Variability Management in Software Product Lines: A Systematic Review

Lianping Chen, Muhammad Ali Babar, Nour Ali
Lero, the Irish Software Engineering Research Centre, University of Limerick, Ireland
{lianping.chen, malibaba, nour.ali}@lero.ie

Abstract

Variability Management (VM) in Software Product Line (SPL) is a key activity that usually affects the degree to which a SPL is successful. SPL community has spent huge amount of resources on developing various approaches to dealing with variability related challenges over the last decade. To provide an overview of different aspects of the proposed VM approaches, we carried out a systematic literature review of the papers reporting VM in SPL. This paper presents and discusses the findings from this systematic literature review. The results reveal the chronological backgrounds of various approaches over the history of VM research, and summarize the key issues that drove the evolution of different approaches. This study has also identified several gaps that need to be filled by future efforts in this line of research.

Keywords: Software product lines, variability management, systematic reviews

1. Introduction

Software Product Line Engineering (SPLE) intends to develop software-intensive systems using platforms and mass customisation [1, 2]. This is achieved through the identification and management of commonalities and variations in a set of systems' artefacts such as requirements, architectures, components, and test cases. In SPLE, variability provides the required flexibility for product differentiation and diversification. Variability refers to the ability of an artefact to be configured, customized, extended, or changed for use in a specific context [3]. Variability Management (VM) encompasses the activities of explicitly representing variability in software artefacts throughout the lifecycle, managing dependencies among different variabilities, and supporting the instantiations of those variabilities [4]. It involves extremely complex and challenging tasks, which needs to be supported by appropriate approaches, techniques, and tools [4, 5]. Systematically identifying and appropriately managing variabilities among different systems of a family are the key characteristics that distinguish SPLE from other reuse-based software development approaches [5].

Given such a vital role of VM in SPLE, there has been a great deal of research in this area of SPLE. Many diverse approaches have been developed with the basic aim of supporting (or automating) various tasks involved in VM at different stages of a product line's life. However, there has been no effort to systematically survey the VM approaches reported in the literature in order to understand their evolutionary paths, inter-relationships, and the motivational issues. Hence, we decided to conduct a Systematic Literature Review (SLR) [6] or Systematic Review (SR) of the literature on VM in SPLE in order to summarize the state of the art in VM research. The specific research questions that motivated our study are:

- What approaches have been proposed for managing variability in software product lines?
- How has the research on developing VM approaches been evolved?
- What are the key issues that have drove the evolution of different VM approaches?

Previously, there have been a few efforts to survey the literature on VM in Software Product Lines (SPL) [7, 8]. However, these efforts were aimed at studying very concrete elements of VM (i.e., modeling [7] and realization mechanisms [8]). Our research has completely different goals as stated before and we have used a systematic and rigorous approach to identifying and selecting the reviewed primary studies. Our study is based on a systematic search of publications from various data sources and follows a pre-defined protocol during the whole process. None of the previous surveys followed a systematic selection process of the reviewed studies; nor did they focus on revealing the chronology of VM research in SPL over the years.

In the rest of the paper, Section 2 describes the research methodology used. Section 3 presents and discusses the results, while Section 4 closes the paper by describing the main conclusions.

2. Research Methodology

As we have mentioned, this study has been carried out according to the SR methodology described in [6]. Since recently many published studies have described the methodology, logistics, benefits, and limitations of SRs in software engineering, we only discuss the key aspects of

the methodology used for the reported research. However, we followed all the stages and steps recommended in Kitchenham's guidelines [6].

2.1. Search Strategy and Data Sources

The search strings used in this review were constructed using the following strategy:

- Derive main terms based on the research question and the topics being researched;
- Determine and include synonyms, related terms, and alternative spelling for major terms;
- Check the keywords in all relevant papers researchers already knew for example [4, 7, 9] and initial searches on the relevant databases;
- Incorporate alternative spellings and synonyms using Boolean "or";
- Link main terms using Boolean "and";
- Pilot different combinations of the search terms.

Following this strategy, we constructed the search strings as bellow:

<<software AND (product line OR product lines OR product family OR product families) AND (variability OR variation OR variant)>>

The final search terms were constructed after a series of test executions and reviews. Due to the varying nature of the search features provided by the main digital sources of literature (such as IEEExplore, SpringerLink, and ACM Digital Library), it was not possible to use a single search string for all the digital sources. Hence, like others [9], we also used different search strings for different sources with the exception of the ACM Digital Library where we had to construct three different search strings. If it was not possible to have syntactically identical search strings for all the searched databases, we made every effort to ensure that the search strings used were logically and semantically equivalent. Three researchers were involved in this process. All of them continuously discussed and refined the search strings until they were fully satisfied with the capability of the used search strings.

