A

Separation of Concerns in Feature Diagram Languages: A Systematic Survey

ARNAUD HUBAUX, University of Namur, Belgium
THEIN THAN TUN, The Open University, UK
PATRICK HEYMANS, University of Namur, Belgium

The need for flexible customisation of large feature-rich software systems, according to requirements of various stakeholders, has become an important problem in software development. Among the many software engineering approaches dealing with variability management, the notion of Software Product Line (SPL) has emerged as a major unifying concept. Drawing from established disciplines of manufacturing, SPL approaches aim to design repertoires of software artefacts, from which customised software systems for specific stakeholder requirements can be developed. A major difficulty SPL approaches attempt to address is the modularisation of software artefacts, which reconciles the user's needs for certain features, and the development and technical constraints. Towards this end, many SPL approaches use feature diagrams to describe possible configurations of a feature set. There have been several proposals for feature diagram languages with varying degrees of expressiveness, intuitiveness, and precision. However, these feature diagram languages have limited scalability when applied to realistic software systems. This paper provides a systematic survey of various concerns of feature diagrams and ways in which concerns have been separated. The survey shows how the uncertainty in the purpose of feature diagram languages creates both conceptual and practical limitations to scalability of those languages.

Categories and Subject Descriptors: A.1 [**General Literature**]: Introductory and Survey; D.2.13 [**Reusable Software**]: Domain engineering; D.2.13 [**Reusable Software**]: Reuse models

General Terms: Design, Documentation

Additional Key Words and Phrases: Software Product Line, Feature Diagram, Separation of Concerns, Variability

1. INTRODUCTION

First introduced by Kang et al. [1990], a *feature diagram* (FD) is a graphical representation of similarities and differences in a family of software systems, called a software product line (SPL). Similarities and differences in an SPL are expressed in terms of features. In particular, an FD defines a hierarchy of features and constraints on their selection. The valid combinations of features define the *products*, also called *configurations*, provided by the SPL. In other words, each product is a valid instance of the FD. Traditionally, FDs are graphically represented as trees where nodes denote features and edges represent top-down hierarchical decomposition of features.

Figure 1 presents an example of graphical tree-shaped FD of an eVoting SPL. The *and*-decomposition of the root feature *Voting* implies that all its sub-features, namely, *Encode*, *VoteValues* and *Default VoteValue*, have to be selected in a valid product. Sim-

Author's address: A. Hubaux (ahu@info.fundp.ac.be), PReCISE Research Centre, Faculty of Computer Science, University of Namur, Belgium; T. T. Tun (t.t.tun@open.ac.uk) The Open University, United Kingdom; P. Heymans (phe@info.fundp.ac.be), PReCISE Research Centre, Faculty of Computer Science, University of Namur, Belgium;

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

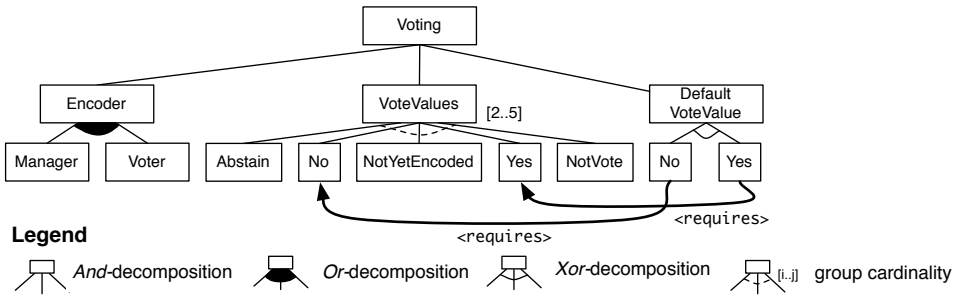

Fig. 1. Sample FD of the PloneMeeting eVoting component.

ilarly, the *or*-decomposition of the *Encoder* feature means that at least one of its child features, namely, *Manager* and *Vote*, has to be selected. The *xor*-decomposition of the *Default Vote Value* means that one and only one of its sub-features (*No* and *Yes*) must be selected. Cardinality-based decompositions are often used, such as for *VoteValues* in our example. $[2..5]$ indicates that at least two, and at most five, of the child features of *VoteValues* should be selected in a valid product. Additionally, the $<$*requires*$>$ links in the diagram denote constraints. For instance, the selection of *No* under *Default vote values* requires the selection of *No* under *Vote Values*. FDs also support $<$*excludes*$>$ links (not illustrated here), which denote the mutual exclusion.

This particular view of software systems having a core set of common features, and differing only in limited ways, is a powerful conceptual tool to characterise many of the products aimed at the mass market. The apparent simplicity of FDs in describing feature relationships, and their suitability as a communication device for stakeholders, are often cited as an important reason for their popularity [Kang et al. 1998]. As a result, there has been much academic and practical interest in designing precise and expressive FD languages, automated reasoning tools to support the languages, and their application to real-life problems [Steger et al. 2004; Jensen 2007; Dordowsky and Hipp 2009; Wenzel et al. 2009].

As FDs grow, a recurrent problem begins to emerge: They have limited scalability [SCA 2009]. When describing an SPL with a realistic number of features, typically in their hundreds and even thousands [Berger et al. 2010], the diagrams often become too large for human cognitive abilities, too imprecise for automated reasoning, and too overloaded to be purposeful. Evidence for limited scalability of FDs comes from practice (difficulties in creating, updating, and interpreting FDs) [Reiser and Weber 2006], and from academic research (difficulties in reasoning about large FDs) [Schobbens et al. 2006].

Recognising this limitation, many SPL approaches have proposed different ways to separate concerns, for instance, by splitting the FD according to the stakeholders, the development stages, or runtime dependency. Although the need for separation of concerns in feature modelling is recognised, there is no consensus about what the main concerns of FDs are and how these concerns may be managed. Two issues are addressed in this paper. First, what are the important concerns that should be recognised and separated in FDs? For instance, FDs may be used to describe design options, choices of user functionality, and legal constraints. There are many other legitimate concerns that may be taken into account in FDs. This paper provides a list of possible concerns for FD languages.

Second, what guidelines are provided by the existing feature diagramming techniques for achieving separation of concerns in FDs? Separated concerns may also in-

teract and overlap with each other, therefore the question of how to make sense of separated concerns is also important. This paper provides a discussion of separation of concerns techniques in FD languages.

In addition to these two issues, we will examine the level of formality involved in the FD language and the (possibility of) tool support for achieving separation of concerns.

Other surveys have already studied different aspects of FDs. Schobbens et al. [2006] survey their formal semantics. Chen and Babar [2009] examine the extent to which scalability is addressed by feature modelling approaches. Chen et al. [2009] also provide a systematic review of variability management approaches in software product line engineering. Finally, Benavides et al. [2010] conduct a literature review that covers twenty years of automated analysis on FDs. Our survey is different in the sense that we systematically review concerns and their separation in FD languages.

The rest of the paper is organised as follows. Section 2 discusses the research method used to elicit concerns and their separation techniques in FD languages. Section 3 reports on the execution of the research method. Sections 4 to 7 summarise our findings for each of the research questions formulated in Section 2.1. Section 8 discusses the findings of the survey and threats to its validity (Section 9). Concluding remarks can be found in Section 10.

2. RESEARCH METHOD

The research method we have followed to collect and review papers is inspired by the guidelines for performing a literature review proposed by Kitchenham [2004]. Her method provides systematic ways to identify, evaluate and interpret research data. Our method deviates from Kitchenham's in that we intentionally leave out a detailed quantitative *meta-analysis* to favour an in-depth qualitative analysis, primarily because the data we are handling is largely non-statistical.

This section begins with the presentation of our research questions followed by the description of the survey protocol. It then details the survey material and the data collection forms we used to harvest data systematically.

2.1. Research Questions

This survey addresses four questions. The first two questions focus on the elicitation of concerns, their separation and composition. The last two questions evaluate the degree of formality and the tool support that is provided. They are formulated as follows.

RQ1. *What are the main concerns of FDs?* We expect there are several FD languages with different notions of "concerns". We will define what is meant by concern in the context of this survey, and list those concerns.

RQ2. *How are concerns separated and composed?* Not only that there are different ways of separating concerns, there may be different ways of composing concerns too. The survey will cover those as well.

RQ3. *What is the degree of formality?* In order to discover whether separation and composition techniques are amenable to automated processing, we will have to assess the formalism used to define them.

RQ4. *Is there (an opportunity for) tool support?* Separation of concerns in FDs is not only a conceptual problem, but a practical problem. Part of the solution is the degree to which automation is available and possible when managing concerns in FDs.

