Visualizing Variability Models Using Hyperbolic Trees

R. Bashroush, A. Al-Nemrat, M. Bachrouch, H. Jahankhani

School of Computing, IT and Engineering,
University of East London,
London, United Kingdom
{rabih, ameer, hamid2}@uel.ac.uk

Abstract. Software Product Line Engineering (SPLE) has emerged in recent years as a viable way to maximize reuse when designing a family of related products. One of the main tasks conducted during the SPLE process is Variability Management (VM). VM is about identifying commonality among the different products being developed while capturing and cataloging variability. In real-life projects, VM models tend to encompass a very large number of variants reaching in many projects the order of thousands. Visualizing these models has been a major challenge for tool developers. In this work, we present our MUSA CASE tool which uses hyperbolic trees for representing VM models and supports gesture based interaction (using multi-touch interfaces). The tool has been successfully used to develop a large scale case study.

Keywords: Software Product Lines, Variability Management, Feature Modeling, Hyperbolic Trees.

1 Introduction

Software Product-line Engineering (SPLE) has emerged as a major strategy for maximizing reuse when a family of related software systems is developed. In this approach, commonality-variability analysis [1] (Variability Management - VM) of the member products is a major phase of the process and plays an important role in its success.

One of the main challenges within VM is the handling and visualizing "industry-size" models which usually comprise a large number of variability points along with the dependency relationships that exist among them. The challenge comes from the large amount of information captured within a model (business related, dependency and relationships, etc.) as well as the current techniques and I/O devices used to visualize the model which do not inherently scale [13].

The MUSA CASE tool was designed to overcome these challenges [13], [14]. MUSA is based on our successful work on multiple-perspective based variability management which provides a rich modeling framework while using the concept of separation-of-concerns to alleviate the problem of information overloading. MUSA implements this theory using a mind-mapping modeling approach, hyperbolic trees, over the state-of-the-art in HCI, the multi-touch Microsoft Surface [2]. This provides

a scalable solution that taps on the latest in Natural User Interface (NUI) [3] design providing an intuitive and large display for VM. In addition, the MUSA solution provides interfaces over other multi-touch platforms including Windows 7 (using its native multi-touch support).

The theory behind MUSA is highlighted in section 2. An overview of the MUSA CASE tool is then presented in section 3. Finally, section 4 ends with related work and conclusion.

2 Technical Background

The Four Views Model (4VM) forms the theoretical foundation upon which MUSA is designed as a Proof-of-Concept. The original version of the 4VM can be found here [4] and to appear here [5].

It is generally agreed that different stakeholders have interest in considering different views of the product line variability model [4],[6]. So, it is important for a VM mechanism to be able to extract and present relevant information about the family model in dedicated views for different groups of stakeholders (users, system analysts, developers, etc.). This could considerably contribute to alleviating the graphical overload when showing all the information in one view (as compared to using multiple views). This is one of the core concepts behind 4VM.

The 4VM proposes a four view presentation of the feature model which are discussed below.

2.1 Business View

The Business View is aimed at the project business and management stakeholders. It acts as a portal for inputting and presenting information related to:

- *Feature implementation time*: This indicates when a given feature is to be implemented. Planning for future releases of products, the features to be implemented in these products, and the timing, is a key step for the success and sustainability of a product line
- *Feature Cost/Benefit analysis*: information related to the effort needed and cost involved in realizing features as well as their foreseen benefit. This provides valuable input to the overall project costing and the product versioning process
- Open/Closed sets of features: it is rarely the case that the architect is furnished with a system's comprehensive and complete set of features upfront. Rather, features are continuously added (and modified) to the initial feature model over time. Designing a system around an open and changing set of features is a very challenging task. To overcome this problem, some industries designate some features as closed, meaning that they can't be changed (core features), while others are designated as open, meaning they can be modified by developers.
- Negative features: these are the features that are not mean to be supported by the system (e.g. for security reasons) as opposed to supported features.

These properties are usually specified and used by the project managers to carry out system-wide business analyses which support decision making such as when to introduce features within a product line; what features are feasible from a business perspective, etc.

2.2 Hierarchical & Behavioral View

The Hierarchical and Behavioral View is the view provided by most existing feature modeling techniques. In this view, information related to the structure of the feature model and the behavior of the features is captured. Among other potential users, this view is mainly targeted at architects and developers.

2.3 Dependency and Interaction View

Due to the size and complexity of feature dependency and interaction within real-life systems, a separate view is created within the 4VM to model these relationships. The Dependency and Interaction View is complementary to the Hierarchical and Behavioral View. We define feature dependency and feature interaction as follows:

- *Feature Dependency*: a feature-to-feature dependency where the inclusion of one or more features affects one or more features within the system.
- *Feature Interaction*: a feature-to-architecture dependency where the inclusion of one or more features affects the architecture structure (different component sets and/or configurations, etc.).