We searched the primary studies in these digital sources (1. IEEExplore; 2. ACM Digital library; 3. Citeseer library (Google); 4. ScienceDirect; 5. EI Compendex / Inspec; 6. SpringerLink; and 7. Web of Science). As an indication of inclusiveness, the results were checked for three known relevant papers (i.e. [4, 7, 10]). All three relevant papers were found in the search results. Apart from those electronic databases, we also manually checked two sources for primary studies: (1. SPLC conference series' proceedings that are not available online; and 2. SEI's technical reports on SPL). One of the well known papers, Feature-Oriented Reuse Method (FORM) [11], was not brought by our authormatic search. An investigation revealed that it did not use any of the keywords used in our automatic search string. We acquired that paper and included in our review. We searched the papers from December 2007 to January 2008. That means the papers published after that date were not reviewed.

Give our limited resources, it was not possible for us to cover all the potential publication venues of SPL literature. That was why we targeted those forums where SPL researchers are expected to publish their work. So the VM approaches published in venues other than the ones we searched for our SR. We purposely left out the approaches that only tackle variability at the implementation stage. A collection of variability implementation mechanisms is reported in [8]. The quality of search engines could have influenced the completeness of the identified primary studies. That means our search may have missed those studies whose authors would have used other terms to specify variability in software product line or would not have used the keywords that we used for the searches in the title, abstract, and keywords parts of their papers.

2.2. Study Selection

We found 628 papers from all sources after removing the duplicates. The papers were downloaded into an Endnote library where all duplicates were removed. We used a staged study selection process. In the first stage, a paper was included if it:

- introduces an approach to dealing with some aspect of VM in SPLE or;
- reports an evaluation of a VM approach.

The paper was excluded if:

- it does not deal with VM in SPLE.

After the first stage, 261 papers were selected. In the second stage, we limited the publication venues to international journals (not include magazines) and proceedings of SPLC and PFE series. 70 papers were left at this stage. During the third stage, we investigated each of the 70 papers, and excluded every paper if it:

- only focuses on variability implementation mechanisms;
- introduces approaches in a particular domain, which do not have generic applicability;
- only presents concepts or conceptual framework (e.g. [12]) instead of concrete and well formed approach;
- only addresses a particular quality attribute;
- is a secondary study;
- is a short paper.

After this stage, there were 34 papers, which were included for extracting and analysing the data.

2.3. Data Extraction and Synthesis

We fully read each of the 34 papers for extracting the required data. We used a predefined form consisting of a number of attributes for extracting and storing the data. These attributes were expected to be required in order to answer our main research questions. This paper does not include the data extraction form because of space limit. Three researchers were involved in extracting and verifying the data. Since most of the selected studies were grounded in qualitative research, a meta-analytical approach was not suitable for synthesizing the data. We decided to manually review and link the extracted data. We decided to group the issues that the reviewed studies claimed to address. We used the affinity diagram [13] process for grouping the issues. This approach fitted well with our data analysis requirements as it allows mutually inclusive categories.

The grouping of issues was performed by two researchers in face to face meetings. The issues for each paper were written down on post-it notes, which were plotted on a white board. The post-it notes were grouped on the white board. Each group was given a name. If one paper raised issues in more than one groups, a duplicate post-it for that paper was put under each group. These groups were further clustered to higher-level groups. Then, we used descriptive statistics (e.g. sum, average) for analysing the data.

Table 1: Distribution of studies based on publication venues

Venue	Type	#
SPLC	C	12
PFE	W	6
SCP	J	4
IEE Proceedings-Software	J	2
TSE	J	1
SoSyM	J	1
SPE	J	1
RE	J	1
JSS	J	1
IMDS	J	1
Computer Networks	J	1
Annals of Software Engineering	J	1
Advanced Engineering Informatics	J	1
SEI Technical Report	TR	1
Total		34

3. Results and Discussion

3.1. Demographic Data

Table 1 gives an overview of the studies according to publication venues. We notice that the SPLC and PFE series have the largest number of papers, followed by Journal of Science of Computer Programming (SCP), which had a special issue on VM in 2004. All of those 4 papers appeared in that issue. Regarding the year of publication, as shown in table 2, the first paper we identified was in 1990. Our SR revealed two peak periods of VM publications in 2002 and 2004, with 9 and 11 papers published respectively. It should be noted that the proceedings of 2001 edition of PFE were published in 2002, from which 3 papers were included. The paper in 2008 was fetched by our search, because the search phase finished in January 2008, and the paper was available online at that time.

Table 2: Number of papers in each year

Year	1990	1998	2000	2002	2003	2004	2005	2006	2007	2008	Total
#	1	1	3	9	1	11	2	1	4	1	34

3.2. Overview of the Reviewed Approaches

We found that two of the reviewed 34 papers presented the same approach. Hence, our analysis is based on 33 approaches. If an approach has a name, we have used that name. Otherwise, we have given a name to an approach for this study by using the first author's surname followed by the publication year, for example Muthig'02. Table 3 lists each of the 33 approaches in chronological order of publication. In this review, the approach, their origin and short description have been identified based on what is stated in each of the reviewed paper. In Table 3, the column "approach" contains the name of the approach given by the authors or by us, the column "paper" provides the reference to the paper (s) that has reported the approach. The last column gives a short description of each approach.

