The effects of visualization and interaction techniques on feature model configuration

The Effects of Visualization and Interaction Techniques on Feature Model Configuration

Mohsen Asadi[1*], Samaneh Soltani[1], Dragan Gašević[2], Marek Hatala[1]

[1]*Simon Fraser University, Canada,*[2]*Athabasca University, Canada*
{ssoltani, masadi, dgasevic, mhatala}@sfu.ca

Abstract

A Software Product Line is a set of software systems of a domain, which share some common features but also have significant variability. A feature model is a variability modeling artifact which represents differences among software products with respect to variability relationships among their features. Having a feature model along with a reference model developed in the domain engineering lifecycle, a concrete product of the family is derived by selecting features in the feature model (referred to as the *configuration process*) and by instantiating the reference model. However, feature model configuration can be a cumbersome task because: 1) feature models may consist of a large number of features, which are hard to comprehend and maintain; and 2) many factors including technical limitations, implementation costs, stakeholders' requirements and expectations must be considered in the configuration process. Recognizing these issues, a significant amount of research efforts has been dedicated to different aspects of feature model configuration such as automating the configuration process. Several approaches have been proposed to alleviate the feature model configuration challenges through applying visualization and interaction techniques. However, there have been limited empirical insights available into the impact of visualization and interaction techniques on the feature model configuration process. In this paper, we present a set of visualization and interaction interventions for representing and configuring feature models, which are then empirically validated to measure the impact of the proposed interventions. An empirical study was conducted by following the principles of control experiments in software engineering and by applying the well-known software quality standard ISO 9126 to operationalize the variables investigated in the experiment. The results of the empirical study revealed that the employed visualization and interaction interventions significantly improved completion time of comprehension and changing of the feature model configuration. Additionally, according to results, the proposed interventions are easy-to-use and easy-to-learn for the participants.

Keywords: Software product line engineering, controlled experiment, tools.

1. INTRODUCTION

Mass customization is defined as "a large-scale production of products tailored to individual customers' needs" [1] . The key factors for the success in mass-customization are *reusability* and *flexibility* – i.e., adapt software products to fit into different requirements [1]. An efficient approach to realizing mass-customization in software engineering is through Software Product Line Engineering (SPLE). SPLE targets the development of a set of software systems that satisfy requirements of a specific domain and that share common features. SPLE contributes into mass-customization by providing mechanisms for managing variability and commonality in order to address flexibility and reusability, respectively.

There are two lifecycles in SPLE, namely, *domain engineering* and *application engineering*. *Domain engineering* (i.e. *development for reuse*) is concerned with the process of understanding the target domain and developing a relevant, suitable and comprehensive representation of the concepts in that domain [3]. Moreover, the commonality and variability of the domain are identified and explicitly modeled. The result of this stage is a *family* of software systems, defined by a number of core assets. The two key assets produced in domain engineering are a variability model (commonly defined as a feature model) and reference models. On the other hand, *application engineering* receives the reusable assets (i.e. variability and reference models) developed in domain engineering and creates an application instance based on a given set of requirements. The application is produced by binding the variation points (i.e., choosing suitable variant(s) for every variation points) and by instantiating the right elements of the representation of the domain model (i.e. reference models) [6].

1.1. Problem Statement

Developing a product line and deriving a product based on the target application requirements is a *creative* process which requires high levels of cognitive resources from software developers. Software developers need to explore and compare alternative solutions in order to come up with the best solution satisfying both functional and non-functional requirements. According to empirical evidence in the creativity support tools [8] and software engineering [47][48] research areas, one way to assist software developers in their tasks is to provide an efficient visualization and interaction support to enhance software developers' cognition of complex models (e.g., variability models) [5]. Many researchers in software product lines developed a wide variety of tools with innovative visualization and interaction techniques for assisting software developers in different tasks of software product line lifecycle [5][11][41]. For example, in the context of feature-oriented software product line, *Feature Modeling Tool* [35], *FeatureIDE [10]*, *Feature Model Plugin (fmp) [26],* and *faMa Tool Suit* [39][40] use an indented list and a tree structure to represent feature models; *FeatureMapper* [36][37] uses color for representing mapping between features and design and implementation models; *CIDE* [11] applies background colors to show code annotations; *FeatureVisu* [12] applies clustering layout to illustrate the degree of relatedness of features elements. Also, decision oriented product line tools like *Dopler* [13] apply various graphs, trees, and coloring techniques to depict a tabular and hierarchical decision view and effect view for application engineers.

Despite many efforts to apply visualization and interaction techniques in software product lines, only a few empirical insights have been gained regarding the impact of visualization and interaction techniques on improvement of developer tasks. In this article, we aim to contribute to the body of empirical evidence on the effectiveness of visualization and interaction techniques when applied to software product line engineering. Considering a variety of approaches in software product lines (e.g. feature oriented, decision oriented, orthogonal variability, etc.) and diversity of tasks (domain and application engineering lifecycles), it would be nearly impossible to empirically investigate efficiency of visualization and interaction techniques for all the approaches and tasks in a single study. Hence, we narrow the scope of our study to feature-oriented product lines and the configuration process (i.e. feature selection) due to their significance reported in the research literature [7] . The research problem that we target within the scope of feature-oriented product line configuration is: *To what extent visualization and interaction interventions enhance the configuration process for application engineers?* The significant parts of the research question are *visualization and interaction interventions* and *measures for facilitating developers' tasks.*

To form an effective set of visualization and interaction intervention, we utilize existing theories in cognitive science [81][82] and software engineering such as the physics of notation [15] and creative support tools principles [8][33] and existing knowledge of visualizing the feature model configuration [14][11][10][38]. The detail of derivation process and the visualization and interaction intervention set were explained in Section 3. Although we have considered many resources to develop the visualization and interaction intervention set, we do not claim that this set is complete and the best set for the configuration. This set forms a basis for empirical investigation of the impacts of the visualization and interaction techniques in the configuration process and can be revised with researchers in software product line community for further studies.

To measure how well the introduced intervention set assists developers in the feature oriented configuration process, we utilized the ISO 9126 quality model [52][53], an international standard which defines a hierarchy of software quality attributes including functionality, reliability, usability, efficiency, maintainability, and portability at the highest level which later are refined into more quantifiable attributes. The reasons behind the selection of the ISO 9126 model are an extensive use of the standard in software engineering literature [54][55][56] for a similar objective (i.e., quantifying attributes for the evaluation products qualities) and fitness of the standard to the context of our study (i.e., exploring interventions effects in the configuration process). Following the guidelines of the ISO 9126 standard, we identified two tasks which developers perform during configuration of a feature model and can be used for measuring this quality attribute: *comprehension* (understanding and interpreting) of the configured feature model; and *changing* a feature model configuration. The comprehension task in the context of configuration of feature models refers to understanding a configuration model (i.e., selected and deselected features) as well as interpreting relations among selected features and functional and non-functional requirements of a target application. The changing task refers to decisions that drive the change of a feature model (i.e., selecting and deselecting the features), based on the stakeholders' requirements. Hence, the introduced visualization and interaction interventions were evaluated for their effects on the comprehension of the feature model configurations and the change of feature model configurations. Consequently, the research question is further refined into following questions:

1. To what extent do the introduced visualization and interaction interventions help software developers in the comprehension tasks in the context of the feature model configuration?
2. To what extent do the introduced visualization and interaction interventions help developers in the changing tasks in the context of the feature model configuration?
3. To what extent do the introduced visualization and interaction interventions improve the usability of the base tool?

1.2. Contributions and outline

To answers the research questions mentioned in the previous section, we devised an empirical study following principles of empirical studies in software engineering [59][60]. In our study, we observed the participants while performing comprehension and change tasks. We defined *time on task* and *accuracy* (i.e., a lower number of errors) metrics for measuring helpfulness of visualization and interaction interventions in comprehension and changing tasks. These metrics were chosen for operationalizing the effect of the introduced interventions because reducing developers' errors while developing a best configuration based on the given requirements and performance is crucial for customers and software developers, respectively. Also, in these two metrics are common for measuring the effectiveness of information visualization techniques in cognitive tasks[82]. After we defined the metrics, we hypothesized that the visualization and interaction interventions decrease time on task and improve accuracy (i.e., a lower number of errors) of developers when performing comprehension and changing of feature models. To test our hypotheses, we conducted a controlled experiment that was aimed at investigating impacts of visualization and interaction interventions. The results of our data analysis revealed that employing the proposed visualization and interaction interventions decreases the time developers spend on the comprehension and changing tasks of feature models. Moreover, the visualization and interaction interventions make the tool easier to learn and easier to use for software developers.

The paper provides the following contributions:

1. Identifies a set of visualization and interaction interventions that can have a positive effect on comprehension and changing tasks related to the configuration of feature models. The introduced set can form a basis for further investigation of the visualization and interaction interventions in the configuration process.
2. Designing, conducting, and reporting an empirical study on comprehension and changing tasks of the feature model configuration process. The methodology can be used by other researchers in software product line engineering for investigating similar research questions related to feature model configuration and configuration process of other approaches such as decision-oriented product lines [13].

The reminder of the paper is organized as follow: Section 2 introduces the notion of feature models, the configuration process, and relevant software quality background. The proposed interventions are introduced in Section 3. Research questions and hypotheses of our study are outlined in Section 4. Section 5 describes the experimental design, while Section 6 reports the results of the study. Discussion of the findings and related works is provided in Section 7, while Section 8 discusses the threats to validity of our empirical findings. Finally, we conclude the paper and point out future works in Section 9.

2. BACKGROUND

2.1 Extended Feature Model

In SPLE, feature models are a prevalent modeling language used for representing variability in the domain. The notion of feature is commonly used to represent a logical unit of behavior specified by a set of functional and non-functional requirements [43][44][45]. Therefore, a feature shows functional and nonfunctional properties of the system. Feature models also formalize commonality and variability relations of the features. A feature model has a tree-like structure whose root node represents a domain application and other nodes represent the features of products of the domain. Features in the feature model typically classified as (see Figure 1):

- *Mandatory feature*: the feature must be included in the description of its parent feature;
- *Optional feature*: the feature may or may not be included in its parent description;
- *Alternative feature group*: one and only one of the features from the feature group can be included in the parent description;
- *Or feature group*: one or more features from the feature group can be included in the description of the parent feature.

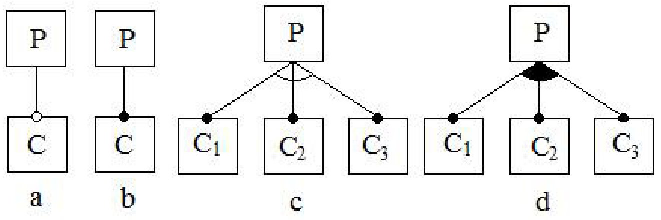


Figure 1: Different types of features in feature model a) Optional feature, b) Mandatory feature, c) alternative feature group, and 4) Or feature group

Although features represent both functional and non-functional characteristics of a family, explicit representations of a non-functional aspect have been neglected. In the other words, in many approaches, a feature in a feature model represents the functionality of a family without depicting non-functional aspects of the functionality. Recently, several efforts have been proposed in order to integrate non-functional aspects of features in feature models [4][6][3][46], referred to as extended feature models. In these works, quality attributes are defined for features and their values show the level of quality provided by the features. Examples of quality attributes are *implementation cost, time,* and *customer importance*. In an extended feature model, one can define the *security* for the *deposit money* feature and specify the level of provided security by the feature by assigning proper values into the security property of the *deposit money* feature. In our study we consider extended feature models to investigate the impact of visualization and interaction interventions for the task of configuring the feature model based on both functional and non-functional requirements.

2.2 Feature Model Configuration Process

Having developed a feature model in domain engineering, in application engineering, the most appropriate features for a target product are selected. Selection of both the most suitable and allowed combination of features (called Configuration Process) is an important issue in SPLE. Software developers can find the configuration process to be a cumbersome task for two reasons: 1) feature models may contain hundreds of features, which affect their maintainability; 2) selecting proper features requires close consideration of many factors such as technical limitations, implementation costs, and stakeholders' requests and expectations [3]. To make the most suitable decisions, one should consider all constraints and business objectives of the stakeholders and apply them in the selection process [4]. In the configuration process, software developers start with requirements (including both functional and non-functional requirements) and select a set of features satisfying the requirements. In selecting a feature for a product not only should the functional requirements be satisfied, but non-functional requirements should also be satisfied.

In order to handle the complexity of the configuration process, some techniques like *stage configuration* [9] were proposed which configuration decisions can be made in several stages. Also, several automation techniques have been proposed which employ different formalisms for developing configured feature models such as Constraints Satisfaction Problem (CSP) [69][70], Genetic Algorithm [71], Planning techniques[72], Binary Decision Diagrams [73]. However, due to the complexity of problems and need for considering functional and non-functional requirements, these approaches have scalability problems and require well defined inputs [72].

2.3 Software Quality Standard

Software quality plays an important role in the software product success. The ISO 9126 standard was defined to provide developers with a quality model for evaluating software-intensive products. To assess the quality of a software product by using ISO 9126, one should select the quality attributes and sub-attributes to be evaluated, identify the appropriate direct and indirect measures to be applied, identify the relevant metrics, and then interpret the measurement and the measurement result in an object manner [52].