In this view, logic design is proposed to capture the dependency and interaction relationships. Once the relationships are modeled, standard logic algorithms (and SAT solvers) can be used to simplify the models.

2.4 Intermediate View

Finally, the intermediate view has been introduced in an attempt to bridge the gap between feature modeling and the architecture design. This gap exists between the two domains due to the fact that the feature model is based on end-user and stakeholder concerns while the architecture structure is designed to accommodate technical concerns.

To bridge this gap, the intermediate view proposed attempts at injecting design decisions into the feature model to take it one step further towards the architecture domain. As such, it may be regarded as an intermediate stage between feature model and system architecture.

3 Implementation

MUSA was funded as a Proof-of-Concept project to demonstrate the theoretical foundation provided in 4VM. The MUSA system provides an end-to-end variability

management solution as shown in Figure 1 below. MUSA provides a rich and collaborative interface to elicit and manage requirements and variability from stakeholders while allowing for appropriate access to the variability model to different teams including: implementation, testing and deployment teams. In addition, MUSA automates model verification (with the use of SAT solvers) and maintains consistency among the different views with the help of a centralized Database (as shown in Figure 1). MUSA is considered among the very first CASE tools to move into the NUI space in order to overcome scalability issues.

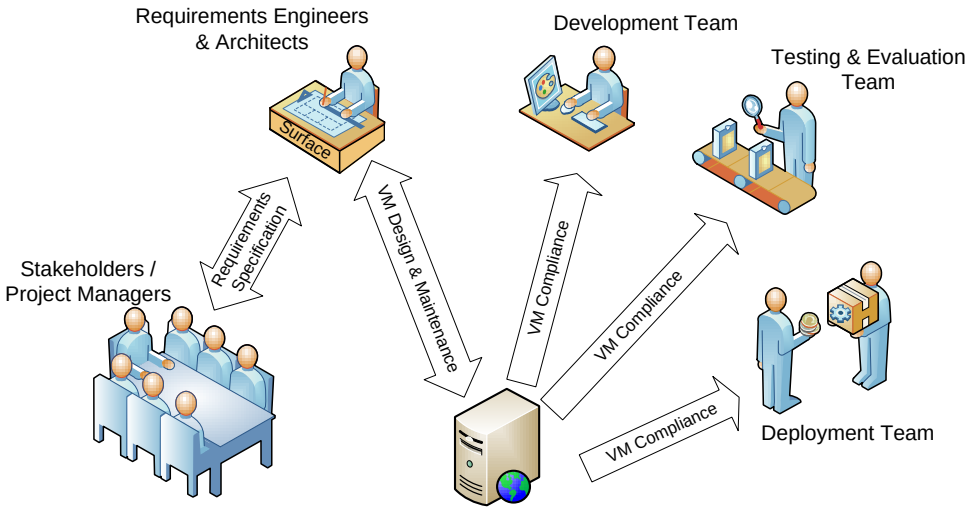

Fig. 1. The end-to-end MUSA System overview

For example, with MUSA, users can user different gestures such as: pinching (for expanding nodes), panning (by moving two fingers on the screen to shift the model), three finger gesture (to center the model at the root node), etc.

In addition, one of the main advantages of MUSA over other CASE tools within the domain of VM (see next section) is scalability. This is made possible with the adoption of hyperbolic trees [15] to represent VM rather than normal trees and other structures within the Euclidean space.

Hyperbolic trees (a.k.a. hypertree) is a visualization method that maps graphs into the hyperbolic geometry. The result effect is similar to a fish-eye lens view where nodes in focus are placed in the center and given more room, while out-of-focus nodes are compressed near the boundaries. Focusing on a different node brings it and its children to the center of the screen, while compressing out of focus nodes.

The advantage of this is that the standard tree suffers from visual clutter when the number of child nodes grow exponentially (in the order of 2^n for binary trees and much quicker for other types of trees), thus, requiring an exponential amount of space

to be displayed appropriately. However, hyperbolic trees employ hyperbolic space which provides more room compared with Euclidean space. This is because increasing objects' size in Euclidean space would cause objects to increase linearly in size compared to hyperbolically in the hyperbolic space [15].

Figure 2 (using the MS Surface) and figure 3 (using Windows 7) below show an example VM of a case study developed with the MUSA toolset. In these figures, we notice color coding is used to distinguish between optional (blue) and mandatory features (yellow).

