Computer Science and Information Technology 8(4): 81-89, 2020 http://www.hrpub.org
DOI: 10.13189/csit.2020.080402

An Enhanced Scalable Design Approach for Managing Large Scale Variability in Software Product Lines (SPLs)

Muhammad Garba*, Muhammad Nura Malami, Muhammad Muazu, Abubakar Ahmad Aliero, Dalhatu Mohammed, Bashar Umar Kangiwa

Department of Computer Science, Kebbi State University of Science and Technology, Aliero, Nigeria

Received June 14, 2020; Revised October 26, 2020; Accepted October 30, 2020

Cite This Paper in the following Citation Styles

(a): *[1] Muhammad Garba, Muhammad Nura Malami, Muhammad Muazu, Abubakar Ahmad Aliero, Dalhatu Mohammed, Bashar Umar Kangiwa , "An Enhanced Scalable Design Approach for Managing Large Scale Variability in Software Product Lines (SPLs)," Computer Science and Information Technology, Vol. 8, No. 4, pp. 81 - 89, 2020. DOI: 10.13189/csit.2020.080402.*

(b): *Muhammad Garba, Muhammad Nura Malami, Muhammad Muazu, Abubakar Ahmad Aliero, Dalhatu Mohammed, Bashar Umar Kangiwa (2020). An Enhanced Scalable Design Approach for Managing Large Scale Variability in Software Product Lines (SPLs). Computer Science and Information Technology, 8(4), 81 - 89. DOI: 10.13189/csit.2020.080402.*

Abstract Variability management remained the main challenge for software product line (SPL) adoption since it needs to be efficiently managed at different levels of the SPL development process (for example, requirements analysis, software design, implementation etc.). With the increase in size and complexity of product lines, and the more holistic systems approach to the design process, managing ever-growing variability models has become a challenge and more difficult to handle. Accordingly, tool support for variability management has been gathering increasing momentum over the last two decades and can be considered a key success factor for developing and maintaining SPLs. This work presents a new tool support that exhibits a number of features that enable it to deal with large models. The new tool adopts the Separation of Concerns design principle by providing multiple perspectives to the model, each conveying a different set of information. It can comprise more than 1000 features particularly, showing the browser (structural) View, which is displayed using a mind-mapping visualisation technique (Hyperbolic trees); the development/Edit view where a new feature can be created either based on existing feature or from scratch; the business view where the information related to the project management, cost/benefit analysis, close/open sets of features and others are presented and the dependency view which is displayed graphically using logic gates.

Keywords Software Product Line Engineering, Feature Modelling, Variability Models, Software Scalability, Visualization

1. Introduction

Software Product Line Engineering (SPLE) is a paradigm of software engineering for creating a portfolio or a collection of similar software products with variations in their features and functions. The products can be software, such as images or systems with software inside. A typical example of these includes; Airplanes, Automobiles, Ships, Cameras, Mobile phones, Computers and Tablets among others [1-3].

SPLE technique provides a systematic way for the reuse of software assets. These assets are the software artefacts or resources associated with the products under development. The artefacts include, but are not limited to requirements

analysis, design specification, Software Implementation, configuration, test plan, test cases, etc. The assets are then engineered to be shared across the entire product line. i.e. to be used in any product. Therefore, SPLE is a technique that optimises the reuse of existing software assets in creating multiple applications that share a lot of features while still exhibiting differences [4]. SPLE allows for a planned reuse of artefacts among the software systems under development.

Some of the key advantages of software Product Line (SPL) development over one at a time system development are productivity gains (the core assets and architecture are reused), decrease time-to-market of products, Large-scale productivity, low-cost production, increase products quality and reliability and Increase customer satisfaction [5].

This paper presents a new tool suite based on mind mapping technique that uses hyperbolic trees as a better way of exploiting smaller screen surfaces to represent a large amount of data and features without graphical overloading. The study is an implementation of theoretical work on the Four View Model for Variability Management (4VM), which proposes the distribution of feature modeling information into four views with each view dedicated to a particular theme and group of stakeholders.

The remaining paper is structured as follows: In section II, multiple views Perspective: the theoretical foundation is discussed. Section III presents the background of the new tool. Section IV provides a detailed description of the New Tool Support. In section V, the case study used has been discussed. Section VI presents the tool demo tips. In section VII, related work is highlighted. Finally, section VIII rounds off the paper with the conclusion.