2.2. Survey protocol

The survey protocol is divided into four main steps that go from the selection of papers to their analysis. This process is depicted in Figure 2. Starting from the top of the

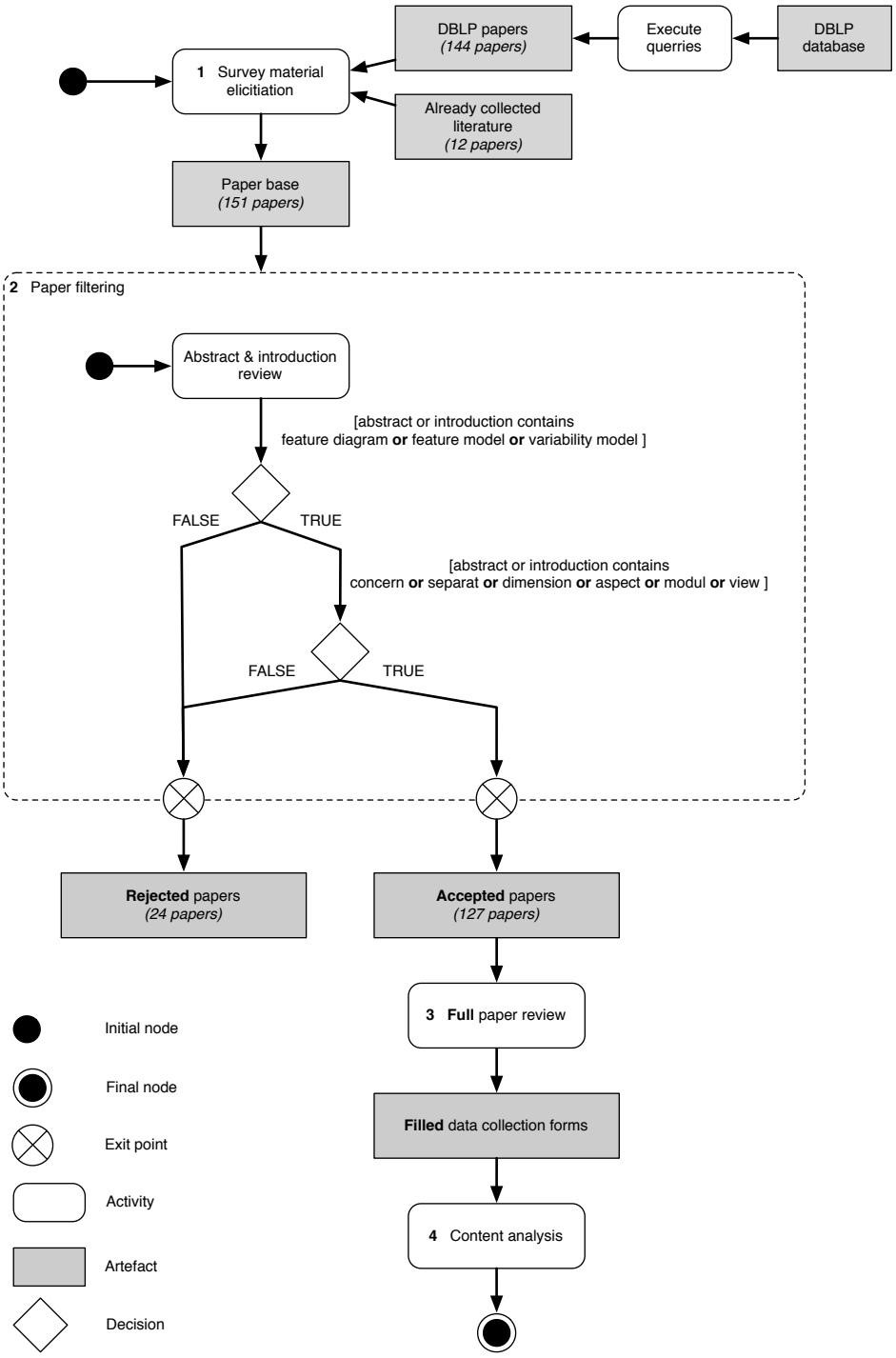

Fig. 2. Survey protocol

diagram, the survey material is composed of the whole set of papers indexed by DBLP [1] and papers that we know are highly relevant to the research questions. We detail in Section 2.3 how the material was searched and papers selected.

The second step is the filtering process during which papers are kept for a complete review. The filtering is based on the search for keywords in the abstract and introduction. In essence, papers that do not refer to feature modelling and separation of concerns in general are discarded.

The third step consists in the complete review of the remaining papers. As advocated in [Kitchenham 2004], we define data collection forms, discussed in Section 2.4, to systematise this task. The paper review was split between the first two authors of this paper. Their task was to review and fill out the forms for every paper on their list. The final step is a qualitative analysis meant to answer our four research questions.

Our research method combines the completeness of automated search with the reliability of manual reviews. This combined approach helps us minimize the subjectivity and human effort involved. Reliance on only one of the two techniques can have major weaknesses. For instance, the fully automated search by Chen *et al.* [Chen and Babar 2009] initially missed important publications such as FORM [Kang et al. 1998] because the terms did not match.

2.3. Survey Material

Our survey is restricted to (1) research papers published in peer-reviewed workshops, conferences and journals, (2) books and other manuscripts, such as technical reports, cited by the peer-reviewed papers, and (3) material accompanying commercial and non-commercial automated tools.

We considered two input sources for our survey: The DBLP computer science bibliography and a collection of papers the authors have amassed in their research on SPL over several years. The search on DBLP was executed through the following five queries on the titles of the papers:

Q1. `program*.famil*`
Q2. `software*.famil*`
Q3. `product*.famil* concern*|separat*|dimension*|aspect*|modul*|view*`
Q4. `product*.line* concern*|separat*|dimension*|aspect*|modul*|view*`
Q5. `feature*|variabil* concern*|separat*|dimension*|aspect*|modul*|view*`

The `.` makes two strings undividable, meaning that only white spaces can separate the left-hand side from the right-hand side. The `*` is the classical wild card substituting any sequence of non-space characters. Finally, the `|` denotes the boolean *or* and the white space the boolean *and*. The first query is to find papers with titles containing terms such as program family, program families, and programs families. The second query is to find papers with titles containing terms such as software family and software families. The third query is to find papers with titles containing terms such as product families and concern, or product families and separation, or product families and dimension. The fourth and fifth queries are similar to the third, but uses terms such as product line and feature variability, instead of product families.

During sample tests of the queries, the version of *Q5* presented above returned more than 600 papers, many of which are false positives. To reduce this number, we automatically excluded all the papers that were not published at one of the venues returned by the first four queries. This way, we reduced the number of papers from $600+$ to 69.

Research question 1		
Field	**Description**	**Possible values**
Concern	Concerns addressed in the paper	Free form text
Stakeholders	Stakeholders targeted by the concerns	Free form text
Artefacts	Artefacts targeted by the concerns	Free form text
Development stages/tasks	Stages in the software development cycle targeted by the concerns	Free form text
Scope	The scope of the concern determines to what extent the concern is related to the software itself or its environment	*some of* {internal, external}
Target domain	Type of domain to which the concern applies	Free form text
Rationale	Motivation for the need for the concerns	Free form text
Evaluation	Type of evaluation conducted to prove the relevance of the concerns	*some of* {illustration, controlled experiment, case studies, survey research, ethnographies, action research, mixed approach}

Research question 2		
Field	**Description**	**Possible values**
Separation techniques	Techniques used to identify and separate concerns in a FD	Free form text
Composition techniques	Techniques used to compose the concerns of a FD	Free form text

Research question 3		
Field	**Description**	**Possible values**
Formalised artefacts	Artefacts discussed and formalised in the paper	Free form text
Level of formality	Level of formality of the notation used	*one of* { ⊞, **+**, ◯, **-**, ⊟ }
Means of formalisation	Type of formalisation technique used	Free form text

Research question 4		
Field	**Description**	**Possible values**
Supported activities	Activities supported by the tool	Free form text
Implementation techniques	Techniques used to implement the activities	Free form text
Degree of automation	To what extent the activity is/can be automated	*one of* {**+**, ◯, **-**}
Implementation Status	Level of maturity of the implementation	*one of* {mature, prototype, proof of concept}

Fig. 3. Data collection forms

2.4. Data Collection Forms

To systematically collect data answering the research questions, we created four data collection forms. To evaluate their clarity and completeness prior to the beginning of the actual review, we conducted a pilot test, as recommended in [Kitchenham 2004]. Five papers were thus randomly selected and assigned to the two researchers. Their task was to fill out the forms and report on their fitness. The adapted forms used for the survey are presented in Figure 3. For each field, we provide a short description and the type of expected input. The main result, after some iterations of the form, was that both researchers were producing similar reports on the five papers.

[1]See `http://www.informatik.uni-trier.de/~ley/db/`.

The possible values for the evaluation field in **RQ1** are taken from Shull et al. [2007]. We added the illustration criterion to account for cases where the authors do not explicitly characterise the type of validation they performed. In the level of formality in **RQ3**, ✚ means that a complete formal semantics is provided. **+** means that a formal semantics is provided but some aspects are still informally defined. ◯ means that only the abstract syntax is defined by a meta-model or formal grammar. **-** means that only the concrete syntax is introduced. ▬ means that only informal descriptions are proposed. Similarly in **RQ4**, **+** means that the activity is fully automated. ◯ means that the automation of the activity is partial and still requires manual processing. **-** means that no automation is provided.

3. EXECUTION

We now report on the execution of the paper elicitation process. First we focus on the results returned by the queries, and then discuss the total number of papers sorted by venues.

The five queries were executed on DBLP database on 13 December 2010, using *CompleteSearch* developed by Bast [2010]. We developed shell scripts to automatically extract papers and merge the results of the various queries. The aggregate result of five queries is a total of 144 papers.

