MUSA: A Scalable Multi-Touch and Multi-Perspective Variability Management Tool

Muhammad Garba, Adel Noureddine, and Rabih Bashroush
School of Architecture, Computing and Engineering
University of East London, United Kingdom
Email: u1036584@uel.ac.uk, a.noureddine@uel.ac.uk, r.bashroush@qub.ac.uk

***Abstract*—Variability management is one of the main activities in the Software Product Line Engineering process. Common and varied features of related products are modelled along with the dependencies and relationships among them. With the increase in size and complexity of product lines and the more holistic systems approach to the design process, managing the ever-growing variability models has become a challenge. In this paper, we present MUSA, a tool for managing variability and features in large-scale models. MUSA adopts the Separation of Concerns design principle by providing multiple perspectives to the model, each conveying different set of information. The demonstration is conducted using a real-life model (comprising of 1000+ features) particularly showing the Structural View, which is displayed using a mind-mapping visualisation technique (hyperbolic trees), and the Dependency View, which is displayed graphically using logic gates.**

I. Introduction

Variability Management is one of the essential activities that determines the success of the software product line engineering process [1], [2]. The process entails the identification and cataloguing of the common and varied features of products within a product line. Feature models have been widely recognised as a viable approach to identify, capture, represent and visualise such commonality and variability within the product line. Feature models are information models that describe a set of possible products presented in a single coherent model [3], [4]. Because of the increased demand for software products, variability models tend to grow very large in size (comprising thousands of features in several cases) and increase in complexity due to the myriad of relationships that could exist among the features in the model, making it a challenge and error-prone to handle, manage and visualise these large-scale models using current existing approaches [5], [6], [7], [8]. However, in a real-life context, feature relationships can emerge in a number of ways. For instance, some variation points depend upon other variation points in a non-hierarchical way, and thus cannot be easily represented in a tree structure. Product line developers are being challenged by the scalability of dependency management within variability models. An excessive amount of time and effort is being spent on fixing dependencies in order to ensure valid derivation of products [9], [10].

To address these challenges, we present a new version of MUSA, a variability management tool developed to provide effective ways to deal with the challenges faced during: (1) the creation, representation and visualisation of large-scale variability models and (2) the definition and visualisation of constraints and dependency relationships among variants and their variation points in a large-scale product line. It treats the dependency and constraints, relationships separately from the main variability representation, where simple and complex relationships can be modelled graphically using logic gates. We demonstrate these capabilities of MUSA using a small case study as well as an industrial case study of more than 1000 features.

II. Background of the MUSA Tool

MUSA (A Multi-touch Variability Modelling Solution for Software Product Lines) is designed to implement our theoretical work [6], [11] on multiple perspective-based variability management, which provides a successful modelling framework while using the concept of Separation of Concerns to alleviate the problem of information overloading. As stakeholders have interest in different views of a product line variability model [12], it is important for a variability model to be able to represent and extract relevant information without overloading the graphical representation of the model. The Four View Model for Variability Management (4VM) aims to alleviate this overload [5]. We follow 4VM in the design and implementation of our MUSA tool. The model proposes the distribution of feature modelling information into four views with each view dedicated to a particular theme and group of stakeholders. The views are: Business, Hierarchical and Behavioural, Dependency and Interaction and an Intermediate View.

Business view is where the information related to the project management, cost/benefit analysis, closed/open sets of features and others is presented. Hierarchical and Behavioural View is where the different features are organised (usually presented in a tree structure) along with the behaviour attached to each feature. Dependency & Interaction View presents the dependency and interaction relationships among features (e.g. inclusion, exclusion, etc.). And the Intermediate View is where some design decisions are introduced into the feature model to take it one step further towards the architecture domain in an attempt to bridge the gap between the feature model and the system architecture.

