Visual Guidance for Product Line Configuration Using Recommendations and Non-Functional Properties

Juliana Alves Pereira¹, Jabier Martinez², Hari Kumar Gurudu¹, Sebastian Krieter¹, Gunter Saake¹
¹University of Magdeburg, Germany;²Sorbonne University, UPMC Paris 6, France
{juliana.alves-pereira, hari.gurudu, sebastian.krieter, gunter.saake}@ovgu.de; jabier.martinez@lip6.fr

ABSTRACT
Software Product Lines (SPLs) are a mature approach for the derivation of a family of products using systematic reuse. Different combinations of predefined features enable tailoring the product to fit the needs of each customer. These needs are related to functional properties of the system (optional features) as well as non-functional properties (e.g., performance or cost of the final product). In industrial scenarios, the configuration process of a final product is complex and the tool support is usually limited to check functional properties interdependencies. In addition, the importance of non-functional properties as relevant drivers during configuration has been overlooked. Thus, there is a lack of holistic paradigms integrating recommendation systems and visualizations that can help the decision makers. In this paper, we propose and evaluate an interrelated set of visualizations for the configuration process filling these gaps. We integrate them as part of the FeatureIDE tool and we evaluate its effectiveness, scalability, and performance.

CCS CONCEPTS
• Software and its engineering → Software product lines; Extra-functional properties; Software functional properties; • Human-centered computing → Visualization;

KEYWORDS
Software Product Lines, Feature Model, Visualization, Configuration, Recommendation Systems

ACM Reference Format:

1 INTRODUCTION
A Software Product Line (SPL) is a set of common and optional features that are developed to satisfy the specific needs of a particular market segment [26]. A selection of optional features determines a product configuration which is used to derive a product with certain functional and non-functional properties (NFPs). Both functional and NFPs are identified after a thorough domain analysis. As running example, we present in Section 2 the smart-home Product Line (PL) where there are several functional features (e.g., an alarm can be silent, visual or siren) and NFPs (e.g., the cost of the features).

In SPL Engineering (SPE), the product configuration process aims to define, for each customer, a valid and complete configuration satisfying its requirements (both functional and non-functional). This interactive process is not trivial [16] and it is problematic mainly for the following reasons:
• In real-world SPLs, the number of features can be numerous with several types of variability relations, constraints among features, and NFPs. According to an industrial survey [8], the comprehension and visualization of models capturing the features of the SPL are reported as an issue for around 60% of the participants. Another survey on variability management tool support [6] shows that 17% of the tools explicitly discuss having limitations related to visualization.
• The constraints that can exist between the features (e.g., requires or excludes relations) can be numerous as well. Although tool support can trigger feature constraints without user involvement based on logical rules, the user may easily lose control of the consequences of the selections. The number of features and constraints impact the interactive configuration process both in the time that it takes and in the quality of the results.
• The selection of functional features can determine a valid configuration responding perfectly to the functional needs of the product. However, it might have non desired NFPs (e.g., slow performance or too expensive). Although automatic approaches for obtaining configurations that optimize NFPs exist [27], they focus on techniques to derive configurations in a single step, not allowing users to interact with the configuration process.

There are many SPL tools that aim to provide configuration support [6, 29] and several recommendation heuristics have been proposed to guide the order of feature selection [21]. However, no visual support is provided to decision makers to focus on likely relevant configuration options including recommendations. Thus, we address this challenge by providing the following contributions:
• A set of interrelated visualizations: 1) 5-star view to suggest feature selections. 2) Feature’s graph to visualize features constraints and the positive or negative impact of NFPs. 3) NFP’s graph to visualize the impact of an NFP over features and others NFPs.
• The implementation is publicly available as an extension of the SPL tool FeatureIDE [40].
• An empirical evaluation of the approach in terms of effectiveness, scalability and performance on eleven SPLs.

This paper is structured as follows: Section 2 presents relevant background information. Section 3 presents related work. Section 4
describes our proposed approach. Section 5 presents the evaluation. Finally, Section 6 concludes the paper and outlines future work.

2 BACKGROUND

In this section, we formally define the terms extended feature model, product configuration and the focused view.

Extended Feature Model. A common representation model for a Feature Model (FM) is a feature diagram. As an example, consider the smart-home feature diagram in Figure 1 where boxes denote features, and links illustrate the interdependencies between them. There are common features found in all products of the PL, known as mandatory features, such as illumination, and variable features that allow the distinction among products in the PL, referred to as optional and alternative features, such as security and the group sensor, respectively. In addition, FMs often contain cross-tree constraints (CTCs) defining feature interdependencies and restricting feature combinations (e.g., sensor → alarm). Furthermore, dashed boxes extend FMs with extra information about features (i.e., NFPs). This type of models where NFPs are included is called Extended Feature Model (EFM) [7].

