visualization algorithms for feature models in concern-driven software development. In *Companion Proceedings of the 14th International Conference on Modularity*, MODULARITY Companion 2015, pages 39–42, New York, NY, USA, 2015. ACM.

[S10] M. Kaur and P. Kumar. Spotting the phenomenon of bad smells in mobilemedia product line architecture. In *Contemporary Computing (IC3), 2014 Seventh International Conference on*, pages 357–363, Aug 2014.

[S11] R. E. Lopez-Herrejon and A. Egyed. Towards interactive visualization support for pairwise testing software product lines. In *Software Visualization (VISSOFT), 2013 First IEEE Working Conference on*, pages 1–4, Sept 2013.

[S12] R. Rabiser, D. Dhungana, W. Heider, and P. Grnbacher. Flexibility and end-user support in model-based product line tools. In *Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on*, pages 508–511, Aug 2009.

[S13] F. Loesch and E. Ploedereder. Optimization of variability in software product lines. In *Software Product Line Conference, 2007. SPLC 2007. 11th International*, pages 151–162, Sept 2007.

[S14] S. Urli, A. Bergel, M. Blay-Fornarino, P. Collet, and S. Mosser. A visual support for decomposing complex feature models. In *Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference on*, pages 76–85, Sept 2015.

[S15] J. Martinez, T. Ziadi, R. Mazo, T. F. Bissyand, J. Klein, and Y. L. Traon. Feature relations graphs: A visualisation paradigm for feature constraints in software product lines. In *Software Visualization (VISSOFT), 2014 Second IEEE Working Conference on*, pages 50–59, Sept 2014.

[S16] G. Botterweck, S. Thiel, D. Nestor, S. b. Abid, and C. Cawley. Visual tool support for configuring and understanding software product lines. In *Software Product Line Conference, 2008. SPLC '08. 12th International*, pages 77–86, Sept 2008.

[S17] D. Sellier and M. Mannion. Visualising product line requirement selection decision inter-dependencies. In *Requirements Engineering Visualization, 2007. REV 2007. Second International Workshop on*, pages 7–7, Oct 2007.

[S18] G. Botterweck, S. Thiel, C. Cawley, D. Nestor, and A. Preuner. Visual configuration in automotive software product lines. In *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, pages 1070–1075, July 2008.

[S19] S. Duszynski, J. Knodel, and M. Becker. Analyzing the source code of multiple software variants for reuse potential. In *Reverse Engineering (WCRE), 2011 18th Working Conference on*, pages 303–307, Oct 2011.

[S20] C. Pietsch, T. Kehrer, U. Kelter, D. Reuling, and M. Ohrndorf. Sipl – a delta-based modeling framework for software product line engineering. In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 852–857, Nov 2015.

[S21] T. F. L. de Medeiros, E. S. de Almeida, and S. R. de Lemos Meira. Codescoping: A source code based tool to software product lines scoping. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, pages 101–104, Sept 2012.

[S22] S. Duszynski and M. Becker. Recovering variability information from the source code of similar software products. In *Product Line Approaches in Software Engineering (PLEASE), 2012 3rd International Workshop on*, pages 37–40, June 2012.

[S23] S. Duszynski, J. Knodel, M. Naab, D. Hein, and C. Schitter. Variant comparison - a technique for visualizing software variants. In *Reverse Engineering, 2008. WCRE '08. 15th Working Conference on*, pages 229–233, Oct 2008.

[S24] Mohsen Asadi, Samaneh Soltani, Dragan Gasevic, Marek Hatala, and Ebrahim Bagheri. Toward automated feature model configuration with optimizing non-functional requirements. *Information and Software Technology*, 56(9):1144 – 1165, 2014. Special Sections from Asia-Pacific Software Engineering Conference (APSEC), 2012 and Software Product Line conference (SPLC), 2012.

[S25] Tom Huysegoms, Monique Snoeck, Guido Dedene, Antoon Goderis, and Frank Stumpe. Visualizing variability management in requirements engineering through formal concept analysis. *Procedia Technology*, 9:189 – 199, 2013. {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.

[S26] Andreas Pleuss and Goetz Botterweck. Visualization of variability and configuration options. *STTT*, 14(5):497–510, 2012.

[S27] Pablo Trinidad, Antonio Ruiz-Cortés, David Benavides, and S. Segura. Three-dimensional feature diagrams visualization. In *2nd SPLC Workshop on Visualisation in Software Product Line Engineering (ViSPLE)*, page 295–302, Limerick, Ireland, Sep 2008. Irish Software Engineering Research Centre (Lero), Irish Software Engineering Research Centre (Lero).

[S28] Florian Heidenreich, Ilie Savga, and Christian Wende. On controlled visualisations in software product line engineering. In *Software Product Lines, 12th International Conference, SPLC 2008, Limerick, Ireland, September 8-12, 2008, Proceedings. Second Volume (Workshops)*, pages 335–341, 2008.

[S29] Christian Kästner, Salvador Trujillo, and Sven Apel. Visualizing software product line variabilities in source code. In *Software Product Lines, 12th International Conference, SPLC 2008, Limerick, Ireland, September 8-12, 2008, Proceedings. Second Volume (Workshops)*, pages 303–312, 2008.

[S30] André Heuer, Kim Lauenroth, Marco Müller, and Jan-Nils Scheele. Towards effective visual modeling of complex software product lines. In *Software Product Lines - 14th International Conference, SPLC 2010, Jeju Island, South Korea, September 13-17, 2010. Workshop Proceedings (Volume 2 : Workshops, Industrial Track, Doctoral Symposium, Demonstrations and Tools)*, pages 229–238, 2010.

[S31] K. Wnuk, B. Regnell, and L. Karlsson. What happened to our features? visualization and understanding of scope change dynamics in a large-scale industrial setting. In *Requirements Engineering Conference, 2009. RE '09. 17th IEEE International*, pages 89–98, Aug 2009.

[S32] Ciarán Cawley, Goetz Botterweck, Patrick Healy, Saad Bin Abid, and Steffen Thiel. A 3d visualisation to enhance cognition in software product line engineering. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Yoshinori Kuno, Junxian Wang, Renato Pajarola, Peter Lindstrom, André Hinkenjann, Miguel L. Encarnação, Cláudio T. Silva, and Daniel Coming, editors, *Advances in Visual Computing: 5th International Symposium, ISVC 2009, Las Vegas, NV, USA, November 30-December 2, 2009. Proceedings, Part II*, pages 857–868, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.