The MUSA tool suit was initially implemented on the Microsoft Surface platform and Windows 7, with a touch

pack platform. It uses hyperbolic trees and supporting gesture-based interaction (multi-touch interaction) for representing and visualising the variability models, which makes it a powerful solution for creating and managing large-scale product lines. However, the initial version of MUSA was developed as a prototype due to some limitations with the surface platform such as hardware issues inherited from surface technology and software issues such as platform dependency.

In this paper, we present the new version of MUSA based on mind mapping technique that uses hyperbolic trees as a better way of exploiting smaller screen surfaces to represent large amount of data and features without graphical overloading. We also added the dependency view which uses logic gates to input complex relationships.

III. The MUSA Tool

The new version of MUSA tool is implemented in Java and uses XML files to input/output data. It provides two different collaborative interfaces (i.e. views) for managing variability models, and their consistencies are maintained with the help of a centralised database (see Figure 1). The Development/Browser View is the default view when the application is initially launched (see Figure 2). Main functionalities concerned with this view are: (1) Product line variability models are represented using a hyperbolic browser; (2) a new feature tree for managing variability can be created either based on existing feature models or from scratch; and (3) feature models can be modified (i.e. feature behaviour), such as changing a particular feature name, its properties and description, adding and deleting features, etc.

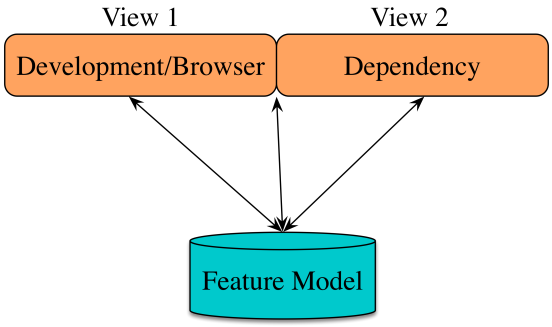

Fig. 1. Description of MUSA's architecture.

The hyper-tree browser uses hyperbolic geometry to place nodes around the root and provides smooth and continuous animation of the tree so that users can bring other nodes into focus by clicking, tapping on or dragging them [13]. The advantage of using hyperbolic trees is reducing visual clutter compared to standard trees when the number of child nodes grow exponentially. The former employs hyperbolic space which provides more room than Euclidean space. Using hyperbolic trees give our MUSA tool an important advantage in scalability. The tool can display models with large number of features, counting more than 1000+ features in our case study (see Section IV).

When focusing on a particular node, MUSA places it at the centre of the screen with all its children, while nodes out of focus reduce in size and are displayed towards the edge of the view. On double tapping a feature node, the option menu with a number of possible options pop-up; this can be used to add a new feature to the existing tree, delete a feature from the tree or view dependency relationships that exist among the features. Users can also use different gestures, such as pinching (for expanding nodes), panning (by moving two fingers on the screen to shift the feature model) or three fingers to centre the model to its root node.

From the dependency perspective, a separate view is proposed within the MUSA tool, using Logic Design to capture and model the dependency relationships. Once the user makes his/her selection of features from the browser view, the dependency model will take the user-selected feature set as an input and verifies it against the model, pointing out any dependency relationships associated with that feature, while at the same time if no relationship for that selection exists, a new window in the dependency view opens to create new dependencies if needed. This provides simplicity in managing dependency relationships within large and complex variability models. We used three basic Logic gate symbols, from which a user, such as an architect, can generate and resolve any (from simple to complex dependency) relationships (see Figure 3).

IV. Case Study

In this section, we present the main features of the new MUSA tool using a product line case study. The later consists of more than 1,000 features. We aim to show how effective our approach is in terms of managing and visualising large-scale variability models. The use of this case study enables us to determine and assess the extent to which MUSA satisfies the design needs as compared to other tools available today.

As shown in Figure 2, MUSA's browser view shows all features of the case study in a hyperbolic tree. By default, the root of the tree is centred, while further leaf names are hidden (but their connections remain in order to provide a visual feedback for the user). The user can cycle through the features by swiping on any direction with a mouse or directly on a touchscreen. Selecting a feature will centre the screen over it, zooming if necessary and displaying more connections to related features. Double clicking anywhere on the background will centre the view back to the root of the model.