Figure 1: An EFM for a smart-home PL (adapted from [10]).

Product Configuration. Product configuration refers to the process of making consecutive decisions to (de)select a set of variable features from an FM. Configuration tools usually represent the FM structure in an outlined tree hierarchy form [6, 29, 30] and the features with check boxes, where decision makers check manually the (un)required features to configure a product. These decisions should cover the product requirements and comply with the FM constraints. A configuration can be classified as partial or complete, as well as valid or invalid. A configuration is complete if each feature is (de)selected, while a configuration is partial if it requires additional (de)selection. In addition, a valid configuration must not violate the FM constraints, otherwise it is invalid. Finally, there are two ways to configure a product: from scratch and from base. A configuration process from scratch starts with no feature (de)selected (except mandatory features), while a configuration from base starts with a partial configuration.

The Focused View. According to Pereira et al. [30], the current configuration tools are still not able to efficiently guide the interactive configuration process of large-scale SPLs due to the high information workload and the complexity of decision making. To overcome this challenge, Pereira et al. [31] proposed a focused view over the configuration. The focused view reduces significantly the users’ information density in each configuration step by showing a restricted view of a partial configuration. It focuses on the relevant configuration space (i.e., part of the configuration that is currently (de)selected). The essential idea of this mechanism is that a feature tree hierarchy represents the features’ degrees of abstraction. The higher a feature is located in the tree, the higher its level of abstraction is. In contrast, leaf features (i.e., features without any children) are the most detailed and technical features. Consequently, a user should first decide among the most abstract features before going into detail. To support this intuitive process, Pereira et al. implement a level-wise view on the feature tree. This view automatically expands and shows only the direct sub-features from selected features. With the focus on direct sub-features, the user is able to focus on one particular choice at each-time.

3 RELATED WORK

In this paper, we focus on visualizations to guide the configuration process [25] and we acknowledge several related works in this field.

Czarnecki et al. [13], as part of their work in probabilistic FMs, proposed a visualization for recommending features during the interactive configuration of a product based on existing configurations. The visualization represents a score associated to each feature. Regarding scores, Pereira et al. [33] studied different algorithms to be used as scores for recommender systems in SPL configuration. They tuned the parameters of these algorithms automatically using empirical data and showed that BRISMF (Biased Regularized Incremental Simultaneous Matrix Factorization) outperformed the others. We took this algorithm and we extended the visualization aspects which was very limited in both Czarnecki et al. and Pereira et al. works. In addition, both works did not consider NFPs as part of the configuration process.

Martinez et al. [20] presented FROGs (Feature Relations Graphs), a visualization paradigm based on radial ego networks that inspired two of our visualizations. Their work uses a simple algorithm for calculating the score based on the co-occurrence of the features, they did not consider NFPs and we extended it to show other relevant information. In addition, several works on visualization during configuration suggest the importance of reducing the information density and using techniques to bring the attention to the user to the relevant features (e.g., focused view in Pereira et al. [33], incremental browsing and increasing the size in Botterweck et al. [9] or manually introducing visibility conditions in Dhungana et al. [14]). We also follow these principles in our approach while this was not present in Martinez et al. work.

Schneeweiss and Botterweck [35] presented Feature Flow Maps which focused on NFPs. This visualization helps to comprehend the contribution of each selected feature in the global value of an NFP. However, the positive or negative impact between features is not visualized. This visualization can be complementary to ours. Bagheri et al. [4] proposed to use colors inside the feature diagram to encourage or discourage features based on an interactive gambling process. The NFPs are considered by showing the interdependencies among them and the impact of each feature in the NFPs. We propose a set of visualization paradigms that are alternatives to this work but both can be complementary as well.

Finally, several authors proposed automatic approaches to automatically configure a product optimizing NFPs [27]. These approaches provide a set of possible solutions and Murashkin et al. [24] proposed a visualization to help the users to select a solution among the ones automatically found. Contrary to this, our visualization focuses on staged and gradual configuration processes.
to support the interactive selection of features. In addition, our approach can be combined with Murashkin et al. approach in order to allow users to see further information about features and NFPs for the set of allowed solutions. Other approaches, such as Temple et al. [39], narrow the configuration space by automatically discovering extra constraints related to specific requirements. This can be also complementary to our approach.

4 THE PROPOSED VISUALIZATIONS

To represent functional and non-functional features’ interdependencies, as well as features’ relevance on SPL configuration tools, we provide a set of three visualization components: 1) 5-star view, 2) feature’s graph, and 3) NFP’s graph. The visualizations allow the user to freely explore the functional and NFPs related to the product-under-configuration in order to choose the features that best meet their requirements. In addition, filters can be applied simultaneously to NFPs. Filters define NFPs thresholds from a minimum to a maximum value that fulfill the product’s non-functional requirements. Thus, the set of features in the visualizations are adapted to satisfy the set of predefined filters. By providing this mechanism, we can additionally reduce the decision makers configuration workload. Next, we describe each visualization component shown in Figure 2.

