A Discussion of Three Visualisation Approaches to Providing Cognitive Support in Variability Management

Ciaran Cawley, Patrick Healy, Goetz Botterweck

Lero
University of Limerick
Limerick, Ireland
ciaran.cawley@lero.ie, patrick.healy@lero.ie, goetz.botterweck@lero.ie

Abstract: Variability management in software intensive systems can be a complex and cognitively challenging process. Configuring a Software Product Line with thousands of variation points in order to derive a specific product variant is an example of such a challenge. Each configurable feature can have numerous relationships with many other elements within the system. These relationships can impact greatly on the overall configuration process. Understanding the nature and impact of these relationships during configuration is key to the quality and efficiency of the configuration process. In this paper we present an overview of three visual approaches to this configuration which utilise information visualisation techniques and aspects of cognitive theory to provide stakeholder support. Using an industry example, we discuss and compare the approaches using a set of fundamental configuration tasks.

1 Introduction

Software Product Line (SPL) engineering claims to realise significant improvements in time-to-market, cost, productivity, and system quality [1]. Establishing a core set of assets from which different software product variants can be subsequently derived is the primary principle underlying the expected benefits. Industrial sized product lines are interesting examples of software intensive systems where there remain difficult challenges in terms of variability management [2, 3].

Information Visualisation techniques have provided a variety of ways for stakeholders to view, comprehend and manage large amounts of related information [4, 5]. However, although recent work has attempted to incorporate these into the domain of variability management [6-8], there appears to be a lack of such research in the literature.

Configuring a Software Product Line with thousands of variation points in order to derive a specific product variant is a challenging process. Each configurable feature can have numerous relationships with many other

elements within the system. These relationships can impact greatly on the overall configuration process. Understanding the nature and impact of these relationships during configuration is key to the quality and efficiency of the configuration process.

In his work on *cognitive support* in software engineering, Walenstein developed a framework to guide the design of tools so that they take advantage of principles from *distributed cognition* [9]. In this paper, we use the principles espoused in this framework and attempt to realise them through the employment of a variety of information visualisation techniques in order to provide support for a number of variability configuration tasks.

We present three different visualisation approaches that, as their basis, combine principles from cognitive theory with information visualisation techniques to address variability configuration issues. As well as a traditional 2D approach, a 2.5D and 3D approach is explored. The effectiveness of 2D versus 3D is a widely debated area but work such as that by Risden [15] provides interesting evaluations showing certain situations can benefit from a 3D approach.

The rest of this paper is structured as follows. Section 2 discusses related work. Section 3 presents the concepts underpinning the approaches taken and outlines the tasks that our approaches aim to support. Section 4 presents the implementations of the three approaches taken while Section 5 provides a discussion on how those approaches support the given tasks. The paper finishes with plans for future work and conclusions.

2 Related Work

Feature modelling is a prevalent mechanism for describing variability in SPL's [10]. These models are typically represented using hierarchical tree views or simple graphs. Tools such as pure::variants [11] and Gears [12] are examples that use such representations. Although these views are familiar and intuitive, there is a lack of evidence supporting their effectiveness in relation to large scale product lines.

The DOPLER [7] tool also employs lists and hierarchical trees but allows for more sophisticated graph layouts to be visualised. These, however, follow traditional node-link diagram approaches and do not employ 2.5D or 3D visual environments.

Tools such as VISMOOS [13] and MUDRIK [14], although not variability management tools, offer an insight into the use of 2.5D/3D and the possibilities of increased cognitive support within software engineering

tools. However, these tools concentrate on comprehension alone and not on process support.

Information visualisation techniques described and analysed by Ware [5] and Card et al. [4], offer expert opinion on the application of various visual mechanisms. Ware also offers a *theory of augmented thinking using visual queries on visualisations* - cognitively, constructing a visual query entails identifying a visual pattern that will be used by a mental search strategy over a graphical visualisation.

Walenstein provides a set of principles [9] which aim to guide the design of software engineering tools to maximise cognitive support.

