Toward Automated Feature Model Configuration With Optimizing Non-Functional Requirements

Mohsen Asadi[a], Samaneh Soltani[a], Dragan Gasevic[a,b], Marek Hatala[a], Ebrahim Bagheri[c]

[a] *Simon Fraser University, Canada*
[b] *Athabasca University, Canada*
[c] *Ryerson University, Canada*

Abstract

Context: A Software Product Line is a family of software systems that share some common features but also have significant variabilities. A feature model is a variability modeling artifact, which represents differences among software products with respect to the 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 binding the variation points in the feature model (called configuration process) and by instantiating the reference model.

Objective: In this work we address the feature model configuration problem and propose a framework to automatically select suitable features that satisfy both the functional and non-functional preferences and constraints of stakeholders. Additionally, interdependencies between various non-functional properties are taken into account in the framework.

Method: The proposed framework combines Analytical Hierarchy Process (AHP) and Fuzzy Cognitive Maps (FCM) to compute the non-functional properties weights based on stakeholders' preferences and interdependencies between non-functional properties. Afterwards, Hierarchical Task Network (HTN) planning is applied to find the optimal feature model configuration.

Result: Our approach improves state-of-art of feature model configuration by considering positive or negative impacts of the features on non-functional properties, the stakeholders' preferences, and non-functional interdependencies. The approach presented in this paper extends earlier work presented in [1] from several distinct perspectives including mechanisms handling interdependencies between non-functional properties, proposing a novel tooling architecture, and offering visualization and interaction techniques for representing functional and non-functional aspects of feature models.

Conclusion: our experiments show the scalability of our configuration approach when considering both functional and non-functional requirements of stakeholders.

Keywords: Software Product Lines, Feature Model Configuration, Stakeholders' Preferences, Non-functional Interdependencies

1. Introduction

Software Product Lines Engineering (SPLE) aims at developing a set of software systems that share common features and satisfy the requirements of a specific domain [2]. SPLE decreases development costs and time to market, and improves software quality

through strategic reuse of assets within a domain of interest. A technique adopted in SPLE for managing reusability is commonality and variability modeling through which common assets and their variabilities are formalized.

A software product line lifecycle encompasses a *domain engineering* process and an *application engineering* process. In the *domain engineering* process, a comprehensive formal representation of the products of the domain is developed. This includes a variability model and the core assets of the product family. *Feature models* are among the prevalent variability modeling techniques in SPLE and represent variability in terms of the differences between the features of the products that belong to a software family. A feature is a logical unit of behavior specified by a set of functional and non-functional requirements [3].

On the other hand, the *application engineering* process is responsible for capturing the requirements of the target application, deriving a concrete product from the variability model through a configuration process, and deploying the product into users' environment [4]. Using feature models as variability modeling tools, the configuration process selects a suitable set of features to satisfy the stakeholders' requirements.

1.1. Open Problem

The focus of existing research in software product lines has been mostly on modeling and managing product lines, while there are only a few works which in particular concentrate on the effective utilization of product lines (also known as the *configuration problem*) during application engineering. In the configuration problem, an application engineer receives a feature model and the target application requirements and attempts to select a subset of the features that optimize the requirements. Solving the configuration problem is challenging for an application engineer due to following reasons:

- There are several types of variability relations and integrity constraints between the features [2];
- The number of possible configurations has exponential growth. Even in small feature models the number of possible configurations can be very high. Industrial feature models may consist of hundreds of features which increases the complexity of feature model configuration [5].
- Features may have either positive or negative impact on the different business concerns of a product, and hence expose different quality attributes [6]. We refer to business concerns of a product (e.g., *security* and *customer satisfaction*) and quality attributes of a product (e.g., *performance* and *cost*) as non-functional properties (NFP). For example, a feature may have a negative impact on *security*, but a positive impact on *customer satisfaction* or it could have *high performance* but *low reliability*.
- In addition to functional requirements, stakeholders may have several constraints and preferences over non-functional properties during the product derivation. For example, one stakeholder may ask for a product with *high security*, *high customer satisfaction*, and specific *cost*; and can mention that *customer satisfaction* is more important than *security*; which would make the configuration process more difficult and complex [5, 7].

- Finally, there are interdependency relations between non-functional properties such that an increase or decrease in the value of one non-functional property may lead to an increase or decrease in the value of another non-functional property. For example, increase in the value of *security* may decrease the value of *performance* for the stakeholders [8].

There are a number of algorithms for product line configuration [9, 10, 11, 12], which aim at assisting application engineers in solving the configuration problem. However, to our knowledge, there has been little coverage for the consideration of non-functional requirements in the configuration algorithms. Some techniques have addressed the configuration problem by transforming the feature model configuration problem into a Constraints Satisfaction Problem (CSP), and have used CSP-solvers to build optimal configurations [5, 13]. The main problem with these techniques is time inefficiency. Other techniques solve this problem by applying approximation algorithms, but their final configurations are only partially optimal [14, 15]. To our knowledge, almost all these works except [13] only support limited types of NFPs (i.e., quantitative NFPs such as footprint and cost) and do not consider qualitative NFPs (e.g., security). Moreover, no work has considered the preferences of stakeholders in terms of the relative importance between non-functional properties in the process of feature model configuration; and relative importance varies depending on the stakeholders standpoint and application domain [16]. Relative importance of non-functional properties is especially important for the stakeholders and software designers who are able to define the relative importance among the available functional and non-functional options but have difficulty in deterministically picking their choice from those options [16]. Thus, a product line configuration technique should not only be able to operate over deterministic functional choices, but should also be able to operate when the relative importance between both functional and non-functional properties are given. Finally, to the best of our knowledge, no feature model configuration approach has taken interdependencies between non-functional properties into account.

1.2. Contribution and Approach Overview

Existing challenges in the configuration problem faced by application developers motivated us to develop an automated method for selecting a set of features that would fulfill both the stakeholders' functional and non functional requirements and preferences. To this end, we adopted and integrated the *Analytical Hierarchy Process (AHP)* [17], *Fuzzy Cognitive Map (FCM)* [18], and *Preference-based Planning* techniques[19].

AHP is a well-known pairwise comparison method used to calculate the relative ranking of different options based on stakeholders' judgments [17, 20]. FCM [18] is an extension of cognitive maps which incorporates fuzzy causal functions to represent fuzzy relations among objects in a complex system. FCM have been widely used several domains for modeling and decision making [21, 22, 23]. Hierarchical Task Network (HTN) planning is a popular planning technique, which is suited for domains with hierarchical task decomposition [24, 19]. The HTN Planning technique generates plans from a developed hierarchical network of domain tasks and actions [25].

The general overview of the proposed approach is illustrated in Figure 1. As shown in the figure, our approach captures functional requirements and non-functional requirements of the stakeholders for a final application. Non-functional requirements are captured in

terms of relative importance of non-functional properties along with constraints over the non-functional properties (Section 3.3). Afterward, we employ an Analytical Hierarchy Process (AHP)[17] to calculate the local weights of non-functional properties. To incorporate the value related interdependencies during the feature selection, we employ Fuzzy Cognitive Map(FCM) to compute the overall influence between non-function properties and then calculate the global non-functional properties weights by integrating result of AHP and FCM (Section 4.1.1). Next, ranks of the features based on the importance of non-functional properties and their values assigned to the features is obtained via a utility function defined in section 4.1.2.

We generate the HTN planning domain and problem from the feature model and stakeholders' requirements (Section 4.2). First, preprocessing steps are preformed to handle optional and OR relation for transformation and then we produce domain predicates, operators, and tasks according to the proposed transformation rules. We apply SHOP2 planner [26], an HTN-based planning system widely used for planning problems, to identify an optimal plan (i.e., a plan with the best overall cost). To produce the final configuration, the features chosen by the SHOP2 planner are selected and represented in the visual view of our tool.