4.1 5-Star View

The 5-star view represents features’ relevance over a focused view of the FM (Sec. 2) plus a color highlighting notation. It uses the state-of-the-art BRISMF recommender algorithm adapted by Pereira et al. [33] to display the score in the form of a 5-star scale as shown in Figure 3.

From this view, the user will be able to (de)select features of interest to configure a product. Features’ scores are computed after each user interaction. Each score is normalized in a scale from zero to five and associated with an unselected feature on the focused view. This view reflects a generalization of knowledge that is inferred from previous configurations, and ensures the consistency of complete and partial configurations.

In addition, a highlighted view guides the user through a final valid configuration. It shows the user which decisions are necessary to finish the configuration process by highlighting the corresponding features. To find these features, we use an algorithm based on the propositional formula of an FM. In order to satisfy the propositional formula, the decision-maker would either need to deselect a blue parent feature or select one of its green children features (Fig. 3, sensor is in blue and all features with stars except alarm are in green). Naturally, (de)selecting one of those features automatically satisfies the corresponding clause. After a clause is satisfied by the decision-maker’s (de)selection, the focus automatically changes to the next unsatisfied clause. Thus, with the help of the 5-star view, the user can efficiently explore the configuration space and finish the configuration process preventing undesired feature selections.

4.2 Feature’s Graph

To represent the impact of undefined features (i.e., non-(de)selected) over a set of relevant features and NFPs, we extend the FRoGs view proposed by Martinez et al. [20]. Similar to FRoGs, each graph is a radial ego network representation associated with a specific target feature displayed in the center (e.g., sensor in Fig. 4a). We extend FRoGs notation by showing to decision makers how the selection of feature affects the features’ scores of a set of relevant features and the values of its NFPs. Also, while FRoGs proposes to use all the features in the FM, we reduce the information density by using only the features of the focused view. Then, the set of $n$ relevant features and $m$ NFPs are displayed around $f_c$ with a constant separation of $2\pi/(n + m)$. This separation allows to uniformly distribute all features and NFPs around the circle. Each graph displays different circular sectors (i.e., zones) associated with two main perspectives: features’ relevance scores and values of NFPs.

Perspective: Features’ Relevance Scores. A set of relevant features are displayed in this perspective depending on their computed relevance score (Sec. 4.1). Features’ relevance scores are associated with a minimum value of 0 and a maximum value of 5. Features are positioned in the range of these values on the graph. To represent different ranges of values, the zones in this perspective are displayed with different shades of the yellow color. The light yellow of the positive zone contrasts with the dark yellow of the negative zone. Features in the positive zone are closest to $f_c$ meaning that, if $f_c$ is selected, these features are more likely to be selected (i.e., the extreme of this zone is represented for the maximum value of relevance score, which is 5). Features in the negative zone are furthest from $f_c$ meaning that if $f_c$ is selected these features are more likely to be deselected (i.e., the extreme of this zone is represented for the minimum value of relevance score, which is 0). As an example, Figure 4a shows the impact of the feature sensor on the relevance score of nine relevant features. Relevant features are the set of undefined features displayed in the focused view (Fig. 3), as well as sensor dependent features. As we can see in this example, the selection of the feature sensor potentially encourages the selection of fire, alarm, siren, and manual located in the extreme of the positive zone. Although the sub-features of alarm (i.e.,
silent, visual, and siren) are not displayed in the focused view, the target feature sensor requires alarm. Consequently, at least one of alarm sub-features need to be selected if sensor is selected. Therefore, we classify these features as relevant. A relevance score for a relevant feature is updated over time depending on $pc$.

We use $\circ$ and $\bullet$ for required and excluded features constraints, respectively. If it is neither a requires nor an excludes constraint, we use $\cdot$. On the one hand, the notation $f_i$ illustrates that there is no occurrence of $f_c$ without $f_i$ given $pc$. However, when a set of required features are linked by a dashed line, at least one feature must be selected if $f_c$ is selected, otherwise exactly one of those features must be selected. For example, the selection of sensor requires the selection of exactly one of the features manual, and automatic, and the selection of at least one of the features fire, flood, and glass. On the other hand, the notation $f_i$ illustrates that there is no occurrence of $f_c$ and $f_i$ together given $pc$.