Of these 144 papers, 122 papers appeared in proceedings and 22 in journals. Table I synthesizes the distribution of the papers by events and journals. We removed 5 papers that are clearly irrelevant to our survey, and added 12 papers that we know are relevant but whose titles were not matched by the queries. This raised the count of papers to 151. Our qualitative analysis is based on the remaining 151 papers.

Table I. Distribution of papers by events and journals.

Event	**# Papers / Event**
SPLC	21
VAMOS	10
ICSE, SAC	7
GPCE, SEKE	6
ICSR, MODELS, OOPSLA	5
AOSD, HICSS	4
APSEC, COMPSAC, ECBS, PFE, RE	3
ARES, DAGSTUHL, ICSOFT, PROLAMAT	2
ACISICIS, CSSE, ECOOPW, EMISA, GCSE, ICAISC, ICECCS, ISPE, ISPW, IWSAPF, MKWI, REFSQ, SERA, SERP, SIGSOFT, UML, VVEIS, WCRE, WICSA	1
Journal	**# Papers / Journal**
TAOSD	5
SIGSOFT, TSE, JUCS, EOR	2
CONCURRENCY, IBIS, IEE, IJMR, IJSEKE, JSS, RE, SCP, SOPR, SPIP, TLSDKCS	1

From the 151 papers, 15 were not included because they did not match our criteria. 9 more were discarded because they were extended by longer papers. Our qualitative analysis is based on the remaining 127 papers. Sections 4 to 7 build upon these papers to respectively answer **RQ1** to **RQ4**.

4. CONCERNS

Two types of concerns are proposed in the literature: (1) concerns that group features together according to some given criteria; and (2) concerns that differentiate relationships between features.

4.1. Feature groups

Concerns grouping features can be further separated into three categories, as reported in Table II.

Table II. Concerns separating groups of features.

References	Concern
Functional and non-functional property	
[Hallsteinsen et al. 2006]	Adaptivity
[Thompson and Heimdahl 2003]	Fault tolerance
[Colyer et al. 2004]	Flexibility
[Etxeberria and Mendieta 2008; Hallsteinsen et al. 2006]	Functionality
[Etxeberria and Mendieta 2008]	Quality
Facet	
[Saleh and Gomaa 2005a; Elsner et al. 2008]	Artefacts
[Choi et al. 2009]	Binding
[Kang et al. 1990; Kang et al. 1998; Lee et al. 2002; Kang et al. 2002]	Capability
[Choi et al. 2009]	Configuration
[Kang et al. 1990; Kang et al. 1998; Lee et al. 2002; Kang et al. 2002; Classen et al. 2008; Hartmann and Trew 2008; Tun et al. 2009; Moreira et al. 2005]	Context
[Savolainen and Kuusela 2001]	Declarative
[Savolainen and Kuusela 2001]	Deductive
[Kang et al. 1990; Kang et al. 1998; Lee et al. 2002; Kang et al. 2002]	Domain technology
[Pohl et al. 2005]	External variability
[Thompson and Heimdahl 2003]	Hardware platform
[Kang et al. 1990; Kang et al. 1998; Lee et al. 2002; Kang et al. 2002]	Implementation techniques
[Pohl et al. 2005]	Internal variability
[Grünbacher et al. 2009]	Market needs
[Hartmann and Trew 2008; Grünbacher et al. 2009; Reiser and Weber 2006]	Multiple product lines
[Fey et al. 2002; Schobbens et al. 2006]	Non-primitive feature
[Choi et al. 2009]	Operational dependency
[Reiser and Weber 2007; Mannion et al. 2009; Grünbacher et al. 2009]	Organisational structure
[Deelstra et al. 2009; Metzger et al. 2007; Moreira et al. 2005; Kircher et al. 2006; Classen et al. 2008; Tun et al. 2009; Grünbacher et al. 2009]	Provided variability (solution space)
[Deelstra et al. 2009; Metzger et al. 2007; Kircher et al. 2006; Tun et al. 2009; Thompson and Heimdahl 2003; Classen et al. 2008]	Required variability (problem space)
[Choi et al. 2009]	Structure
[Choi et al. 2009]	Traceability
[Pohl et al. 2005]	Variability in time
[Pohl et al. 2005]	Variability in space
Configuration process	
[Kang et al. 1990; Lee et al. 2002; Kang et al. 2002; Lee et al. 2004; Fey et al. 2002; Hubaux et al. 2008; Schmid and Eichelberger 2008]	Binding time
[Lee et al. 2009]	Binding state
[Czarnecki et al. 2005; Classen et al. 2009]	Configuration levels
[Czarnecki et al. 2004]	Configuration stages
[Hubaux et al. 2009]	Configuration tasks

4.1.1. Functional and non-functional property. Some authors represent concerns that group features according to non-functional, a.k.a. *quality*, aspects such as *adaptivity*,

fault tolerance, *flexibility* or arbitrary *qualities* that the system possesses [Hallsteinsen et al. 2006; Colyer et al. 2004]. Others group them by *functional* requirements provided by the system [Etxeberria and Mendieta 2008; Hallsteinsen et al. 2006].

4.1.2. Facet. Several approaches group features according to the artefacts of which they model the variability. Some of these approaches directly embed variability into artefacts such as UML models [Ziadi et al. 2004; Gomaa and Shin 2008; Heidenreich et al. 2010]. These approaches do not consider variability models as first-class citizens, in the sense that variability is not the main concept that drives the construction of models and other artefacts. Instead, they treat variability as an additional concept in UML models. Conversely, other approaches handle the variability model and base model separately. For instance, orthogonal variability modelling (OVM) [Pohl et al. 2005] models variation points individually and links these to base models. Czarnecki *et al.* [Czarnecki et al. 2005; Czarnecki and Pietroszek 2006] separate FDs from the base UML model, Classen et al. [2010] formally relate an FD to a behavioural model and reason about them both, and Heidenreich et al. [2010] map FDs to other models such as use cases, class diagrams or statecharts. Our work focuses only on approaches that recognize FDs as first class citizens. In this context, a concern is defined as the mapping between the FD and a set of models.

Choi et al. [2009] divide FDs according to five views. By views, they mean ways to characterise extensions to FD languages as opposed to describing what the main concerns of FDs are. In the *structure view*, feature relationships such as aggregation and generalisation are described. In the *configuration view*, mandatory, optional and alternative features, together with feature cardinality, and group cardinality are described. In the *binding view*, feature binding units representing groups of features bound together are described. In the *operational dependency view* the interactions between features are addressed through the identification of dynamic dependencies among them. In the *traceability view*, the implemented-by relationships between features are shown. These views are, in a sense, particular to concerns of developers. Pohl et al. [2005] distinguish between *external variability*—relevant to customers—and *internal variability*—relevant to developers. A similar distinction is also made by several other authors when recognizing that there are two kinds of variability: *Product line variability* and *software variability*. The former is concerned with the *"ability of a software system or artefact to be efficiently extended, changed, customized or configured for use in a particular context"* [Svahnberg et al. 2005]. The latter is concerned with *"the variation between the systems that belong to an SPL in terms of properties and qualities, like features that are provided or requirements that are fulfilled"* [Metzger et al. 2007].

Furthermore, in addition to human agents, such as customers, a number of environmental factors may also affect variability. The *context* concern, or the operating environment concern, addresses attributes of the environment in which the application is used (e.g., geographic region or legal system) that determine implicit relationships between features. The *domain technologies* gather features related to a technology that is specific to a given domain (such as aviation). Features in this concern are hardly reusable from one domain to the other. In contrast, the *implementation technologies* layer addresses technologies that are not specific to a particular domain (e.g., the computer realm). For instance, this category points to features used to describe display format or database technologies. Business and management concerns also drive the decomposition of the variability along the *market* needs as well as business constraints [Grünbacher et al. 2009]. In collaborative development environments, view mechanisms [Hubaux et al. 2011] can be used to create projections of FDs that are specific to individual development teams. At a fundamental level, some versions of the

semantics of FDs (e.g. [Schobbens et al. 2007]) add dummy features whose presence is required for formal processing. These features, being a technicality, are usually not shown to the users.

Some feature variability is specific to system properties. Savolainen and Kuusela [2001] distinguish between *deductive* and *declarative* properties. *Deductive* properties regroup the functionalities of the system as extracted from the requirements. *Declarative* properties, on the other hand, determine constraints over these requirements (e.g., optional vs. mandatory vs. alternative choices). These latter are part of the domain knowledge that constrains the design space. Thompson and Heimdahl [2003], echoing Lutz [2008], have discussed properties related to *hardware platform* and *fault tolerance*.

Finally, since software systems are likely to change over time, FDs may also have to evolve. Pohl et al. [2005] differentiate between *variability in time*—denoting changes to artefacts over time—and *variability in space*—denoting static variability of artefacts.

4.1.3. Configuration process. A major concern during the configuration process is the time at which a variant is bound, i.e., selected. The list of possible binding times varies from one project to the other and often depends on the technologies that implement the SPL. They typically include *design-time*, *compile-time*, *load-time* and *run-time*. Lee et al. [2009] provide a further analysis of feature-binding time analysis. They add the feature *binding state* view that determines when a given feature can be bound; it includes *inclusion*, *modification*, and *activation* rule states.