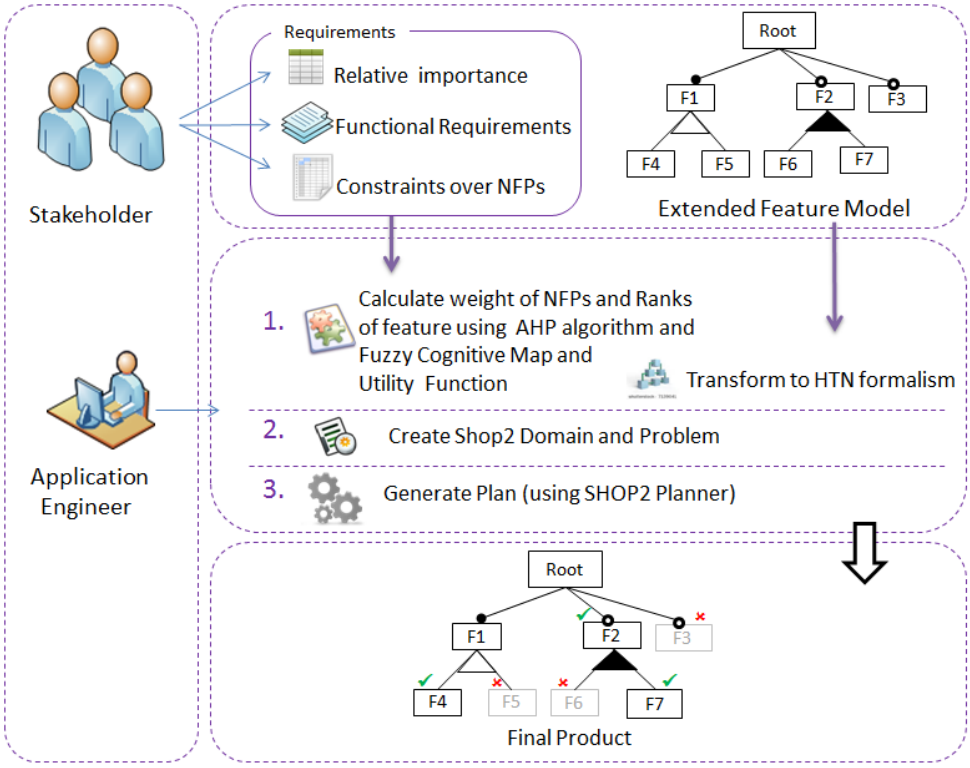


Figure 1: Automatic feature model configuration process in application engineering

The main contributions of this paper are as follows:

- Proposing a complete approach for configuring feature models which includes an easy-to-understand formalism for capturing the stakeholders' preferences over non-functional properties represented in terms of relative importance; utilizing the analytic hierarchy process and fuzzy cognitive maps to calculate the weightes of NFPs;

transforming a feature model and stakeholders' preferences and constraints into a planning domain and problem by considering both functional and non-functional requirements;

- Developing a tool to support the proposed approach for feature model configuration.
- Conducting several experiments to evaluate the performance and usefulness of the proposed approach and the developed tool.

The proposed approach has been evolved over our previous works [6, 27, 7, 1]. In [6, 27] we only concentrated on the relative importance over qualitative non-functional properties and adapted an AHP approach to prioritize features in feature models. In this work, we cover both qualitative and quantitative non-functional properties and propose a utility function to compute the ranks of features. In [7, 1] we applied HTN for feature selection based on business concerns and non-functional properties, respectively. Here, we perform feature selection based on qualitative and quantitative non-functional properties.

Specifically with respect to our previous approach presented at SPLC2012 [1], we extend the approach from several perspectives. The main extension was the integration of interdependencies between non-functional properties and employing fuzzy cognitive maps for resolving dependencies (Section 4). In this line, we proposed a technique which considers both preferences of stakeholders over non-functional and impacts of non-functional properties on each other to compute the ranks of features. According to the study performed in [28], handling interdependencies between non-functional properties is a critical issue in many industrial companies. To best of our knowledge no approach in software product lines (including our work at SPLC12) has tackled this issue before. Additionally, to improve the usability of the approach, we applied visualization and interaction techniques to the supporting tool set(Section 5).

The paper is organized as follow: Section 2 introduces the fundamentals and basic domain concepts including multi-criteria decision making and hierarchical task networks. Next, the definition of configuration problem is given in section 3. The details of the proposed approach is explained in section 4. Section 5 introduces the developed tool and explains its characteristics. After evaluating the proposed approach and developed tool in terms of scalability and effectiveness in section 6, we highlight the related works and compare our work with them in section 7. Finally, section 8 concludes the paper and mentions the open issues and future works.

2. Background

Our configuration approach is based on techniques in multicriteria decision making and hierarchical planning which are reviewed in this section.

2.1. Hierarchical Task Network (HTN) Planning

Given an initial world description, goal conditions, and actions, a planning problem is to find a plan leading from an initial start state to the goal. Hierarichal Task Network(i.e., HTN) Planning is among several planning techniques proposed in artificial intelligence, which fits well with domains consisting of low level actions and high level tasks. High level tasks are hierarchically refined into lower level tasks and finally into actions. HTN

planning consists of a planning domain, planning problem, and an output plan [29]. Figure 2 shows a simple example of an HTN domain [24] and initial states; in this example the goal is traveling between home and park. As shown in the figure, the initial task (i.e., travel(me, home, park)) is decomposed into two sub-tasks (i.e., travel-by-foot, and travel-by-taxi) and finally each sub-task is refined into some actions.

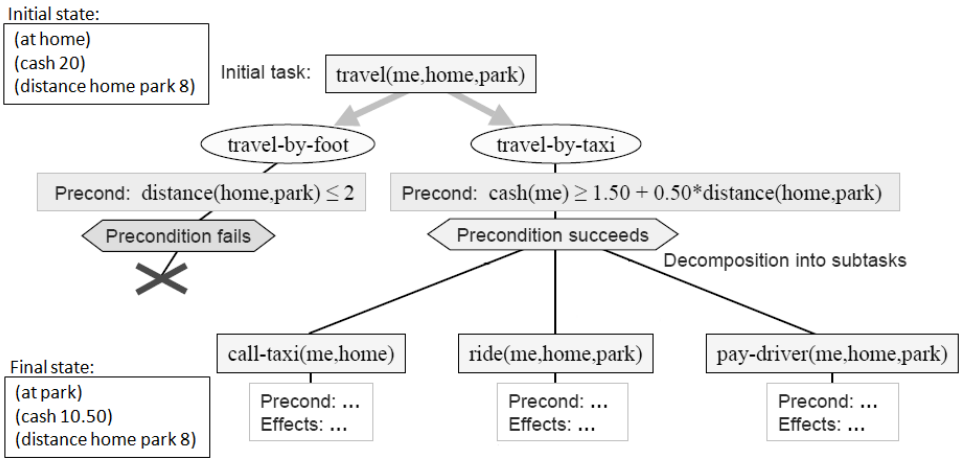


Figure 2: An example of HTN domain [24]

An *operator* (denoted as o) represents a low level action (e.g., ride(me, home, park) in Figure 2), which can be executed in the domain and is defined as a quintuple $o = (name(o), pre(o), eff(o), del(o), value(o))$. $pre(o)$ defines a pre-condition (e.g., $cash(a) > 1.50 + 0.50 \times distance(x, y)$ as a precondition for operator pay-driver) for every operator, which represents required conditions for performing the operator. The effect of performing the operator (e.g., $cash(a) \leftarrow cash(a) - 1.50 + 0.50 \times distance(x, y)$ as an effect of performing operator pay-taxi) can also be represented by using a post condition $eff(o)$. $del(o)$ or operator's delete list shows what becomes false after performing the operator. For every operator, an optional value $value(o)$ can be defined, which shows a required cost for the execution of that operator. The total value of an output plan is the sum of the values of the operators in the plan.

The *task* construct (denoted as t) represents higher level activities in HTN and can recursively be decomposed into the lower level tasks, and finally operators. In HTN, only operators can be executed and tasks can only be reduced into sub-tasks and operators [24]. Refinement of a task into sub-tasks is done using one or more methods (denoted as m_t) corresponding to the task. So, every method defines how a task is decomposed into lower level tasks or operators. A method is a quadruple $M = (name(m), task(m), pre(m), dec(m))$ where $task(m)$ is a parent task, $pre(m)$ is a pre-condition, and $dec(m)$ is a list of sub tasks into which the parent is decomposed. A precondition $pre(m)$ defines a required condition for decomposing the parent task. A method is applied only when its precondition is satisfied [29]. For example, method travel-by-taxi can be defined as following:

- *task*: travel(a, x, y)
- *precond*: $cash(a) > 1.50 + 0.50 \times distance(x, y)$

- *subtasks*: call-taxi(a, x) → ride(a, x, y)→ pay-driver(a, x, y)

The planning problem describes characteristics of a required plan - the objective, initial state, and constraints. As a result of applying a planning technique, a plan containing a sequence of actions that satisfies the objective and the constraints defined in the planning problem is produced. The HTN planning formulates the plan by recursively decomposing the tasks into sub tasks until it reaches the primitive tasks, which can be performed [24].

2.2. Multi-Criteria Decision Making

There are several multi-criteria decision making algorithms including: 1) the weighted sum model (WSM) which is the earliest method [30]; 2) the weighted product model (WPM) [31] which is a modification of the WSM; 3) and analytic hierarchy process (AHP) which is a later development and widely used in several applications [32]. AHP enables ranking objects based on the defined relative importance between them; the study by Karlsson et al. [33] for comparing six prioritization techniques revealed that AHP was the most promising technique as it provides the most trust worthy results; and the study in [6] shows that AHP is easy to use and does not need too much effort from stakeholders.

In AHP, stakeholders defines their preferences over each pair of decision objects. The preferences are expressed in terms of relative importance between the objects. According to stakeholders relative importance on decision objects, a square matrix $RI_{|N|\times|N|}$= $\{RI[i,j] = \alpha \mid 1 \leq i,j \leq |N|$, α is relative importance of object i to object j, and $|N|$ shows the number of objects on which the stakeholder defines the relative importance $\}$ is created. The value of α (i.e., degree of importance) in each cell may be 1, 3, 5, 7, or 9 corresponding to equality, slight value, strong value, very strong or extreme value, respectively [34]. According to the AHP algorithm, the elements of the matrix RI are normalized by dividing the elements of each column by the sum of the elements of that column. In order to estimate the principal eigenvector, the averages of normalized column are calculated. The principal eigenvector for each matrix, when normalized, becomes the vector of priorities for that matrix [17].