3 Concept

3.1 Visualisation Techniques

The application of visualisation techniques to address the complexity issues that exist in configuring high-variability systems is the core activity of this work. Figure 1 shows the Visual Reference Model [4] described by Card et al. This visual model provides the basis for our visualisation approaches. A data meta-model, outlined in Section 4, was developed to describe a software product line and the relationships that can exist between its various elements. At a high level, an instance of this model comprises the "Data" as illustrated in Figure 1. The primary area of effort in our work is the development of appropriate "Views" that work over an abstracted representation of that data.

Visualisation techniques developed and analysed within the visualisation community [4, 5] are leveraged within our approaches. It is proposed that by using such techniques, the expertise and experience of that community can be brought to bear on the complexity challenges that exist in variability management. For example, some key concepts/techniques used are: Details on Demand; Multiple Synchronised Views and Focus+Context.

In addition, motivated by work such as that carried out by Risden and Robertson [15, 16], these techniques are employed in three different visual

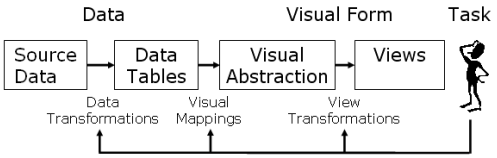

Figure 1. Visualisation Reference Model

environments - 2D, 2.5D and 3D. These three environments allow a comparative evaluation to be performed, showing strengths and weaknesses of each and where one environment might be more useful than others for specific tasks.

3.2 Cognitive Theory

Using a set of principles developed by Walenstein [9] that aim to increase cognitive support in software engineering tools and Ware's theory of augmented thinking using visual queries on visualisations [5], we chose a set of visualisation techniques that aim to realise those principles.

3.3 Relationship-Centric Approach

Literature that reports on industrial issues, requirements and tools [2, 3, 17], motivates an important aspect of our approach - a large percentage of the complexity that challenges the configuration process is due to the quantity and varying nature of the relationships that exist between the different configurable elements and the constraints that these relationships impose. With this in mind, the underlying concept used by our visualisation approaches aims to focus on the relationships that exist and not on what they relate.

3.4 Application

As the end result of this work is to provide support to stakeholders during the feature configuration stages of SPL product derivation, we set out the tasks for which this support is being provided.

The activity of configuring a *feature* is the fundamental task challenging a stakeholder during the feature configuration process. At a basic level, this involves the ability to either include or exclude a *feature* from the product under derivation. We would also add that the ability to include/exclude *features* in groups based on higher level requirements is also a fundamental task (we refer to these higher level requirement groupings as *decisions* later). Whereas these tasks may seem simplistic, it is the knowledge/understanding (cognition) of the stakeholder that allows these tasks to be performed correctly. Again, drawing on work carried by others [2, 3, 17], we outline a set of simple cognitive tasks that aim to support the activity of the primary task – to decide which *features* should be included and which should be excluded.

1. Identify / Locate a configuration *decision*
2. Understand the high-level impact of a *decision* inclusion (perception of scale and nature of the impact - implements/requires/excludes)

3. Identify / Locate a specific *feature*
4. Identify a specific *feature's* context - parent *feature*, alternative/supporting *features*, *sub-features*
5. Understand the high-level impact of a *feature* inclusion - a specific *feature's* constraints (requires/excludes relationships)
6. Identify the state of a *feature* - included/excluded and why.

It is these cognitive tasks that our visualisation approaches target in terms of providing an interactive visual environment. In Section 5 we discuss each of these tasks in the context of the three approaches.

4 Approaches

4.1 Overview

In this section, we firstly enumerate and describe the principles and techniques utilised across all three visualisation approaches. Following this, we briefly outline the data meta-model used to describe the data and introduce the industry example used for our discussion in Section 5. We finally present the three visual environments: 2D; 2.5D and 3D as three different approaches to using visualisation to support feature configuration.

4.2 Visualisation and Cognitive Support

Based on Walenstein's [9] and Ware's [5] work, we use seven principles to guide our choice of visualisation techniques.

1. Simple query patterns
2. Simplified / Reduced interface
3. Reduced tasks
4. Rapid information recovery
5. Optimisation
6. Distribution
7. Specialisation

In summary, the first three are concerned with simplifying the user interface as much as possible. The next two, 4 and 5, are concerned with easy and efficient discovery and calculation of data. The last two, 6 and 7, are concerned with separating data and/or processing ("thinking") so that it is cognitively easier to understand that data.