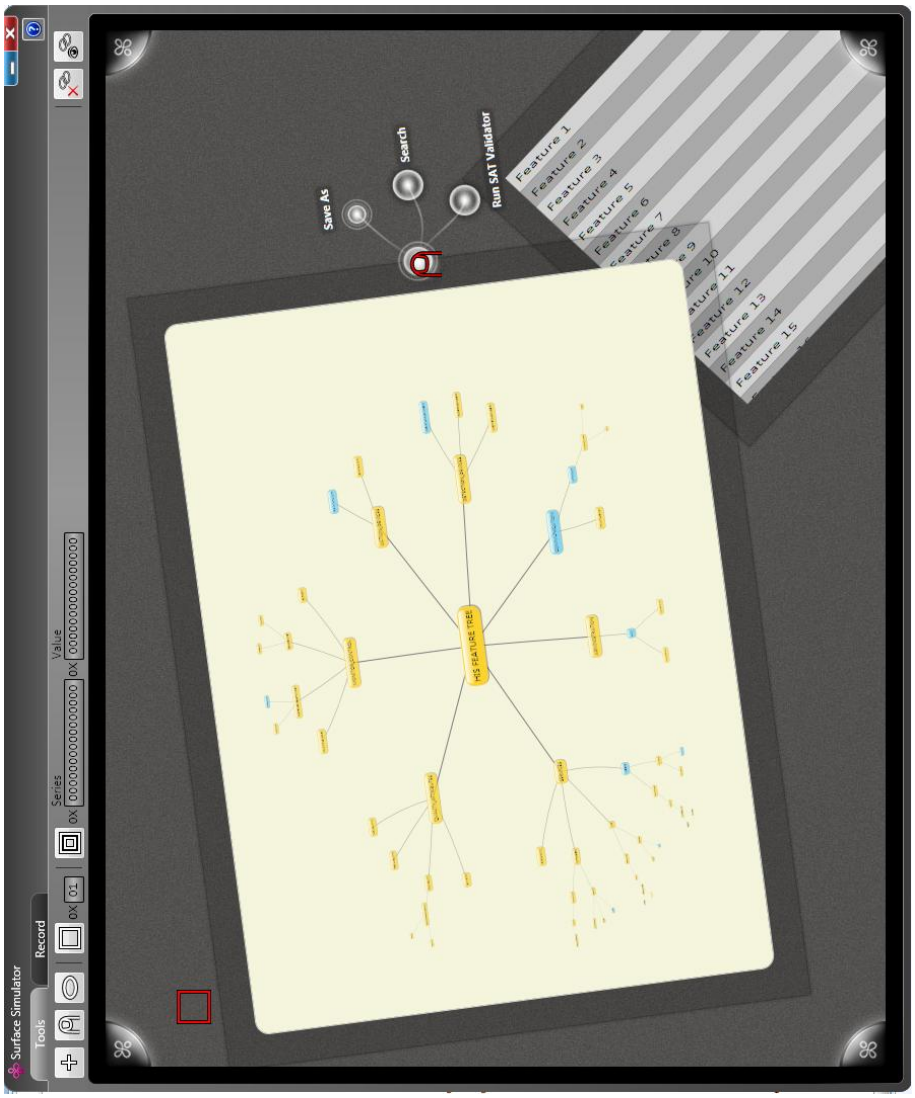

Fig. 2. MUSA over the MS Surface Interface

Fig. 3. MUSA over the Windows 7 Interface

4 Conclusion and Related Work

Over the past few years, a number of VM approaches have been developed ranging from research techniques to commercial products.

On the research techniques front, Sinnema et al [7] introduced the COVAMOF framework and toolset which uses the COVAMOF variability view (CVV) to represent the view of variability for the product family artefacts. The graphical notation used is based on a simple 2D, unidirectional tree that becomes cumbersome to use as soon as the number of variants exceeds about the 50. The Feature Modelling Tool [16] was created as a plugin to visual studio (Figure 4 below). Yet again, in practice, the tool would be difficult to use and manage as soon as the number of variants exceeds 60 or 70. Other tools include FeaturePlugin (an eclipse plugin) by Antkiewicz and Czarnecki [8] and Kubmang by Asikainen et al [9].

