Visualisation of Variability in Software Product Line Engineering

Daren Nestor[1], Luke O'Malley[2], Aaron Quigley[2], Ernst Sikora[1], Steffen Thiel[1]
Lero – The Irish Software Engineering Research Centre

[1]*University of Limerick*
Limerick, Ireland

[2]*University College Dublin*
Dublin, Ireland

{ daren.nestor | luke.omalley | aaron.quigley | ernst.sikora | steffen.thiel }@lero.ie

Abstract

Using a product line approach allows companies realize significant improvements in time-to-market, cost, productivity, and quality. One fundamental problem in software product line engineering is related to the fact that a product line of industrial size can easily incorporate several thousand variation points. This makes variability management and product derivation tasks extremely difficult. This paper elaborates on the idea of using visualisation techniques to support and improve the effectiveness of these tasks. A reference model that helps to frame the visualisation research and important areas that affect the use of visualisation techniques in software product line engineering are presented.

1. Introduction

In software product line engineering similarities between products are exploited to reduce the amount of work involved in product derivation. Software product line engineering has rapidly emerged as an important software development paradigm during the last few years. Developing products based on a product line approach allows companies to build a variety of systems with a minimum of technical diversity and to realize significant improvements in time-to-market, cost, productivity, and quality [1].

One fundamental problem with many software product line engineering approaches is that they do not scale well for industrial size product lines. This is due to the fact that industrial size product lines can easily incorporate thousands of variation points and configuration parameters for product customization (see e.g., [2]). Managing this amount of variability is extremely complex and requires sophisticated modelling techniques. Current variability modelling techniques have proven useful to represent product variation in various application domains (e.g., [3] [4] [5] [6]). However, these techniques have mainly been applied in prototypical settings in which relatively few variation points and variants had to be managed. Consequently, there is a need for appropriate techniques for "industry-size" software product lines with a high number of variants which can support product line stakeholders in performing their tasks.

This paper elaborates on the idea of using information and software visualisation techniques to achieve the economies of scale required to support variability management in industrial product lines. Information visualisation is the use of computer-supported, interactive, visual representations of abstract data to amplify cognition [7]. Software visualisation is a sub-area of information visualisation that focuses on both the human understanding and effective use of computer software [8].

Visualisation has proven useful to amplify cognition in a number of ways, for example, by increasing the "memory" and "amount of processing" available to users, by supporting the search for information, and by encoding information in a manipulable medium [7]. The authors are convinced that visualisation techniques can also amplify the cognition about the large and complex data sets created and used in industrial software product line engineering. The exploration of the potential of visual representations such as trees and graphs combined with the effective use of human interaction techniques such as dynamic queries, direct manipulation, and details-on-demand when applied in a software product line context is a challenging research task. Improving the effectiveness of significant product line activities such as variability management and product derivation is a major goal of this research.

The remainder of this paper is organised as follows: Section 2 introduces variability modelling approaches and discusses representation techniques commonly used in product line engineering. Section 3 explains some of the limitations of current variability

representations and motivates the challenges of applying software visualisation techniques in that area. Section 4 presents a reference model that helps to frame the visualisation research in software product lines. Section 5 discusses three areas that are of particular importance to this research. Section 6 outlines future research activities. Finally, Section 7 summarises the conclusions of the paper.

2. Variability Modelling

2.1 Terminology

Variability refers to the ability of a software product line development artefact to be configured, customized, extended, or changed for use in a specific context [9]. It thus provides the required flexibility for product differentiation and diversification within the product line. Variability can evolve both in time and space. *Variability in time* is the existence of different versions of an artefact that are valid at different times. It thus denotes the evolution of an artefact over time. *Variability in space* is the existence of an artefact in different shapes at the same time. By this we mean that the same basic asset can be used differently in different products. *External variability* is variability visible to (and thus selectable by) customers. *Internal variability* is variability that is hidden from customers but visible to particular stakeholders of the product line effort, for example, product developers.

Variation points identify locations in product line artefacts at which variation will occur [10]. The *binding time* refers to the point in a product's lifecycle at which a particular variant for a variation point is bound to the system, e.g. pre- or post-deployment. A *realisation mechanism* refers to the technique that is used to implement the variation point [10]. *Product derivation* is the process of constructing products from product line artefacts, partially by exploiting variation points and variants.