According to ISO 9126, software quality attributes are categorized into internal and external quality attributes. The former refers to the attributes that can be measured purely based on software features (e.g., length of software document), while the latter refers to those quality attributes that assess how well a software system relates to its environment (e.g., reliability or maintainability) [58].

Usability and *maintainability* as two key quality attributes investigate how well a software system can be used by its users [52]. The usability attributes evaluate the effort needed for the use of a software product when the software is used under specific condition. Usability is viewed as a composition of the ease-of-learning and ease-of-use sub-quality attributes. Ease-of-learning investigates how easy a user can learn to use the software product and ease-of-use refers to how much effort is required from a user for using the software product.

The maintainability attributes are concerned with evaluating how well models developed by a software product can be understood and changed [57]. In this paper, we investigate maintainability as a composition of two sub-quality attributes namely *changeability* and *understandability* (N.B., maintainability can also include other attributes such as analyzability). According to the ISO 9126 standard these sub-characteristics are defined as follows: *Changeability* is the possibility and ease of change in a model developed using a software product when modifications are necessary; and *Understandability* (*comprehension* or *interpretation*) is the prospect and likelihood of the software model developed using a software product to be comprehended and interpreted by its users.

3. Visualization and Interaction Interventions

3.1 Identification of the visualization and interaction interventions

Following the principle of the GQM approach proposed by Basili [74], we presented the goal of this activity as follow:

Analyze <u>existing visualization and interaction literature and theories</u> for the purpose of <u>identifying effective visualization and interaction techniques</u> for <u>feature model configuration</u> from the point of view of <u>application engineers</u>.

We refined this goal into the one to identify the visualization and interaction interventions which: 1) visualize both functional and non-functional aspects of features in a feature model; 2) represent variability and constraints among features; and 3) provide advanced user interface interactivity for developers with feature models when conducting configuration.

In order to achieve aforementioned objectives (i.e. identifying a set of visualization and interaction interventions for configuration process), we adopt *top down* and *bottom up* approaches. In the top down approach, we investigated existing theories on visualization techniques [33][65][81][82]; theoretical foundations in software engineering about visualization [15], and literature on creativity support tools [8][75]. The results of this investigation lead to a set of principles and criteria which guided us in selecting proper visualization and interaction interventions.

Existing cognitive science theories [81][82] in information visualization emphasize the role of visualization and interaction techniques in amplifying cognition. These theories posit that tools augment human cognitive capabilities and enable human to accomplish more. Furthermore, distributed cognitive theory [82] considers external representation (e.g., visualization) as a part of cognitive system and suggests that cognitive tasks are accomplished by means of an interplay between perceptual processes acting with external representations (i.e. visualization) and processes acting with internal cognitive representations [82]. Moreover, the cognitive load theory suggests that unproductive (i.e., extraneous) load is caused by the limited capacity of working memory, which is predicted by the lack of schemas – i.e., prior knowledge – available in the long-term memory [82][83]. When there are no schemas that can easily be retrieved from long-term memory and activated into the working memory– i.e., information a human interacts with is not learned and stored into the long-term memory – different cues, mnemonics, and scaffolds should be used to reduce the cognitive load. Given the size of feature models, their complete storing into long-term memory can hard or sometimes impractical. This means that an extraneous load can easily emerge when developers interact with feature models that are not stored into their long-term memory, and easily activated into their working memory. Therefore, visualization and interaction techniques during a configuration process can become an important factor to improve the cognition of developers when configuring feature models.

According to Moody [15], the notations for a visual language are divided into *primary* and *secondary notations*. The primary notation defines a set of symbols and their literal meaning. For example, in the context of feature models, primary notation includes notation for defining features, mandatory, optional, OR, alternative features, and cross-tree constraints. Secondary notations, on the other hand, refer to the use of visual variables that are not formally specified in the notation of a software language in order to reinforce or clarify the meaning of a specific (set of) constructs or tasks, e.g. the use of color to highlight information [15]. In our study, we do not intend to create new visual notations for feature models, but we want to apply visual variables to add additional meanings to features (i.e., adding non-functional aspects) and provide additional information during the configuration tasks of comprehension and changing. Consequently, from the principles introduced in [15][23][33][65], we only concentrate on a subset of those related to visual variables. The identified principles are shown in table 1. We should note that we do not claim that these principles are neither a complete nor the best set of principles for visualization and interaction, but they provide a comprehensive criteria set for investigating visualization and interaction interventions fitted to the feature model configuration.

After identifying the principles fitted to the domain of feature model configuration, we followed a bottom up approach to identifying visualization and interaction techniques relevant to feature model configuration tasks of comprehension and changing. For this purpose, we investigated the visualization surveys in software engineering [5][17][18], software product line engineering [76], and the existing tools and visualization and interaction interventions for feature model configuration [14][10][38][40]. Afterward, we analyzed the identified set of visualization and interaction techniques with regard to principles identified from top-down approach and we selected a subset of these interventions for our empirical study. Next section explains the selected intervention set.

Table 1: Visualization and interaction principles based on theoretical investigation

Principle	**Description**	**Application in the Configuration Process**
Data oriented objective	The visualization and interaction technique should depend on the underlying data.	Data in the feature model configuration tasks includes features, variability relations, non-functional and functional aspects of the features.
Task oriented objective	The visualization and interaction technique should depend on the nature of the task. This includes both the domain of the task and the kind of interactions which a user makes with the presented information. For example, whether the user only views the information or makes changes as well.	The main task is the configuration of a feature model which includes both viewing and changing in the feature model.
Adequacy for complex models	The technique should include explicit mechanisms for dealing with the complexity of model.	A feature model may include a large number of features with complex dependency relationships. The approach should be able to manage this complexity.
Holistic view of data	The technique should provide a holistic view of the source data or raw material which designer is going to work with.	The approach should provide a holistic view to feature models including functional and non-functional aspects.
Low threshold, high ceiling, wide walls	The technique should be easy for novices, but possible for expert to create complex artifacts.	Application engineers with different levels of expertise should be able to perform configuration tasks.
Support exploration	The technique should support and suggest a wide range of explorations and enables the user to navigate through different possible solutions.	The technique should support exploration of different feature selection and the impact on the overall quality of the product.
The primary of shape	The technique should use shapes as a primary factor for distinguishing between different concepts of the language.	Feature model languages used different shapes to represent feature and variability relations. The technique should utilize shapes to present further information about feature models such as non-functional aspects.
Visual expressiveness	A visualization technique should use a full range and capacities of visual variables (i.e. shape, texture, brightness, size, and color) to communicate information.	Combination of visual variables should be utilized to encode the required information related to feature model configuration.

3.2 Visualization and Interaction Intervention Set

In this section, we explain the selected visualization and interaction intervention set. These interventions only concentrate on secondary notations where we apply visual variables to show additional information enhancing the cognition of software developers during the configuration process. In visual analytics techniques, the possible options for visualization are selected based on the target task and underlying data [88]. In our case, the task is the configuration process and the data is the feature model and associated data (i.e. non-functional properties). Considering these two aspects, in the following we discuss possible options and select the proper one.

Explicit vs implicit representations: In the former case, hierarchy relations between nodes are illustrated by explicit links between nodes (i.e., node-link representation). In the latter case, the hierarchy relations are shown by special arrangement of nodes [11] (i.e., space-filling representation). Examples of implicit representation are Tree-maps [29] and SunBurst [30]. The cone-trees [31] and space trees [32] are examples of explicit representation. Based on the adequacy for complex models principle introduced in Table 1 as well as considering the size of feature models and the weaknesses of implicit visual rendering of large trees, we decided to employ the explicit representation.

3D vs 2D representation interventions: 3D visualization due to the use of three dimensions is more appropriate for showing large scale hierarchies. However, a 3D representation may possess the following problems: 1) 3D representations are suitable mechanisms for showing information but changing information in a 3D representation can be a cumbersome task; 2) some researchers indicate that it is not desirable to consider a third dimension when 2D is enough for representing information (i.e., feature model, non-functional properties, and configuration) [16]; 3) 3D visualizations may have problems which include intensive computation, more complex implementation, hard user adaptation, and orientation [17]; 4) 3D visualization increases difficulty of performing actions and complexity of interactions [18][19]; and 5) finally, occlusion as a common problem in 3D representation may distort software developers during the configuration [20]. Considering these problems with 3D representation and task-oriented objective principle (with the task being configuration of feature models), and 2D is sufficient for feature models, we selected 2D representation.

Use of color intervention: In information visualization, color is used to group the items and show more information. The literature reviews show that a proper use of color can enhance and clarify a presentation [22][23]. Also, the visual expressiveness principle from Table 1 emphasizes on employing several visual variables (e.g., color) for adding more meaning to the notations. Features include both functional and non-functional aspects where the functional aspect is shown with a feature notation and the non-functional aspect is not emphasized. We used color to show the non-functional properties related to each feature in a feature model. When using colors, several challenges – e.g., how many colors should be used and how color will be used – should be addressed [86]. To this end, we assigned predefined colors to each non-functional property, the number of colors equals to the number of non-functional properties. However, software developers can define desired colors for the non-functional properties. Following the rules by Few [86], we made sure that the selected colors were consistent. Having indicated colors corresponding to the non-functional properties, the coloring can be applied to the features of the feature model for showing the non-functional properties with which they were annotated. Since a feature may be annotated with more than one non-functional property, we need a technique to represent all colors related to the non-functional properties accompanied with feature representation. Therefore, colored squares are inserted in the bottom of features (tree nodes) representing the non-functional properties assigned to the features.

Use of texture intervention: In addition to showing non-functional properties related to each feature, other information to convey is the level of the non-functional property for each feature. To represent this information we used other visual variable, i.e. texture for representing a symbol inside a color square (i.e., *"H"* as high value, *"M"* as medium, and *"L"* as low). The other option would be to use the size for representing different values for non-functional properties. However, using size requires users to make a semantic connection between size and values.

Unfolding intervention: Based on the holistic view of data principle and in order to present all features having a specific non-functional property, we designed an unfold intervention. The intervention unfolds the features and expands the feature tree to show all features having a selected non-functional property. This intervention makes it easy for software developers to quickly discover which non-functional properties are covered with the specific features; which features contain a specific non-functional property; or how the coverage of non-functional properties are in the feature model.

Detail on demand intervention: This type of interaction is used when more details about an item or a group are needed [33]. More information about features (e.g., a list of the non-functional properties, the values of the non-functional properties, feature attributes, feature description, and feature constraints) is shown in a detail view. By moving the mouse over the features in the tree or by selecting the features, a detail view is updated and information about the feature is shown.

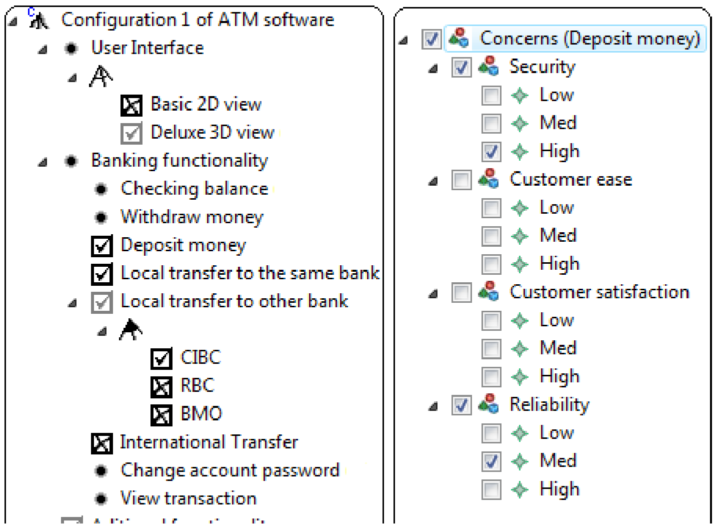
Filtering intervention: Industrial feature models contain many features organized in different levels of granularity. Unfolding the whole feature model may cause a visual clutter. Moreover, software developers may not need to view the entire feature model. Therefore, filtering intervention can help manage the complexity of the feature model by initially showing high-level features (top three levels). Next, an incremental browsing can be provided for the users by two options: 1) *Filter-level* – software developers can set the level of the feature model tree in which they are interested; then, the tool unfolds all the features between the tree root and the indicated level; and 2) *Fisheye Tree Filter* – using this option, software developers can explore the feature model by unfolding sub-features of the features they are interested in. When software developers select a feature for further exploration, the intervention closes other nodes in the same level in order to create more space to show the opened nodes. To assist software developers for identifying features which require further unfolding, the number of children of each node can be shown along with their name.

Zooming intervention: Zooming in and zooming out interventions provides the user with capabilities of concentrating on a specific part of a model and reaching to an overall view of data, respectively. In the context of the feature model configuration, the zooming intervention can be used to view either the features of interest or the overview of the whole feature model. Moreover, auto-zooming is used to fit the tree to the screen after expanding.

Highlighting intervention: Highlighting is an important visualization intervention to enhance human cognition in data-intensive applications with concentrating on critical information and filter out irrelevant information for the given task [11]. In the context of configuration this intervention is beneficial to provide additional information like status of features (i.e., selected, deselected, or undecided). Additionally, highlight intervention can be applied in combination with search so that the tool highlights the found features in the tree based on the user search.