The use of easy to identify/learn *iconography* and *colour encoding* is used to help realise principles 1 and 2. Using icons like ticks, crosses and coloured icons allow multiple data items to be encoded in the visualisation.

The *multiple windows* technique helps to realise principles 2, 6 and 7. Using multiple windows, information can be separated in a way so as to reduce individual views and group related information.

Focus+Context techniques such as *fisheye* / *degree of interest* help to realise principle 2. These techniques are concerned with providing the overall context within which one is working while still allowing a stakeholder focus on specific tasks/data elements.

Details on Demand and *Distortion* help to realise principles 2 and 4. These techniques aim to hide/distort uninteresting information at any given time but allow easy and rapid unveiling of that information when required.

Pan & Zoom also helps realise principles 2 and 4 simplifying navigation around large data displays and allowing focus and exploration of specific data areas.

4.3 Meta-Model

The data meta-model briefly mentioned in Section 3 is used as the basis for our visualisation approach. It consists of three separate but integrated meta-models and describes a product line in terms of ***D**ecisions*, ***F**eatures* and ***C**omponents*:

- A *decision* model captures a small number of high-level questions and provides an abstract, simplifying view onto *features*.
- A *feature* model describes available configuration options in terms of “prominent or distinctive user visible aspects, qualities, or characteristics” [18].
- A *component* model describes the implementation of *features* by software or hardware components.

These three models are interrelated. For instance, making a *decision* might cause several *features* to become selected, which in turn require a number of *components* to be implemented. The details of this meta-model are out of scope for this paper and the interested reader is guided to a previous publication [19] for further information.

To evaluate and provide an example for the purpose of discussing the visualisation approaches, we have created such a DFC model using the configuration database of a large industrial system provided by a commercial software development company. This example contains a subset of the

configuration at approximately 1500 *features* and 1000 implementing *components*. The system itself is a transport management system for large companies such as freight forwarders.

One high level function of this system is to provide transportation documents that are required when moving goods internationally. This functionality is modelled as a *decision* with eight implementing *features*. One such implementing *feature* is "Commodities". This *feature* provides the ability to use and maintain sets of commodity codes that are required to identify the types of goods being transported for documentation purposes. This *feature* requires three other distinct *features* to be included. It is this example *decision* that we use to illustrate the visualisations below.

4.4 2D Approach

As discussed in the related work section, using 2D approaches such as matrices and graphs to visualise *feature* models is the traditional way to allow *feature* exploration and model manipulation [7, 20]. In our 2D approach we provide a linear horizontal tree as the basis upon which we apply a number of visualisation techniques to support the configuration process. The tree view was implemented using the *prefuse* visualisation toolkit [21].

Figure 2 presents a screenshot from our Eclipse based [22] tool showing our 2D visualisation. In all three approaches, a supporting synchronised view is used. This view in the left of the figure presents a simple list view of the *decisions* that identify the high level functionality/requirements that the system implements.

Through selection of a *decision* in the supporting view by mouse-click, the main tree view in the centre of the figure displays all implementing *features*, their location within the *feature* model and their immediate sub-*features*. *Animation* is employed during the tree view transition from its previous visual state to preserve the context. The tree itself is a *degree of interest* tree and automatically displays *features* of interest (path to current node, sibling nodes and child nodes) to the current selection and hides all other *features*. The combination of multiple windows and *Degree of Interest* aim to provide the *Focus+Context* described earlier.

Colour encoding is employed to highlight what *features* directly implement (amber) the selected *decision* and what *features* are required (blue) or excluded (red) by those implementing *features*. A colour encoded icon (sphere) to the left of the label of a highlighted *feature* identifies if the