Requires and excludes notations are consequences of formalized and inferred constraints. Inferred constraints are represented by adding an extra circle in the node with the color associated to the type of the constraint (i.e., $\bullet$ for requires and $\circ$ for excludes). An inferred constraint is a condition that is not explicitly formalized in the FM by the feature tree and the CTCs, but that exists because of logical rules [20]. Figure 4a shows an example of inferred constraint for the features silent, visual, and siren. In the FM shown in Figure 1, there is a CTC sensor requires alarm. Consequently, by looking at the feature tree, the feature alarm requires at least one of the features silent, visual, and siren. Therefore this implication is not explicitly formalized neither in the feature tree nor in the CTCs. We display all the dependent features of $f_c$ and its relation types by analyzing the propositional formula of the FM. Figure 4 shows the complete legend of the graph as it is shown in the tool.

Perspective: Values of NFPs. A set of NFPs are displayed in this perspective depending on their values over the feature $f_c$. NFP values are categorized into two main categories: quantitative and qualitative. Quantitative NFPs are properties represented as a numeric value (e.g., cost and response time), otherwise qualitative NFPs are represented using an ordinal scale, such as low, medium, and high (e.g., security and maintainability). There are a large number of quantitative and qualitative NFPs reported in the literature [11, 19] and the same property can vary for each domain, application scenario, environment, and stakeholder. There are authors [34, 37, 41] who propose the use of functional metrics to a quantifiable measurement of NFPs, and even authors [4, 5, 18] who propose the use of domain expert judgement to assign qualitative or quantitative NFP values. In this work, as we focus on the visualizations, we assume that those values were already specified by using state-of-the-art techniques.

Since the types of quantitative NFPs are quite different, their range of values may suffer a wide variation (e.g., 100-60,000ms for response time and $500-850$ for cost). In this scenario, the range of values should be normalized so that each NFP contributes approximately proportionately to their minimum and maximum values. Thus, we use a simplest feature scaling method (also called unity-based normalization) to rescale the range of NFP value between any arbitrary minimum and maximum value to a common scale in the range of $[0, 1]$. Our aim is that the corresponding normalized values allow the comparison of different types of NFPs.

An NFP is displayed in the zones by considering a scale ranging from a negative impact of 0 to a positive impact of 1. For qualitative NFPs, we automatically mapped them onto real values to be handled as quantitative properties in a scale of $[0, 1]$. To represent the six different zones in the graph, we consider six qualitative levels [3]: high [0.85, 1], medium [0.68, 0.85] and low [0.51, 0.68] positive; and low [0.34, 0.51], medium [0.17, 0.34] and high [0, 0.17] negative.

To represent different ranges of values, the zones in this perspective are displayed with different shades of the blue color (Fig. 4a). As much light is the color of the zone where the NFP is positioned much positive influence it has over $f_c$ (e.g., security), while as much dark the color much negative the influence (e.g., response time). The proposed graph adapts dynamically in accordance with
the amount of information to be displayed (i.e., if there are no NFP associated to \( f_c \), then the graph fully represents the perspective of features’ relevance scores).

### 4.3 Non-Functional Property’s Graph

Apart from interacting with the list of undefined features, the user can interact with NFPs displayed in the feature’s graph view. This view is associated with a NFP \( p_c \) displayed in the center (e.g., cost in Fig. 4b). It shows the effect of \( p_c \) over relevant features and other NFPs. Each graph displays different circular zones associated with two perspectives: features’ NFP values and NFPs interdependencies.

**Perspective: Features’ NFP Values.** A set of relevant features are displayed in this perspective depending on their \( p_c \) value. The zones in this perspective are associated with a minimum to a maximum value of \( p_c \). As must light the color where the feature is displayed, much positive the influence of \( p_c \) over the feature, otherwise much negative the influence. As an example, by visualizing the functional features perspective in Figure 4b, the decision maker has an overview of which features are cheaper (e.g., manual) and more expensive (e.g., visual).

**Perspective: NFPs Interdependencies.** The set of NFPs are displayed in this perspective depending on their effect on \( p_c \). To measure the effect of an NFP on \( p_c \), we use the Pearson Correlation Coefficient (PCC) also known as bivariate correlation. PCC measures the degree of covariation between two NFPs. For example, given two NFPs cost and security, if, on average, expensive products have high security and low cost, we say that cost and security are correlated. To measure the correlation between NFPs, our approach needs to find all valid combinations of features that contribute to both NFPs. However, the number of feature combinations in an FM may be exponential to the number of features that contribute to both NFPs. Therefore, we compute PCC using the subset of previous configurations from past users. Thus, PCC is the covariance of two NFPs divided by the product of their standard deviations. The equation is given as:

\[
\begin{align*}
r &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad \text{cov}(X, Y) = \frac{\text{cov}(X, Y)}{\text{var}(X) \cdot \text{var}(Y)}
\end{align*}
\]

where \( x_i \) and \( y_i \) are the values that \( p_c \) and \( p_l \) assumes for each given previous configuration. \( \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \) is the sample mean (analogously for \( \bar{y} \)). Thus, the value \( r \) is obtained based on the subset of \( n \) valid previous configurations (i.e., samples), then it is an estimation of the true value of the correlation of all valid configurations. It assumes the values between [-1,1], where -1 implies a negative linear correlation (i.e., \( p_l \) decreases as \( p_c \) increases), a value of 0 implies that there is no linear correlation between \( p_l \) and \( p_c \), and 1 implies a positive linear correlation (i.e., \( p_l \) increases as \( p_c \) increases). Thus, the degree of correlation is more significant as it is closer to 1 (direct correlation) or -1 (inverse correlation).