3.3 Implementation of Interventions

In order to investigate the effect of the visualization and interaction interventions, we identified a tool which provides basic representation and interaction techniques; implemented the visualization and interaction interventions in the tool; and conducted a controlled experiment to confirm or reject the impact of the introduced interventions on the configuration process. Among existing tools, we selected a well-known feature model plug-in called fmp [26] for the following reasons: 1) fmp supports very basic visualization (i.e., it uses an indented list to show feature models) and interactivity; however, it provides a good facilities to define a feature model and to manage its variability; 2) the tool is open-source and is accessible for academic researchers, so that we could add the interventions to the tool easily; 3) the tool provides all primary notations of the feature model including notation for mandatory, optional, or, alternative relations. This makes the tool a convenient platform for implementing the visualization and interaction interventions which are secondary notations for clarifying meaning and adding extra information; 4) Finally, our goal was not the comparison of different tools in software product lines, but rather we aimed to empirically validate if the identified set of visualization and interaction interventions could help software developers in the configuration process. One approach for evaluation would be studying the existing tools with respect to visualization interventions which we tried to make an effort in section 3.4.
The *fmp* tool is an Eclipse plug-in developed by the research group of Krzysztof Czarnecki at the University of Waterloo. *Fmp* supports editing and configuring feature models [26]. The screen shot of the tool is shown in Figure 2a. The tool provides some limited interactivity functionality such as search for a feature by name and expanding composite features.



- Configuration 1 of ATM software
  - ● User Interface
    - ☒ Basic 2D view
    - ✓ Deluxe 3D view
  - ● Banking functionality
    - ● Checking balance
    - ● Withdraw money
    - ☑ Deposit money
    - ☑ Local transfer to the same bank
    - ✓ Local transfer to other bank
      - ☑ CIBC
      - ☒ RBC
      - ☒ BMO
    - ☒ International Transfer
    - ● Change account password
    - ● View transaction

- ☑ Concerns (Deposit money)
  - ☑ Security
    - ☐ Low
    - ☐ Med
    - ☑ High
  - ☐ Customer ease
    - ☐ Low
    - ☐ Med
    - ☐ High
  - ☐ Customer satisfaction
    - ☐ Low
    - ☐ Med
    - ☐ High
  - ☑ Reliability
    - ☐ Low
    - ☑ Med
    - ☐ High

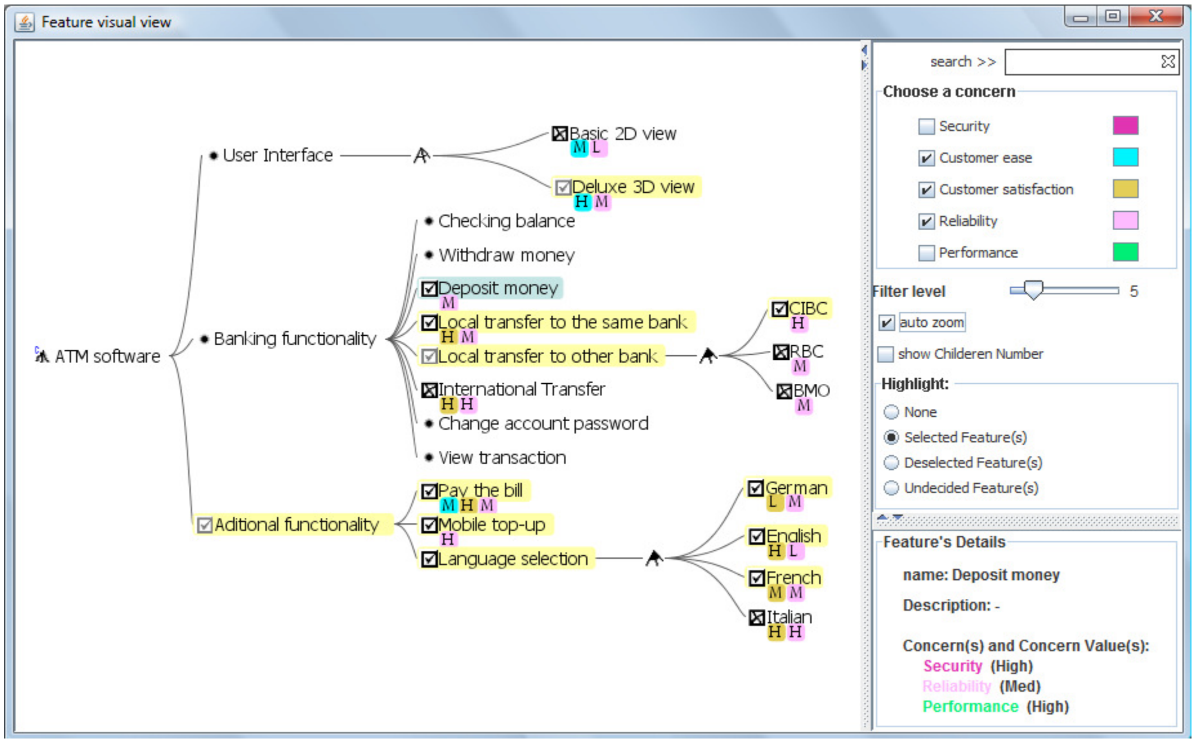
Figure 2: Screen shot of Feature Model Plug-in tool (a) Feature Model, (b) non-functional properties related to the *Deposit money* feature

Fmp supports standard feature models and does not include any support for non-functional properties along with the features. However, we considered extended feature models which include both functional and non-functional aspects of features. Hence, we extended the *fmp* tool to incorporate non-functional properties into the feature model (see Figure 2) and enabled configuration based on both functional and non-functional requirements. The non-functional properties are represented in the separate view shown beside the feature model view. By clicking on the features, the non-functional properties related to each feature are represented in the non-functional view and software developer can edit them or annotate features with new non-functional properties.

During configuration process, application engineers can view both the functionality of a feature (shown in feature view – Figure 2(a)) and its non-functional properties (shown in non-functional view – Figure 2(b)) and decide to select or deselect the feature based on the requirements.

To implement the interventions in the fmp tool, we used prefuse [19], a Java graphic library with different tree and graph views. One of the 2D tree-views in prefuse provides means to fulfill all of our requirements discussed above. We used this tree as an initial representation and extended it for our purposes (i.e., visualizing feature models, non-functional properties, and configurations). In this tree, that provides a layout of this visualization, nodes represent the features of the feature model and links show the composition and variability relations between features. A screenshot of the tool is shown in Figure 3. Influenced by Shneiderman's information visualization interaction mantra [33] (i.e. "overview first, zoom/filter, details on demand") and similar to Keim et al [87], we defined feature model configuration interaction mantra as 1) show overall feature model configuration, 2) show undecided features, 3) zoom, filter, and analyze further, and 4) details on demand for features.

In the reminder of this paper, we refer to *fmp* tool as a *basic tool* and refer to extended version of *fmp* with the visualization and interaction interventions as a *visually-enhanced tool*[a].



search >>

Choose a concern

[ ] Security	🟪
[✓] Customer ease	🟦
[✓] Customer satisfaction	🟨
[✓] Reliability	🟪
[ ] Performance	🟩

Filter level 5

[✓] auto zoom

[ ] show Childeren Number

Highlight:

- ○ None
- ● Selected Feature(s)
- ○ Deselected Feature(s)
- ○ Undecided Feature(s)

Feature's Details

name: Deposit money

Description: -

Concern(s) and Concern Value(s):

Security (High)
Reliability (Med)
Performance (High)

Figure 3: The implementation of visualization and interaction interventions

[a] Demo of the tool is available at: http://www.sfu.ca/~masadi/Demo%201.1.html

3.4 Investigating Feature Modeling Tools w.r.t Visualization and Interaction Interventions

In our study, one of the main sources of identifying visualization and interaction interventions was the existing tools in feature-oriented software product lines. We selected the tools by investigating previous studies on the evaluation of feature model tools [79][80] as well as main software product lines research venues such as the SPLC conference. This section evaluates a set of the tools used for identifying interventions against the selected interventions. Our criteria for considering tools for evaluation were: 1) Tools provide configuration support and employ visualization and interaction techniques during configuration; 2) tools are designed to support feature model configuration; 3) There are available resources related to tools to explore tools configuration capabilities. For example, we did not have access to GEAR and RequiLine, hence, we could not analyze these tools. In the following we briefly introduce each tool.

Feature model plug-in (fmp) [25]: The fmp tool is an Eclipse plug-in which supports editing and configuring feature models [25]. The tool uses an indented list for visualizing feature models and facilitates the configuration process by providing configuration options for software developers. The tool provides some limited interactivity functionality such as search for a feature by name and expanding composite features. Fmp supports standard feature models and does not include any support for non-functional properties along with the features.

pure::variants [24] is a commercially robust Eclipse plug-in developed by Pure-Systems company. The tool uses an indented list along with a visual representation of feature models. The tool supports fast navigation to feature by double clicking on the displayed entry. It also shows the properties of a feature when the feature is selected. In addition to a tree representation, during the configuration, a matrix based representation is provided in the tool.

Feature Modeling Tool [35] represents feature models by using an indented list and a tree structure (non-commercial tool). The tool is integrated into Visual Studio IDE, by using the Microsoft DSL tools. It can support a visual representation and manipulation of features and integrity constraints (i.e. include or exclude relations among features). With respect to configuration support, the tool provides an intended list for visualizing feature. The tool does not provide mechanisms for representing the non-functional aspects for features.

S^2T^2 tool[78]*: Botterweck et al.* provided a tool to support feature model configuration (non-commercial tool). They employed colors for visualizing dependencies between features and feature states. Additionally, features are decorated by iconic representations of common cardinalities.

XFeature [38]: This tool is a commercial eclipse plug-in developed by an association of P&P Software Company with the Swiss Federal Institute of Technology (ETH). The tool provides an outline view of feature models (a holistic view of the model) and shows extra information of a feature in a separate property view. In order to present that a feature has attributes, the tool renders a small circle along with the feature. The tool supports zooming in and zooming out and uses texture for showing annotation over features.

faMa Tool Suit [39][40] are also used for editing and configuring feature models (a non-commercial tool). The tool suite provides an explicit representation of feature models in 2D representation. It supports extended feature models and represents non-functional aspects of features along with their functional aspects.

FeatureIDE [10]*:* The tool is an eclipse plug-in which provides explicit representations of feature models and their configurations (non-commercial tool). The tool includes a configuration editor to create and edit configurations and has a support for deriving valid configurations. Color is used to shows difference between abstract features and concrete features. Also, during the configuration process, a combination of color and texture is used to support user in creating a valid configuration. Features whose selection or elimination turns an invalid configuration into a valid one are marked with different color. Colored checked boxes in the advanced configuration view are used to define different states of feature (i.e., selected, deselected, and undecided).

Visual and Interactive Tool for Feature Configuration (VISIT-FC) [14][5]*:* This tool utilizes visualization techniques in order to enhance developers' tasks such as feature model configuration. The tool uses a 2D tree for representing feature models. The tool provides detail of demand, incremental browsing, and focus+context capabilities and uses colors to show different configuration states (i.e. selected, deselected, and undecided).

Table 11: Evaluation of feature model configuration tools against visualization and interaction intervention set

	Explicit Representation	2D Representation	Use of Color	Use of texture	Unfolding	Detail on demand	Filtering	zooming	Highlight
Feature Model Plug-in (fmp)	+	+	–	–	+	+	–	–	–
Pure::Variant	+	+	–	+	+	+	+	+	–
Feature Modeling Tool	+	+	–	+	+	+	–	–	–
S2T2	+	+	+	+	+	+	–	–	–
XFeature	+	+	+	+	+	+	–	+	–
Fama Tool suite	+	+	+	+	+	+	–	–	–
Feature IDE	+	+	+	+	+	+	–	–	–
VISIT-FC	+	+	+	+	+	+	+	+	+

Table 11 shows the evaluation of feature model configuration tools against the visualization and interaction interventions. All the selected tools support explicit representation and apply 2D representation for visualizing the configuration space. Pure::Variant provides a matrix representation in addition to a tree representation during the configuration process. Detail on demand and unfolding are other interventions which are supported by all tools. Among identified intervention, highlighting, zooming, and filtering have received less attention from the developers of these tools. Results of our empirical study show empowering the tools with these interventions helps software developers in the configuration of feature models.