Fuzzy Cognitive Maps (FCM), proposed by Kosk [18], is an extension to cognitive maps which incorporates fuzzy causal functions to represent fuzzy relations among objects in a complex system. FCM handles dependency relations between options considering the weight on the dependency relations. FCM have been widely applied for several decision making areas [21, 22, 23] with dependencies and feedback over decision options. A FCM is a graph which consists of nodes representing the objects in the system and wighted edges illustrating cause-effect relations between system objects. The values in FCM are fuzzy; hence objects (nodes in graph) receive the values in [0, 1] and weights of the relations take a range of [-1, 1]. Figure 3 shows a sample of fuzzy cognitive map where o_i shows object i in the system and e_{ij} illustrates the weight of cause-effect relations between object i and object j. We define fuzzy cognitive map as the following [21, 18]:

Definition 2.1. ***(Fuzzy Cognitive Map)****. A fuzzy cognitive map is a quadruple FCM = (N, E, f, V) where 1) N is a set of objects in the system; 2) E is a matrix representing weights between objects in the system; 3) f is hyperbolic-tangent threshold function defined as* $f(x) = (1 - e^{-x})/(1 + e^{-x})$*; 3)* $V = \{V^t\}$ *is a set of state matrices where* V^0 *is matrix showing initial values of the objects and* V^{t+1} *is computed as* $V^{t+1} = f(V^t + V^t \times E)$

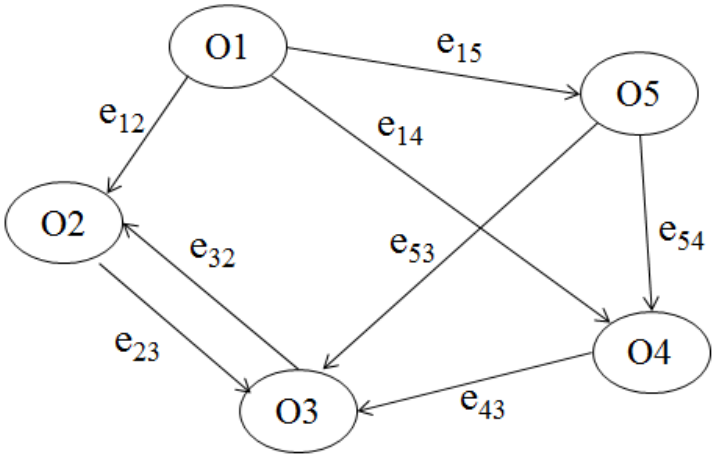


Figure 3: A simple fuzzy cognitive map.

3. Detail Definition of Configuration Problem

Given a product line is developed for an entire domain, different products could potentially satisfy diverse functional and non-functional requirements. An important challenge in product lines is how to efficiently reuse the assets developed in the domain engineering lifecycle to derive a product which satisfies both functional and non-functional requirements. This challenge is referred to as the *configuration problem*. More precisely, in the context of feature oriented software product line engineering, the configuration problem is defined as:

Definition 3.1. ***(Configuration Problem)****.Given a feature model and a set of target application requirements, what set of eligible features should be selected in order to optimally satisfy the requirements of the target application?*

According to the definition, feature models and requirements form the input of the configuration problem. The solution to the configuration problem should not violate the constraints imposed by the feature model and at the same time should conform, as much as possible, to the functional and non-functional requirements. This section provides an understanding of the main components of the configuration problem and the way that these components are presented in this paper. With respect to the requirements, we concentrate on non-functional requirements. We first develop a model of non-functional properties based on the existing literature. We then discuss how the non-functional model can be integrated into feature models and used for defining stakeholders' requirements. We should point out that we do not claim that the given definition of the configuration problem is complete, but it is the basis for the proposed configuration approach described in Section 4.

3.1. Non-Functional Properties and Interdependencies

Diversity of non-functional properties and their values in different products have been less center of attention in existing software product line approaches. Recently, a number of studies [5, 13, 6] investigated the notion of non-functional properties in the context of feature-oriented software product lines. Also, non-functional properties were investigated

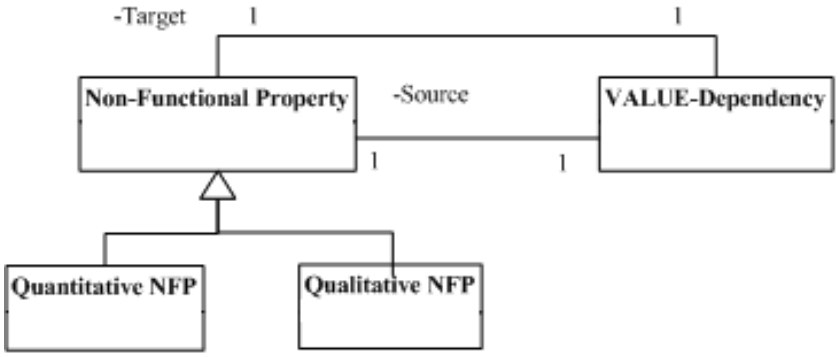


Figure 4: Non Functional Property Model

in the context of other variability modeling languages such as the Orthogonal Variability Model (OVM) [35]. Based on the existing knowledge, we have developed a non-functional property model (see Figure 4) which is used in this paper. In the developed model, based on the nature of non-functional properties, we categorize them into two main categories: *Quantitative* and *Qualitative*.

A *quantitative* property is countable, measurable, or comparable and can be shown as a numeric value [36]. Cost, Performance, and Memory usage are examples of quantitative properties. Metric based values are defined for quantitative non-functional properties and can be measured for a product. For example, performance can be measured for and assigned to a specific feature.

After measuring a non-functional property value for features, the NFP value of a product is computed by aggregating the NFP value of features involved in the product. Based on the nature of non-functional properties, different aggregation functions can be applied [35, 37, 38]. An aggregation function is a mathematical function such as: mean, max, min, summation, multiplication, to name a few. For example, to compute the non-functional values of a product, for some non-functional properties such as cost and response time, the values are summed; while for others like availability and reliability values need to be multiplied. Table 1 adapted from [35] shows a set of quantitative non-functional properties and their corresponding aggregation functions.

Table 1: Quantitative properties and corresponding aggregation functions

NFP	Unit	Aggregation function
Response Time (q_{rt})	msec	$\sum_{i=1}^{n} q_{rt}(f_i)$
Cost (q_c)	$	$\sum_{i=1}^{n} q_c(f_i)$
Availability (q_{av})	percent	$\prod_{i=1}^{n} q_{av}(f_i)$
Accuracy (q_{ac})	percent	$\prod_{i=1}^{n} q_{ac}(f_i)$

Qualitative properties are the second category of non-functional properties which cannot be exactly measured [6]. The qualitative non-functional properties such as *customer satisfaction* or *user friendliness* can be described using an ordinal scale consisting of a

set of predefined qualitative values which we refer to as qualifier tags. For example, *High negative*, *Medium negative*, *Low negative*, *Low positive*, *Medium positive*, and *High positive* can be possible qualifier tags defined for *customer satisfaction*. A qualifier tag represents a possible impact of functionality on a qualitative non-functional property.

Non-functional properties are not independent and they affect each other in a complex way [8]. Unlike functional interdependencies, non-functional interdependencies influence a large part of the functionality or other non-functional properties [28]; hence, it is crucial to take into consideration the non-functional interdependencies. Furthermore, non-functional interdependencies increase the complexity of selecting and deselecting features in the software product line engineering.

An interview study by Brentsson et al. [28] showed that one of the most common inter-dependencies between non-functional properties in a business to consumer domain is value related dependencies. A value related dependency means that satisfying a non-functional property affects the value of another non-functional property for the customer [8]. Value related dependencies are further refined into *positive* and *negative* values. For example, satisfying international sale non-functional property improves customer satisfaction while satisfying security has negative impact on the performance. Figure 6 shows dependency relations in the non-functional model which is defined between a source non-functional property and a target non-functional property.

3.2. Extended Feature Model

Having defined the non-functional model in the previous section, we now define one of the components of configuration problem, i.e. the extended feature model.

The core notion in feature models is the *feature*. Bosch et al. [3] defined a feature as "a logical unit of behavior specified by a set of functional and non-functional requirements." We consider this definition as we intend to configure the feature model based on both functional and non-functional requirements. A feature model provides a formal and graphical representation of features as well as the variability relations, constraints, and dependencies defined over the product lines' features. It has a tree-like structure [39] in which features are typically classified as:

- **Mandatory feature:** if a parent feature is selected, its mandatory child feature must be also selected in the configuration process.
- **Optional feature:** if a parent feature is selected, its optional child feature may or may not be selected in the configuration process.
- **OR feature group:** one or more features in the OR feature group must be selected in the final configuration of the feature model.
- **XOR feature group:** one and only one of the features in the XOR feature group must be selected in the final configuration of the feature model.