Search in MUSA is straightforward by 'touch and hold' on any space (or right clicking) which brings up the search box. In the popup box, users can type the desired search keyword and a list of potential features will be displayed. Touching or clicking any result will centre the view to that particular feature. Figure 4 illustrates the search process.

Adding or removing a feature in the model can be achieved by double tapping or clicking on a feature node. A menu will appear with options to add or remove features. If for instance, the Add button is selected, the user will be prompted with a window where he can type the name of a feature such as `TestFeature` and select its type as *mandatory, optional*

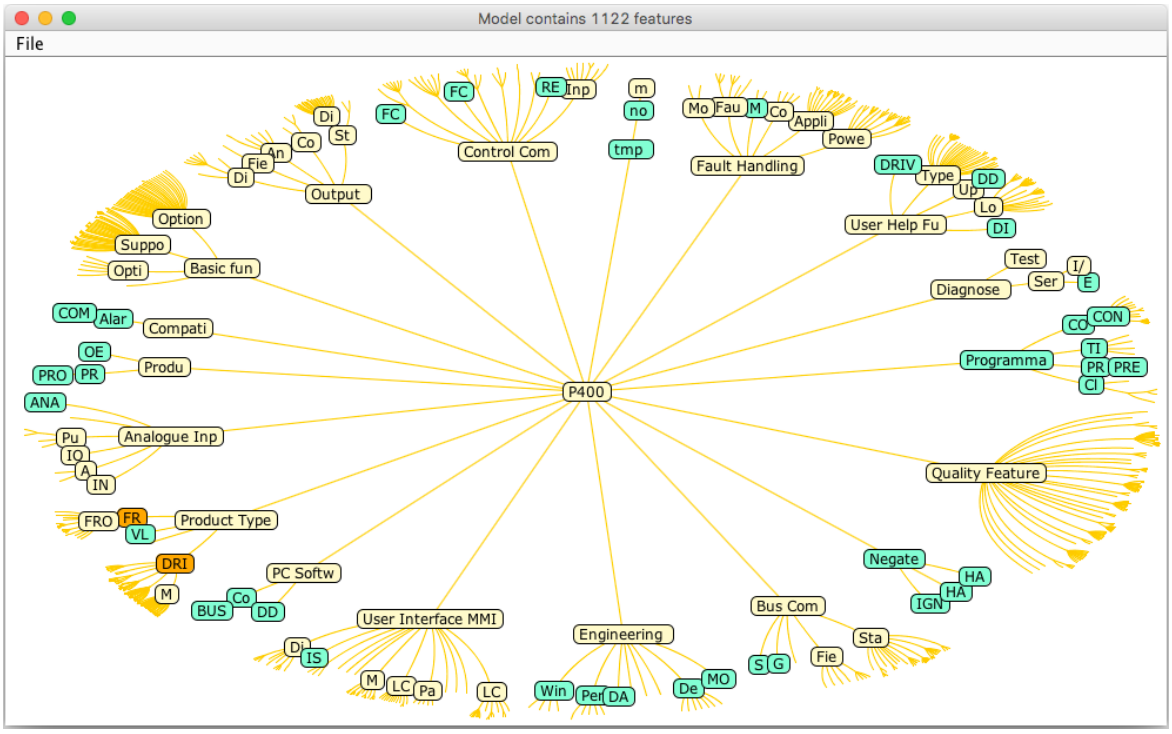

Fig. 2. MUSA's Browser View.