Czarnecki et al. [2004] and Czarnecki et al. [2005] propose a suite of *staged configuration* approaches where FDs are specialised in a stepwise fashion, and instantiated according to the stakeholders' interests at each development stage. With *specialisation* they refer to a process in which variation points in FDs are removed. In other words, a more specialised FD has fewer variation points than its originating FD. A fully specialised FD has no variability. A *configuration*, on the other hand, is an *instantiation* of an FD. Staged configuration has been formalised in the dynamic semantics of FDs by Classen et al. [2009].

With *multi-level* staged configuration, Czarnecki et al. [2005] refer to a sequential process in which an FD is configured and specialised alternately by stakeholders in the development stages. For instance, a stakeholder will instantiate an FD by selecting features that are relevant to its requirements. The instance of the model is then used to specialise the FD by removing parts of the model that are no longer available. The resulting FD is then instantiated by another stakeholder, and so the process repeats itself.

Hubaux et al. [2009] generalize multi-level staged configuration and propose a combined formalism that maps views on a FD [Hubaux et al. 2010b] to *tasks in a workflow* that models the configuration process. They thus achieve better separation of concerns between the configuration process and the model. Unlike multi-level staged configuration, they also allow complex processes to be modelled. In the same vein, [White et al. 2009] describes a model of configuration steps.

4.2. Feature relationships

The second type of concern determines the nature of the relationship between two or more features. Table III collects the different kinds of relationships we accumulated. Relationships defined as propositional formulas [Batory 2005] that are not explicitly qualified are left out of this table.

Cho et al. [2008] identify, based on [Lee and Kang 2004], several feature relations and feature dependencies. These are: *Aggregation* relationship, *generalisation* relationship, *required configuration* dependency, *excluded configuration* dependency, *usage*

Table III. Concerns separating relationship among features.

References	Concern
[Cho et al. 2008]	Aggregation relationship
[Lee et al. 2002]	Composed-of
[Cho et al. 2008]	Concurrent activation dependency
[Fey et al. 2002]	Conflict
[Cho et al. 2008]	Excluded configuration dependency
[Cho et al. 2008]	Exclusive activation dependency
[Classen et al. 2008; Tun et al. 2009]	Entailment
[Lee et al. 2002; Cho et al. 2008]	Generalisation/specialisation
[Czarnecki et al. 2008]	Hard constraint
[Lee et al. 2002]	Implemented by
[Chen et al. 2006a]	Interaction
[Cho et al. 2008]	Modification dependency
[White et al. 2009]	Point configuration constraint
[Fey et al. 2002]	Provided by
[Tun et al. 2009]	Quantitative constraints
[Cho et al. 2008]	Required configuration dependency
[Cho et al. 2008]	Sequential activation dependency
[Czarnecki et al. 2008]	Soft constraint
[Cho et al. 2008]	Subordinate activation dependency
[Cho et al. 2008]	Usage dependency
[Chen et al. 2006a]	Weaving
[Metzger et al. 2007]	X-links (i.e., cross-links)

dependency, *modification* dependency, *exclusive activation* dependency, *subordinate activation* dependency, *concurrent activation* dependency, and *sequential activation* dependency. The *conflict* and *provided-by* relationships proposed by Fey et al. [2002] are similar to the original *excludes* and *requires* constraints defined in FODA.

Chen et al. [2006a] divide requirements into candidate features for crosscutting requirements and identify two types of feature dependencies, called *interaction* and *weaving*. Interactions are derived from the flows between responsibilities (e.g., methods) included in different containers (e.g. classes). Weaving refers to the flows between responsibilities inside the same container.

Lee et al. [2002] use *composed-of*, *generalisation*/*specialisation* and *implemented-by* relationships. This indicates that FDs also show the refinement relationships between artefacts.

Czarnecki et al. [2008] propose using probabilistic FDs in the context of mining FDs from existing systems. In this approach, both *hard constraints* (constraints imposed by the model) and *soft constraints* (conditional probabilities on the selection of features) can be expressed and reasoned about.

[Classen et al. 2008] use an essential relationship that must hold between the components of a requirements document: The logical *entailment*. They build upon the work of Jackson, Zave and others [Jackson 1995; Zave and Jackson 1997; Gunter et al. 2000] to give a precise definition of a feature as a triple composed of *requirement* (R), *specification* (S) and *domain assumptions* (W) such that S and W entail R: $S, D \vdash R$. It means that for a set of assumptions W on the application domain, and a system S, the requirements R must be satisfied. Tun et al. [2009] further constrain the relationship between S, D and R with *quantitative constraints* in order to generate configurations that satisfy both the elicited requirements and other quantitative constraints (e.g., maximum memory footprint).

X-links denote cross-links that capture constraints between the provided and required provided variability.

Point configuration constraints were introduced by White et al. [2009] to denote constraints external to the FD that, for instance, require a given feature to be selected at

all times (e.g., safety constraint). They determine which configurations are allowed at a given point in time.

Here again the concept of relationship is rather imprecise. It is alternatively used to specify the parent/child relationship (e.g., aggregation or composition), to constraint feature selection (e.g., requires or excludes), and to provide dependency information (e.g., implemented by or usage). Furthermore, the semantics of these relationships is for the most part completely overlooked.

However, implementation-specific relationships differ from implementation specific feature groupings. Relationships restrict the selection by enforcing constraints imposed by business or technological decisions. Groups present domain options that are related to each other. Even though concerns can focus on implementation-specific features, they only expose choices and do not import extraneous constraints from the implementation layer.

5. SEPARATION AND COMPOSITION TECHNIQUES

Techniques for separating and composing concerns are often closely related. However, for consistency of presentation, they will be discussed separately here.

5.1. Techniques for Separating Concerns

In the same way that we have many possible concerns of FDs, there are also many ways to separate concerns including aspect-oriented, view-oriented, tabular, visualisation, staged-configuration, requirements-oriented approaches. Many of these approaches are summarised in Table IV.

Although aspect-oriented programming was initially proposed as a mechanism for separating concerns in program code, many authors have argued that the idea can be extended to analytical models of software systems, including FDs. The granularity of these *aspects* ranges from aspects at the code level [Saleh and Gomaa 2005a] to modelling artefacts [Noda and Kishi 2008]. The separation techniques are as diverse as the elements involved in the separation.

Table IV. Separation techniques identified in this survey.

References	Separation technique
[Saleh and Gomaa 2005a; Lee et al. 2006; Noda and Kishi 2008; Colyer et al. 2004; Elsner et al. 2008]	Aspect
[Mendonça et al. 2008]	Configuration spaces
[Hubaux et al. 2010b]	Configuration views
[Kang et al. 1990; Kang et al. 1998; Lee et al. 2002; Kang et al. 2002; Thompson and Heimdahl 2003; Reiser and Weber 2006; 2007; Mannion et al. 2009]	Hierarchical layering
[Czarnecki et al. 2005; Classen et al. 2009]	Level
[Thompson and Heimdahl 2003]	n-Dimensions
[Batory et al. 2003]	Origami matrix
[Czarnecki et al. 2005; Classen et al. 2009]	Stage
[White et al. 2009]	Step
[Hubaux et al. 2009]	Task

Batory et al. [2003] transpose an approach for multi-dimensional separation of concerns [Tarr et al. 1999] in SPLs, which recognises that features may be partitioned in a number of ways (dimensions) and calls the results of each partitioning *units*[2]. For instance, the features of a text editor are regarded as a dimension and the characteristics of the supported languages are another. Another example would have two classes in

[2]Although this approach is focused on features, rather than FDs, the way concerns are separated is of interest to this survey.

the first dimension—a singly-linked list and a doubly-linked list—and the operations in the second dimension—insert and delete operations. They propose using the *origami matrix* for describing the relationships between dimensions. Batory et al. [2003] make two important claims about this approach: (1) it prevents possible invalid combinations of elements; for instance, it does not permit the selection of a doubly-linked list with operations for singly-linked, because folding always has to happen between rows or columns, and not between cells, (2) complexity of n dimensions can be reduced to the complexity of one dimension by folding them.

As discussed in Section 4.1, the concerns for configuration *levels*, *stages*, *steps*, and *tasks* also come with their separation mechanisms. Hubaux et al. [2010b] propose a more generic technique for separation that includes them under the concept of *view*. A view is a simplified representation of an FD that has been tailored for a specific stakeholder, role, task, or, to generalize, a particular combination of these elements.

The vertical separation of concerns in FDs (e.g., from requirements to implementation decisions) corresponds to the generic notion of *hierarchical layering*. Thompson and Heimdahl [2003] use general *dimensions* that can capture both the operational decomposition (e.g., hardware platform) and quality aspects of the system (e.g., fault tolerance).