2.2 Variability Modelling Approaches

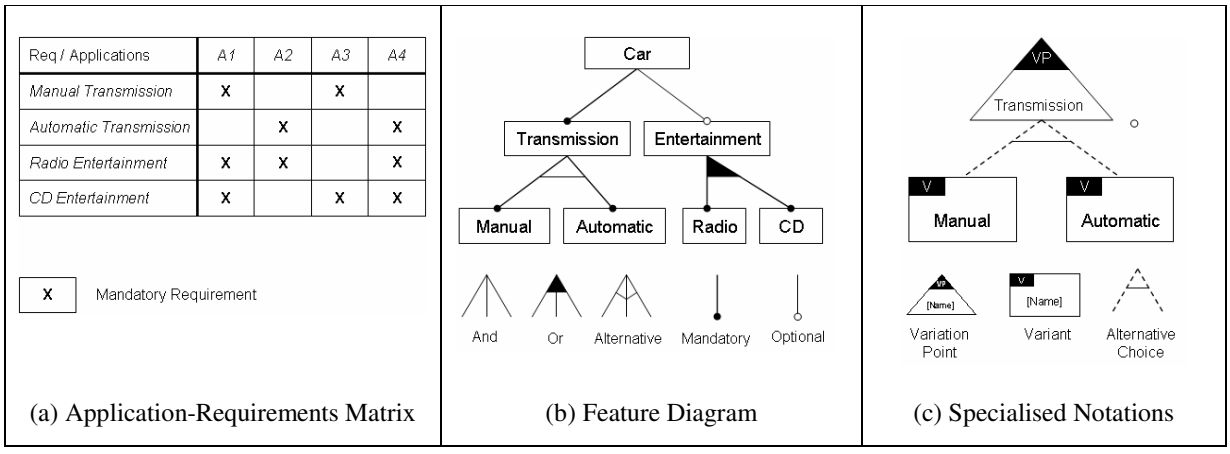
Over the last few years, numerous models and approaches have been proposed for representing variability information in various development phases of a software product line approach, especially in requirements engineering (e.g., [3] [4]) and architecture design (e.g., [5] [6]).

In this context, many researchers have suggested the integration of product line variability into traditional development artefacts such as feature models (e.g., [3]), use case models (e.g., [4]), architecture variability models (e.g., [5]), and class variability diagrams (e.g., [6]). Other researchers have emphasized the need for separating variability information from the original development artefacts in order to support the communication of variability and to improve consistency (e.g., [11]).

Recently, the formalisation of variability emerged as an important research area, especially to support tool development and automation in software product line engineering. Existing approaches deal, for example, with the formalisation of feature models (e.g., [12] [13]) and the formal analysis of dependencies among features (e.g., [14] [15]). Further, a number of tools for variability modeling have been developed (e.g., [15] [16] [17]).

2.3 Variability Representations

Common notations used in existing approaches and tools for representing variability in product lines are application-requirements matrices (e.g., [1]), feature diagrams (e.g., [3]), and UML-based models with specialised notations (e.g., [5] [11] [18]).



(a) Application-Requirements Matrix (b) Feature Diagram (c) Specialised Notations

Figure 1: Variability Representations

An *application-requirements matrix* represents common and variable requirements of a set of applications (or products) in a matrix (see Figure 1a). The left column of the matrix usually shows the requirements of the applications considered in the product line. The applications themselves are given in the top row, whereas the body of the matrix provides information about which requirements are mandatory for a certain product.

A *feature diagram* is a graphical representation of a feature model which shows a hierarchically structured set of features of the product line (see Figure 1b). Features are represented as nodes and relationships between features as links. Possible relationships between features are usually categorized as "And" (all subfeatures must be included), "Or" (one or more subfeatures can be included), "Alternative" (only one subfeature can be included), "Mandatory" (required feature), and "Optional" (potential feature). A feature diagram is typically represented as a tree where primitive features are leaves and compound features are interior nodes.

Finally, several researchers have proposed approaches to model variability by using *UML-based or specialised notations*. The notations usually contain specific extensions or symbols for denoting the existence of variation points and variants. These models are often created from metamodels that define how to build valid model instances. A sample notation introduced in [11] is given in Figure 1c.