The approaches described in the reviewed papers are quite diverse, in terms of goals, philosophy of approach design, techniques used for modeling variability, process support, and so on. For example, a large majority of them are feature oriented approaches, like Feature-Oriented Domain Aanalysis (FODA) [14] and its extensions. Some approaches are architecture-centric such as Hoek'04 [15], Koalish [16], and Thiel'02 [17]. Some of the approaches are configuration-based like Krueger'02 [18], COVAMOF [19], Koalish [16], and Kumbang [20]. Others attempt to extend UML to model variability like VPM [10] and Halmans'03 [21]. Some of the approaches focus on the separating variability representation from the representation of various SPLE artifacts such as Bachmann'04 [22] and Muthig'02 [23]. The developers of some approaches emphasize the importance of notation independency and customizability to facilitate ease of adoption like Schmid'04 [4]. The main focus of FAST

[24] is providing process support without prescribing a specific VM model. Some VM approaches mainly focus on supporting the identification of variability and commonality such as DRM [25], and Moon [26]. While Loesch'07 [27] only focuses on variability optimization in terms of identifying and removing obsolete variabilities from SPL assets.

Table 3: List of approaches reviewed

Approach	Paper	Short description
FODA	[14]	Feature-oriented domain analysis
FORM	[11]	Feature-oriented, extended FODA for design and implementation
FAST	[24]	Process focused
SPLIT	[28]	Software product line integrated technology
KobrA	[29]	Component based
Muthig'02	[23]	Model-driven architecture, decision model based
Thiel'02	[17]	Architecture centric, extended IEEE P1471
Krueger'02	[18]	Configuration-based, operate on the file system level
Ferber'02	[30]	Feature interaction and dependencies
Fey'02	[31]	Feature modeling, enhance usability and usefulness
Becker'02	[32]	Comprehensive variability modeling
Capilla'02	[33]	Include numerical values in feature modeling
Salicki'02	[34]	Variability description and usage
FDL	[35]	Feature description language
Halmans'03	[21]	Extended use case diagram
Bachmann'04	[22]	Orthogonal variability modeling
VPM	[10]	Extended UML, modeling variation point
Jansen'04	[36]	Relates features to a component role model
Koalish	[16]	Product configuration based, architecture-centric
COVAMOF	[19, 37]	First-class representation of complex dependencies, modeling dependency interactions, product derivation
Schmid'04	[4]	Customizable, notation independent, full life cycle
RequiLine	[38]	Tool support for RE, feature model based
CBFM	[39]	Staged configuration, cardinality based feature model
CONSUL	[40]	Feature model based, full life cycle support
Hoek'04	[15]	Architecture-centric, any-time variability
DRM	[25]	Scenario, goal and feature-oriented
Moon'05	[26]	Rational (objective) C&V identification
Ye'05	[41]	Modeling feature variability and dependencies, two views
Brown'06	[42]	Weaving behavior to feature models
Schobbens'07	[43]	Formalization of feature diagram, generic semantics
Loesch'07	[27]	Optimization of variability
Reiser'07	[44]	Multi-level feature trees
Kumbang	[20]	Domain ontology for modeling variability

We did not intend to propose a taxonomic classification of VM approaches. Considering the significantly diverse nature of the reported approaches, we believe that it would be quite difficult to find a classification scheme that can help categorize the current VM approaches satisfactorily (i.e, cleanly, succinctly, meaningfully, and completely). Rather, our objective was to analyse the reported approaches to provide the SPL community (i.e., researchers and practitioners) with an overview of the VM approaches in SPLE from two key view points:

- Chronological view, which sketches the proposition of the approaches over the history of VM research;
- Issues view, which sketches the issues different VM approaches claim to address over the years.

We have identified various variability models used by different approaches. We also looked into the software development life cycle stages, which have attracted most of the research efforts.

3.3. Chronological Overview

Figure 1 shows the chronological history of the reviewed approaches and their relationships with each others. In Figure 1, the continuous lines indicate that the latest approaches are either based on, or receive inspiration from, the previous approaches. The dotted lines circle the approaches that share at least one of the authors of the papers in which the approaches have been

published. The shadowed boxes indicate the approaches that were not included in this SR because of paper selection criteria as mentioned in Section 2.2.

This SR has revealed that researchers contributed the largest number of VM approaches in 2002 (9 approaches) and 2004 (11 approaches). We have placed the approaches in different years based on the publication date of the paper reviewed. Hence, the development date of an approach may be different.

Figure 1 shows that FODA [14] has the largest number (13) of approaches that were based on it. Three approaches were based on Koala [45]. FORM [11], Muthig'02 [23] CBFM [39], Koalish [16], SPLIT [28], and Ferber'02 [30] has contributed to the creation of at least one VM approach. There are several approaches that were not based on any previous approach as shown in Figure 1.

FODA focuses on feature-oriented requirements engineering. FODA was extended by FORM to support the VM for the design and implementation phases in 1998. After that, many other approaches have been based on the feature modeling or its extensions. For example, Ferber'02 [30] has proposed to use a separate view to represent the feature dependencies and interactions. Ye'05 [41] has extended Ferber'02's approach by extending the meaning of the view (i.e. a view is more than a diagram). RequiLine [38] has extended feature model to model reusing features, which might be optional in one domain and mandatory in another. This extension has been used to develop a tool for managing variability in requirements.