Having a diversity of approaches in itself is an advantage. However, since the purposes and concerns of FDs are not clear, the concerns to which these techniques are to be applied remain a question. What is required, in our view, is a clarification in the purposes of FDs, a systematic way to separate, and compose FDs, according to well-defined criteria. The Jackson-Zave framework (and related) for requirements engineering gives a systematic way to separate requirements, specifications and domain properties. We believe that FD languages will benefit such a framework. For instance, in our work in [Classen et al. 2008; Tun et al. 2009], we have separated variability in requirements from variability in the system context and from the software itself. There are clear criteria for separation and relating properties in requirements, system context and the software, which has been formally defined by [Gunter et al. 2000].

5.2. Techniques for Composing Concerns

Separation of concerns is only one side of a two-sided story: Having separated concerns, there is a need to reason about how concerns might be related and negotiated. These techniques are summarised in Table V.

Table V. Composition techniques identified in this survey.

References	Composition technique
[Batory et al. 2003]	Cartesian combination
[Mendonça et al. 2008]	Configuration plan
[Metzger et al. 2007; Tun et al. 2009]	Crosscutting constraints
[Hubaux et al. 2009]	Feature configuration workflow
[Reiser and Weber 2006; 2007; Mannion et al. 2009]	Resolution rules
[Czarnecki et al. 2005]	Specialisation
[Saleh and Gomaa 2005a; Chen et al. 2006a]	Weaving
[Elsner et al. 2008]	Workflow weaving

Mendonça et al. [2008] present algorithms that aggregate features into coherent groups, called *configuration spaces*, and then computes possible *configurations plans* that determine the sequence in which these groups can be configured.

In their origami matrix approach, Batory et al. [2003] use *Cartesian combination* of the classes. In their example, cartesian combination of the classes and operations gives four possible programs (a two by two table). Units can be “folded” along each dimension: If the operation dimension is folded, there are two available classes, each

with insert and delete operations. The class dimension can also be folded in the same way.

Resolution rules define default behaviours to resolve conflicts in, for instance, multi-level feature trees [Reiser and Weber 2007] and viewpoint-oriented variability models [Mannion et al. 2009]. Essentially, these rules define conflictual patterns and the corresponding resolution strategies. For instance, if feature f is optional in a concern and mandatory in the other, f becomes mandatory in the aggregate version of the FD.

Crosscutting constraints appear to be a basic yet powerful and formal way of handling concern composition. Hubaux et al. [2009] top these constraints with a *workflow* that further constrains the *order* in which concerns can be composed.

As for aspects, concern *weaving* techniques highly depend on the artefacts involved and the abstraction level. Faced with this disparity of techniques, we could not elaborate a meaningful comparison strategy.

Elsner et al. [2008] argue that current workflow languages are not adequate in expressing feature-based modularisation of software systems. They propose an aspect-oriented approach to separate out the base workflow from additional workflows, which can be weaved into the base workflow when additional features are selected.

Regarding the composition of concerns, we see a deep synergy between SPL and requirements engineering research. For instance, techniques on viewpoints [Easterbrook and Nuseibeh 1996], inconsistency management [Spanoudakis and Zisman 2001], model synthesis [Uchitel and Chechik 2004], and model merging [Brunet et al. 2006; Acher et al. 2009] may shed new light on how concerns of FDs should be managed.

6. FORMALISATION

In terms of level of formality, most of the FD languages we surveyed are largely informal. We believe that the question of having an expressive and succinct formal FD language is no longer a major problem. The major problem is that most of the approaches to separation and composition of concerns are weakly formalised. Only 19 out of the 127 papers obtained a score of ✚ (complete formal semantics given) or + (partial formal semantics given). 10 other papers obtained ◯ (abstract syntax given) because they provide metamodel or abstract syntax of the concepts they use. Some of the papers we reviewed formalise concepts that are not directly related to FDs (e.g., [Ahmed and Capretz 2006]). In general, what we observed is an informal and loose definition of the concept of concern.

Adittionally, the very nature of concern varies. Besides the distinction between group concerns and relationships, more fundamental differences can be noticed. For group concerns, some seem to be "attributes" characterizing features like the binding time. Others make a clear separation between FDs achieving different objectives like provided vs. required variability. Finally, a concern can be captured by a link to an external model. The same observation holds for relationship concerns that can either denote a specific type of constraint or represent a form of conceptual consistency like *entailment*.

There is apparently no determination to reach a unified definition, which prevents a rigorous comparison of the approaches. To systematise the way the problem is tackled and to converge on a solution, the following questions should be answered:

— *What are the types and subtypes of concerns?* Different types of concern exist. Tables II and III lay the foundation stone by collecting two of them, i.e. feature *groups* and feature decomposition *relationships*. Future work should tell which more fine grained types are needed, and should categorize them more precisely. However, a more pressing matter might be to produce solid empirical evidence that these con-

cerns are actually relevant in practice. Therefore, further investigation is needed to populate a repertoire of relevant concerns.

— *How is a concern defined?* Several papers suggested formal definitions of a concern (e.g., [Batory et al. 2003; Reiser and Weber 2006; Hubaux et al. 2009; Mendonça et al. 2008; Hubaux et al. 2010b]). However, these definitions focus on feature groups with simple parent/child relationships and boolean constraints. Efforts are still needed to integrate and extend these definitions to accommodate the different types of concerns (e.g., using an attribute vs. using a link to an external model that contains equivalent information).

7. TOOL SUPPORT

Many of the tools we came across are little more than simple diagramming tools or standard aspect code-weaving tools. Table VI synthesises tools that provide built-in support for separation of concerns on top of a variability model. For the most part, they are prototypes or proofs of concept.

The concerns they address range from code over UML models to binding time. The separation and composition techniques also vary widely. This disparity in functionalities and purposes renders their comparison irrelevant. It is also another symptom of the very exploratory state in which multi-concern FDs stand.

Overall, meaningful and efficient reasoning about multi-concern FDs remains an area for further research. How separation of concerns, and their recomposition should be supported by an automated tool is under-explored. This is a likely consequence of the limited research on concern formalisation. Better specification of the problem would lead to more robust and efficient tool support.

Despite the apparent difficulties, SPL engineers in various industries have been successfully producing commercial software used by many customers. Insightful reports on how SPL engineers actually manage concerns in FDs would have a positive influence on research. Yet, as Hubaux et al. [2010a] discuss in their preliminary review on the application of FDs in practice, reports in the literature on the actual use of FDs are scarce. The absence of incentive for practitioners to publish experience reports and restrictions imposed by non-disclosure agreements are among the possible causes. In this context, it is hard to tell how commercial tools like pure::variants [Beuche 2008], which enable the addition of numerous attributes to a feature, can serve as a sound basis for concern modelling.

8. DISCUSSION

This survey has shown that when FD languages were first introduced, their main concern was about distinguishing features that are particular to certain members of the product family from those that are common across all members. FDs address something that few other modelling languages do: To treat groups of programs as having common functionality with only minor varying features. This could encourage reuse across programs, and design more stable architectures. As Figure 1 shows, the concepts used in the language were also simple: Mostly with AND (aggregation/consist-of) and OR (alternatives/optional/mutually exclusive OR) relationships, together with some notion of feature attributes and "requires" or "excludes" relationships [Kang et al. 1990].

Having said that, even the seminal report by Kang et al. [1990] has a fundamental ambiguity about the notion of variability. The report is not precise about many possible meanings of variability: Whether variability is in the requirements, in the design, or in the running system; or whether it is inside the software or out in the system context; or whether it is to be determined from the perspective of the user or the developer. At times Kang et al. [1990] treat FDs as a "domain analysis" tool, suggesting

Table VI. Tools supporting separation of concerns in variability models.

References	Tool name	Concerns	Separation	Composition
[Bashroush et al. 2008]	4VM	Business, Behavioural, Relationship, Architecture	Separate models	Composition rules
[Brown et al. 2003]	ADLARS	System, Task, Component	Separate models	Transformation
[Batory and Börger 2008]	AHEAD	Feature	Feature	Functional composition
[Apel et al. 2006]	AML	Feature	Aspects, Features	Weaving
[Ubayashi and Nakajima 2007]	AspectVDM	Context	Aspects	Weaving
[Loughran and Rashid 2004]	Framed Aspects	Frame	Frame	Composition rules
[Batista et al. 2008]	MaRiPLA	Artefacts	Separate models	ATL
[White et al. 2009]	MUSCLE	Step	Step	/
[Gomaa and Shin 2004; 2008]	PLUSEE	Artefacts	Separate models	OCL
[Alves et al. 2007]	PLP	Aspects	Aspects	Refactoring
[Ahmed and Capretz 2006]	SPF_BET	Business	Dimensions	Fuzzy Logic
[Saleh and Gomaa 2005b]	SPLET	Feature	Aspects	Weaving
[Alférez et al. 2009]	VML4RE	Requirements	Separate models, Features	Model transfortmation
[Zhang et al. 2001]	XVLC	Artefacts	Breakpoint	X-Frame
[Reiser and Weber 2006]	*NA*	Organisational structure	Hierarchical layering	Resolution rules
[Ebraert et al. 2009]	*NA*	Change	Change	Algorithm
[Tun et al. 2009]	*NA*	Provided, Required, Context	Separate models	Crosscutting constraints
[Metzger et al. 2007]	*NA*	Provided, Required	Separate models	Crosscutting constraints
[Völter and Groher 2007]	*NA*	Artefacts	Separate models	Transformation, Weaving
[Schmid and Eichelberger 2008]	*NA*	Binding time, Constraint	Meta-variability	/

that the feature analysis should focus on application domain; at other times, they indicate that FDs can be used to describe relationships between features of different binding times: Compile-time, load-time and run-time. An FD is also a *"communication medium between users and developers"* [Kang et al. 1990], a description of the *"user's understanding of the general capabilities of applications in a domain"* [Kang et al. 1990], and a tool for communication between developers and designers. The list of concerns recognised by the surveyed approaches, in particular by Lee et al. [2002] and Kang et al. [1998], is very comprehensive, ranging from cost of features, platform to organisational structure.