2. Multiple Views Perspective: The Theoretical Foundation

It is generally agreed that different stakeholders have an interest in viewing different aspects (views) of the product line variability model. So, it is important for a variability management 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 (compared to multiple views) [6, 7]. This forms the basis of the 4VM model and is discussed in more detail below.

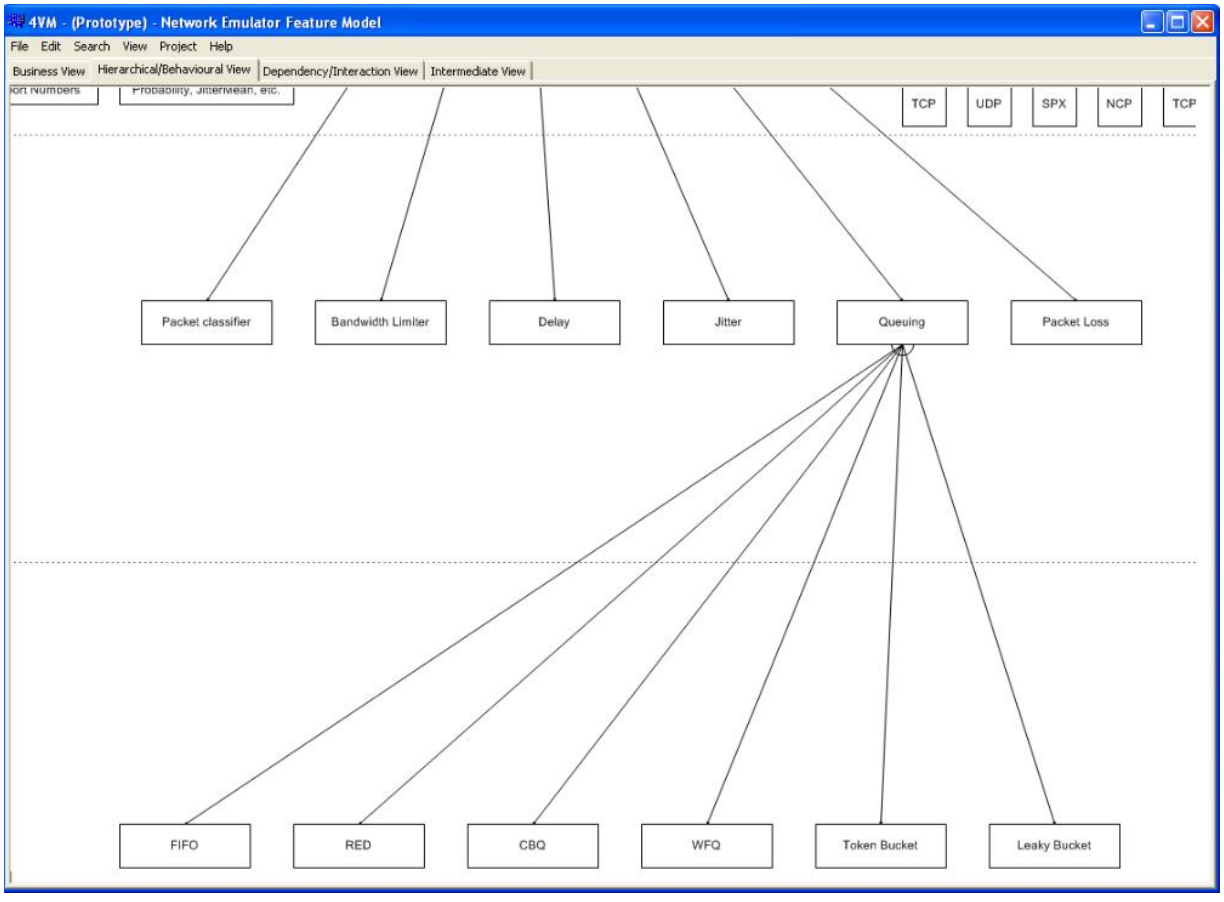

Figure 1. 4VM - Hierarchical & Behavioural view (R. Bashroush et al.)