CBFM [39] and Schobbens'07 [43] have tried to give formal semantics to feature models in order to improve precision and semantics. Brown'06 [42] has integrated behavioral variability into feature models. Reiser'07 [44] has introduced multi-level concept to feature trees in order to manage highly complex product families. Moon'05 [26], DRM [25], and Jansen'04 [36], have also employed feature model. While, we expect that the authors of these papers were aware of FODA while developing their respective approaches, however, they did not mention whether or not their approaches were based on FODA [14].

Another notable branch of work has been inspired by Koala [16]. Koala is a component model and an architectural description language that uses components and interfaces to specify the logical structure of a software system. Hoek'04 [15] was inspired by Koala. Koala resolves all variability at compile time. Hoek'04 differs from Koala by supporting any-time variability (i.e. support the resolution of variability in any particular point in the software life cycle, except phases earlier than architecture). Koalish [16] has extended Koala with explicit variability modeling constructs developed in the product configuration domain [46]. Like Koala, Koalish does not abstract to features. Kumbang [20] has combined Koalish with concepts from feature modeling for modeling a SPL from both features and architectural points of views. Kumbang is a jointpoint between the FODA inspired and the Koala inspired approaches. Figure 1 also shows two relatively small branches of VM approaches. In one branch, Salicki'02 [34] is based on SPLIT [28]. While Schmid'04 [4] is highly influenced by Muthig'02 [23]. Like Muthig'02 [23], Schmid'04 is also notation independent and uses decision model. However, Schmid'04 is more focused on customizability.

As shown by the dotted lines in Figure 1, FODA [14] and FORM [11] share the same first author (i.e., Kang). Kumbang [20] and Koalish [16] share the same authoring team. KobrA [29] and Muthig'02 [23] have two common authors (i.e., Atkinson and Muthig). Halmans'03 and Bachmann'04 have one common author (i.e., Pohl). COVAMOF, Jansen'04, and Bachmann'04 also have one common author (i.e., Bosch). Both Pohl and Bosch were also present in Bachmann'04 [22], which emphasizes the separation of the representation of variability from the representation of the various SPLE artifacts. This idea served as the foundation of several subsequent VM approaches, like Orthogonal Variability Model (OVM) [47].

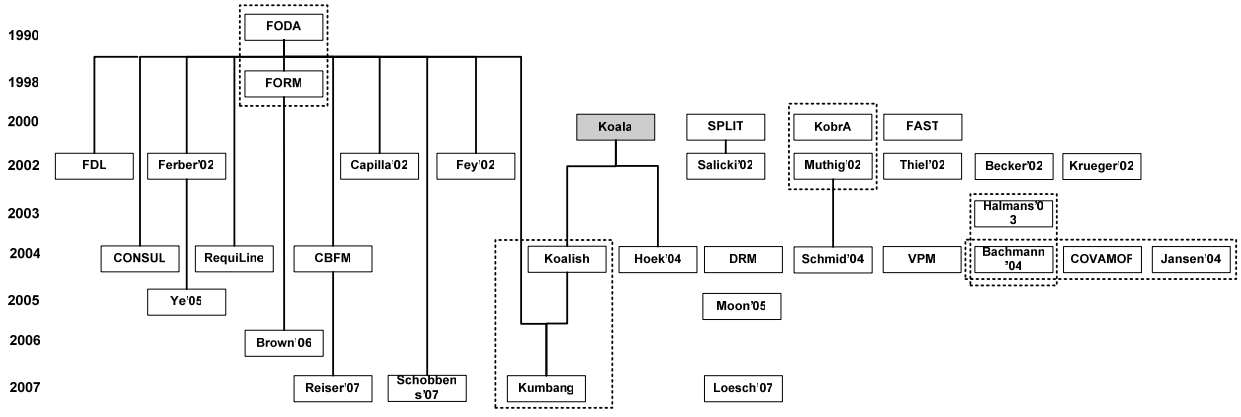

Figure 1: VM Approaches reviewed

3.4. Issues Reported or Addressed

In order to identify the key issues that have motivated the development of different VM approaches, we extracted and analyzed the issues claimed to be addressed by the developers of the reviewed approaches. While identifying the issues claimed to be the motivators for each of the surveyed approaches, we tried to stick to what the authors stated in their papers instead of making our own inferences. We scrutinized the papers following the chronological order, from the oldest one to the latest one, during the issue analysis process. We only counted the prominent issues that were known to be still open at the time the approach was published, that motivated the proposition of the approach, and that appear to distinguish the proposed approach from already existing approaches.

We analyzed the extracted data for grouping the identified issues using the affinity diagram [13] process as mentioned in Section 2.3. We classified the found issues into 10 groups. Table 4 lists the groups of issues and the respective approaches which were developed to address those issues. We explain each group of the issues in the following paragraphs. Each group of issues is referred to with a code, e.g., IG-1 and IG-2.

Variability modeling (IG-1): The issues categorized in this group concern the ability of a modeling approach to satisfactorily capture, organize, and represent variability. The kinds of issues claimed to be addressed by various approaches are: deficiency of feature modeling, no uniform representation of variability throughout the entire lifecycle, inability to model the details about variation point for re-users to build variants, inability in describing complex dependencies and dependency interactions, lack of precision and formalization of variability modeling, deficiency in communicating variability to customers. Table 4 shows that this group of issues has motivated the creation of the largest number of VM approaches.