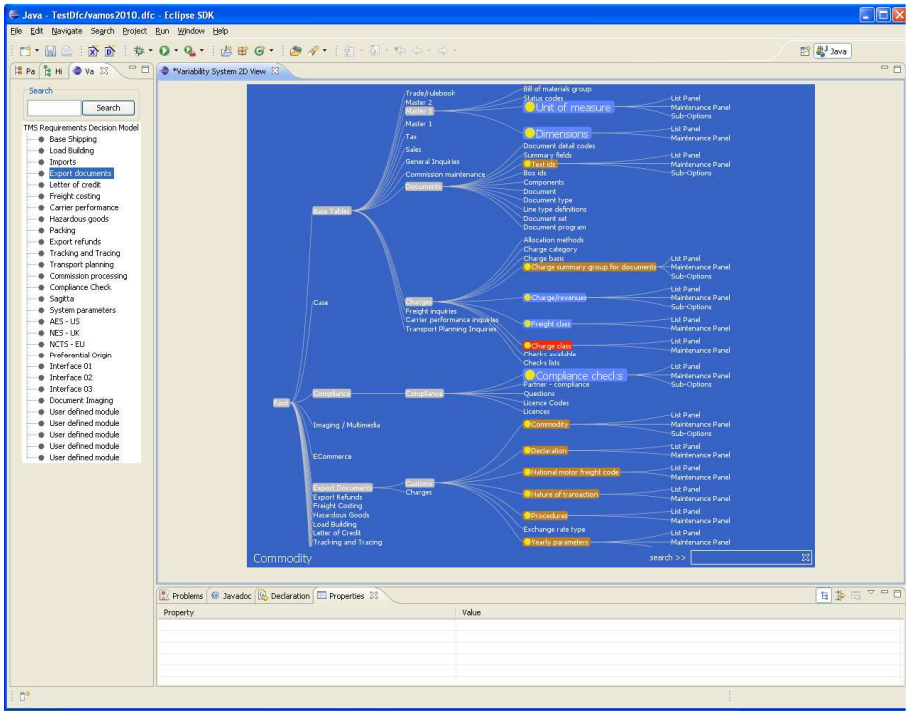

Figure 2. 2D Visualisation Approach

feature has been included (green), eliminated (grey) or is un-configured (yellow).

The stakeholder can explore the tree through mouse-clicks on nodes of interest. Again the tree, using smooth animation, automatically expands and collapses nodes depending on the selected node of interest. The collapsing/hiding of nodes while exploring the tree can be stopped at the will of the stakeholder to allow manual collapsing and expanding of branches. Using the mouse, the stakeholder can perform full *zoom* and can also *pan* the entire tree in any direction. These functions aim to implement the *Details On Demand* principle.

4.5 2.5D Approach

2.5D is a term that describes the use of 3D visual attributes in a 2D display [23]. For example, adding 3D attributes such as perspective (e.g. making certain objects smaller to indicate distance) and occlusion (e.g. overlapping objects to indicate layers) to a 2D display can be described as creating a 2.5D display.