The Four View Model (see Figure 1) for Variability Management (4VM) proposes the distribution of feature modeling information into four views with each view dedicated to a particular theme and group of stakeholders. These proposed views are:

- *Hierarchical & Behavioral View*: this view presents the way the different features are organized (usually presented in a tree structure) along with the behavior attached to each feature.
- *Dependency & Interaction View:* In this view, the dependency and interaction among features are presented.
- *Intermediate View*: where some design decisions are injected 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.
- *Business View:* where the information related to the project management, cost/benefit analysis, closed/open sets of features, etc. are presented.

These requirements are in the form of information and relationships that should be captured in a feature model. The Four Views Model (4VM) is built around these concerns. More concerns can be added to the list in the future to accommodate special application domain or enterprise requirements (e.g. feature evolution, etc.).

3. Background of the New Tool

In the previous section, several requirements that need to be captured within a feature model were identified and discussed. In this section, the new tool designed to implement the theoretical work [6, 8, 9] on multiple perspective-based variability management is Introduced. The tool is a Multi-touch-based Variability Modelling Solution for Software Product Lines that proposes a four-view presentation of the feature model to address all the issues and concepts identified in the previous section. The theoretical background provides a successful modelling framework while using the concept of Separation of Concerns to alleviate the problem of information overloading. As stakeholders have an interest in different views of a product line variability model, a variability model or tool support needs to be able to represent and extract relevant information without overloading the graphical representation of the model.

The new tool for Variability Management aims to alleviate this overload. This study follows 4VM in the design and implementation of our new tool. The tool 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 development/edit view, the browser view, the dependency view and business view. In section V, each of these views is discussed in detail and example views are taken from the tool running on real-life case study.

The new tool suite (see Figure 2 for the architectural description), 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 the tool 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 [10, 11].

4. The New Tool Support

The new version of the tool is implemented in JavaScript and uses CSS files to input/output data. It provides four different collaborative interfaces (i.e. views) for managing variability models, and their consistencies are maintained with the help of a centralised database (see Figure 2). The Development/Browser View is the default view when the application is initially launched (see Figure 3). 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 based on existing feature models; 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.

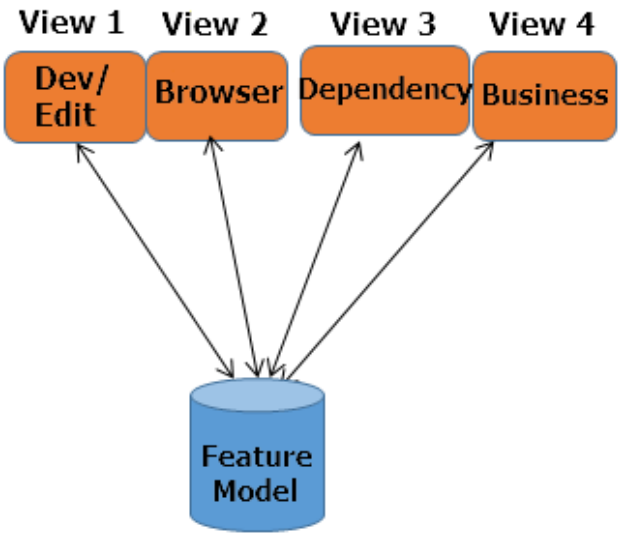

Figure 2. Description of New Tool Architecture

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. The advantage of using hyperbolic trees is reducing

visual clutter compared to standard trees when the number of child nodes grows exponentially. The former employs hyperbolic space which provides more room than Euclidean space. Using hyperbolic trees gives our tool an added advantage in scalability. The tool can display models with a large number of features, counting more than 1000+ features in our case study (see Section V).

In order to clearly highlight the new enhancement, a summary table of comparison between the first version of the work and current one has been presented:

Table 1. Comparison between Tool V1 & V2

Features supported	Tool Version 1	Tool Version 2
Multi-Platform support	Yes	Yes
Support to be run as a web application	No	Yes
Support JavaScript and uses CSS files to input/output data	No	Yes
Support the business view	No	Yes
Multiple views	No	Yes
Modelling and management of variability	Yes	Yes
Multiple views	No	Yes
Innovative visualisation technique	Yes	Yes
Support for new view: Dev/Edit	No	Yes
Feature interaction	Yes	Yes
Identifying alternative features	No	Yes
Support for multitouch	Yes	Yes

When focusing on a particular node, the tool places it at the center 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 center the model to its root node.