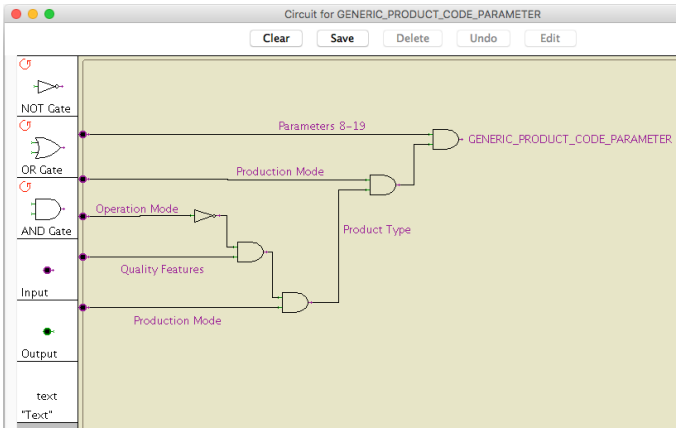

Fig. 3. MUSA's Dependency View.

or alternative. The same menu displays an option to view dependencies in a different view. On tapping or clicking on the dependency option, the Dependency View will open showing the selected feature with all its associated relationships. From this view, different kinds of dependency relationships can be created, edited or modified using Logicf visualisation (see Figure 3).

V. Conclusions

We present a new version of MUSA tool that exhibits a number of features that enable it to deal with large-scale systems. MUSA adopts the idea of Separation of Concerns design principle by providing multiple perspectives to the model, each conveying a distinct set of information. The tool was demonstrated on an industrial case study consisting of more than 1,000 features. The demonstration conducted to show the Structural View, which is displayed using a mind-mapping visualisation technique (hyperbolic trees), and the Dependency View, which is graphically represented using Logic gates.

Acknowledgment

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 received further funding under the Challenge Fund scheme at the University of East London (2010-2011).

References

[1] *Software Product Lines: Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[2] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[3] David Benavides, Sergio Segura, and Antonio Ruiz-Corts. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 35(6):615 – 636, 2010.

[4] Krzysztof Czarnecki and Ulrich W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

[5] R. Bashroush. A nui based multiple perspective variability modeling case tool. In MuhammadAli Babar and Ian Gorton, editors, *Software Architecture*, volume 6285 of *Lecture Notes in Computer Science*, pages 523–526. Springer Berlin Heidelberg, 2010.

