Research Tool to Support Feature Configuration in Software Product Lines

Ciarán Cawley, Patrick Healy, Goetz Botterweck
Lero
University of Limerick
Limerick, Ireland
{ ciaran.cawley | patrick.healy | goetz.botterweck } @lero.ie

Steffen Thiel
Department of Computer Science
Furtwangen University of Applied Sciences
Furtwangen, Germany
steffen.thiel@hs-furtwangen.de

Abstract— **Configuring a large Software Product Line can be a complex and cognitively challenging task. The numerous relationships that can exist between different system elements such as features and their implementing artefacts can make the process time consuming and error prone. Appropriate tool support is key to the efficiency of the process and quality of the final product. We present our research prototype tool which takes a considered approach to feature configuration using visualisation techniques and aspects of cognitive theory. We demonstrate how it uses these to support fundamental feature configuration tasks.**

Keywords-visualisation; variability management; software product lines;

I. INTRODUCTION

Configuring a Software Product Line (SPL) 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 [1].

Information Visualisation techniques have provided a variety of ways for stakeholders to view, comprehend and manage large amounts of related information [2, 3]. However, although recent work has attempted to incorporate these into the domain of variability management [4-6], there appears to be a lack of their explicit consideration in current tools.

In this paper, we present a research prototype tool, which combines aspects of cognitive theory with specific visualisation techniques to provide alternative interactive views on the underlying data.

II. TOOL

The tool has been implemented as an Eclipse Plugin [9] providing a set of synchronised views that allow the loading, exploration, comprehension and manipulation of the underlying data models. These interactive views are designed with the aim of providing cognitive support to the stakeholder during feature configuration. Three distinct approaches have been employed - 2D, 2.5D and 3D.

A. Meta-Model

A data meta-model 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" [11].
- 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 implementing *features* to become selected, which in turn require a number of *components* to be implemented. The meta-model also defines intra-model relationships such as *feature requires feature* or *feature excludes feature*. The details of this meta-model are out of scope for this paper and the interested reader is guided to a previous publication [10] for further information.

B. Task Support

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 (*decisions*) is also a fundamental task. Whereas these tasks may seem simplistic, it is the knowledge/understanding (cognition) of the stakeholder that allows these tasks to be performed correctly. Drawing on work carried out by others [1, 12], 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.

C. *Interactive Views*

1) 2D Approach: Using 2D approaches such as matrices and graphs to visualise feature models is the traditional way to allow feature exploration and model manipulation [5, 10]. 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 [13].

Figure 1 presents a screenshot from our Eclipse [9] based tool showing our 2D visualisation. For this 2D approach (and also for the subsequent 2.5D and 3D 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 *Focus+Context*.

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 *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