To represent the different ranges of correlations, NFPs are positioned in the range of the values [-1,1] on the graph. Thus, NFP’s closest to \( p_c \) has a positive effect on \( p_c \), meaning that for a positive value of \( p_c \) there is a positive effect of \( p_l \) (i.e., the extreme of this zone is represented for the maximum value of PCC correlation, which is 1). NFPs furthest from \( p_c \) have a negative effect on \( p_c \), meaning that for a positive value of \( p_c \) there is a negative effect of \( p_l \) (i.e., the extreme of this zone is represented for the minimum value of PCC correlation, which is -1). As example, Figure 4b shows that the NFPs response time and performance have a higher negative effect in the configuration over a positive effect of the NFP cost (i.e., cheaper products have a high response time and low performance).

### 5 EVALUATION

To empirically evaluate our approach, we extend the state-of-the-art tool FeatureIDE with the proposed visualizations described in Section 4. In this section, we aim at deriving answers to the following research questions:

- **RQ1 (Effectiveness).** Does the set of proposed visualizations support the configuration process of realistic PLs?
- **RQ2 (Scalability).** Does the set of proposed visualizations avoid information overload?
- **RQ3 (Performance).** How long does it take to construct the set of visualizations when increasing the complexity of the problem?

All the experiments were performed on an Intel Core @3.30GHz with 8GB of RAM.

#### 5.1 Approach Effectiveness

We empirically analyze the effectiveness of our approach to reduce the complexity of the interactive configuration process by assisting the decision makers to reasoning on a small set of relevant information. We designed a preliminary experiment in which 10 participants used the FeatureIDE configuration tool to configure the DELL laptops PL\(^1\) [23]. This PL is suitable for our experiment because the domain is easy to understand and a realistic dataset of configurations was available. Table 1 (in the seventh row) summarizes the main characteristics of the DELL PL.

Firstly, we developed a requirement specification for a gamer DELL laptop\(^2\). Laptops for gaming differs from laptops targeting other type of customer profiles (e.g., atom netbooks would not meet the needs of a gamer laptop due to the high processing demand of current games, which require a high performance). In this context, if the product obtained by the user is significantly different from the expected ones, we assume that our approach is not appropriately guiding the user. We use Recall = \(|c_l \cap c_j|/|c_j|\) to evaluate the similarity between a user configuration \( c_j \) and an expected configuration \( c_l \) (i.e., a valid set of truly relevant features known from the specified requirements). We consider as \( c_j \) the expected product configuration much similar to \( c_l \).

The experiment consisted of two groups with 5 Master’s students without SPL experience and with knowledge in computer games to make sure they are on the same level of background. One group used the FeatureIDE configuration tool without the proposed visualizations, while the other group used FeatureIDE with our visualization components. Moreover, in order to balance knowledge of participants, we conducted a training session of 1 hour to introduce the basic concepts of PL configuration and the tool support. After the training session, we asked the participants to perform a configuration task of a gamer DELL laptop by following the requirement specification. All configurations were created from

---

\(^1\)We adapt the DELL laptops PL found at http://www.splot-research.org/. The complete FM with a dataset of 33 real configurations can be found at [28].

\(^2\)Further details about the product specification and the configuration task carried out by the participants are available at [28].
We investigate whether the implemented visualization components were somehow confusing during the task (P7 and P8); (ii) the graph is showing in new windows, instead of in the configuration view (P9 and P10); and (iii) the automatic (de)selection applied to validate the configuration makes me lose control of the consequence of my selections (P2 and P3); (iv) I cannot track the non-functional requirements (P1, P2, P3, and P5). However, the positive point was unanimous about the easy process to (de)select a feature through check boxes. Moreover, some of the participants (P1, P4, and P5) mentioned that the automatic (de)selection of dependent features is a favorable point. On the other hand, the five main concerns reported by the participants P6-10 were: (i) I was not sure where to start (P3); (ii) the relationships among features are not explicitly clear in the configuration view; as it is in the FM editor view (P1 and P4); (iii) the automatic (de)selection applied to validate the configuration makes me lose control of the consequence of my selections (P2 and P3); (iv) I need support to reason over competitive features (P4); and (v) I cannot track the non-functional requirements (P1, P2, P3, and P5).