4. Research Questions and Hypotheses Formulation

The main purpose of our research was to assist software developers in the configuration process by enhancing their cognition of a feature model by using visualization and interaction techniques. At this point of our work, the following questions were raised: 1) Do the employed visualization and interaction interventions enhance the comprehension of the feature model and assist software developers in a configuration process? 2) Do the employed visualization and interaction interventions improve usability aspect (ease-of learning and ease-of-use) of the basic tool? That is, we were interested to know the effects of the visualization and interaction techniques on the performance of software developers. Moreover, we intended to know if the employed visualization and interaction techniques improve usability of the basic tool (i.e. ease of learning and ease of use). Consequently, we designed and executed an empirical evaluation to find out whether applying visualization helps software developers in their tasks during a configuration process and increases usability aspects.
To evaluate the impacts of the interventions using empirical study, first, we considered the ISO 9126 standard to derive the quality attributes suitable to our research questions. The maintainability and usability quality attributes were selected based on the first and second research questions, respectively. Next, the understandability and changeability sub-quality attributes from maintainability were chosen and two following tasks were devised to enable measuring the effects of visualization and interaction interventions on these quality attributes:

- *Understanding and interpreting feature model configuration* (Comprehension task): During a feature model configuration, software developers should have a good insight into the feature model and integration of its non-functional properties allowing for efficient comprehension of models (time-wise). Also, an effective representation causes to avoid the misunderstanding of relations between features and non-functional properties, and incorrect decisions during the configuration process.
- *Comprehending change to perform* (Changing task)*:* Stakeholders' requirements change during the development of applications. Changes in requirements urge changes in a configured feature model. Therefore, understanding required changes in the configured feature model and making right decisions to select or deselect features based on the changed requirements is common activity in a configuration process. Thus, a representation that facilitates making changes and helps to point out the impacts of change easily, correctly, and quickly is desirable.

In order to operationalize the evaluation of the proposed visualization and interaction interventions during these two tasks derived from understandability and changeability quality attributes, we defined two metrics for each quality attribute including *time efficiency* and *accuracy* (i.e., having fewer errors) of developers while performing the tasks. Accuracy is calculated according to the number of correct answers for the comprehension and changing tasks. Therefore, according to the types of two tasks (i.e., comprehension and changing) and measurement variables (i.e., time and accuracy), we formulized the following hypotheses:

- ***H1:*** *The visualization and interaction interventions significantly decrease time spent by software developers while performing the comprehension task on a configuration of a feature model.*
- ***H2:*** *Visualization and interaction interventions significantly increases the accuracy of software developers (in terms of fewer errors) while performing the comprehension task on a feature model.*
- ***H3:*** *Visualization and interaction interventions significantly decrease time spend by software developers while performing the changing task on a configured feature model.*
- ***H4:*** *Visualization and interaction interventions significantly increases software developers' accuracy (in terms of fewer errors) while performing the changing task on a configured feature model.*

With respect to the usability quality attribute, we considered ease-of-use and ease-of-learning sub-qualities. Afterwards, we derived four metrics for each of these quality attributes and defined a Likert-point scale measuring system. We formulated the following hypotheses regarding usability of visually-enhanced tool.

- ***H5:*** *A feature model configuration tool with the employed visualization and interaction interventions is significantly easier to use than the tool without interventions when performing maintainability tasks (i.e. changing and comprehension).*
- ***H6:*** *A feature model configuration tool with the employed visualization and interaction interventions is significantly easier to learn than the tool without the interventions when performing maintainability tasks (i.e. changing and comprehension).*

The above hypotheses are summarized in Table 2. They are categorized based on the quality attributes derived from the ISO 9126 standard.

TABLE 2: FORMAL DEFINITIONS OF THE EXPERIMENT HYPOTHESES
visually-enhanced tool refers to implementation of visualization and interaction interventions in the basic tool

QA	Sub-QAs	Hypotheses
Maintainability	Comprehension	H1: Completion time(basic tool) > Completion time(visually-enhanced tool) H2: Number of correct answer(basic tool) < Number of correct answer (visually-enhanced tool)
	Changeability	H3: Completion time(basic tool) > Completion time(visually-enhanced tool) H4: Number of correct answer(basic tool) < Number of correct answer (visually-enhanced tool)
Usabi lity	Ease of use	H5: Effort to use (visually-enhanced tool) < Effort to use (basic tool)

	Ease of learning	H6: Effort to learn (visually-enhanced tool) < Effort to learn (basic tool)

5. Method

Figure 4 illustrates the method followed the design and executing of the empirical study. The method consists of the study design and study execution phases. In this section, we explain different activities of these two phases.

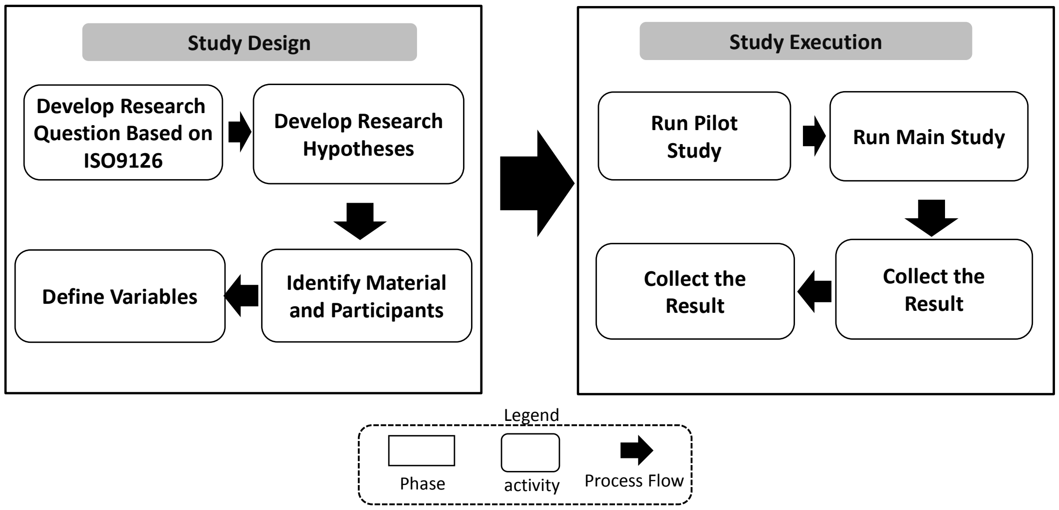


Figure 4: The experiment method

5.1 Study Design

The purpose of our study is to evaluate the efficiency of the visualization and interaction interventions based on the ISO 9126 quality evaluation framework. Accordingly, a number of research questions corresponding to quality aspects defined in ISO 9126 (e.g. usability and maintainability) were described. The types of our research questions are causality-comparative questions [59], which aim to investigate relationships between different causes (i.e. the relation between the employed visualization and interaction interventions in visually-enhanced tool and the basic visualization and interaction techniques in basic tool). The hypotheses corresponding to the research questions formulated the comparative cause-effect relationships in terms of quantitative variables (e.g. time and number of correct answer). As a result, a controlled experiment is more suited to our research questions and hypotheses than the other existing research methods in empirical software engineering such as case-study or ethnographies. Hence, we designed our study by following the guidelines provided for designing controlled experiments [60]. Given the similarity of the research purpose, the controlled experiment was used in other papers such as [61].
Due to limitation on the number of participants, we conducted a repeated measure experiment in which participants performed tasks with both implementations of the tool (i.e. basic tool and visually-enhanced tool). In a repeated measure experiment (also known as within-subject design), participants' performance is measured before (using basic tool) and after a treatment (using visually-enhanced tool) [51]. Using this method, variability between participants gets isolated and analysis can focus more precisely on the effects of the employed visualization and interaction interventions. We balanced the order of visually-enhanced tool and basic tool, that is, a half of the participants first used basic tool and then visually-enhanced tool and the rest used tools in inverse order. The participants were randomly distributed to these two groups. Moreover, we designed different tasks with different questions for each tool.

5.2 Context

5.2.1 Participants

The experiment was executed at Simon Fraser University in Canada and involved 20 graduate students and trainees (including 10 master's students, nine PhD students, and one postdoctoral fellow) of the Department of Computer Science and the School of Interactive Art and Technology. Since our context was the configuration process in SPLE and we used feature models for configuration, the participants were required to be familiar with software modeling languages such as Feature Models (FM), Unified Modeling Language (UML), and Entity Relationship Diagram (ERD). This constraint (i.e., understanding of the used materials - feature models and non-functional properties) caused limitations in recruiting participants for the study. On the other hand, we aimed at having enough participants to ensure that the p value of less than .05 is considered significant in our data analyses. Therefore, we calculated the sample size for the study using JMP tool[b] by setting power level into 0.95. According to the result, 16 participants was the least number of participants to ensure the 0.95 power level. All participants had a good knowledge of software engineering and based on their answers in a pre-questionnaire they were all familiar with the abovementioned modeling languages. Their background knowledge made them suitable for our study.
In this study, all the participants agreed to take part in the experiment, and received neither financial nor non-financial credits for their participation. The participants who responded to our invitation to take part in the study successfully completed all the tasks of the study. The last point regarding participants is that due to the difficulty of obtaining professional subject, we select students as participants. According to Basili and Lanubile [34] using students in experiments is well-suited.

5.2.2 Materials

In this study, we used basic implementation of Feature Model Plug-in (basic tool) and implementation containing the proposed visualization and interaction interventions (visually-enhanced tool). Participants performed designed tasks using those two implementations of fmp. Moreover, a demo for both basic tool and visually-enhanced tool was created in order to train the participants. The demo explained: 1) all parts of the tools; 2) details of each part; and 3) how users interact with tool and perform the feature model configuration. In the demo, an ATM machine feature model was used which contained six mandatory features, eight optional features, two OR-group features, and one alternative group feature. For experiments, different tasks were designed with different feature models (different from the feature model used in the demo).

In feature modeling, we deal with two types of complexity: the first one is the complexity of a domain for which the feature model is designed, and the second one is the internal structure of the model, as per the general principle identified in [50], and empirically confirmed for feature models in [6]. These complexities may have an influence on both the comprehension and changing tasks in the experiment. The aim of our work (developing visualization and interaction interventions) does not deal with the first type of complexity; hence, we used feature models about domains with which all the participants were familiar with (i.e., the electronic shopping feature model). The second type of complexity has been controlled by designing tasks with different levels of complexity (i.e., so-called simple and complex feature models).
With respect to questionnaires, three forms were created including: 1) pre-questionnaire; 2) the instruction of performing tasks and some questions regarding those tasks; and 3) post-questionnaire. The pre-questionnaire includes questions about the educational background of participants, their familiarity with software modeling languages.

The instruction forms contained instructions for the tasks, two questions for every comprehension task, and one

[b] A statistical tool: http://www.jmp.com/

question for every changeability task. Features in a feature model represent both functionalities and non-functionalities of a family and both aspects are taken into account in configuration process. Hence, data to visualize are both functional and non-functional aspects of a feature model. Considering the *data oriented objective* principle in Table 1, we developed two questions: the first question was concerned with the comprehension of the functionality aspect, and the second question was concerned with the comprehension of the non-functionality aspect. When designing these questions, we considered visualization and interaction principles introduced in Table 1. For example, as shown in samples of the comprehension question in Table 3, for designing the comprehension of the functional aspect, we considered the *support for exploration, holistic view of the functional aspect,* and *representing variability relations between features* principles. To answer this question, developers need to explore different features and relate them to functional requirements, investigate the variability relations between features, and have a holistic view on the feature model. Also, for the comprehension of non-functional aspect the same principles are applied. Similarly, in the changing task, the changes were made based on both functional and non-functional requirements and principles of visualization and interaction are considered in developing of questions. For instance, for providing *medium* and *high international sale*, developers should explore non-functional values of features (e.g., *support for exploration*), consider the variability relations when selecting a desirable feature, and have a holistic view of the overall provided *international sale* non-functional property. However, we combined two aspects (i.e. functionality and non-functionality) in one question in which features must be selected or deselected based on requested functionalities and non-functionalities. Finally, the post-questionnaire encompasses two categories of questions about ease-of-use and ease-of-learning of the tools and each group has four questions. As mentioned in Section 4, in order to evaluate usability of visually-enhanced tool, the ease-of-use and ease-of-learning sub-quality attributes were defined. Next, for each of these sub-qualities, four metrics were defined. A Likert-point scale question corresponding to each metric was created and subjective opinions of participants were collected. We used a 5-point Likert scale including: strongly disagree, disagree somewhat, neither agree nor disagree, agree somewhat, and strongly agree which were encoded to 1, 2, 3, 4, and 5, respectively. Tables 4 and 5 show the questions of the post-questionnaire.

Based on hypotheses 1-4, we required to measure completion time and participants' accuracy (number of correct answers). Hence, *Camtasia Studio* (screen recording software) was employed for logging the participant's activities. We calculated the completion time for each task performed by each participant and number of correct answers in changeability task based on *Camtasia Studio* log files.

TABLE 3: SAMPLE QUESTIONS FOR COMPREHENSION AND CHANGEABILITY TASKS

Task	Aspect	Question
Comprehension and Interpretation	Functionality Aspect	Which sentence(s) is (are) correct about the configuration? (Select one or more) Both Advanced and Basic search options are available on the configured catalog. Customer review, Availability, Detail description, and Documents are selected for product. 3D image, Gallery, and Video are selected Associated assets in the product. Filtering based on Price and Quality is possible and Services and Electronic goods are Product types exist in the catalog.
	Non-functionality (Non-functional properties) aspect	Which of the following statement(s) is (are) correct according to the configuration? (Select one or more) *Performance* is not important At most medium *cost* is required (All the selected feature have low or medium *cost*) At least medium *security* is covered (All the selected features have medium or high *security*)

		For the Expiration feature lower *cost* is more important than higher *security*
Changeability	Both Functional and non-functional Aspects	Change the configuration based on following requirements: For customers, *international sale* and *cost* become very important. Select all allowable feature(s) with medium or high *international sale* and low or medium *cost*. Deselect the features with high *cost* and low *international sale*. The customer also asked that Detail of Address should not be City. City must not be selected even though it covers the non-functional properties.