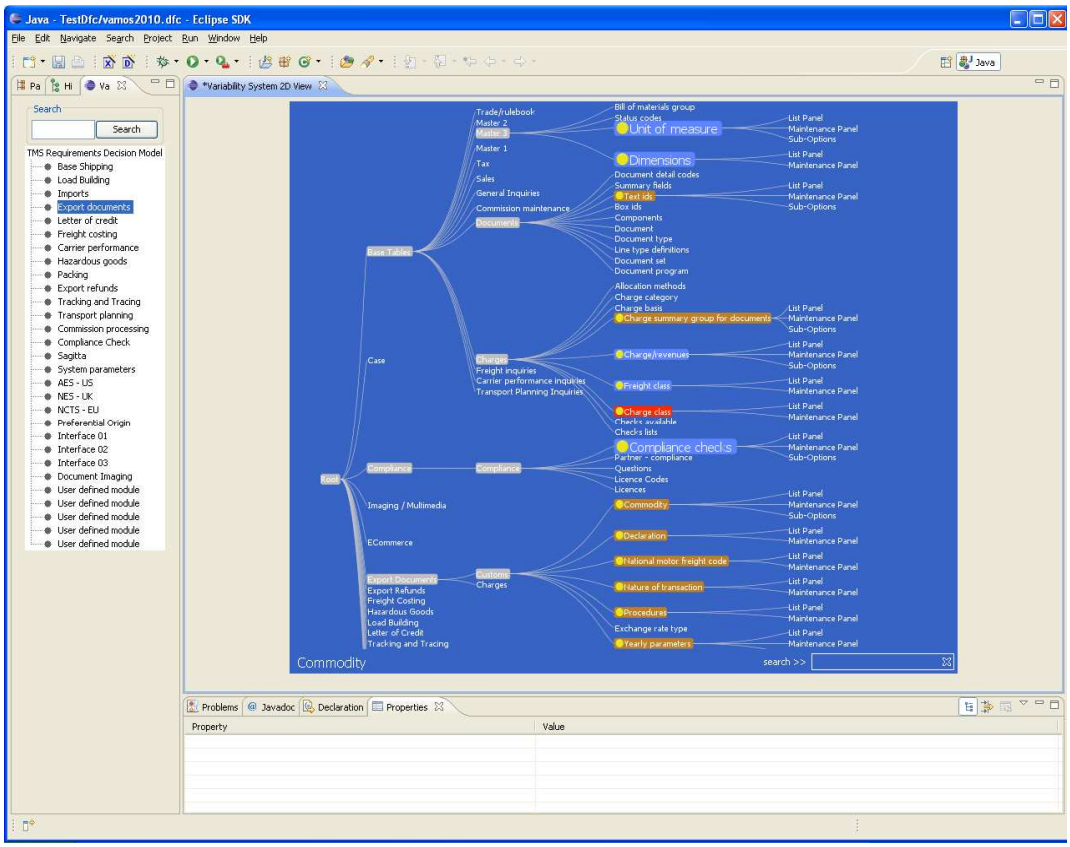

Figure 1 2D Tree View

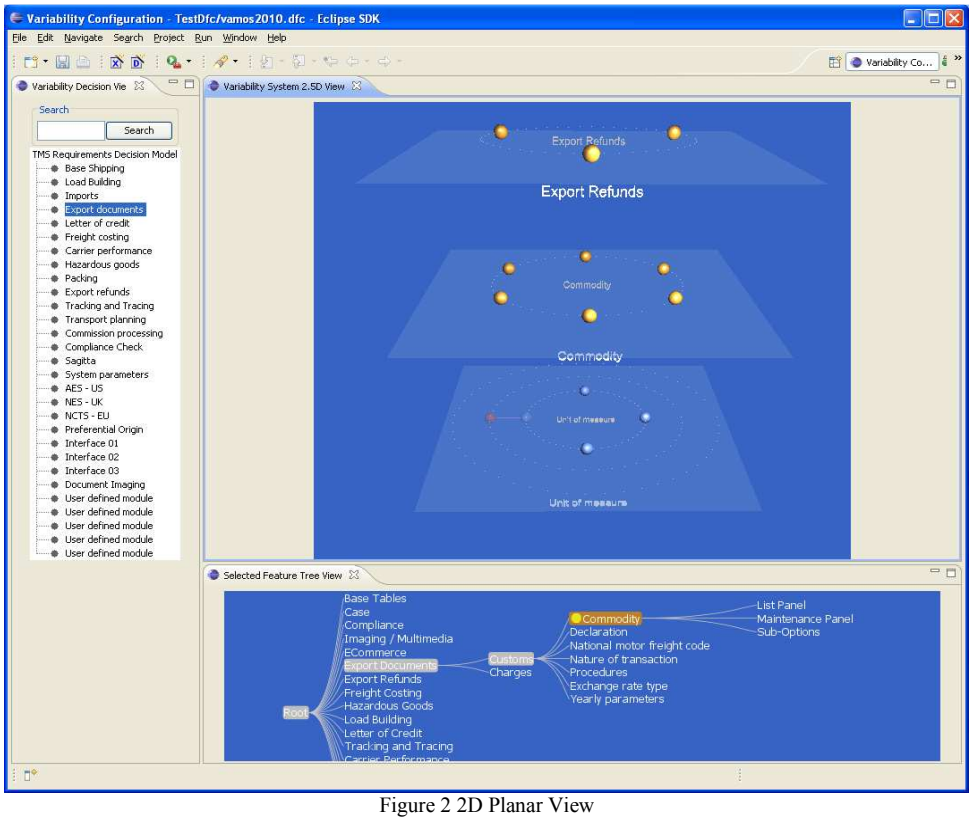

Figure 2 2D Planar View

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.

2) 2.5D Approach: 2.5D is a term that describes the use of 3D visual attributes in a 2D display [14]. 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 2 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 [15], 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 2 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*.

3) 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.

Figure 3 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.

Multiple windows (not shown) are employed to distribute the information and provide the supporting *decision* view.

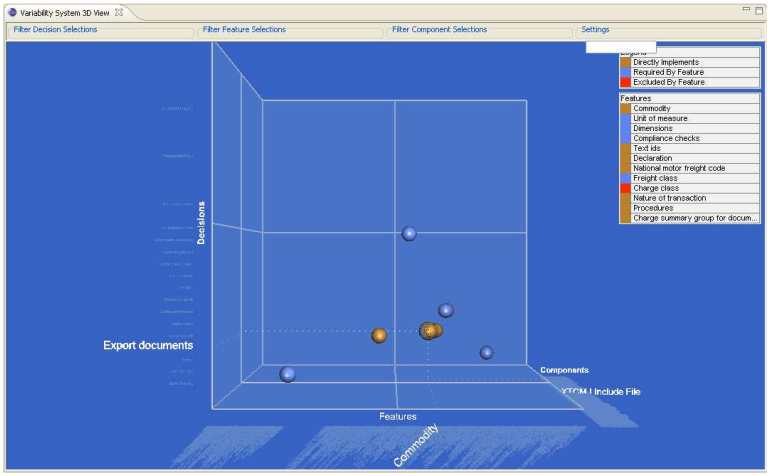

Figure 3 3D Plot View

Figure 3 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* along the horizontal X-axis and a sequential list of all the *components* 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 3, 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.

III. CONCLUSION

In this paper we have presented a research tool prototype that employs aspects of cognitive theory and visualisation techniques to support some of the fundamental but challenging tasks that exist when configuring large software product lines.

ACKNOWLEDGMENT

This work is partially supported by Science Foundation Ireland under grant number 03/CE2/I303-1.

REFERENCES

[1] 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.

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

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

[4] 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.

[5] 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.

[6] 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.

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

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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, 2005.

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

[15] 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.