In addition, we ask the participants the main difficulties they faced using the configuration tool. This analyzes the problems that participants may have to carry out the configuration task by using both tool supports. On the one hand, the five main concerns reported by the participants P1-5 were: (i) I was not sure where to start (P3); (ii) the relationships among features are not explicitly clear in the configuration view; as it is in the FM editor view (P1 and P4); (iii) the automatic (de)selection applied to validate the configuration makes me lose control of the consequence of my selections (P2 and P3); (iv) I cannot track the non-functional requirements (P1, P2, P3, and P5). However, the positive point was unanimous about the easy process to (de)select a feature through check boxes. Moreover, some of the participants (P1, P4, and P5) mentioned that the automatic (de)selection of dependent features is a favorable point. On the other hand, the five main concerns reported by the participants P6-10 were: (i) I was not sure where to start (P3); (ii) the relationships among features are not explicitly clear in the configuration view; as it is in the FM editor view (P1 and P4); (iii) the automatic (de)selection applied to validate the configuration makes me lose control of the consequence of my selections (P2 and P3); (iv) I cannot track the non-functional requirements (P1, P2, P3, and P5). However, the positive point was unanimous about the easy process to (de)select a feature through check boxes. Moreover, some of the participants (P1, P4, and P5) mentioned that the automatic (de)selection of dependent features is a favorable point.

From the average values of Table 2, we can see that the gaps between these two approaches are large for decision makers of this PL. The users of our approach have advantages in terms of the accuracy of the final complete configuration, the number of decisions to make, the number of rollbacks, and the confidence level. The recall for users of our approach is higher than for users of the traditional configuration approach (due to the low recall of P2 and P3), which suggest that the users of our approach will be more confident about the fitness of the final product (see confidence column). Although by using our approach the users took much longer to finalize the configuration process, they are more confident and consequently they performed a relative minor number of rollbacks. Therefore, we believe that the time consumed was due the users’ understanding of the features’ and NFP’s dependencies.

In addition, we ask the participants the main difficulties they faced using the configuration tool. This analyzes the problems that participants may have to carry out the configuration task by using both tool supports. On the one hand, the five main concerns reported by the participants P1-5 were: (i) I was not sure where to start (P3); (ii) the relationships among features are not explicitly clear in the configuration view; as it is in the FM editor view (P1 and P4); (iii) the automatic (de)selection applied to validate the configuration makes me lose control of the consequence of my selections (P2 and P3); (iv) I need support to reason over competitive features (P4); and (v) I cannot track the non-functional requirements (P1, P2, P3, and P5). However, the positive point was unanimous about the easy process to (de)select a feature through check boxes. Moreover, some of the participants (P1, P4, and P5) mentioned that the automatic (de)selection of dependent features is a favorable point. On the other hand, the five main concerns reported by the participants P6-10 were: (i) I was not sure where to start (P3); (ii) the relationships among features are not explicitly clear in the configuration view; as it is in the FM editor view (P1 and P4); (iii) the automatic (de)selection applied to validate the configuration makes me lose control of the consequence of my selections (P2 and P3); (iv) I need support to reason over competitive features (P4); and (v) I cannot track the non-functional requirements (P1, P2, P3, and P5). However, the positive point was unanimous about the easy process to (de)select a feature through check boxes.

Moreover, some of the participants (P1, P4, and P5) mentioned that the automatic (de)selection of dependent features is a favorable point. On the other hand, the five main concerns reported by the participants P6-10 were: (i) I was not sure where to start (P3); (ii) the relationships among features are not explicitly clear in the configuration view; as it is in the FM editor view (P1 and P4); (iii) the automatic (de)selection applied to validate the configuration makes me lose control of the consequence of my selections (P2 and P3); (iv) I need support to reason over competitive features (P4); and (v) I cannot track the non-functional requirements (P1, P2, P3, and P5). However, the positive point was unanimous about the easy process to (de)select a feature through check boxes. Moreover, some of the participants (P1, P4, and P5) mentioned that the automatic (de)selection of dependent features is a favorable point. On the other hand, the five main concerns reported by the participants P6-10 were: (i) I was not sure where to start (P3); (ii) the relationships among features are not explicitly clear in the configuration view; as it is in the FM editor view (P1 and P4); (iii) the automatic (de)selection applied to validate the configuration makes me lose control of the consequence of my selections (P2 and P3); (iv) I need support to reason over competitive features (P4); and (v) I cannot track the non-functional requirements (P1, P2, P3, and P5). However, the positive point was unanimous about the easy process to (de)select a feature through check boxes. Moreover, some of the participants (P1, P4, and P5) mentioned that the automatic (de)selection of dependent features is a favorable point.