TABLE 4: FIRST PART OF THE POST QUESTIONNAIRE

Ease of use	
Q1	The tool is user friendly.
Q2	Using the tool is effortless.
Q3	Interaction with the tool is clear and understandable.
Q4	Overall, I find the tool easy to use.

TABLE 5: SECOND PART OF THE POST QUESTIONNAIRE

Ease of learning	
Q1	I learned to use the tool quickly.
Q2	I easily remember how to use the tool.
Q3	Learning to operate the tool is easy for me.
Q4	I quickly became skillful with the tool.

5.3 Defining Variables

5.3.1 Independent Variables and Confounding Variables

We conducted a single factor experiment where the main factor is visualization and interaction techniques used in the interface (i.e. *visually-enhanced tool*, these techniques have been explained in section 3). Therefore, one independent variable (visualization technique) with two levels (basic tool and visually-enhanced tool) was defined. To assess better the effects of visualization and interaction techniques, we controlled other factors (i.e. confounding variables), which could have an influence on the results of the experiment. We were concerned with the following confounding variables:

- **Complexity of the feature model:** As explained in Section 5.1.2 (materials), we controlled two types of feature model complexity. All the feature models used in the tasks were selected from the electronic shopping feature model. All parts of that feature model are easy to understand (e.g., checkout, catalog, taxation, and password) and participants were familiar with them. Moreover, tasks designed with two levels of complexity: simple and complex feature models. According to the study of Bagheri & Gašević [6] some structural metrics can be considered as good predictors of complexity of feature models and assessing the maintainability of a software product line. They performed a controlled experiment to find out the relation between the proposed structural metrics and maintainability (i.e. analyzability, changeability, and understandability) of feature models. Based on the achieved results the number of features (NF) and number of leaf features (NLeaf) are good indicators for understandability and changeability of a feature model. Hence, a feature model with higher values of the NF and NLeaf metrics can be considered more complex. It is possible that the efficiency of employed visualization techniques on comprehension and changeability tasks varies for different feature models with various level of complexity. Hence, we decided to design feature models with two levels of complexity designed based on metrics identified by Bagheri & Gašević [6]. When designing complex feature models, we find out that for very large feature models, it would require us to take longer time for the experiments so that the participants had enough time to get familiar with the entire models. This would problematic as the experiment would take much longer and would prevent us to control fatigue effect. Thus, the size of the feature models was determined by bearing in mind the balance between the actual size of models and fatigue effect. Information about four designed feature models are shown in Table 6. We have reported the number of total features (NF) and the number of leaf features (NLeaf) as well as the number of different types of features (i.e. Mandatory feature, Optional feature, OR feature group, and Alternative feature group) for each feature model. The participants completed both the comprehension and changing tasks on simple and complex feature models by using both the basic tool and visually-enhanced tools.

TABLE 6: FEATURE MODELS INFORMATION

Task	Info. Complexity	NF	NLeaf	Mandatory	Optional	OR-feature	X-OR
Comprehension	Simple [Password]	11	7	3	0	8	0
	Complex [Catalog]	30	20	5	10	15	0
Changeability	Simple [Taxation]	15	9	7	1	7	0
	Complex [Check Out]	30	23	5	7	15	3

- **Learning effect:** For controlling the learning effect during the experiment, we tried to train participants before running the experiment. To this end, the demo of basic tool and visually-enhanced tool was created and the participants were asked to follow the demo thoroughly. We answered the participants' questions about the tools while they were watching the demo. Moreover, a preparation task was designed to make the participants familiar with the tasks and provide them with an opportunity to interact and practice with both the tools. Hence, we made sure that before performing the real tasks all the participants had been trained properly. The feature model used in the preparation task was totally different from the feature models in the tasks of the experiment reported. Only the structure of the training tasks and the type of asked questions were similar to the real tasks. Therefore, the participants had become familiar with tools and tasks before performing the real tasks.

5.3.2 *Dependent Variables*

The aims of this experiment are: first to find out the accuracy and performance level of software developers by using visually-enhanced tool versus basic tool (Hypotheses 1-4); second to evaluate usability aspects of visually-enhanced tool versus basic tool (Hypotheses 5-6). For measuring the efficiency and accuracy, we used the following metrics respectively: 1) completion time of the tasks; 2) number of incorrect answers to the questions given regarding the tasks. As explained in Section 5.1.2, we logged the activities of users during performing the tasks using Camtasia Studio (a separate log file for each task and participant). The values of these variables (i.e. completion time and number of correct answers) were extracted from log files. With regard to usability, ease-of-use and use-of-learning were defined as dependent variables whose values were captured through post-questionnaires given to the participants.

5.4 Experimental Procedures

The experiment was conducted in the Laboratory for Ontological Research of the School of Interactive Art and Technology at Simon Fraser University and took around one hour per each participant. During the experiment, we followed this procedure:

1. The participants were given the pre-questionnaire, which contained the questions regarding their knowledge level about the context of study (i.e. software product line and feature model).
2. The participants watched the demo of the tools (explained in Section 5.1.2) and their questions regarding different parts of tools were answered.
3. The participants were given the preparation tasks' form containing instruction about the preparation tasks and some questions regarding the performed tasks. They completed the preparation tasks with both the tools in order to get ready for doing real tasks.
4. After the preparation phase, the participants were given the first form containing four tasks in the following order: 1. simple comprehension task; 2. simple changing task; 3. complex comprehension task; and 4. complex changing task. The participants were asked to record their screen during doing tasks by Camtasia Studio.
5. After completing the tasks of the first form, the participants were asked to fill up the first post-questionnaire. They were asked to complete the post-hoc questionnaire about perceived ease-of-use and ease-of-learning of the tool (i.e., visually-enhanced tool or basic tool), with which they worked in the previous stage.
6. The participants were given the second form containing four tasks in the same order as in Step 4. The tasks of the second form were different from the tasks in the first one in terms of required comprehension and changes. The group of participants, who used the basic tool for performing the tasks from the first form, completed the tasks of the second form with the visually-enhanced tool and vice versa.
7. Finally, the participants were asked to fill out the second post-questionnaire related to the tool with which they performed the tasks in step 6.

By finishing the experiment, we acquired three filled questionnaires (one pre-questionnaire and two post-questionnaires) and eight files (recordings of screen capturing) for each participant.

5.5 Pilot Study

In order to verify the experimental parameters and optimize them, we conducted a pilot study with two participants. The participants did not take part in the actual experiment later on. One of the participants performed the tasks with basic tool first and then with visually-enhanced tool, and the other one did the tasks in the reverse order.

As a result of the pilot study, we revised both the tasks and the feature model configurations to make them clearer and feasible. For example, we disambiguated two options of the comprehension questions and one of the changing tasks. Other than these unclarities, the tasks were found to be sufficiently feasible for both the basic tool and visually-enhanced tools; and performing them did not take time more than our expected time.

5.6 Data Analysis

According to the type of study (i.e. repeated measures design), hypotheses (causality-comparative questions), and independent and dependent variables; appropriate tests were identified for the hypotheses. Generally, for repeated measure designs where each participant performs the tasks on all the groups (e.g., the basic tool and visually-enhanced tool groups), the suitable paired test is employed. Since the dependent variables (i.e., completion time, number of correct answers, ease-of-use, and ease-of-learning) are numerical and the independent variable (i.e., visualization technique) is categorical with two levels, we can either use one-tailed paired t-test or Wilcoxon Sign-Rank test. One-tailed paired t-test is a well-known statistical test for comparing the means of two groups of data with normal distribution. It is also suitable for the samples with small sizes. The Wilcoxon Sign-Rank test is a none-parametric statistic test which is an alternative to the paired t-test when the sample is not normally distributed.

To identify appropriate test for the hypotheses, we investigated normality of the involved variables: 1) time completion variable is continuous variable and based on the Shapiro W test it has a normal distribution. Therefore, the one-tailed paired t-test was used for comparing the mean of this variable in two groups (Hypotheses 1 and 3); 2) the number of correct answers, based on the Shapiro W test, was not distributed normally, so we used the Wilcoxon Sign-Rank test for comparing two samples in terms of accuracy (Hypotheses 2 and 4); 3) the collected data regarding ease-of-use and ease-of-learning also did not have normal distribution, so we used the Wilcoxon Sign-Rank test to compare two tools in terms of these variables (Hypotheses 4 and 6).

As mentioned in Section 5.2.1, complexity of a feature model is considered as a confounding variable and we designed two variants for each task – one with simple feature model and one with complex feature model – to handle this variable. As a result, two variant of hypotheses 1-4 were tested - one for simple task (e.g. simple comprehension task) and one for complex task (e.g. complex comprehension task).

6. Experimental Results

In this section, we report on the results obtained in conducting the experiment. To perform the analysis, we used the JMP v8.0.2 statistical tool.

6.1 Descriptive Analysis

In Tables 7 and 8, descriptive statistics (i.e. median, mean, and standard deviation values) for the completion time (measured in seconds) are reported—grouped by complexity of feature model and tools. As the results show, for the comprehension and changing tasks, the mean and median values of the completion time by using the visually-enhanced tool are less than the mean and median values of the completion time by using the basic tool.

TABLE 7: DESCRIPTIVE STATISTICS OF COMPLETION TIME (IN SECONDS) FOR THE **COMPREHENSION** TASKS

Feature model	Tool	Median	Mean	Std Dev
Simple	Basic tool	214.50	218.60	80.21
	Visually-enhanced tool	177.50	186.50	61.60
Complex	Basic tool	376.00	350.65	93.77
	Visually-enhanced tool	248.00	273.05	86.81

TABLE 8: DESCRIPTIVE STATISTICS OF COMPLETION TIME (IN SECONDS) FOR THE **CHANGING** TASKS

Feature model	Tool	Median	Mean	Std Dev
Simple	Basic tool	145.75	190.50	67.64
	Visually-enhanced tool	116.75	150.00	52.46
Complex	Basic tool	260.50	257.40	84.20
	Visually-enhanced tool	173.50	177.50	59.10

Descriptive statistics (i.e. median, mean, and standard deviation values) for the number of incorrect answer are reported in Tables 9 and 10—grouped by the complexity of the feature models and tools. As the results show, for the comprehension and changing tasks, the mean and median values of the number of incorrect answers by using visually-enhanced tool are less than the mean and median values of the number of incorrect answers by using basic tool (except for the simple comprehension task).

TABLE 9: DESCRIPTIVE STATISTIC OF THE NUMBER OF INCORECT ANSWERS FORTHE **COMPREHENSION** TASKS

Feature model	Tool	Median	Mean	Std Dev
Simple	Basic tool	0.50	0.55	0.60
	Visually-enhanced tool	0.00	0.55	0.68
Complex	Basic tool	1.00	0.80	0.89
	Visually-enhanced tool	0.00	0.50	0.82

TABLE 10: DESCRIPTIVE STATISTIC OF THE NUMBER OF INCORECT ANSWERS FOR THE **CHANGING** TASKS

Feature model	Tool	Median	Mean	Std Dev
Simple	Basic tool	0.00	0.70	0.98
	Visually-enhanced tool	0.00	0.55	0.89
Complex	Basic tool	1.50	1.50	1.19
	Visually-enhanced tool	0.00	0.75	0.97

As it is shown in Tables 7 to 10, the standard deviations of dependent variables for visually-enhanced tool are always lower than dependent variables for basic tool (except for the number of incorrect answers in the simple comprehension task). This indicates that the data points vary less from the mean values. Based on this, there is some evidence that the participants' performance is more consistent while using visually-enhanced tool rather than using basic tool.

Figures 5 and 6 show the descriptive statistics of data collected by the post questionnaires about the perceived ease-of-use and ease-of-learning of the two implementations of the tool investigated in the experiment.

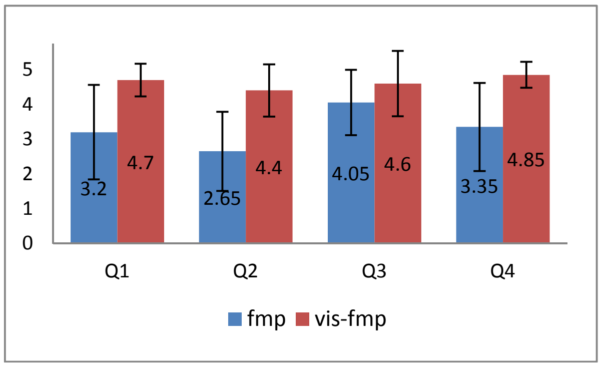


Figure 5: Descriptive statistics of the participants' feedback about the perceived ease of use of the tools

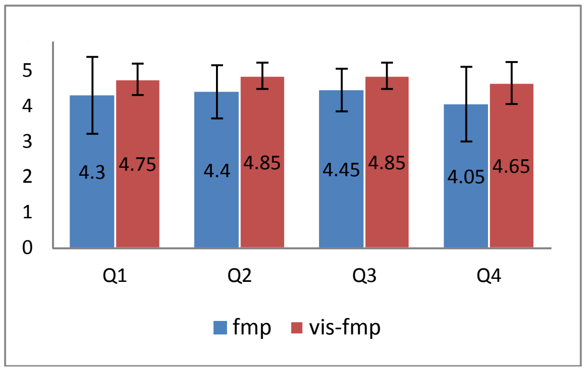


Figure 6: Descriptive statistic of the participants' feedback about the perceived ease of learning of the tools

According to the results, the visually-enhanced tool has the higher mean values for the perceived ease-of-use and ease-of-learning than the basic tool. Further details of the results are discussed in Section 6.3.