In addition to the relations between a parent feature and its child features, a number of relations are defined to represent mutual inter-dependencies (also referred to as *integrity constraints*) between features. The two most widely used integrity constraints are [39]: *requires*-the presence of a given (set of) feature(s) requires the inclusion of another

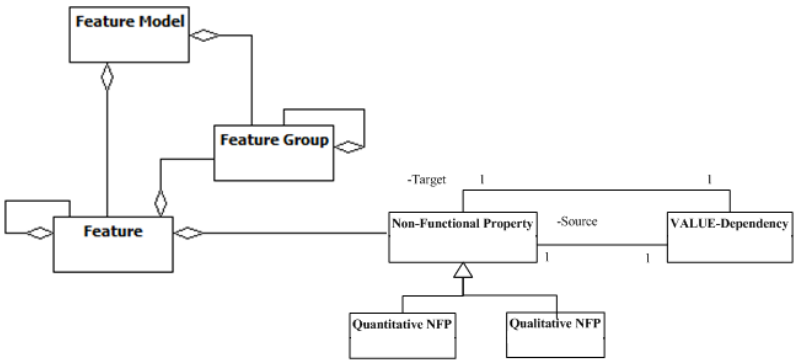


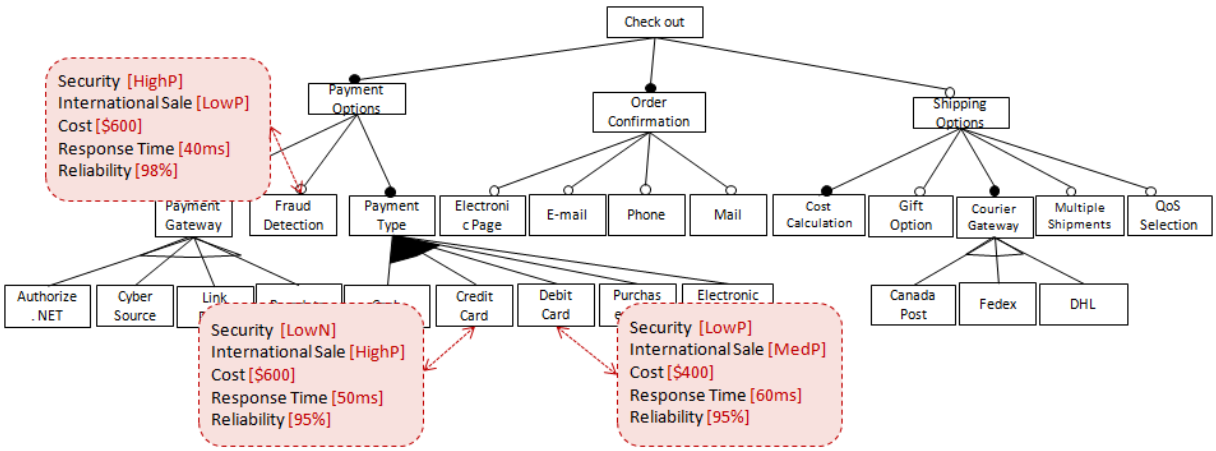
Figure 5: Integration of Feature Model with Non-Functional Property Model

(set of) feature(s); and *excludes*-the presence of a given (set of) feature(s) requires the exclusion of another (set of) feature(s). Batory [40] argued the need for more complex integrity constraints and generalized the constraints into propositional formulas defined over features. In our work we adopt Batory's view over integrity constraints. In feature model tree, features without any children called *atomic* (or leaf) features and features which are decomposed into sub-feature(s) called *non-atomic* (or intermediate) features.

In feature oriented software product line, feature models are mainly used for representing functional variability between different products. Some researchers have extended the feature model notation with non-functional properties to represent the non-functionality aspect [5, 13]. Similarly, we extended the feature model with the notion of non-functional properties which can be either qualitative or quantitative. In this model each feature can be annotated with several non-functional properties and corresponding values of those properties. We should note that our extension is similar to existing works [5, 13] and can be easily rewritten by such notation [5, 13]. Figure 5 shows the extended feature model representing non-functional properties as well as functional properties (i.e. features) and their relationships.

Figure 6 shows non-functional properties employed in the on-line shopping product line. *Response time*, *cost*, *availability*, and *reliability* are quantitative non-functional properties and c*ustomer satisfaction*, *international sale*, and *security* are the qualitative non-functional properties. In our approach, we assume that atomic features (i.e., leaf features) in a feature model have concrete implementations. Non-atomic (i.e., non-leaf) features are used for variability and composition relationships of the atomic features. Hence, non-functional properties are defined for the leaf features. If an intermediate feature contains implementations and non-functional properties, we create a mandatory child feature for the intermediate feature and assign the non-functional properties to the child feature. After identifying domain features, developing a feature model and implementing its atomic features, we can then analyze the impact of features on non-functional properties and annotate them with proper qualifier tags.

For qualitative non-functional properties such as *security*, *international sale*, based on existing domain knowledge, the impact of each feature on non-functional properties can be identified and proper qualifier tags can be assigned to each feature's non-functional properties. On the other hand, quantitative non-functional properties for the features



Features \ NFP	Security	International Sale	Customer Satisfaction	Response Time	Cost	Availability	Reliability
Credit Card	LowN	HigP	MedP	50ms	$600	90%	95%
Debit Card	LowP	MedP	MedP	60ms	$450	90%	95%
Fraud Detection	HighP	LowP	HighP	40ms	$600	80%	98%
⋮							

Figure 6: Integrated Feature Model with Non-Functional Properties

can be measured using a suitable metric and assigned to the features. We assume that some techniques, like those proposed in [41], can be employed to measure non-functional properties for each feature.

For example, as shown in Figure 6, feature Credit Card is annotated with *low negative security*, *high positive international sale*, and *medium positive customer satisfaction* and its estimated *cost*, *response time*, and *availability* are \$600, 50ms, and 90%, respectively.

3.3. Stakeholders' Requirements (Preferences and Constraints)

In this section we explain the other component of the configuration problem, i.e. stakeholders' requirements. More specifically, we provide more detailed explanation of the non-functional requirements. We divide the non-functional requirements into preferences (i.e. stakeholders' priorities with respect to non-functional properties) and constraints over non-functional properties.

Various stakeholders may have different priorities over non-functional properties. For some stakeholders, a subset of non-functional properties may be more important and relevant than the other non-functional properties. Moreover, in some situations, a conflict may arise between requirements defined over non-functional properties that have an opposite behavior, in case of which the stakeholder needs to choose between the competing options. One approach for specifying the priority between non-functional properties is through formalizing their relative importance relations. Relative importance is defined as follow [27].

Definition 3.2. ***(Relative Importance)***. *Relative importance between non-functional properties a and b is: $a \succ^{\alpha} b$ if non-functional property a is more important than non-functional property b with coefficient α.*

Usually, the degree of importance of options (non-functional properties) is represented using values 1, 3, 5, 7, and 9 corresponding to equality, slight value, strong value, very strong and extreme value, respectively [34]. For example, relation *Security* $\succ^3$ *Performance* represents that *security* is slightly more important than *performance*. We refer to the relative importance between non-functional properties as stakeholders' preferences. Prioritizing the stakeholders' preferences, formalized in terms of relative importance of non-functional properties, can have a dramatic impact on the design of the system.

Stakeholders may also define constraints over non-functional properties of a system. For example, a stakeholder may define a constraint that an on-line shopping system has to be at least medium secure and cost less than $1000. Thus, the final system should not have any functionality (i.e., feature) that provides positive impact on security less than the medium level. We assume that a product has some level of qualitative non-functional property (e.g., medium security), if all of its features have at least that level of non-functionality (i.e., all the features must have at least medium security).

4. Automated Feature Model Configuration

In this section, we explain the details of our proposed configuration approach which helps application engineers select features based on their requirements. Our basic idea is to employ planning techniques for solving the configuration problem. To this end, we transform feature models into hierarchical task network formalisms and use a HTN planner to automatically select proper features based on stakeholders' requirements. We also describe how non-functional properties can be optimized during the configuration process, according to the needs of the stakeholders.