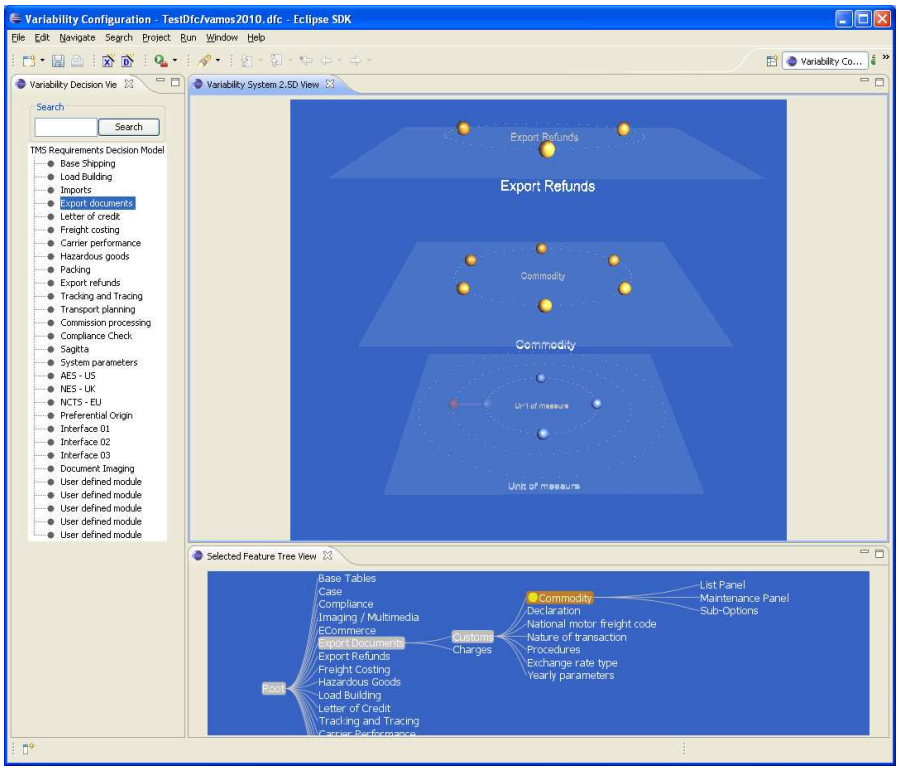

Figure 3. 2.5D Visualisation Approach

Figure 3 presents our 2.5D view. Again, when a selection is made within the supporting *decision* view, the main view displays the implementing *features* along with all *features* that are required or excluded by them.

The view, inspired by Robertson et al.'s cone trees [24], consists of three stacked planes. Each plane provides a circular grouping of spheres. In the top plane, each sphere in the circle represents a grouping of *features*. When any one of those groupings in the top plane is selected (by mouse-click) then all *features* that comprise that grouping are displayed in the middle plane in a similar circular format. In the lower plane, all related (required / excluded) *features* are displayed (for *all features* presented in the middle plane). The innermost circle on the lower plane identifies *features* that are directly related (required, excluded) to *features* in the middle plane. In order of ascending radii, each subsequent circle in the lower plane represents the transitive relationships that exist i.e. required *features* can further require and/or exclude other *features*. In Figure 3 the stakeholder has selected the "Export Refunds" grouping in the top plane which groups six *features*. These six *features* are represented on the middle plane while their related *features* (required, excluded) are represented on the lower plane.

By *hovering* the mouse over any sphere in any of planes, a description of that element will be displayed in the centre of the plane. When a sphere is *selected* in any plane, the circle on which it is presented will rotate so that that sphere is brought to the front with its description displayed underneath. These functions aim to implement *Details on Demand*.

The *colour encoded* sphere acts as the representation of a *feature* and its relationship. An amber sphere indicates a *feature* that implements the current *decision* selection. A blue sphere indicates a required *feature* while a red sphere indicates an excluded *feature*.

Multiple windows (and multiple planes) are employed to separate and distribute *decisions*, *feature* groupings, *features* and *relationships*.

Note that the lower plane displays *all* related *features* for all the *implementing features* in the middle plane. This allows an overview of the impact as a whole for this group of *features*. When a single implementing *feature* is selected in the middle plane, the circles in the lower plane rotate to ensure all related *features* are brought to the front while all other *features* in the plane are *distorted* (made transparent) in order to highlight the ones of interest. *Animation* is again used for all movements to preserve context.