From the dependency perspective, a separate view is proposed within the 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 verify 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 6).

5. Implementation and Case Study Used

In this section, we present the main features of the new 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 the new tool satisfies the design needs as compared to other tools available today.

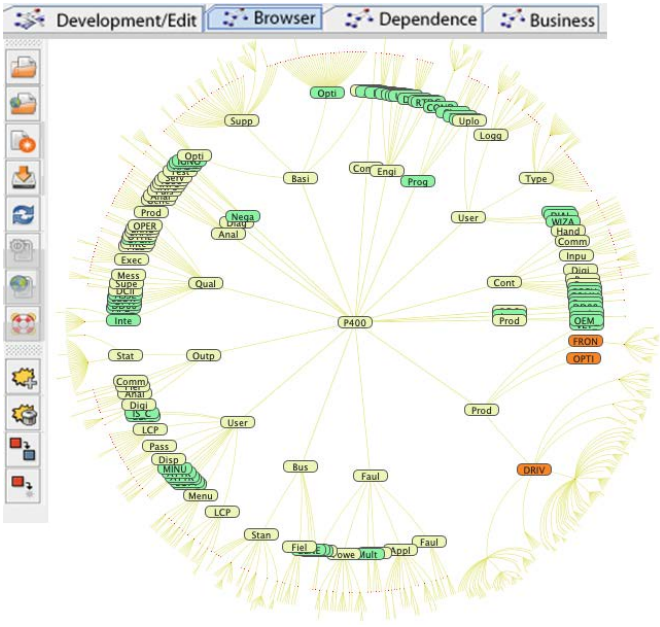

Figure 3. The browser view showing all features of the case study in a hyperbolic tree