4.1. Computing Features Ranks Based on Stakeholders' Preferences

Configuring a feature model based on the stakeholders' requirements and preferences usually means selecting features such that a feature model configuration satisfies the stakeholders' functional requirements and constraints and optimizes their preferences. To optimize the configuration with respect to the preferences, feature ranks must be computed based on their impact on the non-functional properties which may be of different importance for the target stakeholders. Additionally, not only stakeholders' preferences, but also interdependency between non-functional requirements must be considered in the computation of feature ranks. The proposed approach for computing features ranks consists of two main stages: *computing non-functional weights considering stakeholders' preferences and interdependencis between non-functional requirements*; and *computing the ranks of features based on the assigned non-functional properties to features.*

4.1.1. Computing Weights of Non-Functional Properties

To calculate the ranks of features based on relative importance of non-functional properties assigned to the features, we propose a technique which combines Analytical Hetwork Process (AHP) proposed by Saaty [34] and Fuzzy Cognitive Map [18, 42]. The technique consists of:

1. Calculating local weights of non-functional properties by employing AHP over stakeholders preferences;

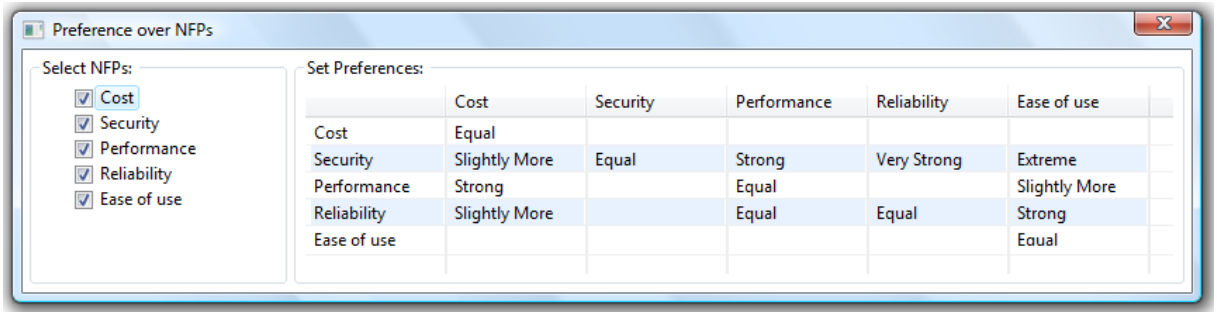


Figure 7: Non-functional properties prioritization

2. Computing the impact of non-functional properties by employing fuzzy cognitive maps on interdependency relations between non-functional properties;
3. Computing the global weights of non-functional properties by integrating the local weights and impacts of non-functional properties.

Computing Local weight of Non-functional Properties. As mentioned in Section 3.3, the stakeholders' preferences are in the form of relative importance over non-functional properties. After the stakeholders specify their preferences, RI matrix and RI matrix with normalized values are created using AHP algorithm. Next, the weight of non-functional properties is calculated based on eigenvalues.

As an example, suppose that a stakeholder defines the relative importance as shown in Figure 7; then, RI matrix and RI matrix with normalized values are created (see Table 2 and Table 3). The final local weights of NFPs are shown in the last column of Table 3.

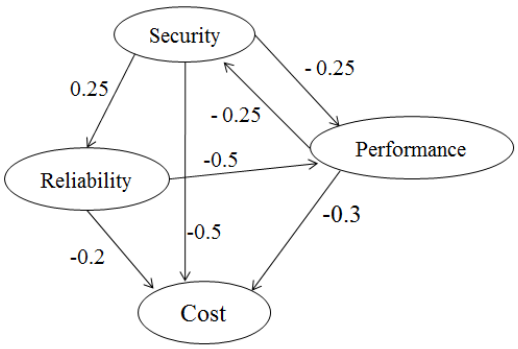
Table 2: Relative importance of non-functional properties.

	Cost	Security	Performance	Reliability	Ease of use
Cost	1.00	0.33	0.20	0.33	3.00
Security	3.00	1.00	5.00	7.00	9.00
Performance	5.00	0.20	1.00	1.00	3.00
Reliability	3.00	0.14	1.00	1.00	5.00
Ease of use	0.33	0.11	0.33	0.20	1.00

Table 3: Normalized values of relative importance of non-functional properties.

	Cost	Security	Performance	Reliability	Ease of use	Sum	importance(Z)
Cost	0.08	0.19	0.03	0.03	0.14	0.47	0.12
Security	0.24	0.56	0.66	0.73	0.43	2.63	0.66
Performance	0.41	0.11	0.13	0.10	0.14	0.90	0.22
Reliability	0.24	0.08	0.13	0.10	0.24	0.80	0.20
Ease of use	0.03	0.06	0.05	0.02	0.05	0.20	0.05

Computing the impacts of non-functional properties interdependencies. As we mentioned in section 3.1, value related interdependencies are the most common dependency types



(a)

	Security	Performance	Reliability	Cost	Ease of use
Security	0	-0.25	0.25	-0.5	0
Performance	-0.25	0	0	-0.3	0
Reliability	0	-0.5	0	-0.2	0
cost	0	0	0	0	0
Ease of use	0	0	0	0	0

(b)

Figure 8: Non-functional properties value related interdependencies. (a)Graph Representation (b) Matrix Representation

between non-functional properties. According to this kind of dependency, an increase or decrease in the value of a non-functional property may lead to increase or decrease of another non-functional property. For example, increasing the value of security in the system causes increase in the value of cost and decrease in the value of performance of a system. Figure 8 shows the interdependency graph whose nodes are non-functional properties and weighted edges are the influences between non-functional properties. The weights are assigned by domain engineers in collaboration with domain experts.

Since non-functional requirements depend on each other and affect the other non-functional requirements, the preferences on non-functional requirements cannot be the only basis for selecting features [20]. Hence, interdependencies between non-functional requirements must be considered in the configuration process. Similar to [21], we formulate dependency graph as a a fuzzy cognitive map and apply proposed techniques in the fuzzy cognitive map [18] to compute the overall influence of the non-functional properties on the other non-functional properties.

In order to compute the overall influence of objects on each other, first the formula[1] given in the definition 2.1 is applied until the state matrix coverage to a fixed point situation (when state vector values do not change for successive iteration) or a limit state cycle (the sequence of the state vector keeps repeating indefinitely) [21]. The final matrix is called steady state matrix. Afterward, according to [21], the steady state matrix (V^{t+1}) is multiplied to a vector which shows the local weights of objects to compute the overall influence of objects on the other objects in the system.

To compute the influence of non-functional properties on each other, we consider that N is a set of non-functional properties and E shows the weights of non-functional properties on each other in which:

- $e_{ij} > 0$ indicates that increase (decrease) in the value of non-functional property n_i leads to an increase (decrease) in the value of non-functional property n_j.
- $e_{ij} < 0$ indicates that increase (decrease) in the value of non-functional property n_i

[1]The formula is a basic formula in the fuzzy cognitive map to compute the state steady matrix. The proofs and details of this formula can be found in [18].

leads to an decrease(increase) in the value of non-functional property n_j.

- $e_{ij} = 0$ indicates no relationship between non-functional property n_i and non-functional property n_j.

V^0 is the initial state of non-functional properties. Since we aim at computing the impact of interdependencies, the initial state matrix is set to identity matrix(i.e. $V^0 = I_{n\times n}$). After calculating steady matrix V^*, the matrix is normalized by dividing each element in the matrix to the largest row sum of V^* [21].

Table 4: Normalized values of the non-functional properties dependency graph.

tanh(x)	Security	Performance	Reliability	cost	Ease of use
Security	-0.4391	0.0362	-0.2915	0.2332	0.0
Performance	0.2181	-0.2332	0.0362	-0.0291	0.0
Reliability	-0.1295	0.0574	-0.4592	0.0072	0.0
cost	0.0	0.0	0.0	0.0	0.0
Ease of use	0.0	0.0	0.0	0.0	0.0

Now for computing the overall influence of non-functional properties(T), similar to [21], we multiply the steady matrix (V^{t+1}) to local the weights of non-functional properties (Z).

$$T = V^t \times Z = \begin{Bmatrix} Security & -0.1558 \\ Performance & -0.0318 \\ Reliability & -0.0320 \\ Cost & 0.0463 \\ Ease\ of\ use & 0.0000 \end{Bmatrix}$$

Computing global weight of NFPs. After computing the weight of non-functional properties (Z) using AHP and computing the influence of each non-functional property (E) using fuzzy cognitive maps, we can compute the overall weights of non-functional properties based on the following formula [21].

$$W = Z + T$$

where W is final weights of non-functional properties; Z is normalized local weights computed by AHP; and T is the overall influence of non-functional properties. The final local weights of NFPs for the given example are as following:

$$W = Z + T = \begin{Bmatrix} Security & 0.5042 \\ Performance & 0.1882 \\ Reliability & 0.0880 \\ Cost & 0.1663 \\ Ease\ of\ use & 0.0500 \end{Bmatrix}$$

4.1.2. Utility Function for Calculating Features' Rank

To calculate the rank of features, both qualitative and quantitative properties should be taken into account. To consider qualitative non-functional properties in feature ranks, they should be mapped to the real values. The first way is that, the stakeholders or application engineers provide a mapping function from qualifier tags onto real values. For example, for *customer satisfaction*, one can define the following mapping function.

$$M_{customerSat.}(QT) = \left\{ \begin{array}{c} -1 \quad High\ negative \\ -0.5 \quad Medium\ negative \\ -0.25 \quad Low\ negative \\ 0.25 \quad Low\ positive \\ 0.50 \quad Medium\ positive \\ 1 \quad High\ positive \end{array} \right\}$$

The other way for calculating the corresponding real-numbers for the qualifier tags of a qualitative non-functional property is to again use the AHP algorithm. In this way, the stakeholders specify the relative importance between the qualifier tags of each non-functional property and AHP calculates the rank of each qualifier tag. The detail of calculating weight of options using AHP is explained in the Section 4.1.1. For example, the stakeholder specifies that for the *international sale* property, *High positive* $\succ^3$ *Medium positive*, which means a feature with high positive impact on *international sale* is slightly more important than the feature with medium positive on international sale. In both the methods (i.e., both the mapping function and AHP), we assume the values are normalized in the $[-1, +1]$ range.