3. Research Challenges for Variability Visualisation

While numerous modelling approaches and representations for variability have been proposed in the past, the visualisation of variability information is still limited in certain cases. To illustrate this assertion we first introduce an example from one of our industrial partners and then discuss problems that arise with existing variability representations.

3.1 E&P Variability Example

E&P is a company that specializes in authentic simulations and interactive tutorials for consumer electronics such as car infotainment systems, mobile phones, and digital imaging systems. The product line approach is effective in this domain because due to the similarities in the structure and behaviour of the different simulations. For example, all simulations have a similar user interface comprising one or multiple (simulated) displays and (simulated) control buttons. The number of variation points is high as the simulations have to be customised for different types of products, different manufacturers, and different product models.

To represent the variability of (a part of) the E&P product line we have applied the Application-Requirements Matrix technique discussed in Section 2.3. In this example, the rows of the matrix contain simulation features. Figure 2 shows an excerpt of the product line features, limited to twelve products and 120 features. Mandatory features are marked as black rectangles. From a visualisation point of view, a representation such as that depicted in Figure 2 supports retrieving certain facts related to variability, for instance, which features are common to most products listed and which features apply only to a small subset of products. This simple and compact visualisation allows us to reason about which features are candidates for common assets, and identify other features as areas where variation points could be introduced.

However, the matrix conveys a very limited set of information. Relational and dependency information are not included, for instance. Information about binding times is missing and it is not clear which features include internal variability and to which implementation structures (e.g. modeules) a particular variation point is assigned. Yet, this information is essential to accomplish variability management and product derivation tasks in a product line approach. Consequently, the tabular visualisation presented in Figure 2 is useful for some tasks but generally not sufficient. More sophisticated visualisation techniques are required in order to satisfy all information needs that arise in variability management and product derivation.

Other variability techniques, such as those discussed in Section 2, allow the addition of some of the information missing from this representation. Yet, graph based visualisations such as feature diagrams or UML models suffer from other limitations. Graph based diagrams can, for example, easily overwhelm stakeholders with information, especially when the diagram includes a high number of nodes, detailed information for each node, and a high number of links. Extensive study is required to retrieve specific information from such diagrams. In addition, since the visualisations are (essentially) static, it is not possible to create customised views, e.g. to support particular product derivation tasks.

3.2 Limitations in Variability Representation

From the example above, it is clear that even with a relatively small product line of twelve products and by applying the existing techniques to represent

variability, an effective variability management is difficult to achieve. There are two main limitations the authors want to expand on: the representation of large variability structures and interacting with the variability representation.