Identifying commonality and variability (IG-2): This group of issues represents a lack of systematic way of identifying commonality and variability. Practitioners have to depend on experience and intuition of domain experts to recognize commonality and variability [26]. How the results of commonality and variability analysis will satisfy an organization's high-level business goals is not directly shown [25].

Process support (IG-3): A general lack of explicit and systematic process support for managing variability is very important but not much raised issue.

Architecture (IG-4): It has been claimed that around 2002 the available VM approaches typically dealt with requirement level and early life cycle stages such as domain analysis, product line scoping without paying sufficient attention to the activities that come later in the life cycle [17, 23] such as designing and realizing product line architectures. Hence, new approaches were developed for VM at the architecture level.

Table 4: Issues by approaches

Issue Groups	Approaches
Variability Modelling (IG-1)	FODA, FORM, Ferber'02, Fey'02, CBFM, Brown'06, Becker'02, Capilla'02, RequiLine, CONSUL, Reiser'07, Ye'05, Muthig'02, Bachmann'04, COVAMOF, Schmid'04, VPM, Koalish, Schobbens'07, Kumbang
Identifying C&V (IG-2)	FODA, Moon'05, DRM
Process Support (IG -3)	FAST, SPLIT
Architecture (IG-4)	SPLIT, KobrA, Muthig'02, Thiel'02
Product derivation (IG-5)	Salicki'02, Jansen'04, COVAMOF, FDL, CBFM, Koalish, Krueger'02
Evolution of variability (IG-6)	FDL, Ye'05, Loesch'07
Tool support (IG-7)	RequiLine, CONSUL
Customizability (IG-8)	Schmid'04
Binding time (IG-9)	Hoek'04
Scalability (IG-10)	Ye'05, Reiser'07

Product derivation (IG-5): The issues in this group concern the lack of methodological and tool support for efficiently and effectively resolving variability to produce particular products. Table 4 shows that this group of issues has motivated the development of the second largest number of VM approaches we reviewed.

Evolution of variability (IG-6): Evolution of variability is a central issue. Some typical evolution scenarios include: adding variation points and variants, removing obsolete variation points and variants, changing relationships among variation points and variants. Despite being a vital group of issues, there have been only three VM approaches that have explicitly claimed to be motivated by these issues.

Tool support (IG-7): Variability modeling can become very complex in an industry setting, an adequate tool support is essential. However, the tool support for VM appears to be very weak when RequiLine and CONSUL were developed. However, it may not reflect the current situation of tooling support for VM.

Customizability (IG-8): This group of issues characterizes the importance of allowing an organization

to keep as many of the existing notations and approaches as possible to support VM [4].

Binding time (IG-9): The ability to resolve variability at any point in the lifecycle is required due to the presence of significantly increased and dissimilar levels of variability in software systems. However, the time at which a variability has be resolved after its introduction was limited to only a single point in the solutions available before Hoek'04 [15].

Scalability (IG-10): When the number of variation points, variants, and the variability relationships and interactions become very large and complex, the complexity of variability modeling may substantially increase to a level where it is uncontrollable. Therefore, scalability of variability modeling is a key issue that should be appropriately addressed [41].

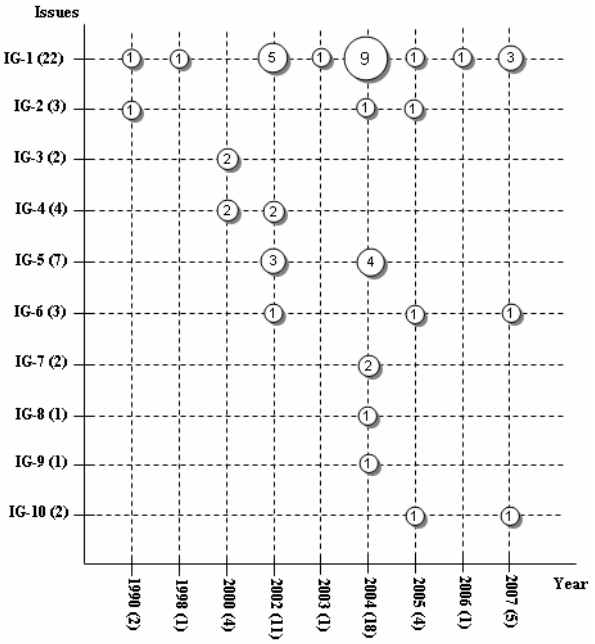

Figure 2: Issues vs. years

Our analysis has also revealed that the issues that have motivated the development of various approaches are time dependent. It is interesting to find that a factor that was an issue in 2000, may not be an issue in 2005 as it might have been solved by 2005 or might not have been an important issue anymore.