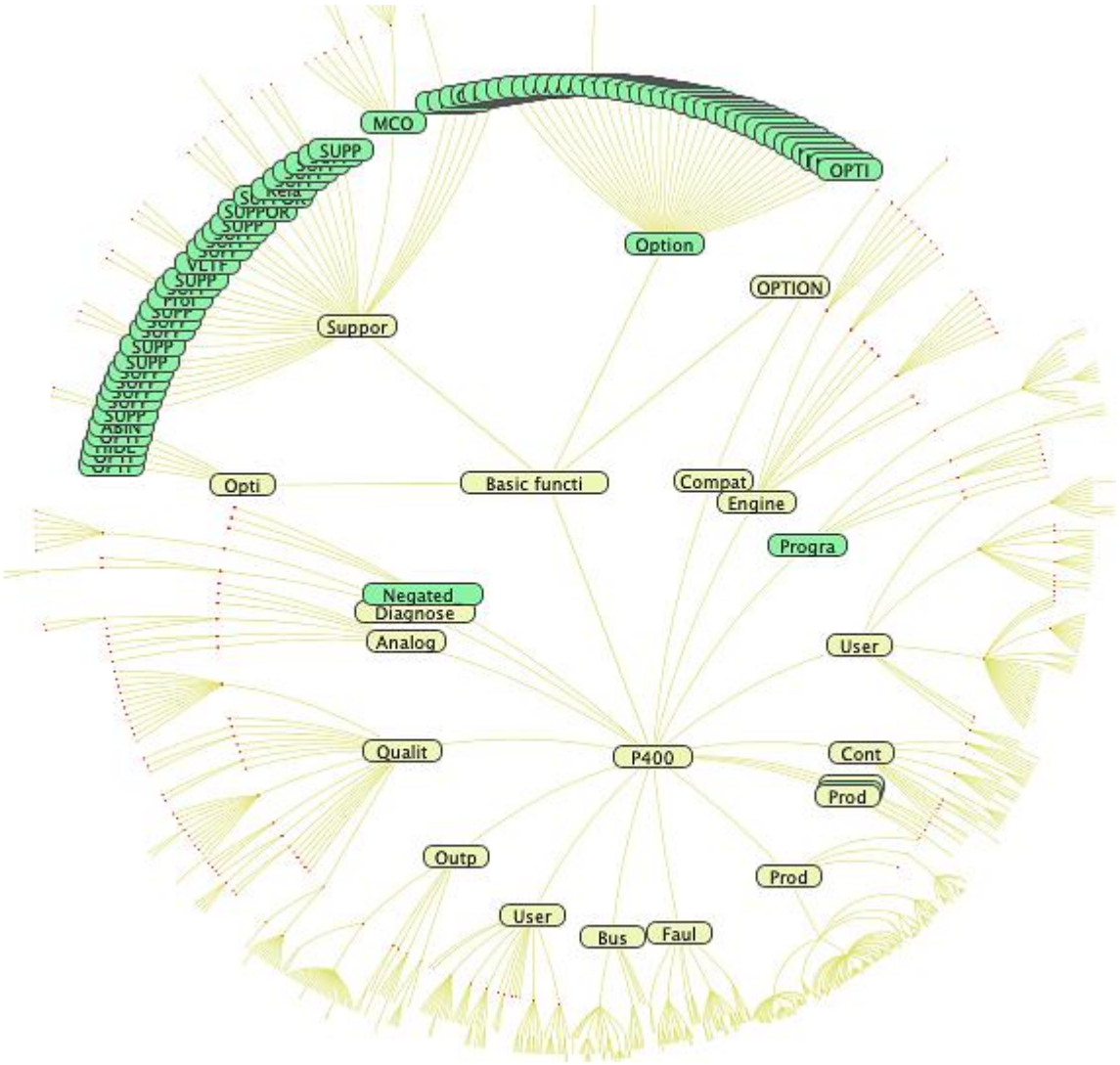

Figure 4. Selecting a feature will center the screen over it- Example 1

As shown in Figure 3, the browser view shows all features of the case study in a hyperbolic tree. By default, the root of the tree is centered, while further leaf names are hidden (but their connections remain to provide visual feedback for the user). The user can cycle through the features by swiping in any direction with a mouse or directly on a touchscreen. Selecting a feature will center the screen over it (See Figure 4 & 5 respectively; Basic Functionality and Type Code String are the focused features), zooming if necessary and displaying more connections to related features. Double-clicking anywhere on the background will center the view back to the root of the model.

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 or alternative.

Similarly, hovering a mouse pointer over a particular feature will display its full name as shown in Fig. 5.

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 Logic circuit visualisation (see Figure 6).