From the average values of Table 2, we can see that the gaps between these two approaches are large for decision makers of this PL. The users of our approach have advantages in terms of the accuracy of the final complete configuration, the number of decisions to make, the number of rollbacks, and the confidence level. The recall for users of our approach is higher than for users of the traditional configuration approach (due to the low recall of P2 and P3), which suggest that the users of our approach will be more confident about the fitness of the final product (see confidence column). Although by using our approach the users took much longer to finalize the configuration process, they are more confident and consequently they performed a relative minor number of rollbacks. Therefore, we believe that the time consumed was due the users’ understanding of the features’ and NFP’s dependencies.

In addition, we ask the participants the main difficulties they faced using the configuration tool. This analyzes the problems that participants may have to carry out the configuration task by using both tool supports. On the one hand, the five main concerns reported by the participants P1-5 were: (i) I was not sure where to start (P3); (ii) the relationships among features are not explicitly clear in the configuration view; as it is in the FM editor view (P1 and P4); (iii) the automatic (de)selection applied to validate the configuration makes me lose control of the consequence of my selections (P2 and P3); (iv) I need support to reason over competitive features (P4); and (v) I cannot track the non-functional requirements (P1, P2, P3, and P5). However, the positive point was unanimous about the easy process to (de)select a feature through check boxes. Moreover, some of the participants (P1, P4, and P5) mentioned that the automatic (de)selection of dependent features is a favorable point.

From the average values of Table 2, we can see that the gaps between these two approaches are large for decision makers of this PL. The users of our approach have advantages in terms of the accuracy of the final complete configuration, the number of decisions to make, the number of rollbacks, and the confidence level. The recall for users of our approach is higher than for users of the traditional configuration approach (due to the low recall of P2 and P3), which suggest that the users of our approach will be more confident about the fitness of the final product (see confidence column). Although by using our approach the users took much longer to finalize the configuration process, they are more confident and consequently they performed a relative minor number of rollbacks. Therefore, we believe that the time consumed was due the users’ understanding of the features’ and NFP’s dependencies.

In addition, we ask the participants the main difficulties they faced using the configuration tool. This analyzes the problems that participants may have to carry out the configuration task by using both tool supports. On the one hand, the five main concerns reported by the participants P1-5 were: (i) I was not sure where to start (P3); (ii) the relationships among features are not explicitly clear in the configuration view; as it is in the FM editor view (P1 and P4); (iii) the automatic (de)selection applied to validate the configuration makes me lose control of the consequence of my selections (P2 and P3); (iv) I need support to reason over competitive features (P4); and (v) I cannot track the non-functional requirements (P1, P2, P3, and P5). However, the positive point was unanimous about the easy process to (de)select a feature through check boxes. Moreover, some of the participants (P1, P4, and P5) mentioned that the automatic (de)selection of dependent features is a favorable point.
We evaluate the performance of our approach by measuring the worst-case scenario regarding scalability. Each feature of each FM (except the DELL laptops) has been annotated with 6 NFPs: cost, response time, performance, security, reliability, and maintainability. The values for quantitative NFPs have been set randomly with a uniform distribution: cost takes real values between $0 and $500; response time takes integer values between 0ms and 60,000ms; performance takes percentage values between 0% and 100%. Finally, the NFPs security, reliability, and maintainability take the following qualitative values [HighN, MedN, LowN, LowP, MedP, HighP] introduced by [3]. For the DELL laptops PL, we consider 3 quantitative NFPs (price, frequency, and rotation) and 2 qualitative NFPs (security and performance). The values for these NFPs were specified by game domain experts. The number of NFPs for these PLs reaches a threshold of six which is the upper bound number suggested by some authors (Mairiza et al. [19] and Sommerville and Sawyer [38]) as the most often used for realistic systems.

To perform our experiment, we developed a computer program that randomly chose a feature to be analyzed and consequently selected. The program repeats the process until obtaining a complete configuration. We performed this experiment by taking into account a set of 1,000 runs. Each configuration was created from scratch and no filter to stakeholders’ requirements was applied. For each feature being analyzed, we collected the number of relevant features plus the number of NFPs being displayed by the graphs and we record the worst case scenario (i.e., with higher information overload). The Graph Data column of Table 1 presents the experiment results for each PL.