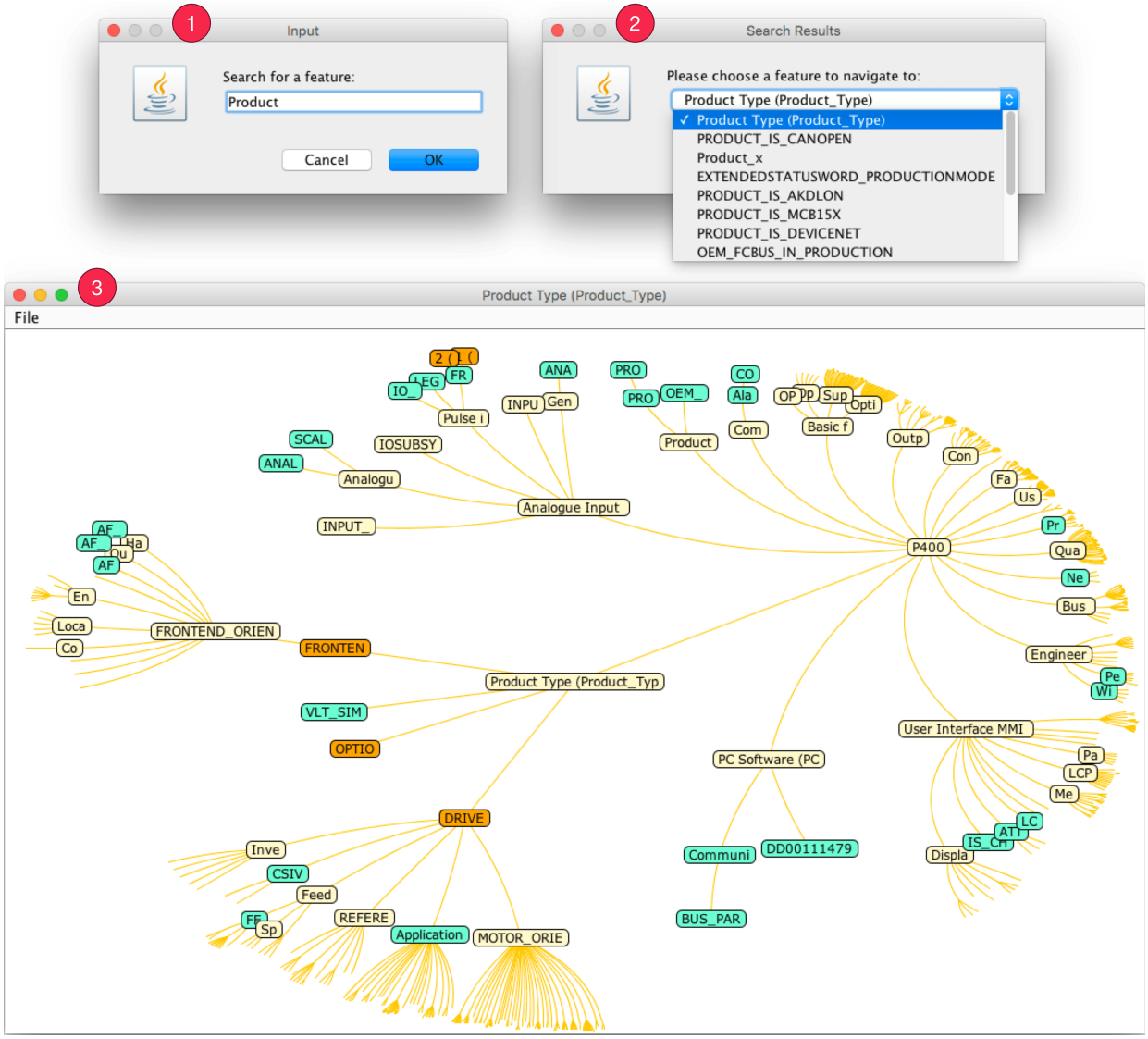

Fig. 4. The search process in MUSA.

[6] Rabih Bashroush, Ameer Al-Nemrat, Mohammad Bachrouch, and Hamid Jahankhani. Visualizing variability models using hyperbolic trees. In *Proceedings of the 23rd International Conference on Advanced Information Systems Engineering Forum(CAiSE Forum 2011)*, 2011. Citation: R. Bashroush, A. Al-Nemrat, M. Bachrouch and H. Jahankhani, ?Visualizing Variability Models Using Hyperbolic Trees?, in Proceedings of the 23rd International Conference on Advanced Information Systems Engineering Forum(CAiSE Forum 2011), London, June 2011..

[7] Goetz Botterweck, S. Thiel, D. Nestor, S. bin Abid, and C. Cawley. Visual tool support for configuring and understanding software product lines. In *Software Product Line Conference, 2008. SPLC '08. 12th International*, pages 77–86, Sept 2008.

[8] F. Loesch and E. Ploedereder. Optimization of variability in software product lines. In *Software Product Line Conference, 2007. SPLC 2007. 11th International*, pages 151–162, Sept 2007.

[9] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M. Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wasowski. A survey of variability modeling in industrial practice. In *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, VaMoS '13, pages 7:1–7:8, New York, NY, USA, 2013. ACM.

[10] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch. Modeling dependencies in product families with covamof. In *Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on*, pages 9 pp.–307, March 2006.

[11] Hassan Gomaa and MichaelE. Shin. A multiple-view meta-modeling approach for variability management in software product lines. In Jan Bosch and Charles Krueger, editors, *Software Reuse: Methods, Techniques, and Tools*, volume 3107 of *Lecture Notes in Computer Science*, pages 274–285. Springer Berlin Heidelberg, 2004.

[12] B. Nuseibeh, J. Kramer, and A. Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *Software Engineering, IEEE Transactions on*, 20(10):760–773, Oct 1994.

[13] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.