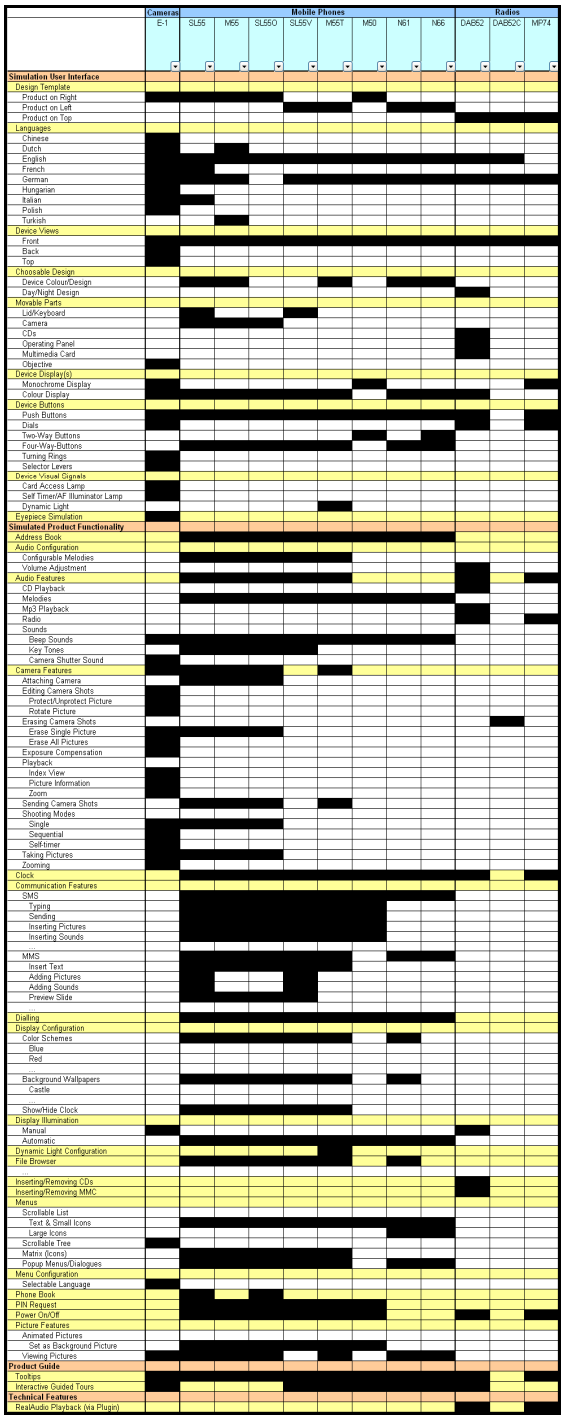


Figure 2: Excerpt of a Feature Matrix

Limited representation of a large variability structures: Existing approaches are limited with respect to representing information about large variability structures. Industrial product lines such as those reported by Nokia [19], Philips [20], and Bosch [2] usually include hundreds of products. This means that these product lines could easily incorporate many thousands of variation points, making it extremely difficult to handle relationships among product line artefacts and to manage systematic product derivation. The extraction of information from existing variability representations is possible for software product lines with only small variability models but severely impaired for large-scale models.

Limited interaction with variability representation: A single, static visualisation can hardly support different product line engineering tasks such as exploring the variation points or deriving product variants. Product line engineers must be able to interact with visualisations and adapt the information that is visualised. Yet, existing approaches provide either no or very limited support to interact with the visualisation.

To overcome these limitations in representing variability, the authors propose to investigate in information and software visualisation techniques (see e.g. [7]). These techniques have proven useful in the comprehension of large amounts of data. However, the use of information/software visualisation techniques in the context of a software product line approach is largely unexplored.

In the following, the authors elaborate on the idea of using visualisation techniques in software product line engineering and introduce a reference model to illustrate potential research areas of investigation.

4. Visualisation Reference Model

According to the work of Card et al. [7], visualisation can be described as "adjustable mappings from data to visual form to the human perceiver.". Figure 3 shows a slightly adapted version of the reference model given in [7] that illustrates these mappings for variability visualisation in the context of software product lines.

In Figure 3, the arrows flow from data on the left to the human perceiver, indicating a series of data transformations. Arrows also flow from the human perceiver at the right into the transformations themselves, indicating the adjustment of these transformations by user-operated controls.

The first type of transformation is *Data Transformations* which map raw *SPL Data* (i.e., data about the software product line (SPL) artefacts, their

variability, and the dependencies among them) into *Data Tables*. Data Tables are relational descriptions of data extended to include metadata (i.e. descriptive information about the data). The usual strategy here is to achieve a set of relations that are more structured than the original data and thus easier to map to visual forms.

Visual Mappings then transform Data Tables into *Visual Structures*. Visual Structures are structures that combine spatial substrates (e.g., nominal or ordinal axis), marks (points, lines, areas, volumes), and graphical properties (e.g., colour, texture or intensity) to encode information. It is important to note that a Visual Mapping preserves the data and that it can be perceived well by the human.

View Transformations create *Views* of the Visual Structures by specifying graphical parameters such as position, scaling, and clipping. As such, View Transformations interactively modify and augment Visual Structures to turn static presentations into visualisations. They exploit time to extract more information from the visualisation than would be possible statically.

Finally, user interaction controls parameters of these transformations, restricting the view to certain data ranges, for example, or changing the nature of the transformation. The visualisation and their controls are used in service of some task. A summary of interaction tasks is given in Table 1.

There are various visualisation techniques that are applicable to the interaction tasks described above.

Filtering of the source data could allow the breaking down of systems into manageable chunks, making it possible for meaningful tasks to be completed on the data (e.g., examining the variability of a particular subsystem of the product line).

Table 1: Visualisation Interaction Tasks

Task	Description
Overview	Gain an overview of the entire collection
Zoom & Pan	Zoom in on items of interest and move about the visualization
Filter	Filter out uninteresting items
Details-on-demand	Select an item or group and get details when needed
Relate	View relationships among items
History	Keep a history of actions to support undo, replay, and progressive refinement
Extract	Allow extraction of sub-collections and of the query parameters

Focus of attention techniques could allow an engineer to keep a mental map of the overall product line artefacts while emphasizing the information of current interest, allowing this information to be communicated or acted upon with greater efficiency and success (e.g., variability dependencies).