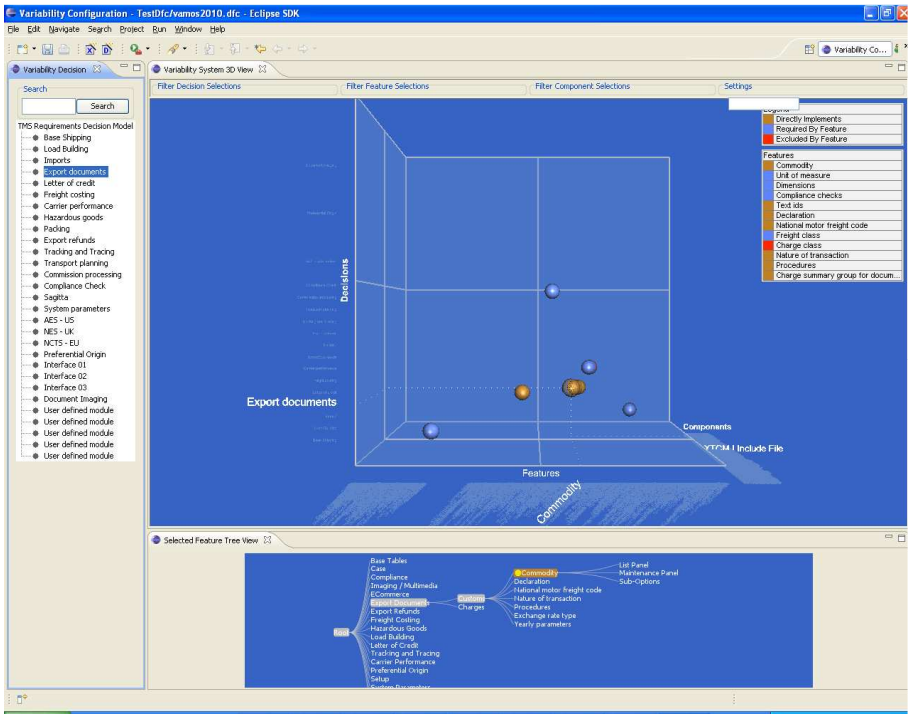

Figure 4. 3D Visualisation Approach

4.6 3D Approach

Differing reports exist on the effectiveness of 3D visualisations to support software engineering but literature suggests that there is acceptance that it can be effective in specific instances [14-16].

Figure 4 presents a 3D view which attempts to provide a self contained representation of all three models (*decisions*, *features* and *components*) and their inter-relationships. However, at any given time, only information of interest is displayed.

As before, *multiple windows* are employed to distribute the information and provide the supporting *decision* view.

Figure 4 consists of a 3D space containing X, Y and Z axes. A sequential list of the *decisions* is displayed along the vertical Y-axis, a sequential list of the *features* is displayed along the horizontal X-axis and a sequential list of all the *components is* displayed along the Z-axis (moving away from the observer).

The key idea here is that a point within this 3D space identifies a relationship between all three models. In other words, a sphere plotted at a particular

point will identify that the *feature* labelled at its X co-ordinate implements the *decision* labelled at its Y co-ordinate and is implemented by the *component* labelled at its Z co-ordinate. In Figure 4, the stakeholder has highlighted the sphere that represents the "Commodities" *feature*. However, in addition to this, by looking at the highlighted labels on the axes, we can see that it also represents the "Export Documents" *decision* that the *feature* implements and the "XTCM.I Include File" *component* that implements the *feature*.

Focus+Context and *Details On Demand* are the main techniques guiding this implementation. We argue that all three models can be perceived to be represented through the listings on each axis. However, the details of any part of any model or its relationships are only displayed when required. For example, when a *decision* is selected there can be a number of implementing *features*. For each implementing *feature*, a sphere is plotted in the 3D space as described above. Other *features* that are required or excluded by those implementing *features* are also similarly plotted as spheres and are given a specific *colour encoding* - required *features* are blue and excluded *features* are red.

Pan & Zoom are combined with *rotation* to allow a full *world-in-hand* manipulation of the view in three dimensions letting the stakeholder position the view depending on the information of interest.

5 Discussion

Our discussion follows a line of argument of how each task identified in Section 3 is supported by each visualisation approach, and discusses the benefits and limitations.

5.1 Identify / Locate Decisions and Features

In all the approaches presented, the supporting *decision* view (displayed to the left of the main views in the figures) provides a simple list of the high level requirements *decisions* that represent the system functionality. Each *decision* groups a set of *features* that satisfy a particular functional need. In our industry example illustration, the "Export Documents" *decision* groups the set of *features* that combine to provide the production of printed documentation to allow the movement of goods. To aid fast identification / location of particular *decisions*, a search field and button are provided which when used will highlight any textual matches.

Further to this, within the 3D approach, the main view also contains a search field where the stakeholder can run a search for *decisions*, *features* or

components. When searching for a *decision*, the resulting matches are highlighted using increased brightness and enlarging of the text of the corresponding Y-axis labels.

When searching for a *feature* within the 3D view, each matching result is rendered as a sphere, identifying the *feature* and also its implemented *decision* and implementing *component*. The stakeholder may also choose to see all required and excluded *features* of the *features* returned by the search. Similarly, the 2.5D view will display a matching *feature* on the middle plane, however, currently it is restricted to only showing the first exact match. This is a limitation that can be addressed in future work.

5.2 Impact of Decision / Feature Selection

When a *decision* is selected, a stakeholder needs to understand what the impact would be on the system in terms of what *features* implement the *decision*, what *features* would subsequently be required and what *features* would be removed from the configuration.