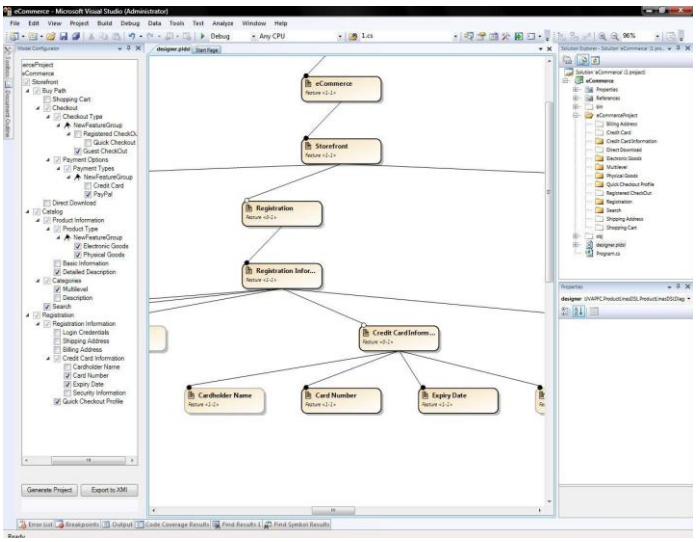

Fig. 4. Feature Modeling Tool [16]

The major challenge for most research techniques is scalability. The scalability issue arises from the graphical modeling techniques traditionally adopted (e.g. trees) and the I/O devices used (standard keyboard, mouse, and monitors). More recently, virtual reality technologies have been reported as being explored as a potential approach for VM. It is hard to see how such techniques could make their way to commercial environments due to the difficulty involved in integrating such approaches within existing industrial development settings.

On the commercial products front, the main tools are from Pure-Systems [11] who have introduced the pure::variants [10] solution and BigLever [12] who developed the Gears toolset. Both are provided as part of a complete modeling framework. These commercial products have managed scalability by largely moving away from graphical representation of models. File system tree like structures and even text

listings (e.g. using MS Excel sheets) have been seen in use. Although such approaches scale and are in industrial use, adopting NUI interfaces such as the one we implemented in MUSA will increase productivity, time-to-market and allow for the creation and management of larger and more complex product families.

Acknowledgments. The work on the MUSA project has been funded by the European RD Fund through INI under the Proof of Concept funding scheme [2008-2010]. It has also received further funding under the Challenge Fund scheme at the University of East London [2010-2011]. We thank all the postgraduate students at the CITE school at UEL who contributed to some of the testing and development of the MUSA toolset as part of their thesis work.

References

1. K. C. Kang, J. Lee, and P. Donohoe, "Feature-Oriented Product Line Engineering," IEEE Software, vol. 19, pp. 58-65, (2002)
2. Microsoft Surface, http://www.microsoft.com/surface/
3. Natural User Interfaces, http://en.wikipedia.org/wiki/Natural_user_interface
4. R. Bashroush, I. Spence, P. Kilpatrick, TJ Brown, and C. Gillan. "A Multiple Views Model for Variability Management in Software Product Lines," Proceedings of the Second International Workshop on Variability Modelling of Software-intensive Systems. Essen, Germany, (2008)
5. US Patent Application No 12/349,797, Inventor: Rabih Bashroush, Title: "Multiple Perspective Feature-based Variability Management", (Patent Pending)
6. B. Nuseibeh, J. Kramer, and A. Finkelstein, "A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification," IEEE Transactions on Software Engineering, vol. 20(10), pp. 760-773, (1994)
7. M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch, "COVAMOF: A Framework for Modeling Variability in Software Product Families." In proceedings of Third Software Product Line Conference 2004, Boston, (2004)
8. M. Antkiewicz and K. Czarnecki, "FeaturePlugin: feature modeling plug-in for Eclipse." In proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange, (2004)
9. T. Asikainen, T. Männistö, and T. Soininen, "Kumbang: A domain ontology for modelling variability in software product families," Advanced Engineering Informatics, Elsevier Science Publishers B. V., vol. 21, pp. 23-40, (2007)
10. D. Beuche, "Variant Management with pure::variants," pure-systems GmbH (2003)
11. Pure-Systems Pure::Variants, http://www.pure-systems.com/Variant_Management.49.0.html
12. "BigLever Software Gears," http://www.biglever.com/solution/product.html
13. R. Bashroush. "A NUI Based Multiple Perspective Variability Modelling CASE Tool," Muhammad Ali Babar, Ian Gorton (Eds.): ECSA 2010. Lecture Notes in Computer Science, Volume (6285), Springer-Verlag Berlin Heidelberg, ISBN 978-3-642-15113-2, August 2010
14. R. Bashroush. "A Scalable Multiple Perspective Variability Management CASE Tool". Proceedings of the 14th International Software Product Line Conference (SPLC), South Korea. September 2010.
15. Lamping, John; Rao, Ramana; Pirolli, Peter (1995). "A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies". Proc. ACM Conf. Human Factors in Computing Systems, CHI. ACM. pp. 401–408.
16. Grupo de Investigacion en Reutilizacion y Orientacion a Objeto (GIRO) – Feature Modeling Tool. Available from: http://www.giro.infor.uva.es/FeatureTool.html[last visited May 2011]