After defining the ranks of non-functional properties and the mapping function for the qualitative non-functional properties, we describe the following utility function to calculate the ranks of each feature based on an extension to [43]. The proposed utility function calculates the rank of features based on their impact on non-functional properties by considering the preferences of stakeholders formulated in terms of the weight of non-functional properties.

Definition 4.1. ***(Utility function)***. *Let us assume there are α quantitative NFPs to be maximized, β quantitative non-functional properties to be minimized, and θ qualitative non-functional properties whose impact needs to be maximized. The utility function for feature f is defined as:*

$$R(f) = \sum_{i=1}^{\alpha} w_i \times \frac{q_i(f)-\mu_i}{\sigma_i} + \sum_{j=1}^{\beta} w_j \times (1 - \frac{q_j(f)-\mu_j}{\sigma_j}) + \sum_{k=1}^{\theta} w_k \times M_k(QT(f))$$

where w is the weight of each non-functional property calculated by S-AHP such that $0 \leq w_i, w_j, w_k \leq 1$ and $\sum_{i=1}^{\alpha} w_i + \sum_{j=1}^{\beta} w_j + \sum_{k=1}^{\theta} w_k = 1$ and $M_k(QT(f))$ returns the real number corresponding to quality tags of k-th non-functional property. μ and σ are the average values and the standard deviation of the quantitative non-functional properties for all atomic features in the feature model.

The overall rank of a product is calculated by aggregating the ranks of selected features for the product. The aggregation function used for calculating the product rank

depends on the aggregation functions which exist over non-functional properties of a feature. As discussed in Section 3.1, some quantitative non-functional properties such as *response time* are additive and the quality of a composition of features is computed by adding up the quality of features [44][37][38]. The second type of aggregation functions defined on quantitative non-functional properties is multiplication where the quality of composition is calculated by multiplying the quality of features involved in the composition. The multiplication type can be converted into an additive type by computing the logarithm values of non-functional values. For qualitative non-functional properties, a qualifier tag assigned to a feature represents a qualitative impact of the feature on the non-functional property. Considering the mapping function that maps the qualifier tags into real numbers, we can calculate the overall impact of a composition of features by adding the impacts of the features involved in the composition. Hence, the aggregation function over the utility functions of features is an additive function and the overall rank of a product is computed as $\mathbb{R} = \sum_{f_i \in P} R(f_i)$. In order to derive an optimal configuration (product) P, we need to select the features, which maximize $\mathbb{R}(P)$.

4.2. Automated Feature Model Configuration Based on HTN planning

Having annotated a feature model with NFPs, the process for deriving a new product starts by selecting and deselecting features based on the stakeholders' requirements reflected through desired features and preferences expressed in terms of relative importance between NFPs. The configuration problem is concerned with selecting features that satisfy the functional requirements (i.e., the requested functionality) and constraints and optimize the preferences.

To automate the configuration process, we define transformation rules to convert a configuration problem to a planning problem. To do so, we develop transformation rules to represent extended feature models in the HTN formalism. The transformation is done in two steps: 1) generating an HTN domain model from a feature model, and 2) generating a planning problem from a configuration problem.

During transformation we need to convert the maximization problem into a minimization problem, because the SHOP2 planner, used in our implementation, works on minimization only (Negating the ranks of features can do this).

4.2.1. Transforming Feature Model into HTN

Considering the analogy between tasks in HTN and features in a feature model, there is a need to define task decompositions to reflect feature relations. The HTN *method* element is used to define decomposition of tasks into sub-tasks or operators. A method contains the parent task, a list of children, and a precondition. Here, we have two options for decomposing a task into sub-tasks:

1. we can define different methods with a common parent task. In this case, an HTN planner (i.e., SHOP2) for decomposing a task into sub-tasks selects just one method for each plan. Hence, this option is suitable for mapping the XOR relation to HTN;
2. we can also define one method for a decomposition in which all children can be considered a list of sub-tasks in the method. In this case, the HTN planner performs the method if and only if the preconditions of all sub-tasks or operators are satisfied. We can then use this option for defining AND-decomposition with mandatory child features.

According to abovementioned, in HTN, a *method* does not support OR-decomposition (i.e., selecting one or more tasks from sub-task list). Moreover, in HTN the concept of *optional* tasks or operators is not defined; that is, in the sub-task list of a method, we can have just mandatory features. In other words, OR group and optional features cannot directly be transformed into HTN formalism. Hence, before transforming a feature model into an HTN domain, a preprocessing step should be done to replace optional and OR features. To this end, similar to [19] for optional goals, every optional feature f_o is replaced with a new feature f_p which is decomposed into two alternative features f_o and f^d where f^d stands for a "dummy" feature (i.e., an operator without any effect) (Figure 9a). Regarding OR-decomposition, every OR group in the feature model is converted into a set of optional features and a mandatory feature (f_{or}) with AND relations between them (Figure 9b). Feature f_{or} is added to ensure that at least one of the features in the OR group is selected (more details are explained in the "Generating Tasks and Methods" subsection). Consequently, an OR group can be easily transformed into HTN using a method with sub-tasks containing "dummy" features.

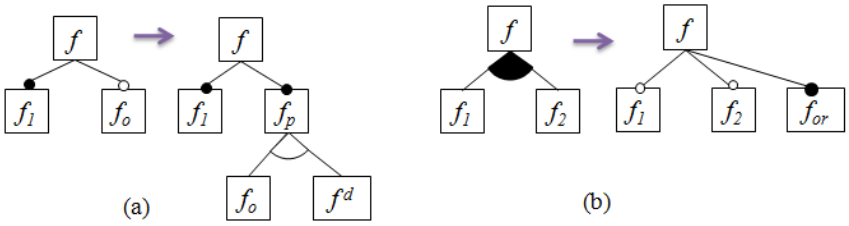


Figure 9: The preprocessing step for transforming feature models into HTN domains: (a) optional features (b) OR feature groups

After performing the above changes in the feature model, we transform the feature model into an HTN domain using the following three types of transformation rules:

- domain predicate transformation;
- operator transformation;
- method and task transformation.

Generating Domain Predicates. In the HTN planning domain, methods and tasks involve *logical expressions* which are the combination of *logical atoms* and *logical connectives*. *Logical atoms* involve a *predicate symbol* plus a list of *terms*. These logical expressions can be used as either precondition formulae or effect formulae. Based on possible preconditions for selecting and decomposing features, we can identify possible domain predicates.

In the feature model, we have two groups of preconditions including: 1) precondition for checking integrity constraints (i.e., requires and excludes); and 2) preconditions for checking the stakeholder's constraints with respect to non-functional properties. To generate these preconditions, we need domain predicates for each quantitative NFPs, qualifier tags of the qualitative NFPs, and atomic features in the feature model.

For example, the v_{secnh} and v_{secph} predicates are created for *high negative* and *high positive* qualifier tags of the *security* non-functional property, respectively; and v_{cost} predicates is created for *cost* NFP. These predicates can be involved in the logical expressions

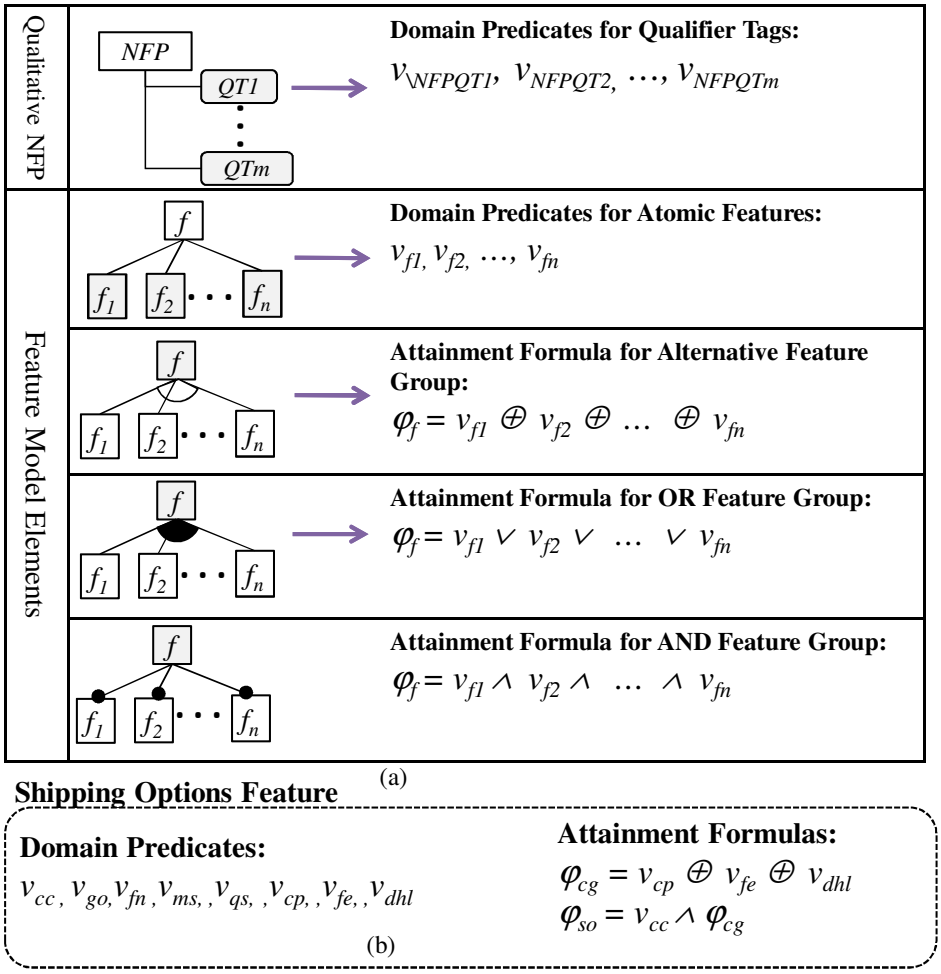


Figure 10: Generating Domain Predicates.(a) transformation rules, (b) generated domain predicates for Shipping Options feature.