6.2 Hypotheses Testing

In this section based on the data and our hypotheses we apply suitable statistical tests (e.g., t-test and Wilcoxon Sign-Rank test). The test results are used to statistically support or reject hypotheses and show if there is statistically significant difference between two tools. In addition to these tests, we also compute the effect size for each statistical test to find out the magnitude of differences. According to guideline of Cohen [85], effect size 0.2, 0.5, 0.8 shows small effect, medium effect, and large effect, respectively.

6.2.1 Basic tool vs. Visually-enhanced tool based on Comprehension Tasks

According to our dependent variables (i.e. completion time and number of incorrect answers), we formalized two hypotheses for the comprehension tasks (i.e. H_1 and H_2). The null hypothesis for H1 is formalized as a follow:

- ***$H_0$1:*** *There is no significant difference between the time completion of the comprehension task done by software developers using basic tool and visually-enhanced tool (including the visualization and interaction interventions).*

Results–First, we report the results for the complex comprehension task. As the distribution of the collected data for completion time was normal, the paired t-test *for the complex comprehension tasks* revealed that there is a *significant difference in the mean values of the completion time* for basic tool (M=350.65, SD=93.77) and visually-enhanced tool (M =273.5, SD= 86.81), t(19) = 5.75, $p < 0.0001$. The effect size is equal to 0.80 which means applying visualization and interaction interventions has a large effect in reducing time required for comprehending complex feature models. Based on the paired t-test *for the simple comprehension tasks*, there is *a significant difference in the mean values of the completion time* for basic tool (M =218.60, SD= 80.21) and visually-enhanced tool (M =186.50, SD= 61.60), t (19) = 4.17, p = 0.0005. Hence, we can reject the null-hypothesis $H_0$1. The effect size is equal with d = 0.70 which shows applying visualization and interaction interventions shows a medium effect in time required for comprehending simple feature models.

The second hypothesis (H2) for the comprehension tasks compares the accuracy of software developers while performing the tasks by using the basic tool and visually-enhanced tools. The corresponding null hypothesis is as follows:

- ***$H_0$2**: There is no significant difference between the accuracy (i.e. fewer errors) of the comprehension task done by software developers using basic tool and visually-enhanced tool (including the visualization and interaction interventions).*

Results–The results of the Wilcoxon Sign-Rank test indicate that for the simple comprehension tasks there is no significant difference between the number of incorrect answers when using basic tool (Mdn=0.5) and Visually-enhanced tool (Mdn=0), Z = 0, p = 1.00. Moreover, for the complex comprehension tasks, no significant difference between the number of incorrect answers when using basic tool (Mdn=1) and visually-enhanced tool (Mdn=0) was found, Z = 14.50, p = 0.40. Therefore, we cannot reject null-hypothesis $H_0$2, which means there is *no significant difference between the number of incorrect answers based on the simple and complex comprehension tasks* for the basic tool and Visually-enhanced tool.

6.2.2 Basic tool vs. Visually-enhanced tool based on Changeability Tasks

We also formalized two hypotheses for the changeability tasks (i.e. H_3 and H_4) based on our dependent variables (i.e. completion time and number of incorrect answers). The null hypothesis for H3 is formalized as a follow:

- ***$H_0$3**: There is no significant difference between the completion time of the changeability tasks done by software developers by using basic tool and visually-enhanced tool (including the visualization and interaction interventions).*

Results–First, we report the results for the complex changeability task. As the distribution of the collected data for completion time was normal, the paired t-test *for the complex changeability tasks* revealed that there is a *significant difference in the mean values of the completion time* for basic tool (M =257.40, SD= 84.20), and Visually-enhanced tool (M =177.50, SD= 59.10), t (19) = 5.25, p < 0.0001. The effect size is equal to 0.78 which shows applying visualization and interaction interventions has a medium effect in reducing time required for changing complex feature models. Based on the t-test for paired samples *for the simple changing tasks*, there is a *significant difference in the mean values of the completion time* for basic tool (M =190.50, SD= 67.64), and Visually-enhanced tool (M =150.00, SD= 52.46), t (19) = 2.79, p = 0.01. Hence, we can reject null-hypothesis $H_0$3. The effect size is equal to 0.32 which shows applying visualization and interaction interventions has a low effect in reducing time required for changing simple feature models.

The second hypothesis (H4) for the changing task compares the accuracy of software developers when applying changes to a feature model by using the basic and visually-enhanced tools. The corresponding null hypothesis is as a follow:

- ***$H_0$4**: There is no significant difference between the accuracy (i.e. fewer errors) of the changing tasks done by software developers by using basic tool (basic implementation) and visually-enhanced tool (including the visualization and interaction interventions).*

Results–The Wilcoxon Sign-Rank test on the number of incorrect answers for the simple changeability task revealed that there is *no significant difference in the number of the incorrect answers* when using basic tool (Mdn=0) and Visually-enhanced tool (Mdn=0) Z = 6.50, p = 0.45. In addition, for *the complex changing tasks* when using basic tool (Mdn=1.5) and visually-enhanced tool (Mdn=0), there is *no significant difference in the number of incorrect answers*, Z = 24.00, p = 0.06. Therefore, we cannot reject null hypothesis $H_0$4 for the simple and complex feature model which means that *for changing tasks, the use of basic tool or visually-enhanced tool does not have an impact on the number of incorrect answer.*

6.2.3 Perceived Ease-of-Use and Ease-of-Learning

In addition to investigating the effect of the visualization and interaction interventions on performing the comprehension and changing tasks of feature model configuration, we collected the participants' feedback about the ease-of-use and ease-of-learning of both the tools. By analyzing two formulized hypotheses H5 and H6, the difference between visually-enhanced tool and basic tool in terms of ease-of-use and ease-of-learning can be revealed. The corresponding null hypothesis for H5 (ease of use) is as follow:

- ***$H_0$5:** Visualization and interaction interventions do not make visually-enhanced tool significantly easier to use than basic tool when performing maintainability tasks (i.e. changeability and comprehension).*

Results–By performing the Wilcoxon Sign-Rank test on the participants' responses about ease of use, the following results were obtained:

- The visually-enhanced tool (Mdn=5) is *perceived significantly more user friendly* than the basic tool (Mdn=3.5), Z =56.50, p = 0.0004. The effect size is equal to 0.59 which means applying visualization and interaction interventions shows a medium effect on increasing perceived user-friendliness of the tool by participants.
- The visually-enhanced tool (Mdn=4.5) is *perceived to need significantly less efforts to use* than effort required for using the basic tool (Mdn=2), Z = 75.50, p = 0.0001. The effect size is equal to 0.67 which means applying visualization and interaction interventions has a medium effect on reducing perceived effort to use of the tool by participants.
- The interaction with the visually-enhanced tool (Mdn=4) is *perceived to be significantly clearer and understandable* than the interaction with the basic tool (Mdn=5), Z =23.50, p = 0.02. The effect size is equal to 0.28 which means applying visualization and interaction interventions has a low effect on increasing perceived clearness and understandability of the tool by participants.
- The Visually-enhanced tool (Mdn=4) is *perceived to be significantly easier to use* than the basic tool (Mdn=5), Z =60.00, p = 0.0001. The effect size is equal to 0.62 which means applying visualization and interaction interventions has a medium effect on perceived ease-of-use of the tool by participants.

Similarly, the null hypothesis for H6 (ease-of-learning) is defined as a follow:

- ***$H_0$6:** Visualization and interaction interventions do not make visually-enhanced tool significantly easier to learn than basic tool when performing maintainability tasks (i.e. changeability and comprehension).*

Results – For the ease of learning questions, we performed the Wilcoxon Sign-Rank test and the following results were obtained:

- There is no significant difference in the perceived amount of time needed to learn either visually-enhanced tool (Mdn=5) or basic tool (Mdn=5), Z =18.00, p = 0.13.
- The Visually-enhanced tool (Mdn=4.5) is *perceived to be significantly easier to remember* how to use than the basic tool (Mdn=5), Z = 18.00, p = 0.03. The effect size is equal to 0.35 which means applying visualization and interaction interventions has a low effect on increasing perceived easer to remember of the tool by participants.
- The Visually-enhanced tool (Mdn=4.5) is *perceived to be significantly easier to learn to operate* than the basic tool (Mdn=5), Z =22.00, p = 0.03. The effect size is equal to 0.37 which means applying visualization and interaction interventions has a low effect on increasing perceived easer to learn of the tool by participants.
- The perceived amount of time needed to be a skillful user is significantly better in visually-enhanced tool (Mdn=4) than basic tool (Mdn=5), Z = 2.08) =28.00, p = 0.01. The effect size is equal to 0.33 which means applying visualization and interaction interventions has a low effect on decreasing perceived time needed to be a skillfule of the tool by participants.

7. Discussion of the Results and Related Work

7.1 Discussion of the Results

The results of the data analysis applied are affirmative for our hypotheses and provide some evidence that the proposed approach (applying visualization and interaction interventions) improves software developers' performance while conducting comprehension and changing tasks. Based on the results, the required time for the comprehension of simple and complex feature models is significantly shorter when using visually-enhanced tool (implementation of fmp with visualization and interaction interventions) than basic tool. The lower time requirements for the comprehension tasks in the visually-enhanced tool can be attributed to several factors. First of all, visually-enhanced tool unlike basic tool provides a single representation of both features and their associated non-functional properties in which non-functional properties of each feature were shown along with the feature. According to cognitive science theories [81][82], effective visualization and interaction techniques during improve the cognition of developers which leads to better performance and higher accuracy. Hence, the selected visualization and interaction interventions improve the cognition of developers when configuring feature models. Moreover, zooming and highlighting properties facilitate focus on specific parts of feature models. This likely has a promising potential for a practical configuration of feature models in industrial settings where software developers deal with large feature models with a high number of features. Therefore, applying visualization techniques have a potential to improve the comprehension of feature models along with their associated non-functional properties.

Moreover, based on the results obtained by using visually-enhanced tool, the completion time for the changing tasks of the feature model configuration is significantly reduced. The facilities, that visually-enhanced tool provides for navigation in the visual view such as detail on demand, zooming, and highlighting may be factors to significant decrease of the time of the changing tasks in visually-enhanced tool in contrast to basic tool where the user does not have these facilities. Hence, the application of the visualization and interaction techniques has a potential to improve time needed for the comprehension and application of changes to feature models. The configuration process includes the comprehension and changing tasks, so decreasing the completion time of these tasks means decreasing the completion time of the configuration process. The results motivate us to apply more visualization and interaction techniques suitable for feature model configuration for further improvement. The results can be beneficial for other researchers in SPLE to apply similar techniques in their tools.

With respect to the participants' feedback on the visually-enhanced tool and basic tool, the results showed that visually-enhanced tool is perceived to be easier to use (i.e., the participants on average ranked 4.64 out of 5 for visually-enhanced tool and 3.31 out of 5 for basic tool). Based on the design of our experiment, there is some evidence that this difference can be attributed to the use of visualization and interaction interventions. This makes a feature model more intuitive and integrates non-functional properties and features in one representation (see Figure 3). However, both the tools were perceived to be easy to learn (i.e., the participants on average ranked 4.78 and 4.30 out of 5 for visually-enhanced tool and basic tool, respectively). This can likely be attributed to the simplicity of learning the feature model structure.

7.2 Related Works

In addition to the tools mentioned in section 3.4, we also studied another group of related works that concentrate on representation of features models. Pablo Trinidad et al. [27] proposed a visualization approach, which visualizes large feature models by using cone trees – a three-dimensional visualization technique to represent hierarchical information. Their work did not cover feature model configuration and it only aimed at presenting and understanding large size feature models. Daren Nestor et al. [41] also employed visualization techniques to support feature model management and product derivation in SPLE. They aimed at improving the tasks such as retrieving a set of features and their attributes, retrieving architectural structures of a feature set and comparing the architecture design to the implementation by using different visualization techniques such as tree-map, graph with a force-directed layout, TreeJuxtaposer, and semantic zoom. David Sellier et al. [42] developed a tool named V-Visualise which visualize the decision model and interdependency between requirements decisions. They tried to provide understandable views of interdependencies between product lines for software developers. The toolset which includes V-visualise have a product derivation tool called V-Resolve and V-Define.