In the worst case, the graph showed up 56 features and 6 NFPs (i.e., 62 nodes) for the Battle of Tanks PL. Although this PL is not the largest, it has few mandatory features and a small height of the feature tree. Consequently, a parent feature has many dependent sub-features and, when they are selected, there are many required features (due the XOR relation). To have insights about the scalability of the resultant visualization, we showed the generated graph with the worst case scenario to the ten participants of our user study (Sec. 5.1) in order to find out if they have any trouble understanding the graph. All participants were able to completely explain the information in the graph. Overall, they described that once the relevant information are in the extreme of the graph (i.e., higher prediction and positive NFP values) and represented for different notations (i.e., , , , and ), they can easily focus on a reduced amount of data and quickly decide which feature would be more relevant to them. However, they complained mainly about the large name of some features and the feature positions in each zone. Overall, although there are some points to be improved in the future regarding usability, our approach was applicable for this worst-case scenario regarding scalability.

5.3 Approach Performance

We evaluate the performance of our approach by measuring the response time of the tool for generating the visualization graphs. We capture the response time during the scalability experiment presented in Section 5.2. For each feature being analyzed, we record the highest time spent for the configuration tool to generate the visualizations (See Time column of Tab. 1). In the worst case, we have around 8 seconds for the Web Portal and e-Shop PLs. Displaying a feature’s graph is an algorithm with order $O(nm)$ where $n$ is the number of features related to the feature in the center (i.e., $f_b$) plus the unselected features showing up in the focused view, and $m$ is the number of NFPs related to $f_b$. Moreover, displaying the NFP’s graph is also an algorithm with order $O(nm)$, here $n$ is the number of unselected features showing up in the focused view, and $m$ is the number of NFPs (except the one in the center). As summary, these worst case scenarios can negatively affect the user experience as visual continuity is not reached. These were worst cases however further efforts can be done to improve the current implementation regarding performance.

5.4 Threats to Validity

Regarding the internal validity, we identify the characteristics of the PL used for evaluating the efficiency and the number of participants as two relevant factors (Sec. 5.1). We used a publicly available PL and we have investigated and created suitable NFPs based on experts opinion and general characteristics of NFPs found in the literature [11, 19]. In addition, we discussed the experiment design with experienced researchers in the PL field. However, additional experiments are required to determine the impact in other scenarios.

Another limitation of this study concerns to the focused view used by the proposed graphs. On the one hand, the focused view reduces the information density. On the other hand, it might cause that the user will lose the “global view”. To minimize this validity threat, we allow the user to expand the focused view in the 5-star perspective but we agree that loosing the global view in the graph perspectives might be problematic. Also, the way that the features are filtered (child features) might cause that the effectiveness of our approach depend on the topology of the target feature model. To simulate the practical configuration task while measuring the scalability and performance of our approach (Sec. 5.2 and 5.3), the set of selected features were chosen randomly from the whole set of relevant features. We carried out 1,000 runs to try to ensure significance of the results of this stochastic process and we reported the worst scenario to prove its feasibility.

Regarding external validity, our experiments to check the scalability rely on eleven large and medium size real-world PLs with different types of features and structures and from a set of diverse domains (Tab. 1). However, the use of other PLs with different characteristics could have impacted our results. We try to minimize this validity threat by documenting the characteristics of the PLs. Still, conducting experiments with additional PLs with realistic NFPs remains part of our future work.

Our approach is designed to work effectively when a large dataset of realistic previous configurations is available. This is a requirement for our recommender system that allows us to compute a set of more reliable feature scores and build the visualizations. We also use this dataset to compute NFPs’ interdependencies. In this work, we assume the use of state-of-the-art approaches to compute NFP values results of the interaction of a valid set of features (i.e., our focus is in the visualizations). We are aware that measuring NFPs might influence the scalability and time performance of our approach. This, however, goes beyond the scope of this paper.

The complete representation (xml files) of all PLs can be found at [28]
6 CONCLUSION AND FUTURE WORK

We have proposed a set of interrelated visualizations to improve the efficiency of decision makers to configure a product. We extended the state-of-the-art configuration tool FeatureIDE with our approach. This tool ensures the consistency of the configured products. In addition, our approach reduces the configuration effort and complexity of decision making by providing a restricted view of the configuration space and by assisting the decision makers to reasoning on a focused set of relevant information about features and NPs. We conducted a set of numerous experiments with eleven state-of-the-art PLs to evaluate three important practical characteristics of our approach: effectiveness, scalability, and performance.

Our experimental results show that the proposed set of visualizations are useful as: (i) it is able to improve the effectiveness of the configuration process in three perspectives: the accuracy of the configuration, the number of decisions to take, and the confidence level of decision makers; and (ii) it is scalable for state-of-the-art PLs. We assume that most real-world problems will be of similar scale to our experiments. As future work, we aim to explore how the number of previous configurations affect the efficiency of the visualizations (i.e., at which point they start to be useful).

ACKNOWLEDGMENT

This work was partially supported by the CNPq grant 202368/2014-9. Martinez’s work is supported by the ITEA3 15010 REVaMP² project: FIT the Ile-de-France region and BPI in France.

REFERENCES