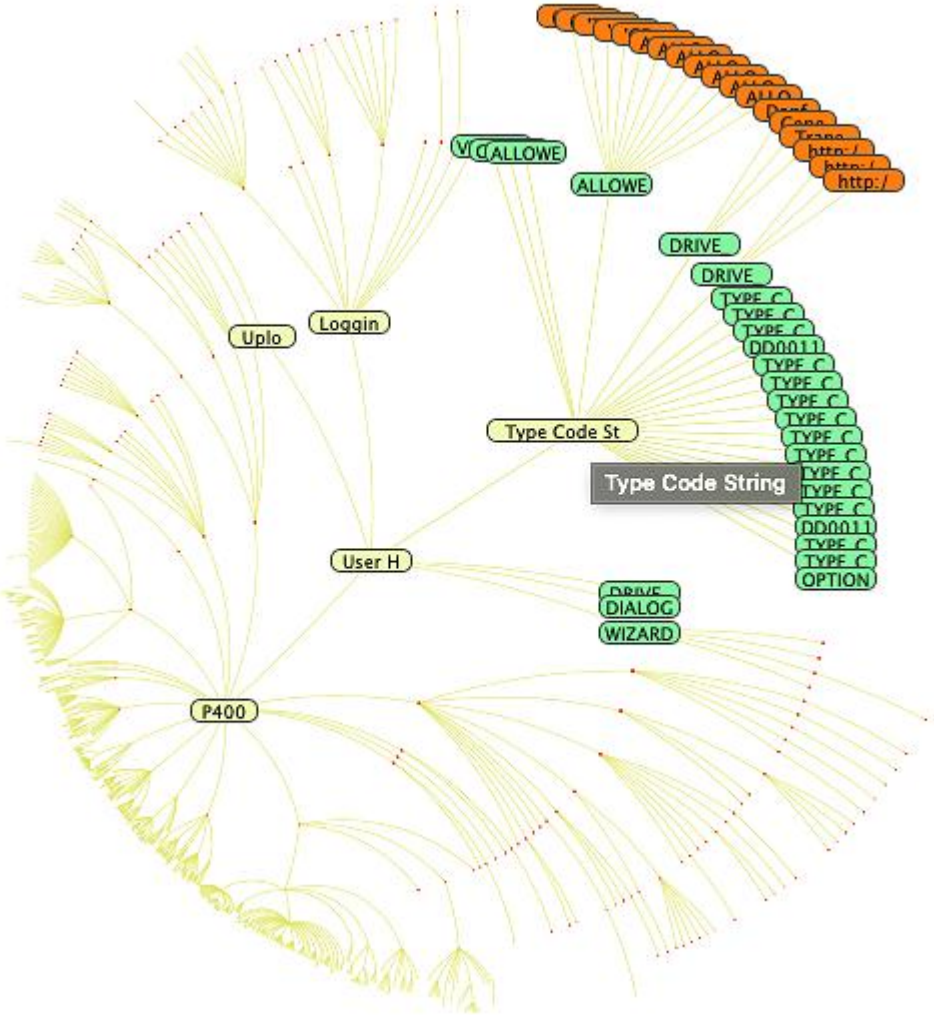

Figure 5. Selecting a feature will center the screen over it- Example 2

Figure 6. Dependency View

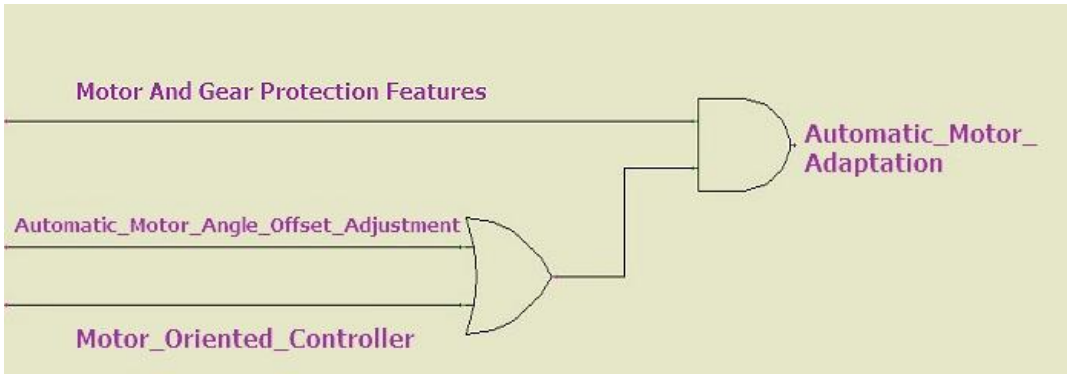

Figure 7. Automatic_Motor_Adaptation requires Motor and Gear protection features and at least one of Auto_Motor_Angle_Offset_Adjustment or Motor_Oriented_Controller