Finally, we broaden the scope of related work to those which focused on the visualization of other software models. The following works are related to our work in that they tried to improve the comprehension of software model by visualizing software model and related attributes of elements in the models. Lange and Chaudron [48] have been working on the improvement of comprehension of UML diagrams and proposed visuals which aim at visualizing the inter-diagram relations (*Meta View*), visualizing the context of a model element—consists of all other related model elements to that element — (*Context View*), and visualizing metrics of each elements on top of the diagram (*Metric View*). They also evaluated their work on comprehension tasks and the results showed that their tool (i.e. the implementation of above views) is successful to reduce the completion time by 20% and increase the correctness by 4.5%. Byelas and Telea [47] have also been working on visualizing software metrics a top of UML diagram. They proposed a solution to visualize Areas Of Interest (AOIs) — as a group of elements in a UML diagram which share common software metrics — on the UML diagram. Similar to our approach, their suggested that their approach helps software developers associate non-functional properties with diagrams and understand models better. They also performed a controlled experiment to show the level of understandability and quality of drawn AOIs. The result of the evaluation revealed that the drawn AOIs using their algorithm are close to good human drawn AOIs. Another approach was introduced by Langelier et al. [62] that supports complex software analysis by employing visualization. They proposed a visualization framework to assist software developers understand the quality of software effectively. Their evaluation of the visualization framework had promising results. In the evaluation, the participants perform a complex analysis task correctly in less than one minute— the tasks need significantly more time to complete without using the visualization framework.

8. Threats to Validity

8.1 Internal validity

For internal validity of our study, we investigate if and to what extent some confounding factors affect our results. Possible confounding factors for our study were as following:

- *Motivation of participants:* Since all the participants, who responded to our invitation, voluntarily participated in the experiment and for that received no material or immaterial compensation, we can exclude the motivation as a confounding factor. Moreover, none of our participants left the experiments.
- *Learning effect in the tasks:* we controlled this threat by creating a demo and introduced both the tools to the participants. The participants watched the demo and asked any questions they have about different parts of the tool. Moreover, we designed the preparation tasks similar to the main tasks with different feature models (in the domain of ATM). All the participants had to perform the preparation tasks before starting with the main tasks.
- *Complexity of feature model:* The other factor which could have affected the internal validity of our experiment is the domain complexity of feature models (i.e. selecting feature models from domains which the participants were not familiar with). The domain of the used feature models was online shopping and all the participants were familiar with the online shopping systems. In addition, by designing tasks with the two level of complexity (i.e. feature models with different number of features and leaf features), we controlled the other type of feature model complexity (more details have been addressed in Section 5.2.1). Therefore, we filtered out this factor as a threat to validity of our study.
- *Carry-over effect:* in a repeated measure experiment this factor occurs when a treatment is effected by previous one. In our experiment, we handled this threat by designing different tasks for each tool. Moreover, we used balance order for performing the task related to each tool.
- *Fatigue effects:* The total time of experiment for each participant was about one hour on average in which the time of performing the tasks was 30 minutes on average (except the time of watching the demo and the preparation task). This time is not too much in contrast with the regular lecture time (considering that the participants are computer science students) which is about 90 minutes. So, we can ignore the effect of fatigue on the result of study because it is not significant.
- *Possible information exchange among the subjects:* To avoid the effect of this threat we asked all

participant to not share the information about tasks with others

8.2 External Validity

The external validity explores whether and to what extent the results of a study can be generalized. In our case, we should investigate whether we can generalize the results and applied them to the industrial context of feature model configuration. The main confounding factor is population [28]. Our population is comprised of master's and PhD students, and a postdoctoral fellow who had experience in software engineering and SPLE. The argument that the participants may not be representative of the real target users (professional software developers) and their performance on the tools is probably not that relevant. However, based on the questionnaire, which investigated their familiarity with modeling languages such as UML, ERD, and feature modeling and based on their experience, they are not far from at least junior industrial software developers. Furthermore, Basili et al. [34] believe that in many cases, the use of students in experiments has little impact on the validity of the results. Since there are some differences between industry and academic contexts, a field experiment should be conducted with the industry subjects to validate the reported results.

The other confounding factor is the type of feature models used in the tasks. One can say that the feature models in the tasks are not representative of industry ones. Although the size of feature models used in the experiment is smaller than industry versions of feature models, all of them are subsets of a large feature model, which is designed for the electronic shopping and which was downloaded from the SPLOT repository[c]. Since we have several tasks (i.e., eight tasks) in the experiment and the participants were supposed to complete all the tasks for a given amount of time (to control the fatigue effect, which could affect the internal validity of our findings, as discussed in Section 8.1), we could not select larger size feature models.

8.3 Construct Validity

Construct validity explores whether the independent and dependent variables provide accurate measurements of what they are intended to measure [6]. In our case, according to our hypotheses, we were interested to investigate the effects of the visualization and interaction interventions on comprehension and changing of a feature model configuration and aimed at generalizing the effects to visualization techniques for feature model configuration in general (see section 7.1). However, we should mention that the experiment does not distinguish between the visualization and interaction implemented in visually-enhanced tool and visualization techniques for feature model configuration. That is, we cannot tell whether the improvement in the completion time and the number of correct answers is because of the investigated visualization and interaction interventions for feature model configuration in general or specific techniques employed in visually-enhanced tool (e.g., 2-dimentional tree). To discriminate the difference, more similar experiments involving other visualization techniques (e.g. 3D representation) are required.

Furthermore, the completion time and the number of correct answers were used as measures in our hypotheses. The completion time is measured by the amount of time that the participants spent on performing comprehension and changing tasks. This is a straightforward and direct measurement mechanism, and hence, poses the least threats to construct validity. With regard to measuring the number of correct answers in the comprehension tasks, the aim of our study was to analyze whether visually-enhanced tool provides a better support than basic tool in the comprehension of the feature model configuration process. Therefore, we decided to use a multi-choice questionnaire where each question admitted one or more correct answers. In particular, each question investigates a comprehension of one feature model where each of its options investigates understandability of a part of the feature model or the whole feature model. With respect to the changing task, the changing questions were designed such that requested changes lead to one correct configuration. We measured the number of incorrect answers by

[c]http://www.splot-research.org

comparing the final answer (i.e. configured feature model) of participants with the correct configuration that we designed in advance. These measurement mechanisms satisfy objective of our study and poses the least threats to construct validity.

Regarding usability variables, the subjective opinions of participants were collected. The main threat of applying a subjective measurement is different attitudes of different participants towards the evaluation of dependent variables. For example, some participants may easily provide high subjective values to the options (usability of basic tool or usability of visually-enhanced tool) that they are evaluating, while others may be reluctant to provide high subjective values. However, since we used repeated measure design in which one participant provides subjective values for both tools, this subjective measurement *exactly* captures what it claims to capture, which is the difference between usability of basic and visually-enhanced tools from users' point of view. Moreover, standard models for measuring ease-of-use and ease-of-learning such as the technology acceptance model [65][66][67] and system usability scale [68] are based on perceived values.

Another potential threat to construct validity is the use of a 5-point Likert scale to collect the subjective opinions of the participants. The problem of Likert scales is that it provides a limited number of ordinal options for evaluation and it may prevent participants from accurately expressing their opinion. Empirical studies, however, have shown that the best number of options for a Likert scale is between 4 and 7 [64]. According to the study [64], in many cases more than 7 options are likely to exceed the discriminative capacity of participants, even though it leads to better psychometric properties. By considering the 5 point Likert scale, we tried to minimize the threats to validity of this measurement mechanism.

9. Conclusion and Future work

One of the main challenges in software product line engineering is the use of variability models to configure products in the application engineering. A feature model represents feasible features and properties of a domain in hierarchical representation. In application engineering, a final product is created by selecting or deselecting of the features based on the target requirements. Configuring feature models is a challenging task because many factors should be taken into account and high cognitive abilities are required. We identified a set of visualization and interaction interventions which help software developers in the configuration tasks by using visual aids. The visualization interventions are employed to represent the feature model and non-functional properties in the feature model. Also, a number of interactivity interventions are embedded in the tool to assist software developers in their tasks on the feature model.

To understand the effects of visualization and interaction interventions on the feature model configuration process, we performed a controlled experiment on the developed tool with respect to common activities in configuration, i.e., changing and comprehension. The results revealed that applying visualization and interaction techniques significantly decreases time required for changing and comprehension configuration tasks of simple and complex feature models. The participants' inputs showed that the ease-of-use and ease-of-learning are better for the visually-enhanced tool implementation (includes visualization and interaction interventions) than the basic tool, which uses a basic tree view without additional visualization and interaction techniques.

For future work, we aim at enhancing the visualization and interaction interventions based on the participants' feedback. In this experiment, we investigated the effects of the employed visualization and interaction interventions on software developers' performance. Hence, other experiments should be performed to compare the effects of different types of visualization and interaction interventions on software developers' performance. We also investigated these interventions collectively. Further interesting steps for experimentation could be include the investigation of each of these interventions individually and record their impact on software developers' performance. Moreover, we intended to add visualization and interaction interventions for mapping between feature models and other artifacts used in SPLE. Afterwards, the visualization and interaction interventions will be evaluated for the mapping representation. Finally, in our current study, we concentrated on the visualization and interaction intervention related to the visualization of hierarchy relations and non-functional properties in feature models. An important aspect in the feature model configuration is inter-dependencies between features and feature attributes. In our future works, we aim at identifying a set of visualization and interaction interventions for representing the inter-dependencies between feature selections, between feature attribute values of two different features, between one feature and another feature's attribute value, and between groups of selection decisions. Next, we will design an empirical study to investigate the comprehension and changeability of these relations by software developers during the feature model configuration process.

Acknowledgement

We wish like to thank Dr. Chris Shaw and Dr. Bernhard E. Riecke for their valuable comments on the first draft of the paper. We also would like to thank all participants for taking part in the experiment.

References

[1] F. Linden, K. Schmid, and E. Rommes, “Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering,” Springer, Berlin, Heidelberg, Paris, 2007.

[2] K. Kang, J. Lee, P. Donohoe, “Feature-oriented product line engineering,” IEEE software 19 (2002) 58-65.

[3] M. Asadi, E. Bagheri, D. Gašević, M. Hatala, “Goal-driven software product line engineering,” In Proceedings of the 26th ACM Symposium on Applied Computing, TaiChung, Taiwan, 2011 (in press).

[4] E. Bagheri, M. Asadi, D. Gašević, S. Soltani, “Stratified Analytic Hierarchy Process: Prioritization and Selection of Software Features,” In Proceedings of the 14th International Software Product Lines Conference, Jeju Island, South Korea, 2010 (Lecture Notes in Computer Science Vol. 6287), pp. 300-315.

[5] C. Cawley, G. Botterweck, P. Healy, S. B. Abid, and S. Thiel, “A 3D Visualisation to Enhance Cognition in Software Product Line Engineering,” in Advances in Visual Computing, vol. 5876, G. Bebis et al., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 857-868.

[6] E. Bagheri, and D. Gasevic, “Assessing the Maintainability of Software Product Line Feature Models using Structural Metrics,” Software Quality Journal (2010), Springer, To appear 2010.

[7] David Benavides, Sergio Segura, and Antonio Ruiz-Cort\&\#233;s. 2010. Automated analysis of feature models 20 years later: A literature review. Inf. Syst. 35, 6 (September 2010), 615-636.

[8] B. Shneiderman. 2007. Creativity support tools: accelerating discovery and innovation. Commun. ACM 50, 12 (December 2007), 20-32.

[9] K. Czarnecki, and U. Eisenecker, “Generative Programming: Methods, Tools, and Applications,” Addison-Wesley, 2000.

[10] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich. 2014. FeatureIDE: An extensible framework for feature-oriented software development. Sci. Comput. Program. 79 (January 2014), 70-85.

[11] J. Siegmund, C. Kästner, S. Apel, C. Parnin, A. Bethmann, T. Leich, G. Saake, and A. Brechmann. Understanding Understanding Source Code with Functional Magnetic Resonance Imaging. In Proceedings of the 36th International Conference on Software Engineering (ICSE), June 2014.

[12] S. Apel and D. Beyer. 2011. Feature cohesion in software product lines: an exploratory study. In Proceedings of the 33rd International Conference on Software Engineering (ICSE '11). ACM, New York, NY, USA, 421-430.

[13] D. Dhungana, P. Grünbacher, and R. Rabiser. 2011. The DOPLER meta-tool for decision-oriented variability modeling: a multiple case study. Automated Software Engg. 18, 1 (March 2011), 77-114.

[14] G. Botterweck, S. Thiel, D. Nestor, S. Abid, and C. Cawley, "Visual Tool Support for Configuring and Understanding Software Product Lines," In Proceedings of the 2008 12th International Software Product Line Conference (SPLC '08). IEEE Computer Society, Washington, DC, USA,

[15] D. Moody. 2009. The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Trans. Softw. Eng. 35, 6 (November 2009), 756-779.

[16] H. Purchase, "Which aesthetic has the greatest effect on human understanding?," in Graph Drawing, vol. 1353, G. DiBattista, Ed. Berlin/Heidelberg: Springer-Verlag, pp. 248-261.

[17] D. Teyseyre, M. Campo, "An overview of 3d software visualization," IEEE Trans. on Visualization and Computer Graphics, 2008. DOI: 10.1109/TVCG.2008.86.

[18] K. P. Herndon, A. van Dam, and M. Gleicher, "The challenges of 3D interaction: a CHI'94 workshop," ACM SIGCHI Bulletin, vol. 26, p. 36–43, Oct. 1994.

[19] D.A. Bowman, E. Kruijff, J.J. Laviola Jr., and I. Poupyrev, "An Introduction to 3D User Interface Design," Presence, vol. 10, no. 1, pp. 96-108, 2001.

[20] M. C. Chuah, S. F. Roth, J. Mattis, and J. Kolojejchick, "SDM: selective dynamic manipulation of visualizations," in Proceedings of the 8th annual ACM symposium on User interface and software technology, New York, NY, USA, 1995, p. 61–70.