This attempt to cater for such diverse purposes unfortunately overloads FDs and confuses the concerns recognised by FDs. We did not find any later work on FD languages remeding this weakness. Rather conversely, there were proposals for even more concerns, such as organisational concerns [Grünbacher et al. 2009], configuration concerns [Czarnecki et al. 2005] and feature interaction concerns [Chen et al. 2006b], to be recognized in FDs. This tendency to expand the scope of FDs by including a growing share of concerns, as this survey reveals, inevitably contributes to increasing the size and complexity of FDs. We believe that a framework for defining, distinguishing and relating these concerns is an urgent research problem for FD languages and methods.

Many of the concern separation and composition techniques are based on the Aspect-Oriented Programming (AOP) paradigm. When FD concerns are treated as "cross cutting" concerns in program design, features have a restricted meaning. For instance, Colyer et al. [2004] regard global properties such as flexibility and configurability as FD concerns that can be separated and weaved using AOP techniques. Although it is clear that some FD concerns can be addressed using AOP techniques, there remains a question about how concerns such as requirements and stakeholder concerns can be handled using the same techniques.

In terms of formalisation, although many of the existing work use semi-formal notations, the formal semantics of FD language is now clear. For instance, Schobbens et al. [2006] has provided a general semantics for FD languages. As shown by Schobbens et al. [2007], from the semantic perspective, a relatively simple FD language can have a high degree of expressiveness and succinctness. Therefore, we have to conclude that making FD languages richer does not necessarily make them more scalable. Although the semantics of FD languages is clear, such formalisms treat the individual features simply as nodes in a graph. There has also been progress in defining the content of a feature formally, for example, using a kind of state transition graph [Classen et al. 2010]. Depending on the analysis to be performed, alternative formalisations of features, accommodating variability, are also possible.

Tool support for defining concerns, and for separating and composing them, is very limited at the moment. Many of the tools focus on visualisation of FDs, allowing partial views of the FDs. Concerns within a FD are difficult to distinguish using the current tool support. For instance, it is difficulty to visualise FDs from the perspective of a particular stakeholder or for a particular stage in the development. Many research questions are still outstanding in this area.

9. THREATS TO VALIDITY

The last step of our survey is a critical look at the research method and the collected results. For each point, the threats to validity are discussed with the presumed consequences on the global validity of the results.

Single bibliographical database. The only bibliographical database we used to collect papers is DBLP. The bias that could have been introduced was however mitigated by (1) the automatic inclusion in DBLP of papers from other electronic li-

braries (ACM, IEEE, Springer...), and (2) the list of papers we manually added. The latter reduced the risk to miss relevant papers not caught by the queries or not encoded in DBLP.

No meta-analysis. We favoured a qualitative approach over a quantitative one. Consequently, we neither completed a meta-analysis nor a sensitivity analysis. A meta-analysis is usually meaningful to aggregate results from various yet similar studies [Brereton et al. 2007]. The variability among the papers we collected simply rendered such aggregation impossible. It turned out that tabulated data [Brereton et al. 2007] was the only way for us to compare the results. This, in turn, made a sensitivity analysis inappropriate.

No external expert. We did not appoint an external expert to control the completeness and consistency of the review. The impact of this decision upon the final results is hard to tell given the broad scope of our investigation. We tried to lessen the possible negative impact by appointing two different researchers with some overlap in their assignments to make sure that their evaluations were consistent and comparable.

Inconsistent content validation. We observed significant differences in the validation of the results presented in the papers. For instance, many papers claimed that their results were validated with case studies. Yet, few of them followed a rigorous approach to empirically validate their results such as [Shull et al. 2007]. That threat affects research on separation of concerns in FDs in general. Our study can only reflect the current state of the art in that domain.

No quality evaluation. To build a complete repertoire of concerns, all papers were equally evaluated without any discrimination regarding the source (e.g., journal, conference or workshop) or venue. A quality evaluation would have excluded several papers that contained valuable input for our repertoire. The overall impact on the quality of our study, however, is hard to predict. In fact, the fairly low level of empirical validation in all the papers led us to believe that no significant gains in quality would have been perceived if we had been more selective over venues.

10. CONCLUSION

In this paper, we have presented the results of a systematic survey of the literature on separation of concerns in feature diagram languages. Our research method is well-founded and covers both quantitative and qualitative aspects. The main research question of the survey has been to identify the most important concerns, or purposes, of feature diagrams, how they are separated and composed in existing SPL approaches, the level of formalism used and the tool support available. The keys findings of the survey are the following:

— The inherent vagueness in the feature abstraction and in the purpose of feature diagrams makes realistic feature diagrams hard to comprehend and analyze. Since a more expressive feature diagram language is unlikely to solve the problem, we have concluded that a clarification in the purpose of feature diagrams, and the meaning of features in various artifacts will go some way towards solving this problem.
— Separation and composition techniques for concerns in feature diagrams have been found to be rudimentary: They range from hierarchical layering of feature diagrams to selective projection of feature diagrams using a visualization tool.
— In terms of formalization, expressive and succinct formal feature diagram languages already exist. However, the extension of this formalization to concerns is still partial.
— Tool support for aspect-oriented approaches seems to offer some opportunity, but so far they have been mainly applied to code, rather than feature diagrams. We have concluded that a firmer conceptual footing for concern definition, separation and com-

position will enhance tool support for separating and composing concerns in feature diagrams.

Based on these findings, we conclude that in order to make feature diagram languages scalable, there is a need for a consensus about the key concerns of feature diagrams and formalism for better separation and composition of concerns in feature diagrams.

In terms of extending the work, there is still a long way to go before an agreed list of most important and valuable concerns can be provided. Additional work is needed to comprehend the relationship between feature diagrams and other types of model. Besides introducing new concerns, many extra separation and compositions techniques mixing features and heterogeneous model elements will need to be considered.

11. ACKNOWLEDGEMENTS

The work is supported by the Interuniversity Attraction Poles Programme of the Belgian State, Belgian Science Policy, under the MoVES project. A significant part of the work was undertaken while T. T. Tun was at the University of Namur, who is now partially supported by a Microsoft Software Engineering Innovation Foundation Award and by the European Research Council project ASAP.

REFERENCES

2004. *Software Reuse: Methods, Techniques and Tools: 8th International Conference, ICSR 2004, Madrid, Spain, July 5-9, 2009. Proceedings*. Lecture Notes in Computer Science Series, vol. 3107. Springer.

2009. *Workshop on Scalable Modeling Techniques for Software Product Lines (SCALE'09) at SPLC'09, San Francisco, USA*.

ACHER, M., COLLET, P., LAHIRE, P., AND FRANCE, R. 2009. Composing Feature Models. In *2nd International Conference on Software Language Engineering (SLE'09)*. Springer, Denver, CO, USA, 62–81.

AHMED, F. AND CAPRETZ, L. F. 2006. Maturity assessment framework for business dimension of software product family. *IBIS 1,* 1, 9–32.

ALFÉREZ, M., SANTOS, J., MOREIRA, A., GARCIA, A., KULESZA, U., ARAÚJO, J., AND AMARAL, V. 2009. Multi-view composition language for software product line requirements. In *SLE*, M. van den Brand, D. Gasevic, and J. Gray, Eds. Lecture Notes in Computer Science Series, vol. 5969. Springer, 103–122.

ALVES, V., MATOS, P., COLE, L., VASCONCELOS, A., BORBA, P., AND RAMALHO, G. 2007. Extracting and evolving code in product lines with aspect-oriented programming. *T. Aspect-Oriented Software Development 4*, 117–142.

APEL, S., LEICH, T., AND SAAKE, G. 2006. Aspectual mixin layers: aspects and features in concert. In *ICSE*. 122–131.

BASHROUSH, R., SPENCE, I. T. A., KILPATRICK, P., BROWN, T., AND GILLAN, C. 2008. A multiple views model for variability management in software product lines. See Heymans et al. [2008], 101–110.

BAST, H. 2010. Completesearch: `http://dblp.mpi-inf.mpg.de/dblp-mirror/index.php`.

BATISTA, T. V., BASTARRICA, M. C., SOARES, S., AND DA SILVA, L. F. 2008. A marriage of mdd and early aspects in software product line development. In *Workshop on Early Aspects (EA'08) collocated with SPLC'08*. 97–103.

BATORY, D., LIU, J., AND SARVELA, J. N. 2003. Refinements and multi-dimensional separation of concerns. *SIGSOFT Software Engineering Notes 28,* 5, 48–57.