Figure 8. Mutually exclusive relationships between features

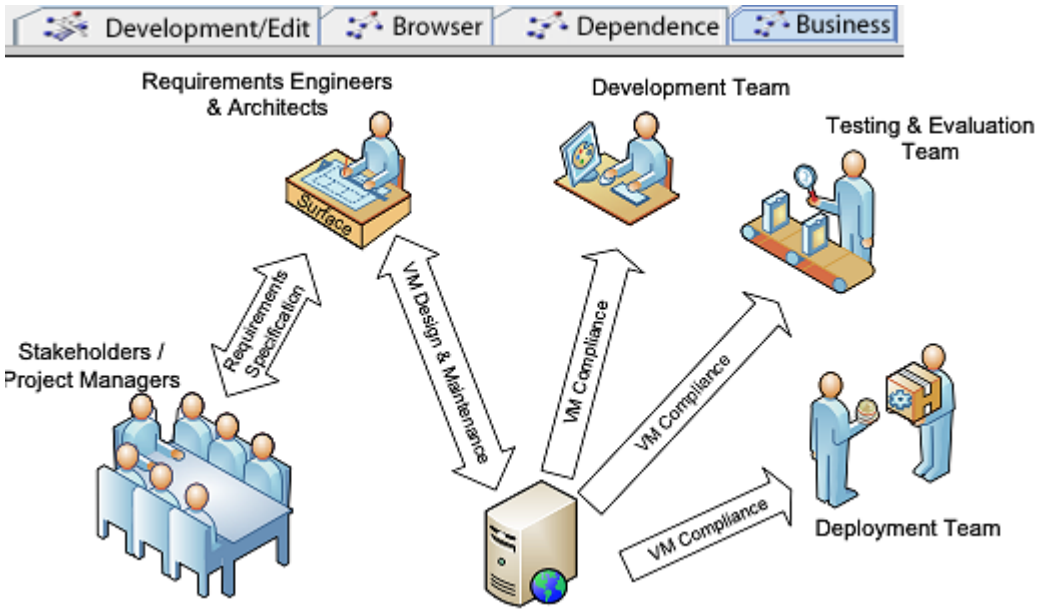

Figure 9. Business View

On the dependency perspective (Figure 6, 7 & 8 respectively), the circuit diagram is for variability dependency that exists within the model, showing a particular feature called Automatic_Motor_Adaptation which is mutually dependent on Motor and Gear protection features, hence, requires it to fulfill its functions. It is also necessary that at least one member be selected from either *Auto_Motor_Angle_Offset_Adjustment* or *Motor_Oriented_Controller* (see Figure 7 for more clarity). The diagram however shows that there exists a conflict between Motor and Gear protection features and Hand_Auto_Combined; therefore, they cannot be chosen for the same product configuration, that is, they are mutually exclusive to each other (see Figure 8 for a break down). Hence, a bi-directional exclusive relationship exists between the two features.

With respect to Business View, the tool 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. Also, the new tool automates model verification with the use of satisfiability solvers (SAT solvers) and maintains consistency among the different views with the help of a centralized Database (see Figure 9).

6. The Tool Demo Tips

A demonstration has been done using a real-life case study and went through the different functionalities of the system from creating and managing variability models, to adding and removing features and using the dependency view.

The demonstration conducted shows the following characteristics of the new tool (please refer to Figure 3 through Figure 8 for details:

- Demonstrate the basic features of the tool such as adding, removing and editing features. In this step, we show the different options available for features within the model. We aim to start from an empty model and interactively create a mid-sized model containing 10-20 features.
- Outline the capabilities of the tool in supporting thousands of features with two feature models (containing 100+ and 1000+ features respectively). The focus is twofold: first, to show the improvements in performance for large feature models, and second to outline the multi-touch support in navigating, creating and editing nodes in a large feature model.
- Demonstrate the search capabilities of the tool, where the tool finds and automatically navigates to the feature. We also explain why hyperbolic trees are better suited to display feature models with a search demo.
- Finally, we demonstrate the dependency view and its capabilities.

7. Related Work

Pure: variants [12] are developed by pure-systems GmbH in Magdeburg, Germany. The tool supports variant management and product configuration based on feature models and has a strong focus on interoperability and extensibility. For example, the tool can be integrated into the Eclipse IDE, used with a web browser, as a command-line client, and even in a custom application. Several extensions to existing commercial-off-the-shelf tools exist, e.g., to DOORS or SAP. Four types of models can be created and managed with pure: variants: (1) Feature Models represent the variability of a system. (2) Family Models represent the variants of assets that can be selected. (3) Variant Description Models are used to store the selected features and their values. (4) Result Models based on 1-3 represent one concrete instance derived from a product line.

Gears [13, 14] are a commercial tool developed by BigLever Software Inc., Austin, Texas, USA. The tool has been developed in Java and supports the three-tiered methodology proposed by Krueger. The tool allows defining arbitrary reusable software assets and a product feature profile that describes products in terms of features. Gears focus on products: feature profiles define the products that can be built from assets and the optional choices that can be made for each product.

EASy-Producer [15] is an open-source research toolset for engineering product lines, variability-rich software ecosystems and dynamic software product lines. It has been applied in several industrial case studies and research projects showing its practical applicability both from a stability and a capability point of view. The toolset consists of an interactive approach to product line definition and configuration through DSLs.

8. Conclusions

This paper presents a new version of an existing tool suite earlier implemented on Microsoft windows and surface platforms. The tool has now been redesigned to overcome the platform dependency related problems. The study is an implementation of a theoretical work on Four View Model for Variability Management (4VM). The redesign was based on mind mapping technique that uses hyperbolic trees as a better way of exploiting smaller screen surfaces to represent a large amount of data and features without graphical overloading. The new tool adopts the Separation of Concerns design principle by providing multiple perspectives (views) to the model, each conveying a different set of information. The proposed views are development/edit, the browser, the business view and the dependency view which uses logic gates to input complex relationships.