[21] prefuse | interactive information visualization toolkit, http://prefuse.org/, Accessed in November 2010.

[22] C. Brewer, "Color use guidelines for data representation," In Proceedings of the Section on Statistical Graphics, 1999.

[23] M. Stone, "Choosing Colors for Data Visualization," Dostupné na internete: http://www. perceptualedge. com/articles/b-eye/choosing_colors. Pdf.

[24] Pure-systems GmbH: Variant Management with pure:variants (2003-2004), http://www.pure-systems.com

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

[26] K. Czarnecki and C. H. P. Kim, "Cardinality-based feature modeling and constraints: A progress report," in International Workshop on Software Factories, 2005.

[27] P. Trinidad, A. Ruiz-Cortes, D. Benavides, and S. Segura. "Three-dimensional feature diagrams visualization," In ViSPLE, pages 295{302. Lero, 2008.

[28] D. N., Chin, "Empirical Evaluation of User Models and User-Adapted Systems". User Modeling and User-Adapted Interaction 11, 1-2 (Mar. 2001), 181-194, 2001.

[29] B. Johnson and B. Shneiderman, "Tree-Maps: a space-filling approach to the visualization of hierarchical information structures," in Proceedings of the 2nd conference on Visualization '91, pp. 284–291, 1991.

[30] J. Stasko and E. Zhang, "Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations," in Information Visualization, 2000. InfoVis 2000. IEEE Symposium on, pp. 57-65, 2000.

[31] G. G. Robertson, J. D. Mackinlay, and S. K. Card, "Cone Trees: animated 3D visualizations of hierarchical information," in Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology, pp. 189–194, 1991.

[32] C. Plaisant, J. Grosjean, and B. Bederson, “SpaceTree: supporting exploration in large node link tree, design evolution and empirical evaluation,” in Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on, pp. 57-64, 2002.

[33] B. Shneiderman, “The eyes have it: a task by data type taxonomy for information visualizations,” in Visual Languages, 1996. Proceedings., IEEE Symposium on, pp. 336-343, 1996.

[34] V. R. Basili, F. Shull, and F. Lanubile, “Building knowledge through families of experiments,” Software Engineering, IEEE Transactions on, vol. 25, no. 4, p. 456–473, 1999.

[35] R. Fernández, M. A. Laguna, J. Requejo, N. Serrano, “Development of a Feature Modeling Tool using Microsoft DSL Tools” GIRO Technical Report 2009-1.ver 1.0, Department of Computer Science, University of Valladolid

[36] F. Heidenreich, J. Kopcsek, and C. Wende, “FeatureMapper: mapping features to models,” in Companion of the 30th international conference on Software engineering, pp. 943–944, 2008.

[37] F. Heidenreich, I. Savga, and C. Wende, “On controlled visualisations in software product line engineering,” in Proceedings of the 2nd International Workshop on Visualisation in Software Product Line Engineering (ViSPLE 2008), collocated with the 12th International Software Product Line Conference (SPLC 2008), 2008.

[38] XFeature:Feature Modelling Tool http://www.pnp-software.com/XFeature/Home.html

[39] FaMa-FW, http://www.isa.us.es/fama/

[40] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés, “FAMA: Tooling a framework for the automated analysis of feature models,” in Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems, 2007

[41] D. Nestor, L. O'Malley, P. Healy, A. Quigley, and S. Thiel, “Visualisation techniques to support derivation tasks in software product line development,” in Proceedings of the 2007 conference of the center for advanced studies on Collaborative research, pp. 315–325, 2007.

[42] D. Sellier and M. Mannion, “Visualising Product Line Requirement Selection Decision Inter-dependencies,” in Proceedings of the Second International Workshop on Requirements Engineering Visualization, pp. 7–, 2007.

[43] D. Batory, “Feature models, grammars, and propositional formulas,” Software Product Lines, p. 7–20, 2005.

[44] Bosch, J., Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach. ACM Press / Addison-Wesley, 2000.

[45] S. Apel and C. Kästner. An Overview of Feature-Oriented Software Development. Journal of Object Technology (JOT), 8(5):49–84, July 2009.

[46] D. Benavides, P. Trinidad, and A. Ruiz-Cortés, “Automated reasoning on feature models,” in Advanced Information Systems Engineering, 2005, p. 491–503.

[47] H. Byelas and A. Telea, “Towards realism in drawing areas of interest on architecture diagrams,” Journal of Visual Languages and Computing, vol. 20, pp. 110–128, Apr. 2009.

[48] C. F. J. Lange and M. R. V. Chaudron, “Interactive Views to Improve the Comprehension of UML Models - An Experimental Validation,” in Proceedings of the 15th IEEE International Conference on Program Comprehension, pp. 221–230, 2007.

[49] I. Sommerville, P. Sawyer, “Viewpoints: Principles, Problems and a Practical Approach to Requirements Engineering,” Ann. Software Eng. 3: 101-130 (1997).

[50] L. C. Briand, J. Wüst, S. V. Ikonomovski, and H. Lounis, "Investigating quality factors in object-oriented designs: an industrial case study," in Proceedings of the 21st international conference on Software engineering, New York, NY, USA, 1999, p. 345–354.

[51] A. Minke, "Conducting Repeated Measures Analyses: Experimental Design Considerations," Annual Meeting of the Southwest Educational Research Association, Austin, (1997).

[52] ISO/IEC Standard No. 14598: Information technology – Software product evaluation; Parts 1–6. International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), Geneva, Switzerland, 1999-2001.

[53] ISO/IEC Standard No. 9126: Software engineering – Product quality; Parts 1–4. International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), Geneva, Switzerland, 2001-2004.

[54] F. Losavio, "Quality Models to Design Software Architecture.," The Journal of Object Technology, vol. 1, no. 4, p. 165, 2002.

[55] B. B. Chua and L. E. Dyson, "Applying the ISO 9126 model to the evaluation of an elearning."In R. Atkinson, C. McBeath, D. Jonas-Dwyer & R. Phillips (Eds), Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference (pp. 184-190).

[56] B. Behkamal, M. Kahani, and M. K. Akbari, "Customizing ISO 9126 quality model for evaluation of B2B applications," Information and Software Technology, vol. 51, no. 3, pp. 599-609, Mar. 2009.

[57] M. Genero, J. Olivas, M. Piattini, and F. Romero, "Using Metrics to Predict OO Information Systems Maintainability," in Proceedings of the 13th International Conference on Advanced Information Systems Engineering, London, UK, UK, 2001, p. 388–401.

[58] T. Korson and J. D. McGregor, "Understanding object-oriented: a unifying paradigm," Communications of the ACM, vol. 33, p. 40–60, Sep. 1990.

[59] S. Easterbrook, J. Singer, M.-anneStorey, and D. Damian, "Selecting Empirical Methods for Software Engineering Research."

[60] B. A. Kitchenham et al., "Preliminary guidelines for empirical research in software engineering," IEEE Transactions on software engineering, p. 721–734, 2002.

[61] A. De Lucia, C. Gravino, R. Oliveto, and G. Tortora, "An experimental comparison of ER and UML class diagrams for data modelling," Empirical Software Engineering, vol. 15, p. 455–492, Oct. 2010.

[62] G. Langelier, H. Sahraoui, and P. Poulin, "Visualization-based analysis of quality for large-scale software systems," in Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering - ASE '05, Long Beach, CA, USA, 2005, p. 214.

[63] N. Fenton, "Software measurement: a necessary scientific basis," IEEE Transactions on Software Engineering, vol. 20, no. 3, pp. 199-206, Mar. 1994

[64] L. M. Lozano, E. Garc´ıa-Cueto, and J. Mu˜niz, "Effect of the number of response categories on the reliability and validity of rating scales," Methodology: European Journal of Research Methods for the Behavioral and Social Sciences, vol. 4, no. 2, pp. 73–79, 2008. [Online]. Available: http://dx.doi.org/10.1027/1614-2241.4.2.73

[65] Davis, F. D. (1989), "Perceived usefulness, perceived ease of use, and user acceptance of information technology", MIS Quarterly 13(3): 319–340

[66] Venkatesh, V.; Davis, F. D. (2000), "A theoretical extension of the technology acceptance model: Four longitudinal field studies", Management Science 46(2): 186–204

[67] Venkatesh, V.; Bala, H. (2008), "Technology Acceptance Model 3 and a Research Agenda on Interventions", Decision Sciences 39(2): 273–315.

[68] Bangor, A., Kortum, P., Miller, J. (2008). An empirical evaluation of the system usability scale. International Journal of Human-Computer Interaction, 24(6), 57 - 594.

[69] D. Benavides, P. Trinidad, and A. Ruiz-Cortés, "Automated Reasoning on Feature Models," *in Proc. 17th Int'l conf. Advanced Information Systems Engineering*, pp. 491–503., 2005.

[70] N. Siegmund, M. Rosenmüller, M. Kuhlemann, C. Kästner, S. Apel, and G. Saake, "SPL Conqueror: Toward optimization of non-functional properties in software product lines," *Software Quality Journal*, 2011.

[71] J. Guo, J. White, G. Wang, J. Li, and Y. Wang, "A genetic algorithm for optimized feature selection with resource constraints in software product lines," Journal of Systems and Software, vol. 84, no. 12, pp. 2208-2221, Dec. 2011.

[72] S., Soltani, M. Asadi, D. Gašević, M. Hatala, E. Bagheri, "Automated Planning for Feature Model Configuration based on Functional and Non-Functional Requirements," *In Proceedings of the 16th International Software Product Line Conference*, Salvador, Brazil, 2012

[73] M. Mendonca, A. Wasowski, K. Czarnecki, and D. Cowan, "Efficient compilation techniques for large scale feature models," in Proceedings of the 7th international conference on GPCE, 2008, p. 13–22.

[74] Victor R. Basili. 1992. Software Modeling and Measurement: The Goal/Question/Metric Paradigm. Technical Report. University of Maryland at College Park, College Park, MD, USA.

[75] B. Schneiderman, Creativity support tools: A grand challenge for HCI researchers. In Engineering the User Interface (M. Redondo, Ed.), Springer, (2009)

[76] A. Pleuss, R. Rabiser, and G. Botterweck. 2011. Visualization techniques for application in interactive product configuration. In Proceedings of the 15th International Software Product Line Conference, Volume 2 (SPLC '11), Ina Schaefer, Isabel John, and Klaus Schmid (Eds.). ACM, New York, NY, USA, , Article 22 , 8 pages.

[77] V. Friedman. Data visualization: Modern approaches.Website, 2007.

[78] Botterweck, G., M. Janota, and D. Schneeweiss, A Design of a Configurable Feature Model Configurator, Proceedings of the 3rd International Workshop on Variability Modelling of Software-Intensive Systems (VAMOS 09), 2009.

[79] M. El Dammagh and O. De Troyer. 2011. Feature modeling tools: evaluation and lessons learned. In Proceedings of the 30th international conference on Advances in conceptual modeling: recent developments and new directions (ER'11), Olga De Troyer, Claudia Bauzer Medeiros, Roland Billen, Pierre Hallot, and Alkis Simitsis (Eds.). Springer-Verlag, Berlin, Heidelberg, 120-129.

[80] Lisboa, L.B., Garcia, V.C., Almeida, E.S., Meira, S.L., Lucrédio, D., Fortes, R.P.: A Systematic Review on Domain Analysis Tools. Information and Software Technology 52, 1–13 (2010)

[81] S. K. Card, J. D. Mackinlay, and B. Shneiderman. Readings in informa-tion visualization: using vision to think, chapter 1, pages 1–34. Morgan Kaufmann Publishers Inc, 25 January 1999.

[82] Zhicheng Liu, Nancy Nersessian, and John Stasko. 2008. Distributed Cognition as a Theoretical Framework for Information Visualization. IEEE Transactions on Visualization and Computer Graphics 14, 6 (November 2008), 1173-1180.

[83] Paas, F., Renkel, A., & Sweller, J. (2004). "Cognitive Load Theory: Instructional Implications of the Interaction between Information Structures and Cognitive Architecture". Instructional Science 32: 1–8

[84] Sweller, J. "Cognitive load during problem solving: Effects on learning." Cognitive science 12 (2), 1988: 257-285.

[85] J. Cohen, 1992. "A Power Primer." Psychological Bulletin, 112, 155– 15

[86] Few, S. (2008, February). Practical Rules for Using Color in Charts. Visual Business Intelligence Newsletter, Perceptual Edge. Retrieved from http://www.perceptualedge.com/articles/visual_business_intelligence/rules_for_using_color.pdf

[87] Daniel A. Keim, Florian Mansmann, Jörn Schneidewind, Jim Thomas, and Hartmut Ziegler. 2008. Visual Analytics: Scope and Challenges. In Visual Data Mining, Simeon J. Simoff, Michael H. Böhlen, and Arturas Mazeika (Eds.). Lecture Notes In Computer Science, Vol. 4404. Springer-Verlag, Berlin, Heidelberg 76-90.

[88] Reddivari, S., "Visual analytics for software requirements engineering," Requirements Engineering Conference (RE), 2013 21st IEEE International , vol., no., pp.389,392, 15-19 July 2013