Aggregation and decomposition allow the data to be either grouped or partitioned in some meaningful manner, allowing for less overwhelming visualisations while keeping the relevant connections between artefacts and products, allowing for more effective product navigation and derivation.

Interactive details on demand allows information to be accessed when required, while keeping them hidden when they are not needed, resulting in effective information communication and preventing information overload (e.g., details on binding times of variation points).

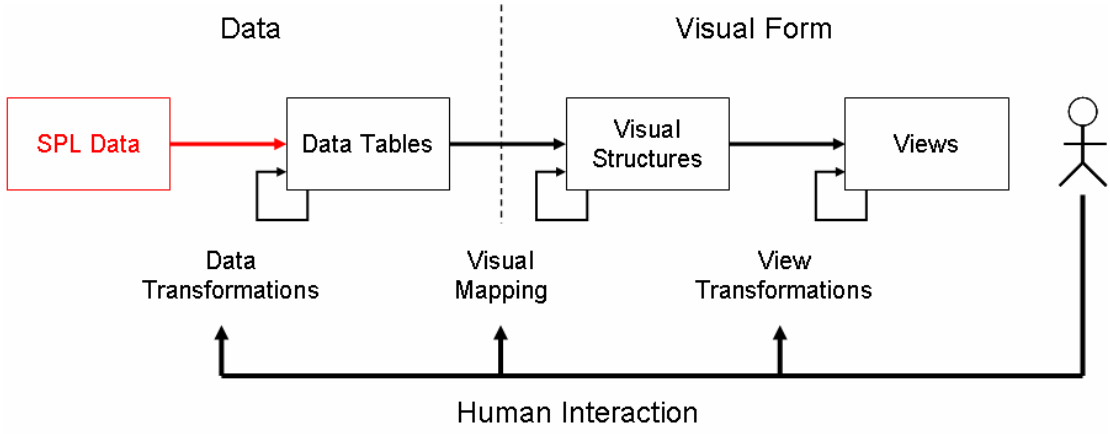


Figure 3: Reference Model for Variability Visualisation (based on [7])

These techniques could be applied to product line use scenarios that have emerged during consultation with our industrial partners. Information such as typical product configurations where products can be categorised into regional variants or customer specific variants would be useful. Purely technical information, such as a representation of assets used to create a product, or visualization of architectural structures, was also considered desirable.

Figure 4 is a concept of visualising the crossover of variation between different products based on set notation. It effectively illustrates the feature set of each product, and makes it easy to identify commonality, variability and exclusivity in the variation sets.

5. Discussion

Three important research directions for further investigation are outlined in this section: presentation of information, task support, and user interaction. The authors believe that these areas will be important to ensure that visualisation will be effective in supporting variability management and product derivation in the context of a product line approach.

5.1 Presentation of Information

An important issue in this area is to overcome the problem of communicating information effectively in a high information density environment. Extracting information from high variability representations can lead to information overload. Hierarchical structures can help in this respect. Common visualisation techniques for hierarchical information structures include listings, outlines, and tree diagrams [7].

Listings are good at providing detailed information on content but poor at presenting structural information.

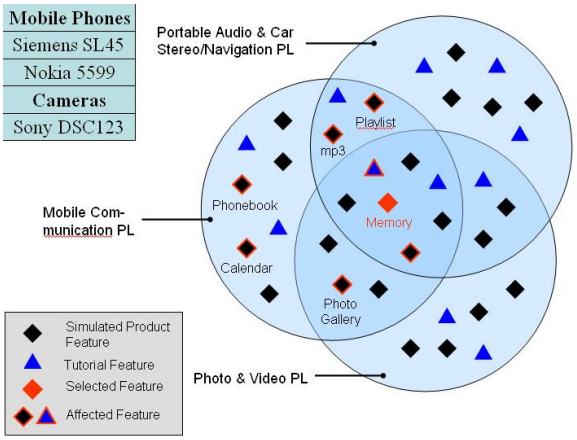


Figure 4: Feature Set Visualisation Concept

Outline methods can provide both structural and content information. Yet the structural information can be extracted only for a few lines at the same time. For both, listings and outlines, the number of lines required to present a hierarchy is equal to the number of nodes. According to Johnson and Shneiderman (see [7], p. 152) listings and outlines are "inadequate for structures containing more than a few hundred nodes." The authors point out that "a great deal of effort is required to achieve a mental model of the structure in large hierarchies using these methods."