BATORY, D. S. 2005. Feature Models, Grammars, and Propositional Formulas. In *9th International Conference on Software Product Lines (SPLC'05)*. 7–20.

BATORY, D. S. AND BÖRGER, E. 2008. Modularizing theorems for software product lines: The jbook case study. *J. UCS 14,* 12, 2059–2082.

BENAVIDES, D., SEGURA, S., AND RUIZ-CORTES, A. 2010. Automated analysis of feature models 20 years later: a literature review. *Information Systems 35,* 6, 615–636.

BERGER, T., SHE, S., LOTUFO, R., WASOWSKI, A., AND CZARNECKI, K. 2010. Variability modeling in the real: a perspective from the operating systems domain. In *Proceedings of the 25th International Conference on Automated Software Engineering (ASE'10)*. ACM, Antwerp, Belgium, 73–82.

BEUCHE, D. 2008. Modeling and building software product lines with pure::variants. In *12th International Software Product Line Conference (SPLC'08)*. IEEE CS, 358–.

BRERETON, P., KITCHENHAM, B. A., BUDGEN, D., TURNER, M., AND KHALIL, M. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software 80,* 4, 571 – 583. Software Performance, 5th International Workshop on Software and Performance.

BROWN, T. J., SPENCE, I. T. A., AND KILPATRICK, P. 2003. A relational architecture description language for software families. In *PFE*, F. van der Linden, Ed. Lecture Notes in Computer Science Series, vol. 3014. Springer, 282–295.

BRUNET, G., CHECHIK, M., EASTERBROOK, S., NEJATI, S., NIU, N., AND SABETZADEH, M. 2006. A manifesto for model merging. In *International workshop on Global integrated model management (GaMMa '06)*. ACM, New York, NY, USA, 5–12.

CHEN, K., ZHAO, H., ZHANG, W., AND MEI, H. 2006a. Identification of crosscutting requirements based on feature dependency analysis. In *14th International Conference on Requirements Engineering (RE'06)*. 300–303.

CHEN, K., ZHAO, H., ZHANG, W., AND MEI, H. 2006b. Identification of crosscutting requirements based on feature dependency analysis. In *14th International Conference on Requirements Engineering (RE'06)*. IEEE Computer Society, Los Alamitos, CA, USA, 300–303.

CHEN, L. AND BABAR, M. A. 2009. A survey of scalability aspects of variability modeling approaches. In *Workshop on Scalable Modeling Techniques for Software Product Lines (SCALE 2009)*. 119–126.

CHEN, L., BABAR, M. A., AND ALI, N. 2009. Variability management in software product lines: A systematic review. In *13th International Software Product Line Conference (SPLC'09)*. 81–90.

CHO, H., LEE, K., AND KANG, K. C. 2008. Feature relation and dependency management: An aspect-oriented approach. In *12th International Software Product Line Conference (SPLC'08)*. 3–11.

CHOI, H., LEE, K., LEE, J., AND KANG, K. C. 2009. Multiple views of feature models to manage complexity. In *Workshop on Scalable Modeling Techniques for Software Product Lines (SCALE 2009)*. 127–133.

CLASSEN, A., HEYMANS, P., AND SCHOBBENS, P.-Y. 2008. What's in a feature: A requirements engineering perspective. In *11th International Conference on Fundamental Approaches to Software Engineering (FASE'08)*. Springer, 16–30.

CLASSEN, A., HEYMANS, P., SCHOBBENS, P.-Y., LEGAY, A., AND RASKIN, J.-F. 2010. Model checking lots of systems: efficient verification of temporal properties in software product lines. In *32nd International Conference on Software Engineering (ICSE'10)*. ICSE '10. ACM, New York, NY, USA, 335–344.

CLASSEN, A., HUBAUX, A., AND HEYMANS, P. 2009. A formal semantics for multi-level staged configuration. In *3rd International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS'09)*. 51–60.

COLYER, A., RASHID, A., AND BLAIR, G. 2004. On the separation of concerns in program families. Tech. Rep. COMP-001-2004, Lancaster University.

CZARNECKI, K., HELSEN, S., AND EISENECKER, U. W. 2004. Staged configuration using feature models. In *3rd International Software Product Line Conference (SPLC'04)*. 266–283.

CZARNECKI, K., HELSEN, S., AND EISENECKER, U. W. 2005. Staged configuration through specialization and multi-level configuration of feature models. *Software Process: Improvement and Practice 10,* 2, 143–169.

CZARNECKI, K. AND PIETROSZEK, K. 2006. Verifying feature-based model templates against well-formedness ocl constraints. In *5th International Conference on Generative programming and Component Engineering (GPCE'06)*. ACM, New York, NY, USA, 211–220.

CZARNECKI, K., SHE, S., AND WASOWSKI, A. 2008. Sample spaces and feature models: There and back again. In *12th International Software Product Line Conference (SPLC'08)*. 22–31.

DEELSTRA, S., SINNEMA, M., AND BOSCH, J. 2009. Variability assessment in software product families. *Information Software Technology 51,* 1, 195–218.

DORDOWSKY, F. AND HIPP, W. 2009. Adopting software product line principles to manage software variants in a complex avionics system. In *Proceedings of the 13th International Software Product Line Conference (SPLC'09)*. San Francisco, CA, USA, 265–274.

EASTERBROOK, S. M. AND NUSEIBEH, B. A. 1996. Using viewpoints for inconsistency management. *Software Engineering Journal 11,* 1.

EBRAERT, P., VALLEJOS, J., AND VANDEWOUDE, Y. 2009. Flexible features: making feature modules more reusable. In *SAC*, S. Y. Shin and S. Ossowski, Eds. ACM, 1963–1970.

ELSNER, C., LOHMANN, D., AND SCHRÖDER-PREIKSCHAT, W. 2008. Towards separation of concerns in model transformation workflows. In *Workshop on Early Aspects (EA'08) collocated with SPLC'08*. 81–88.

ETXEBERRIA, L. AND MENDIETA, G. S. 2008. Variability driven quality evaluation in software product lines. In *12th International Software Product Line Conference (SPLC'08)*. 243–252.

FEY, D., FAJTA, R., AND BOROS, A. 2002. Feature modeling: A meta-model to enhance usability and usefulness. In *2nd International Software Product Line Conference (SPLC'02)*. Springer-Verlag, London, UK, 198–216.

GOMAA, H. AND SHIN, M. E. 2004. A multiple-view meta-modeling approach for variability management in software product lines. See DBL [2004], 274–285.

GOMAA, H. AND SHIN, M. E. 2008. Multiple-view modelling and meta-modelling of software product lines. *IET Software 2,* 2, 94–122.

GRÜNBACHER, P., RABISER, R., DHUNGANA, D., AND LEHOFER, M. 2009. Structuring the product line modeling space: Strategies and examples. In *3rd International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS'09)*. 77–82.

GUNTER, C. A., GUNTER, E. L., JACKSON, M., AND ZAVE, P. 2000. A reference model for requirements and specifications. *IEEE Software 17,* 3, 37–43.

HALLSTEINSEN, S. O., STAV, E., SOLBERG, A., AND FLOCH, J. 2006. Using product line techniques to build adaptive systems. In *10th International Software Product Line Conference (SPLC'06)*. 141–150.

HARTMANN, H. AND TREW, T. 2008. Using feature diagrams with context variability to model multiple product lines for software supply chains. In *12th International Software Product Line Conference (SPLC'08)*. 12–21.

HEIDENREICH, F., SÁNCHEZ, P., SANTOS, J., ZSCHALER, S., ALFÉREZ, M., ARAÚJO, J., FUENTES, L., KULESZA, U., MOREIRA, A., AND RASHID, A. 2010. Relating feature models to other models of a software product line - a comparative study of featuremapper and vml*. *Theory of Aspect-Oriented Software Development 7*, 69–114.

HEYMANS, P., KANG, K. C., METZGER, A., AND POHL, K., Eds. 2008. *Second International Workshop on Variability Modelling of Software-Intensive Systems, Universität Duisburg-Essen, Germany, January 16-18, 2008, Proceedings*. ICB Research Report.

HUBAUX, A., CLASSEN, A., AND HEYMANS, P. 2009. Formal modelling of feature configuration workflow. In *13th International Software Product Lines Conference (SPLC'09)*. 221–230.

HUBAUX, A., CLASSEN, A., MENDONÇA, M., AND HEYMANS, P. 2010a. A preliminary review on the application of feature diagrams in practice. In *Proceedings of the 4th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'10)*. Universität Duisburg-Essen, Linz, Austria, 53–59.

HUBAUX, A., HEYMANS, P., AND BENAVIDES, D. 2008. Variability modelling challenges from the trenches of an open source product line re-engineering project. In *12th International Software Product Line Conference (SPLC'08)*. IEEE Computer Society, 55–64.

HUBAUX, A., HEYMANS, P., SCHOBBENS, P.-Y., AND DERIDDER, D. 2010b. Towards multi-view feature-based configuration. In *16th International Working Conference on Requirements Engineering:Foundation for Software Quality (REFSQ'10)*. Springer-Verlag.

HUBAUX, A., HEYMANS, P., SCHOBBENS, P.-Y., DERIDDER, D., AND ABBASI, E. 2011. Supporting multiple perspectives in feature-based configuration. *Software and Systems Modeling (SoSyM)*. Springer Berlin / Heidelberg (In print).