to generate preconditions of methods and operator; using these preconditions we can check whether a feature covers a required NFPs or not. Moreover, for features Credit Card and Debit Card, we generate domain predicates v_{cc} and v_{dc}, respectively. For each non-atomic feature, a propositional formula (called *attainment formula* [19]) is created according to the features that exist in its sub-tree (Figure 10). For instance, in the on-line shopping feature model, which is shown in Figure 6, the attainment formula for feature Courier Gateway is $\varphi_{cg} = v_{cp} \bigoplus v_{fe} \bigoplus v_{dhl}$, which shows alternative relation between Canada Post, Fedex and DHL features. These formulae are later used as preconditions of the other features in the feature model in order to investigate the integrity constraints (requires and excludes relations among features).

Generating Operators. HTN operators represent the lowest level activities in the domain. Hence, in process of feature model transformation, HTN operators can be generated from atomic (i.e., leaf) features in the feature model tree. Each atomic feature f is transformed into an operator o_f and the rank of the feature calculated by the utility function is transformed into $value(o_f)$. A precondition is defined for each operator based on the non-functional properties and integrity constraints defined over its corresponding

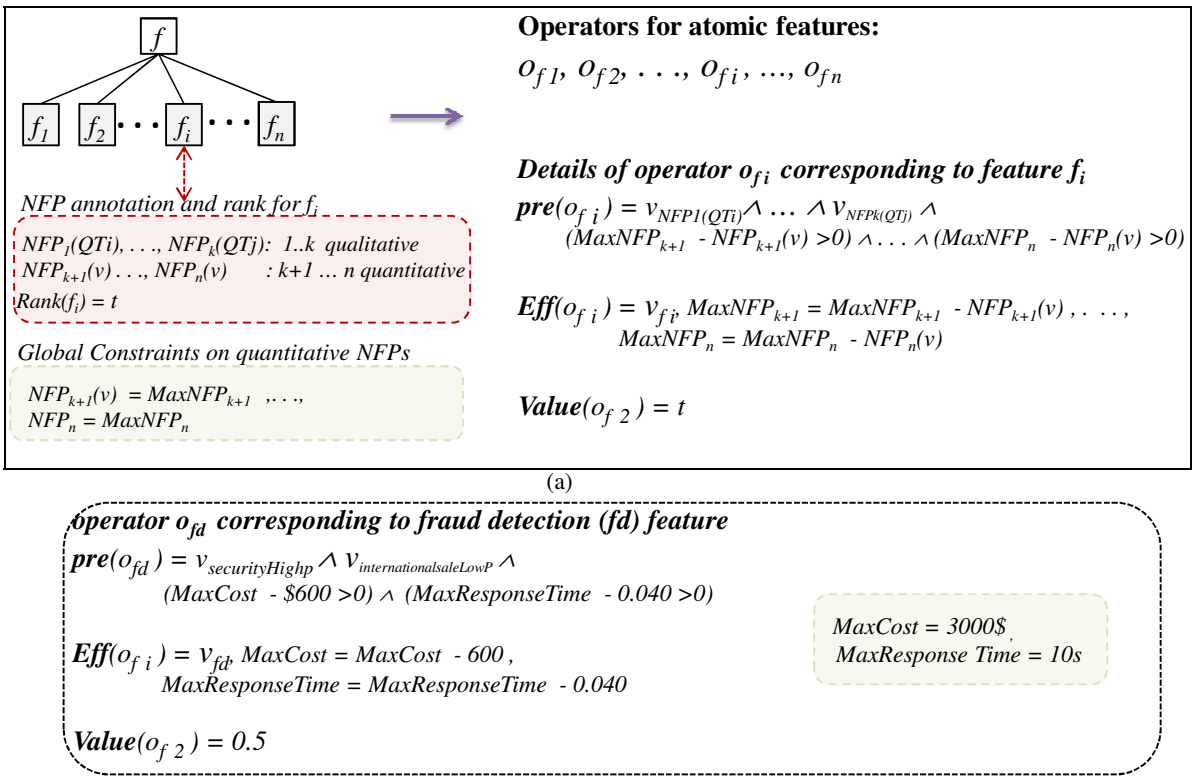


Figure 11: Translating Atomic features into Operators. (a) transformation rules (b) an example of generated domain predicates for fraud detection.

feature. The preconditions are defined as logical AND expressions of:

1. domain predicates corresponding to qualifier tags of qualitative non-functional properties with which the feature is annotated;
2. an evaluation expression to check whether the feature is allowable to be selected or not based on quantitative non-functional properties constraints;
3. a propositional formula which defines the selection constraints (i.e. integrity constraints) of the feature (see Figure 15).

Next, the domain predicate, which corresponds to feature f_i (i.e., v_{fi}) is added as an effect of the operator o_{f_i} (i.e., $eff(o_{fi}) = v_{fi}$). Whenever operator o_{f_i} is performed, its corresponding domain predicate becomes true (i.e., $v_{fi} = true$).

For handling quantitative non-functional property constraints, a logical expression is created showing the maximum available value of each corresponding non-functional property during the planning process. At the first, this value is set by the requested value of stakeholders and added to the initial state of the planning problem. For example, if a stakeholder has a limitation on *cost* (e.g. maximum $3000), we add a logical expression "($MaxCost$ 3000)" to the initial state of the planning problem. Then, in the effect of each operator, this value is updated based on the assigned non-functional property value to the feature. For example, in Figure 11b feature fraud detection has *cost* NFP with value $600; hence, after selecting this feature the maximum available *cost* should be reduced by 600.

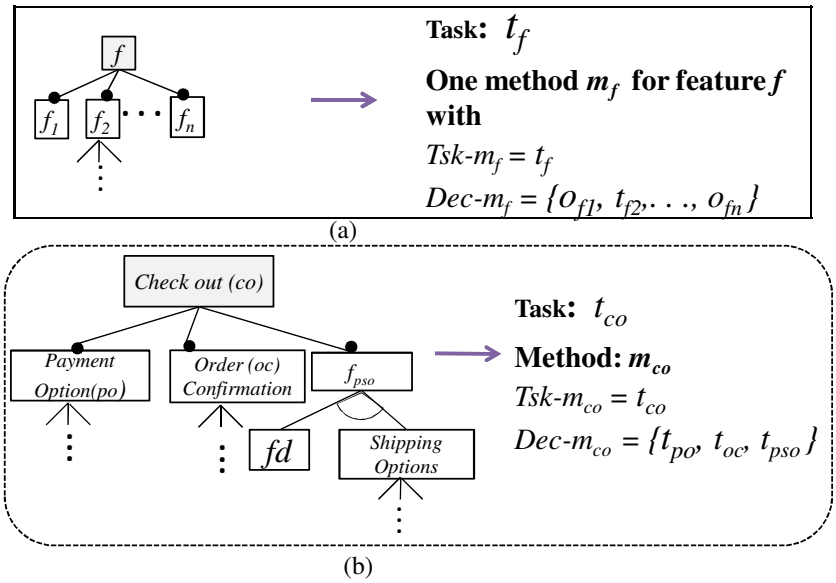


Figure 12: Generating tasks and methods from And Decomposition. (a) transformation rule, (b) example

Moreover, before selecting feature fraud detection we should check whether we have enough available *cost* or not. To this end, we add logical expression like "$(MaxCost - 600 > 0)$" to the precondition formula of feature fraud detection. The value of the maximum available *cost* is updated by adding "$(MaxCost = MaxCost - 98)$" to the effect formula of feature fraud detection.