Figure 2 shows the groups of issues along the years of their appearance in bubble plot. The X axis represents year, and Y axis represents the groups of issues. The number in each parenthesis in front of each issue group represents the total number of approaches, appeared over the years, that claim to be motivated by the issues belonging to that group. The number in the bubble indicates the number of approaches whose creation was motivated by the group of issues (represented by the axis of "issues") in the year (represented by the axis of "year"). The size of a bubble is proportional to the number represented by the bubble. One study may address multiple issues belonging to multiple issue groups as the groups of issues are not mutually exclusive.

Figure 2 shows that variability modeling has attracted most of the attention. FODA was proposed to address the issues related to commonality and variabilities (i.e., called differences at that time) in 1990. More than a decade later, in 2004 and 2005, the C&V identification issues were again raised. These issues emphasized on rational C&V identification (i.e. the decision on C&V should be more rational and objective, rather than relying on experts' opinion). It is interesting to note that the issues of lack of systematic process for VM were raised only before 2002. The lack of systematic approaches to managing variability in architecture was also raised as an important issue to be addressed in 2002 and before. The issues regarding product derivation and evolution of variability were raised and addressed from 2002 until recently. The issues in the groups of customazability, tool support, binding time, scalability were raised as issues from 2004 onward. Only a few studies have addressed the issues in each of these 4 groups. We also observed that almost all the reviewed approaches described the issues related to variability modeling as one of the motivators for developing those approaches. This finding further provides the evidence that variability modeling is the most researched area in SPL, which is also a topic of workshops like VaMos nowadays. We also found that only two approaches claimed to be developed for addressing the issues regarding scalability of variability modeling. It is obvious that scalability related issues in VM have attracted very little attention. Hence, there is a need of paying sufficient attention to these issues.

3.5. Variability Models Used

Since variability modeling is the main area of research on variability management, our study also attempted to identify the kinds of variability models used in the reviewed approaches. It was not a surprise that fourteen approaches used feature models. Almost each of them had extended the initial feature model presented in FODA [14]. The decision modeling was used in six approaches. We also found that 12 other kinds of variability models were proposed by various approaches; however, no other approach used any of them. Though, it is not the claim of the developers of FAST and Loesh'07, our conclusion is that both approaches are independent of any particular way of modeling variability.

The use of feature modeling is evenly distributed over the years starting from 1990. However, the reporting of new variability models reached its peak in 2004 when 6

new models were proposed. The research on variability modeling appears to be concentrated into 2 main streams: extending traditional software development models that are in use in different life cycle stages, called integrated VM; separating the representation of variability from the representation of the various SPLE artifacts [22], called orthogonal VM. Halmans'03 [21] and VPM [10] are the examples of the first stream; while Bachmann'04 [22] and COVAMOF [33, 34] are the examples of the second stream. Each stream has its strengths and limitations. However, a comparative discussion on this topic is not within the scope of this paper.

3.6. Support for Lifecycle Phases

We also reviewed the VM approaches for their support during different phases of the SPLE lifecycle. We used the SPLE lifecycle phases described by Pohl et al. [47]. We also took the view that architecture phase is a part of design phase. We found that Requirements phase of domain engineering attracted the most attention as 23 approaches claimed to tackle VM in Requirements phase. It is followed by design phase. We did not review the papers dealing with variability implementation in this study. Only 1 approach, FAST [24], addressed the test phase.

Except the testing phase, which appears to attract relative little attention in both domain engineering and application engineering, all phases of application engineering have also received a lot less attention than their counterparts in domain engineering.

According to our analysis, only FAST, which mainly contributes to the process aspects of VM, can be considered covering the full life cycle phases, from Requirements engineering to testing. Some approaches, FORM [11], SPLIT [28], KobrA [29], Muthig'02 [23], Schmid'04 [4], CONSUL [40] and etc., also intended to provide comprehensive lifecycle coverage. However, our conclusion is that they do not address the testing phase specifically. One approach, Loesch'07 [27], does not address any particular phase of SPLE lifecycle. However, it appears to tackle the problems often happen in the maintenance phase.

4. Conclusions

In this paper, we present the results from a SR of VM research in SPL. We believe that the results provide interesting insights into the current status of VM research with respect to the historical background of different VM appraochs, the issues that motivated their creation, different variability models used by them, and their support for the different phases of SPLE. This SR has also identified certain gaps that need to be filled by future research. This SR has enabled us to make the following conclusions in order to highlight the areas, which need immediate attention by researchers and practitioners. We believe that more active collaboration between these two communities is expected to result in VM technologies, which would have higher potential of industrial adoption.

There is only little, if any, experimental or detailed comparative analysis to show the relative advantages and disadvantages of different VM approaches. That is why it would be hard to build an evidence-based guidance for selecting a VM approach for specific development situation and context. Hence, there is a vital need of conducting comparative analysis of different approaches in order to provide the practitioners with a qualified portfolio of techniques.

The reviewed VM approaches share significant number of commonalities. However, we have not found a reference model encompassing the large number of different approaches. We assert that there should be a reference model to support model transformation and future research in this area.

Only a few approaches tackle the issues regarding systematic process support for VM. The approaches that do provide process support such as FAST [24] and SPLIT [28] were mainly designed for SPLE in general. The process support specifically for VM appears to be a neglected area of research.