JACKSON, M. 1995. *Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices*. ACM Press.

JENSEN, P. 2007. Experiences with product line development of multi-discipline analysis software at overwatch textron systems. In *Proceedings of the 11th International Software Product Line Conference (SPLC'07)*. Kyoto, Japan, 35–43.

KANG, K., COHEN, S., HESS, J., NOVAK, W., AND PETERSON, S. 1990. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Tech. Rep. CMU/SEI-90-TR-21, SEI, Carnegie Mellon University. November.

KANG, K. C., DONOHOE, P., KOH, E., LEE, J., AND LEE, K. 2002. Using a marketing and product plan as a key driver for product line asset development. In *2nd International Software Product Line Conference (SPLC'02)*. 366–382.

KANG, K. C., KIM, S., LEE, J., KIM, K., SHIN, E., AND HUH, M. 1998. Form: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Engineering 5*, 143–168.

KIRCHER, M., SCHWANNINGER, C., AND GROHER, I. 2006. Transitioning to a software product family approach - challenges and best practices. In *10th International Software Product Line Conference (SPLC'06)*. 163–171.

KITCHENHAM, B. A. 2004. Procedures for undertaking systematic reviews. Tech. Rep. 0400011T.1, Computer Science Department, Keele University (TR/SE-0401) and National ICT Australia Ltd.

LEE, J., KANG, K. C., AND KIM, S. 2004. A feature-based approach to product line production planning. In *3rd International Software Product Line Conference (SPLC'04)*. 183–196.

LEE, K., BOTTERWECK, G., AND THIEL, S. 2009. Aspectual separation of feature dependencies for flexible feature composition. In *COMPSAC (1)*, S. I. Ahamed, E. Bertino, C. K. Chang, V. Getov, L. Liu, H. Ming, and R. Subramanyan, Eds. IEEE Computer Society, 45–52.

LEE, K. AND KANG, K. C. 2004. Feature dependency analysis for product line component design. See DBL [2004], 69–85.

LEE, K., KANG, K. C., KIM, M., AND PARK, S. 2006. Combining feature-oriented analysis and aspect-oriented programming for product line asset development. In *10th International Software Product Line Conference (SPLC'06)*. 103–112.

LEE, K., KANG, K. C., AND LEE, J. 2002. Concepts and guidelines of feature modeling for product line software engineering. In *Proceedings of the 7th International Conference on Software Reuse*. Springer-Verlag, London, UK, 62–77.

LOUGHRAN, N. AND RASHID, A. 2004. Framed aspects: Supporting variability and configurability for aop. See DBL [2004], 127–140.

LUTZ, R. R. 2008. Enabling verifiable conformance for product lines. In *12th International Software Product Line Conference (SPLC'08)*. 35–44.

MANNION, M., SAVOLAINEN, J., AND ASIKAINEN, T. 2009. Viewpoint-oriented variability modeling. *Computer Software and Applications Conference, Annual International 1*, 67–72.

MENDONÇA, M., COWAN, D. D., MALYK, W., AND DE OLIVEIRA, T. C. 2008. Collaborative product configuration: Formalization and efficient algorithms for dependency analysis. *Journal of Software 3,* 2, 69–82.

METZGER, A., HEYMANS, P., POHL, K., SCHOBBENS, P.-Y., AND SAVAL, G. 2007. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In *15th International Conference on Requirements Engineering (RE'07)*. New Delhi, India, 243–253.

MOREIRA, A., RASHID, A., AND ARAÚJO, J. 2005. Multi-dimensional separation of concerns in requirements engineering. In *13th International Conference on Requirements Engineering (RE'05)*. 285–296.

NODA, N. AND KISHI, T. 2008. Aspect-oriented modeling for variability management. In *12th International Software Product Line Conference (SPLC'08)*. 213–222.

POHL, K., BÖCKLE, G., AND VAN DER LINDEN, F. J. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, NJ, USA.

REISER, M.-O. AND WEBER, M. 2006. Managing highly complex product families with multi-level feature trees. In *14th International Conference on Requirements Engineering (RE'06)*. IEEE Computer Society, Los Alamitos, CA, USA, 146–155.

REISER, M.-O. AND WEBER, M. 2007. Multi-level feature trees. *Requirements Engineering 12,* 2, 57–75.

SALEH, M. AND GOMAA, H. 2005a. Separation of concerns in software product line engineering. In *Workshop on Modeling and Analysis of Concerns in Software (WACS'05)*. ACM, New York, NY, USA, 1–5.

SALEH, M. AND GOMAA, H. 2005b. Separation of concerns in software product line engineering. *ACM SIGSOFT Software Engineering Notes 30,* 4, 1–5.

SAVOLAINEN, J. AND KUUSELA, J. 2001. Consistency management of product line requirements. In *5th International Conference on Requirements Engineering (RE'01)*. 40–47.

SCHMID, K. AND EICHELBERGER, H. 2008. Model-based implementation of meta-variability constructs: A case study using aspects. See Heymans et al. [2008], 63–71.

SCHOBBENS, P.-Y., HEYMANS, P., AND TRIGAUX, J.-C. 2006. Feature diagrams: A survey and a formal semantics. In *14th International Conference on Requirements Engineering (RE'06)*. IEEE Computer Society, Los Alamitos, CA, USA, 136–145.

SCHOBBENS, P.-Y., HEYMANS, P., TRIGAUX, J.-C., AND BONTEMPS, Y. 2007. Generic semantics of feature diagrams. *Computer Networks 51,* 2, 456–479.

SHULL, F., SINGER, J., AND SJØBERG, D. I. K. 2007. *Guide to Advanced Empirical Software Engineering*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

SPANOUDAKIS, G. AND ZISMAN, A. 2001. Inconsistency management in software engineering: Survey and open research issues. In *Handbook of Software Engineering and Knowledge Engineering*, K. Chang S., Ed. World Scientific Publishing Co, 329–380.

STEGER, M., TISCHER, C., BOSS, B., MÜLLER, A., PERTLER, O., STOLZ, W., AND FERBER, S. 2004. Introducing pla at bosch gasoline systems: Experiences and practices. In *Proceedings of the 3rd International Software Product Line Conference (SPLC'04)*. Boston, MA, USA, 34–50.

SVAHNBERG, M., VAN GURP, J., AND BOSCH, J. 2005. A taxonomy of variability realization techniques. *Software: Practice and Experience 35,* 8, 705–754.

TARR, P. L., OSSHER, H., HARRISON, W. H., AND JR., S. M. S. 1999. *N* degrees of separation: Multi-dimensional separation of concerns. In *21st International Conference on Software Engineering (ICSE'99)*. 107–119.

THOMPSON, J. M. AND HEIMDAHL, M. P. E. 2003. Structuring product family requirements for n-dimensional and hierarchical product lines. *Requirements Engineering 8,* 1, 42–54.

TUN, T. T., BOUCHER, Q., CLASSEN, A., HUBAUX, A., AND HEYMANS, P. 2009. Relating requirements and feature configurations: A systematic approach. In *Proceedings of the 13th International Software Product Lines Conference (SPLC'09)*. San Francisco, CA, USA.

UBAYASHI, N. AND NAKAJIMA, S. 2007. Context-aware feature-oriented modeling with an aspect extension of vdm. In *SAC*, Y. Cho, R. L. Wainwright, H. Haddad, S. Y. Shin, and Y. W. Koo, Eds. ACM, 1269–1274.

UCHITEL, S. AND CHECHIK, M. 2004. Merging partial behavioural models. In *International Symposium on Foundations of Software Engineering (FSE'04)*. Newport Beach.

VÖLTER, M. AND GROHER, I. 2007. Product line implementation using aspect-oriented and model-driven software development. In *SPLC*. IEEE Computer Society, 233–242.

WENZEL, S., BERGER, T., AND RIECHERT, T. 2009. How to configure a configuration management system - an approach based on feature modeling. In *Proceedings of the 1st International Workshop on Model-driven Approaches in Software Product Line Engineering (MAPLE'09)*. San Francisco, CA, USA, 99–105.

WHITE, J., BENAVIDES, D., DOUGHERTY, B., AND SCHMIDT, D. C. 2009. Automated reasoning for multi-step software product-line configuration problems. In *Proceedings of the 13th International Software Product Lines Conference (SPLC'09)*. San Francisco, CA, USA.

ZAVE, P. AND JACKSON, M. 1997. Four dark corners of requirements engineering. *ACM Transactions on Software Engineering and Methodology (TOSEM) 6,* 1, 1–30.

ZHANG, H., JARZABEK, S., AND SWE, S. M. 2001. Xvcl approach to separating concerns in product family assets. In *GCSE*, J. Bosch, Ed. Lecture Notes in Computer Science Series, vol. 2186. Springer, 36–47.

ZIADI, T., HELOUET, L., AND JEZEQUEL, J.-M. 2004. Towards a uml profile for software product lines. In *Software Product-Family Engineering*, F. van der Linden, Ed. Lecture Notes in Computer Science Series, vol. 3014. Springer Berlin / Heidelberg, 129–139.