In the example introduced in Section 3 and illustrated in Figure 2, Figure 3 and Figure 4 the stakeholder has chosen the “Export Documents” *decision* which has eight implementing *features*.

In the 2D approach (Figure 2), these can be clearly identified by the amber highlighting. Required and excluded *features* can also be easily identified through their blue and red highlighting respectively. By hovering the mouse over any particular *feature* of interest, all related *features* are further highlighted through animated enlargement allowing identification of the required and excluded *features*. Although the tree visualised in this approach is a degree of interest tree whereby only the nodes of interest are displayed, we would suggest that dealing with more than, say, 20 implementing *features*, each requiring and excluding other *features*, would increase the cognitive difficulty - the stakeholder would need to keep a mental map while panning and zooming.

In the 2.5D approach (Figure 3), when the stakeholder selects a *decision*, the top plane in the view is populated with groupings of *features*. In this view the stakeholder can then choose which group of *features* to investigate further, once a group is selected, the implementing *features* are populated onto the middle plane and all their related (required, excluded) are populated onto the lower plane. In Figure 3, the stakeholder has chosen the “Export Refunds” grouping and we can see that there are six implementing *features* which require another four *features* and exclude one. Although there is an additional step in this approach (stakeholder must choose a grouping of *features*), we argue that through this grouping, many more *features* could be represented without increasing the cognitive effort required.

In the 3D approach (Figure 4), when the stakeholder selects a *decision* (or multiple *decisions*), all implementing *features* and their required and excluded *features* are displayed as colour encoded spheres. In addition to this, the implementing *component* of all the *features* is also identifiable on the Z-axis. By hovering over any element of interest (spheres, axis labels), that particular element is highlighted using a number of techniques. In Figure 4, the stakeholder is interested in the "Commodity" *feature*, which has been highlighted along with its implemented *decision* ("Export Documents") and implementing *component* ("XTCM.I Include File"). By performing a mouse-click on any *feature* (sphere), all related *features* (required, excluded) are also highlighted. Using this approach, we would argue that it is easy for the stakeholder to comprehend the overall impact of selecting the *decision* and to also follow that up in order to understand what the impact of including any particular *feature* would be.

5.3 Feature Context Comprehension

The understanding of a *feature's* context (parent *feature*, alternative *features*, *sub-features*) is an important cognitive ability during its configuration. This understanding can inform the stakeholders as to what alternatives might be selected or what sub-options are available and how they would alter the impact of a particular configuration.

Due to the tree visualisation employed as part of the 2D approach, the context is immediately identifiable. The degree of interest, by default, will always show parent, sibling and children of the node of interest.

In both the 2.5D (Figure 3) and 3D (Figure 4) approaches, a supporting tree view along the bottom of the screen is employed which displays the feature of interest in a hierarchical tree. The tree shows the feature's path from the root node along with the sibling nodes of any node on that path. It also shows the child nodes of the node of interest.

As all the approaches essentially employ the same mechanism for this task, there is no evident behaviour that makes one approach better than the other. All three approaches allow quick and easy identification of the feature's context.

6 Future Work

Additional implementation work is planned for both the 2.5D and 3D visualisation approaches. In both approaches, the use of colour encoding will be employed to identify the state of a feature (included/eliminated/unconfigured). Within the 2.5D approach, improved

text rendering and highlighting of selected features is planned. Within the 3D approach, improved text rendering/fisheye will be implemented along with the ability to dynamically separate clustered features.

Following these improvements, an evaluation of all three approaches is planned through the elicitation of expert opinion from senior practitioners in the area of large systems development and management.

7 Conclusion

In this paper we have presented and discussed three visualisation approaches to understanding and configuring variability in a software intensive industry example. We combine principles from cognitive theory and information visualisation techniques and use 2D, 2.5D and 3D visual environments in their implementation. We argue that each approach has benefits and limitations when considering a set of cognitive tasks that support the configuration process. We outline our plans for future work and further evaluation.

8 Acknowledgment

This work is partially supported by Science Foundation Ireland under grant number 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. ed. New York: Springer, 2005.

[2] S. Deelstra, M. Sinnema, and J. Bosch, "Product Derivation in Software Product Families: A Case Study," *Journal of Systems and Software*, vol. 74, pp. 173-194, 2005.