There are only 3 approaches, (FDL [35], Ye'05 [41] and Loesch'07 [27]), which are concerned with evolution of variability. However, we have found that these approaches provide very limited support for evolution of variability. Hence, we can conclude that a systematic approach to provide a comprehensive support for variability evolution is not available.

This SR also revealed that except Ye'05 [41] and Reiser'07 [44] no other VM approach has mentioned the scalability as an issue to be addressed. However, most of the current VM approaches have often been criticized for their inability to scale to large and complex product lines. Hence, researchers need to pay attention to the scalability aspects of VM approaches.

Among all the reviewed approaches, only one approach (FAST [24]) explicitly mentions testing. However, its treatment of testing is very limited as it only provides some strategies and suggestions instead of a systematic and concrete approach. Other quality assurance techniques like inspection and review were not mentioned at all in the reviewed approaches. However, the experiment by Denger and Kolb [48] found that traditional testing and inspection techniques used in single system development were ineffective in identifying variant-specific defects. The adaptation of these quality assurance techniques to effectively handle variability in SPLE is still a key challenge in this area, which needs to be addressed by future research.

5. Acknowledgements

This work is partly supported by Science Foundation Ireland under the grant no. 03/CE2/I303_1. We would also like to thank Prof. Kyo-Chul Kang for his valuable discussion with us on the topic covered in this paper.

6. References

[1] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*: Addison-Wesley, 2002.

[2] J. Bosch, *Design & Use of Software Architectures: Adopting and evolving a product-line approach*: Addison-Wesley, 2000.

[3] F. Bachmann and P. Clements, Variability in Software Product Lines, Software Engineering Institute, Pittsburgh, USA, Technical Report CMU/SEI-2005-TR-012, 2005.

[4] K. Schmid and I. John, A customizable approach to full lifecycle variability management, *Sci. Comput. Program.,* vol. 53, pp. 259-284, 2004.

[5] J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, J. Obbink, and K. Pohl, "Variability Issues in Software Product Lines," in *Softw Product-Family Eng (PFE-4)*: Springer, 2002, pp. 303-338.

[6] B. Kitchenham and S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Keele University, UK EBSE-2007-1, 2007.

[7] M. Sinnema and S. Deelstra, Classifying variability modeling techniques, *Information and Software Technology,* vol. 49, pp. 717-739, 2007.

[8] M. Svahnberg, J. van Gurp, and J. Bosch, A taxonomy of variability realization techniques, *Softw. Pract. Exper.,* vol. 35, pp. 705-754, Jul 2005.

[9] B. Kitchenham, E. Mendes, and G. Travassos, Cross versus within-Company Cost Estimation Studies: A Systematic Review, *IEEE Trans. on Softw Eng,* vol. 33, pp. 316-329, 2007.

[10] D. L. Webber and H. Gomaa, Modeling variability in software product lines with the variation point model, *Sci. Comput. Program.,* vol. 53, pp. 305-331, 2004.

[11] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, FORM: A feature-oriented reuse method with domain-specific reference architectures, *Annals of Software Engineering,* vol. 5, pp. 143-168, 1998.

[12] M. Jaring and J. Bosch, "Representing Variability in Software Product Lines: A Case Study," in *Software Product Lines (SPLC 2)*: Springer, 2002, pp. 219-245.

[13] H. Beyer and K. Holtzblatt, *Contextual design: defining customer-centered systems*: Morgan Kaufmann Publishers Inc., 1998.

[14] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study, SEI Technical Report 1990.

[15] A. van der Hoek, Design-time product line architectures for any-time variability, *Sci. Comput. Program.,* vol. 53, pp. 285-304, 2004.

[16] T. Asikainen, T. Soininen, and T. Männistö, "A Koala-Based Approach for Modelling and Deploying Configurable Software Product Families," in *Softw Product-Family Eng (PFE-5)*: Springer, 2004, pp. 225-249.

[17] S. Thiel and A. Hein, "Systematic Integration of Variability into Product Line Architecture Design," in *Softw Product Lines (SPLC2)*: Springer, 2002, pp. 67-102.

[18] C. Krueger, "Variation Management for Software Production Lines," in *Softw Product Lines (SPLC2)*: Springer, 2002, pp. 107-108.

[19] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch, "COVAMOF: A Framework for Modeling Variability in Software Product Families," in *Softw Product Lines (SPLC3)*: Springer, 2004, pp. 197-213.

[20] T. Asikainen, T. Männistö, and T. Soininen, Kumbang: A domain ontology for modelling variability in software product families, *Advanced Engineering Informatics,* vol. 21, pp. 23-40, 2007.

[21] G. Halmans and K. Pohl, Communicating the variability of a software-product family to customers, *Software and Systems Modeling,* vol. 2, pp. 15-36, 2003.

[22] F. Bachmann, M. Goedicke, J. Leite, R. Nord, K. Pohl, B. Ramesh, and A. Vilbig, "A Meta-model for Representing Variability in Product Family Development," in *Softw Product-Family Eng (PFE-5)*: Springer, 2004, pp. 66-80.

[23] D. Muthig and C. Atkinson, "Model-Driven Product Line Architectures," in *Softw Product Lines (SPLC2)*: Springer, 2002, pp. 79-90.