Tree drawings are excellent at presenting structural information for small hierarchies. However, they make poor use of available display space. To provide a global view on a large hierarchical structure, nodes have to be very small. In this case, content information cannot be presented well. Hence, tree drawings are of limited use for the visualisation of variability, unless the available display space is large.

The presentation of hierarchical information can be improved even if the display space is limited. Visualisation techniques that are based on enclosure rather than on connection, for example, allow improved display space usage. Examples of such techniques are Venn Diagrams and Tree-Maps (see e.g. [21]).

5.2 Task support

The visualisations need to provide direct support for the product line stakeholders' tasks in variability management and product derivation. This requires identification of the tasks, the information required to perform the tasks, and the group of stakeholders who are associated to the tasks. Typical tasks for variability management (VM) and product derivation (PD) are shown in Table 2.

Table 2: Product Line Engineering Tasks

Type	Stakeholder Task
VM	Determine in how many products a particular variant is used
VM	Determine how many versions of a variant are available
VM	Determine the internal/external variability of the product line
VM	Determine the dependencies of a particular variant to other variants
PD	Derive the design for a product variant based on a set of requirements
PD	Derive the design of the minimum product variant (entry level system)
PD	Derive the design of the maximum product variant (high end system)

According to the authors opinion, task support could be best provided by interactive visualisations. Support for different tasks could be achieved by providing different views (e.g., one view to explore structural information and another view to explore content details). In addition, each view could be adjustable for different purposes, as each task has different information requirements.

5.3 User Interactions

Interaction is important to get the most out of a visualisation. Different interactions produce different results, and are effective for different tasks. Simple actions such as zooming and rotation of a visualisation can greatly increase its effectiveness at communicating information.

One of the more powerful possible user interactions are dynamic queries. These support information seeking by allowing the user to adapt a visualisation and to observe the effect of the adaptation immediately (see [7], p. 235). The adaptation is accomplished by adjusting visualisation parameters through user interface elements. A typical application of dynamic queries is the filtering of the set of objects displayed on the screen. For example, the user may influence the set of visible objects by selecting and unselecting check boxes or by moving a slider. A possible use of dynamic queries for the visualisation of variability is allowing the user to turn on and off entities and relationships based on their type and on attribute values (e.g., represent all features of a certain priority that are not assigned to a component; or represent all features that are assigned to product X and product Y but not to product Z).

6. Future Work

The authors envisage their future work in this area to broadly follow the visualisation reference model introduced in Section 4. This will involve the identification of product line data required to be of assistance for product line stakeholders for variability management and product derivation and storing it in an appropriate format.

This data then needs to be mapped to a selection of visual structures (such as those discussed in Section 5). However, an adaptation of existing structures and the development of new visual structures are expected in order to present the data in a coherent manner. It would be desirable that a visual structure can be selected interactively rather than arbitrarily presented as the “best” solution to a given problem.

Views on these structures will then need to be provided to support the identified product line tasks. It is thought that the view transformations to be applied to the visual structures will require interaction (cf. Section 5.3) to provide the task support necessary.

7. Conclusions

This paper has elaborated on the idea of using information and software visualisation techniques to support software product line activities such as variability management and product derivation. Current techniques for representing variability were introduced and it has been discussed that these techniques have trouble scaling to the levels required to allow for their use in the medium and large scale product lines that are prevalent in industrial applications.

Further, a reference model that helps to frame the visualisation research in software product lines has been presented. The model helps to map out the use of techniques from visualisation research to assist with managing variability and product derivation. Promising techniques were mentioned, along with their benefits.

In addition, important areas that could affect the use of these techniques were discussed, and a general road map for moving this research forward was outlined.

In the context of the examples presented, the authors feel that visualisation could be of assistance in many areas. For example, in relation to managing the asset base and the links to current products, visualisation could assist in keeping the required level of understanding as the product line expands. Rather than relying on a paper trail, or on the tacit knowledge and experience of a small number key people, a visualisation toolkit could lower the complexity involved in managing the documentation, application and reuse of assets. However, further research is necessary to explore the full potential of visualisation techniques in the software product line area.