[3] M. Steger, C. Tischer, B. Boss, A. Müller, O. Pertler, W. Stolz, and S. Ferber, "Introducing PLA at Bosch Gasoline Systems: Experiences and Practices," in *SPLC 2004*, Boston, MA, USA, 2004, pp. 34-50.

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

[5] C. Ware, *Information Visualisation: Perception for Design*, 2nd ed.: Morgan Kaufmann, 2004.

[6] F. Heidenreich, I. Savga, and C. Wende, "On Controlled Visualisations in Software Product Line Engineering," in *2nd International Workshop on Visualisation in Software Product Line Engineering (ViSPLE 2008)* Limerick, Ireland, 2008.

[7] R. Rabiser, D. Dhungana, and P. Grünbacher, "Tool Support for Product Derivation in Large-Scale Product Lines: A Wizard-based Approach," in *1st International Workshop on Visualisation in Software Product Line Engineering (ViSPLE 2007)* Tokyo, Japan, 2007.

[8] D. Sellier and M. Mannion, "Visualizing Product Line Requirement Selection Decisions," in *1st International Workshop on Visualisation in Software Product Line Engineering (ViSPLE 2007)* Tokyo, Japan, 2007.

[9] A. Walenstein, "Foundations of cognitive support: Toward abstract patterns of usefulness.," in *9th International Workshop on Design, Specification and Verification of Interactive Systems (DSVIS)*: Springer-Verlag, 2002.

[10] K. C. Kang, J. Lee, and P. Donohoe, "Feature-Oriented Product Line Engineering," *IEEE Software*, vol. 19, pp. 58-65, 2002.

[11] pure-systems GmbH, "Variant Management with pure::variants," http://www.pure-systems.com, Technical White Paper, 2003-2004.

[12] Biglever Software, "Gears," http://www.biglever.com.

[13] O. Rohr, "VisMOOS (Visualization Methods for Object Oriented Software Systems)," University of Dortmund, http://ls10-www.cs.uni-dortmund.de/vise3d/prototypes.html, 2004.

[14] J. Ali, "Cognitive support through visualization and focus specification for understanding large class libraries," *Journal of Visual Language and Computing*, 2008.

[15] K. Risden, M. P. Czerwinski, T. Munzner, and D. B. Cook, "An initial examination of ease of use for 2D and 3D information visualizations of web content," *Int. J. Human-Computer Studies*, pp. 695-714, 2000.

[16] G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins, "Polyarchy Visualization: Visualizing Multiple Intersecting Hierarchies," in *Conference on Human Factors in Computing Systems* Minneapolis, Minnesota, USA.: ACM, 2002.

[17] M. Sinnema, O. d. Graaf, and J. Bosch, "Tool Support for COVAMOF," in *Workshop on Software Variability Management for Product Derivation - Towards Tool Support*, 2004.

[18] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21," Software Engineering Institute, Carnegie Mellon University 1990.

[19] G. Botterweck, S. Thiel, D. Nestor, S. B. Abid, and C. Cawley, "Visual Tool Support for Configuring and Understanding Software Product Lines," in *The 12th International Software Product Line Conference (SPLC08)* Limerick, Ireland, 2008.

[20] C. Cawley, D. Nestor, A. Preußner, G. Botterweck, and S. Thiel, "Interactive Visualisation to Support Product Configuration in Software Product Lines," in *Proceedings of 2nd International Workshop on Variability Modeling of Software-Intensive Systems (VAMOS 2008)* Essen, Germany, 2008.

[21] J. Heer, S. K. Card, and J. A. Landay, "prefuse: a toolkit for interactive information visualization," in *Conference on Human Factors in Computing Systems* Portland, Oregon, USA: ACM New York, NY, USA, 2005.

[22] "Eclipse IDE," http://www.eclipse.org.

[23] C. Ware, "Designing with a 2 1/2D Attitude," *Information Design Journal*, vol. 3, pp. 255-262., 2001.

[24] G. G. Robertson, J. D. Mackinlay, and S. K. Card, "Cone Trees: animated 3D visualizations of hierarchical information," in *Conference on Human Factors in Computing Systems* New Orleans, Louisiana, United States: ACM New York, NY, USA, 1991.