[24] M. Ardis, N. Daley, D. Hoffman, H. Siy, and D. Weiss, Software product lines: a case study, *Softw. Pract. Exper.,* vol. 30, pp. 825-847, 2000.

[25] S. Park, M. Kim, and V. Sugumaran, A scenario, goal and feature-oriented domain analysis approach for developing software product lines, *Industrial Management + Data Systems,* vol. 104, pp. 296-308, 2004.

[26] M. Moon, K. Yeom, and H. S. Chae, An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line, *IEEE Trans. on Softw Eng,* vol. 31, pp. 551-569, 2005.

[27] F. Loesch and E. Ploedereder, "Optimization of Variability in Software Product Lines," in *11th Int'l Softw Product Line Conf*, 2007, pp. 151-162.

[28] M. Coriat, J. Jourdan, and F. Boisbourdin, "The SPLIT method: building product lines for software-intensive systems," in *1st Int'l Softw Produt Line Conf* Denver, Colorado, United States: Kluwer Academic Publishers, 2000, pp. 147-166.

[29] C. Atkinson, J. Bayer, and D. Muthig, "Component-based product line development: the KobrA approach," in *1st Int'l Softw Produt Line Conf* Denver, Colorado, United States: Kluwer Academic Publishers, 2000, pp. 289-309.

[30] S. Ferber, J. Haag, and J. Savolainen, "Feature Interaction and Dependencies: Modeling Features for Reengineering a Legacy Product Line," in *Softw Product Lines (SPLC2)*: Springer, 2002, pp. 37-60.

[31] D. Fey, R. Fajta, and A. Boros, "Feature Modeling: A Meta-Model to Enhance Usability and Usefulness " in *Softw Product Lines (SPLC2)*: Springer, 2002, pp. 198-216.

[32] M. Becker, L. Geyer, A. Gilbert, and K. Becker, "Comprehensive Variability Modelling to Facilitate Efficient Variability Treatment," in *Softw Product-Family Eng (PFE-4)*: Springer, 2002, pp. 294-303.

[33] R. Capilla and J. Dueñas, "Modelling Variability with Features in Distributed Architectures," in *Softw Product-Family Eng (PFE-4)*: Springer, 2002, pp. 49-109.

[34] S. Salicki and N. Farcet, "Expression and Usage of the Variability in the Software Product Lines " in *Softw Product-Family Eng (PFE-4)*: Springer, 2002, pp. 173-210.

[35] A. van Deursen, M. de Jonge, and T. Kuipers, "Feature-Based Product Line Instantiation Using Source-Level Packages " in *Softw Product Lines (SPLC2)*: Springer, 2002, pp. 19-30.

[36] A. G. J. Jansen, R. Smedinga, J. van Gurp, and J. Bosch, First class feature abstractions for product derivation, *IEE Proc. - Software,* vol. 151, pp. 187-197, 2004.

[37] M. Sinnema and S. Deelstra, Industrial validation of COVAMOF, *Journal of Systems and Software,* vol. 81, pp. 584-600, 2008.

[38] T. von der Maßen and H. Lichter, "RequiLine: A Requirements Engineering Tool for Software Product Lines," in *Softw Product-Family Eng (PFE-5)*: Springer, 2004, pp. 168-180.

[39] K. Czarnecki, S. Helsen, and U. Eisenecker, "Staged Configuration Using Feature Models " in *Softw Product Lines (SPLC3)*: Springer, 2004, pp. 266-283.

[40] D. Beuche, H. Papajewski, and W. Schroder-Preikschat, Variability management with feature models, *Sci. Comput. Program.,* vol. 53, pp. 333-352, 2004.

[41] H. Ye and H. Liu, Approach to modelling feature variability and dependencies in software product lines, *IEE Proc. - Software,* vol. 152, pp. 101-109, 2005.

[42] T. J. Brown, R. Gawley, R. Bashroush, I. Spence, P. Kilpatrick, and C. Gillan, "Weaving behavior into feature models for embedded system families," in *10th Int'l Softw Product Line Conf*, 2006, pp. 52-61.

[43] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, and Y. Bontemps, Generic semantics of feature diagrams, *Computer Networks,* vol. 51, pp. 456-479, 2007.

[44] M.-O. Reiser and M. Weber, Multi-level feature trees: A pragmatic approach to managing highly complex product families, *Requir. Eng.,* vol. 12, pp. 57-75, 2007.

[45] R. v. Ommering, F. v. d. Linden, J. Kramer, and J. Magee, The Koala Component Model for Consumer Electronics Software, *Computer,* vol. 33, pp. 78-85, 2000.

[46] T. Soininen, J. Tiihonen, T. Männistö, and R. Sulonen, Towards a general ontology of configuration, *Artif. Intell. Eng. Des. Anal. Manuf.,* vol. 12, pp. 357-372, 1998.

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

[48] C. Denger and R. Kolb, "Testing and inspecting reusable product line components: first empirical results," in *2006 ACM/IEEE Int'l Symposium on Empirical Softw Eng* Rio de Janeiro, Brazil: ACM, 2006.