REFERENCES

[1] K. Pohl, G. Böckle, and F. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Germany: Springer-Verlag Heidelberg, 2005.

[2] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns* (SEI Series In Software Engineering). Massachusetts: Addison-Wesley, 2002.

[3] J.-M. Horcas and P. Monica, "Software product line engineering: a practical experience," in *Proceedings of the 23rd International Systems and Software Product*, Paris, France, 2019, vol. A: ACM, pp. 164--176, doi: https://doi.org/10.1145/3336294.3336304.

[4] J. Bosch, *Design & Use of Software Architectures: Adopting and evolving a product-line approach*. Addison-Wesley, 2000.

[5] P. Clements and L. Northrop, "A Report," in *Proceedings of the 1st Software Product Line Conference (SPLC 2001)*, Software Engineering Institute, Carnegie Mellon University Pittsburgh, 2001.

[6] R. Bashroush, M. Garba, R. Rabiser, I. Groher, and G. Botterweck, "CASE tool support for variability management in software product lines," *ACM Computing Survey*, vol. 50, no. 1, pp. 14:1–14:45, 2017, Art no. 14, doi: http://dx.doi.org/10.1145/3034827.

[7] R. Bashroush, I. Spence, P. Kilpatrick, J. Brown, and C. Gillan, "A Multiple Views Model for Variability Management in Software Product Lines," in *Proceedings of the 2ndnternational Workshop on Variability Modelling of Software-intensive Systems (VaMoS2008)*, Duisburg-Essen, Germany, 2008.

[8] R. Bashroush, I. Spence, P. Kilpatrick, J. Brown, and C. Gillan, "A Multiple Views Model for Variability Management in Software Product Lines," in *Proceedings of the Second International Workshop on Variability Modelling of Software-intensive Systems (VaMoS2008)*, Essen, Germany, 2008.

[9] M. Garba, A. Noureddine, and R. Bashroush, "MUSA: A Scalable Multi-touch and Multi-perspective Variability Management Tool," in *13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, Venice, Italy, 2016: IEEE, pp. 299 - 302, doi: DOI: 10.1109/WICSA.2016.45.

[10] R. Bashroush, A. Al-Nemrat, R. Bachrouch, and H. Jahankhani, "Visualizing Variability Models Using Hyperbolic Tree," in *Proceedings of the 23rd International Conference on Advanced Information Systems Engineering Forum(CAiSE)*, London, 2011, pp. 113-120.

[11] R. Bashroush, "A NUI Based Multiple Perspective Variability Modeling CASE Tool," in *Proceedings of the 4th European Conference on Software Architecture (ECSA '10)*, Copenhagen, Denmark, 2010, vol. 6285: LNCS, pp. 523-526, doi: 10.1007/978-3-642-15114-9_55.

[12] D. Beuche, "Modeling and Building Software Product Lines with pure::variants," in *Proceedings of the 12th International Software Product Lines Conference (SPLC 2008)*, Limerick, Ireland, 8-12 Sept. 2008 2008: IEEE Computer Society, p. 358, doi: 10.1109/splc.2008.53. [Online]. Available: http://www.pure-systems.com/News.20+M5acffda5188.0.html

[13] C. Krueger and P. Clements, "Systems and software product line engineering with gears from BigLever software " in *Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools* Florence, Italy, 2014, vol. Volume 2: ACM.

[14] C. Krueger and P. Clements, "Feature-based systems and software product line engineering with gears from BigLever," in *Proceedings of the 22nd International Systems and Software Product Line Conference SPLC '18*, 2018, vol. 2, pp. 1-4, doi: https://doi.org/10.1145/3236405.3236409.

[15] K. Schmid, H. Eichelberger, and S. El-Sharkawy, "Variability Modeling and Implementation with EASy-Producer," in *23rd International Systems and Software Product Line Conference (SPLC '19)*, Paris, France, 2019, vol. A: ACM, doi: https://doi.org/10.1145/3336294.3342382.