8. Acknowledgements

This work is partially supported by SFI grant no. 03/CE2/I303_1.

References

[1] K. Pohl, G. Böckle, and F. v. d. Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*, 1st ed. New York, NY: Springer, 2005.

[2] M. Steger, C. Tischer, B. Boss, A. Müller, O. Pertler, W. Stolz, and S. Ferber, "Introducing PLA at Bosch Gasoline Systems: Experiences and Practices," Software Product Line Confernce (SPLC-2004), Boston, MA, USA 2004.

[3] K. C. Kang, J. Lee, and P. Donohoe, "Feature-Oriented Product Line Engineering," *IEEE Software*, Vol. 19, No. 4, pp. 58-65, 2002.

[4] G. Halmans and K. Pohl, "Communicating the Variability of a Software Product Family to Customers," *Software and Systems Modeling*, Vol. 2, No. 1, pp. 15-36, 2003.

[5] S. Thiel and A. Hein, "Modelling and Using Product Line Variability in Automotive Systems," *IEEE Software*, Vol. 19, No. 4, pp. 66-72, 2002.

[6] J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, H. Obbink, and K. Pohl, "Variability Issues in Software Product Lines," 4th International Workshop on Product Family Engineering (PFE-4), Bilbao, Spain, October 2001.

[7] S. K. Card, J. D. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*: Morgan Kaufmann Publishers, 1999.

[8] B. A. Price, R. M. Baecker, and I. S. Small, "A Principled Taxonomy of Software Visualization," *Journal of Visual Languages and Computing*, Vol. 4, No. 3, pp. 211-266, 1993.

[9] J. van Gurp, J. Bosch, and M. Svahnberg, "On the Notion of Variability in Software Product Lines," Working IEEE/IFIP Conference on Software Architecture (WICSA 2001), Amsterdam, The Netherlands, August 2001.

[10] I. Jacobson, M. Griss, and P. Jonsson, *Software Reuse. Architecture, Process and Organization for Business Success*: Addison-Wesley, 1997.

[11] S. Bühne, K. Lauenroth, and K. Pohl, "Modelling Requirements Variability Across Produduct Lines," 13th International Conference on Requirements Engineering (RE'05) 2005.

[12] A. G. J. Jansen, R. Smedinga, J. van Gurp, and J. Bosch, "First Class Feature Abstractions for Product Derivation," *IEE Proceedings Software*, Vol. 151, No. 4, pp. 187-197, 2004.

[13] D. Batory, "Feature Models, Grammars, and Propositional Formulas," 9th International Conference on Software Product Lines (SPLC 2005), Rennes, France 2005.

[14] M. Ryan and P. Y. Schobbens, "FireWorks: A Formal Transformation-Based Model-Driven Approach to Features in Product Lines," 3rd Software Product Line Conference (SPLC 2004), Workshop on Software Variability Management for Product Derivation, Boston, MA, August 2004.

[15] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch, "Modeling Dependencies in Product Families with COVAMOF," 13th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2006), Potsdam, Germany, March 2006.

[16] M. Antkiewicz and K. Czarnecki, "FeaturePlugin: Feature Modeling Plug-in for Eclipse," OOPSLA Workshop on Eclipse Technology eXchange, Vancouver, British Columbia, Canada 2004.

[17] D. Beuche, "Variants and Variability Management with pure::variants," 3rd Software Product Line Conference (SPLC 2004), Workshop on Software Variability Management for Product Derivation, Boston, MA, August 2004.

[18] H. Gomaa and M. E. Shin, "Multiple-View Meta-Modeling of Software Product Lines," 8th International Conference on Engineering of Complex Computer Systems (ICECCS 2002) 2002.

[19] A. Maccari and A. Heie, "Managing Infinite Variability in Mobile Terminal Software," *Software - Practice and Experience (SPE)*, Vol. 35, No. 6, pp. 513-537, 2005.

[20] F. van der Linden, "Software Product Families in Europe: the ESAPS & CAFE Projects," *IEEE Software*, Vol. 19, No. 4, pp. 41-49, 2002.

[21] B. Johnson and B. Shneiderman, "Tree-Maps: A Space-Filling Approach to the Visualisation of Hierarchical Information Structures," *IEEE Visualization*, pp. 189-194, 1991.