Generating Tasks and Methods. *Tasks* in the HTN planning domain represent high level activities which need to be decomposed into smaller tasks using a *method*. A feature model also has a hierarchy structure in which features are decomposed into sub-features. Therefore, every non-atomic feature f can be mapped into a task t_f; and method(s) (m_f) can be defined for different types of decomposition in feature model (i.e., AND, OR, and XOR). To create tasks corresponding to features, we use the pre-processed feature model from which OR feature groups and optional features have been removed (c.f. Figure 9). Based on the type of a feature group(i.e., XOR group or AND-decomposition), we may define one or more methods. If a feature f is AND-decomposed into features f_1 to f_n, we define one method m_f which connects corresponding task t_f to tasks or operators corresponding to f_1 to f_n (Figure 12a). As shown in Figure 12a, $Dec - m_f$ is generated which includes the list of tasks and operators corresponding to sub-features of f. Figure 12b illustrates generated HTN formalisms for check out feature. t_{co} denotes the task corresponding to check out feature and $Dec - m_{co}$ denotes the task list of method m_{co} which contains tasks t_{po}, t_{oc}, and t_{pso} which are corresponding HTN representations for features payment option, order confirmation and shipping option, respectively.

For alternative feature groups with n sub-features (i.e., $f = XOR(f_1, f_2, .., f_n)$), n methods are defined with a common parent task t_f corresponding to feature f. Each method connects task t_f to one operator o_{f_i} or task t_{f_i} corresponding to the sub-feature f_i of the parent feature f (see Figure 13a). For example, as shown in Figure 13b, for the shipping gateway feature, a task t_{sg} and three methods m^1_{sg}, m^2_{sg}, and m^3_{sg} corresponding

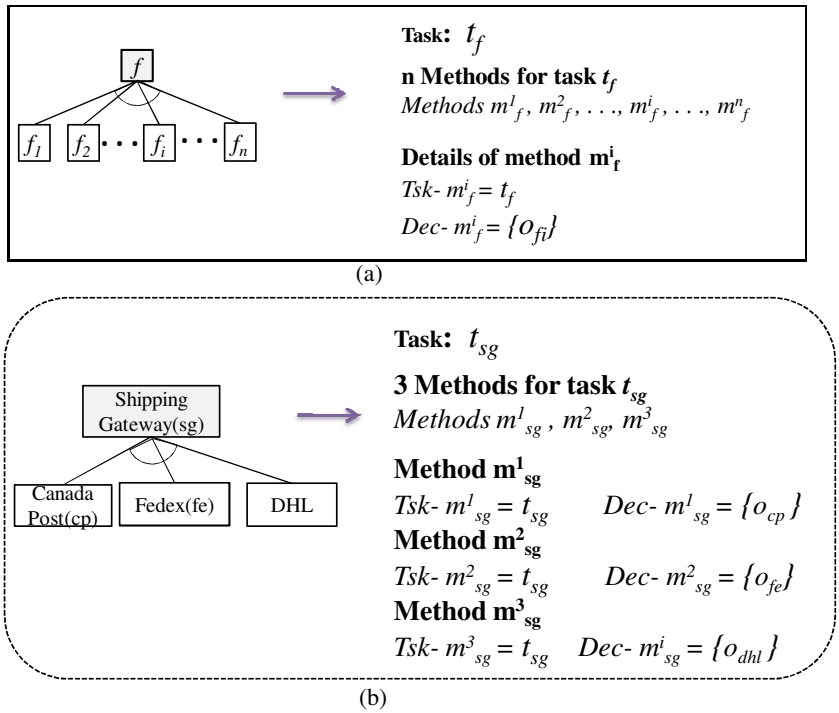


Figure 13: Generating tasks and methods from Alternative Decomposition. (a) transformation rule, (b) example

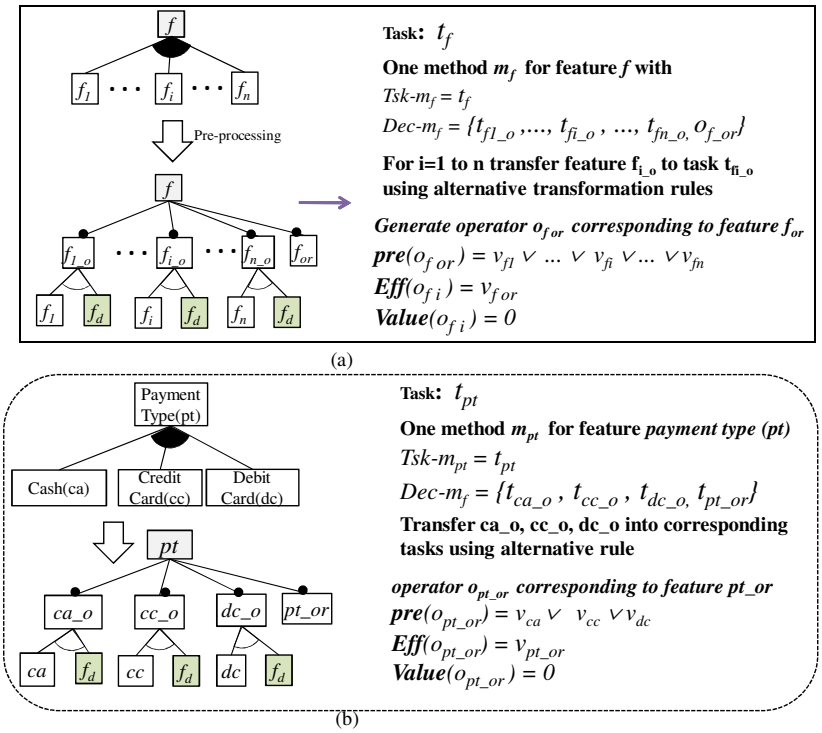


Figure 14: Generating tasks and methods from OR Decomposition. (a) transformation rule, (b) example

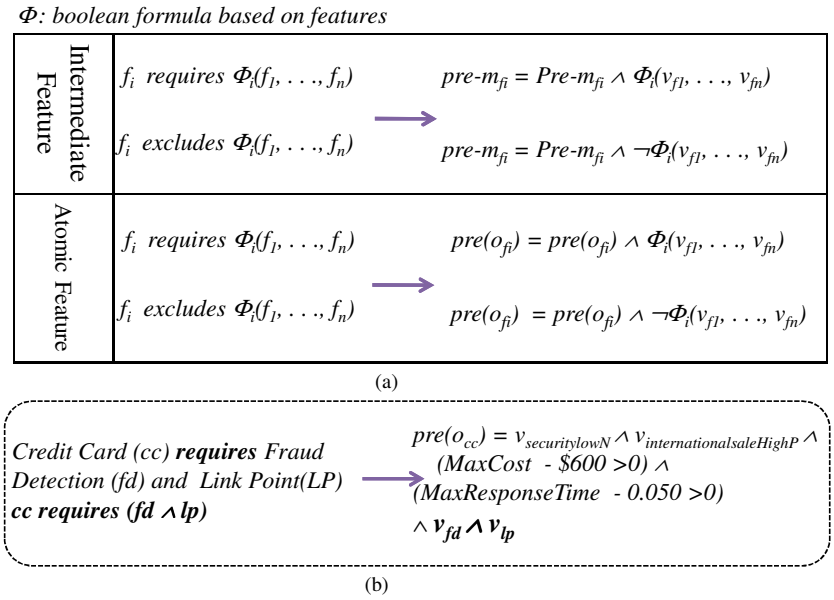


Figure 15: transformation for integrity constraints. (a) transformation rule, (b) example

to canada post, fedex, and DHL features are generated. Next, tasks lists $Dec - m^1_{sg} = o_{cp}$, $Dec - m^2_{sg} = o_{fe}$, $Dec - m^3_{sg} = o_{DHL}$ are produced which contain tasks corresponding to Canada Post, Fedex and DHL sub-features.

Regarding OR feature groups, in the feature model configuration at least one of features in the group must be selected if a parent feature is selected. For example, in Figure 14a if feature f is selected, at least one of its sub-features (i.e., f_1 to f_n) must be selected. To this end, we add a precondition for operator ($o_{f_{or}}$) which is a logical OR expression of attainment formula of the features in the OR feature group (see Figure 14a). Figure 14b depicts the generated HTN formalisms for the payment type feature.

Generating Preconditions from Integrity Constraints. methods and operators preconditions are the means to represent integrity constraints in the feature model. If atomic feature f *requires* a set of features f_1 to f_n, expressed via boolean formula $\phi(f_1, ..., f_n)$, precondition of operator o_f is extended with AND conjunction of $\phi(v_{f1}, ..., v_{fn})$. If atomic feature f *excludes* a set of features f_1 to f_n, expressed via boolean formula $\phi(f_1, ..., f_n)$, precondition of operator o_f is extended with conjunction of $\neg\phi(v_{f1}, ..., v_{fn})$. Figure 15b shows change in the precondition of operator o_{cc} due to constraints: credit card require fraud detection and link point. For intermediate features the precondition of corresponding methods are updated.

4.2.2. Planning Process

After defining the planning domain, the feature model configuration problem is transformed into the HTN planning problem. The transformation is done by considering the constraints over the NFPs and setting their corresponding domain predicates as true to form initial state. For example, if a stakeholder asks for at least *medium security*, and a specific *cost* (e.g., $1000), the domain predicates $v_{secph} = true$ and $v_{secpm} = true$ and the rest of qualifier tags are set to false; the logical atom "($cost1000$)" is also added as an initial state. Next, the set of required atomic features (F'_A) and the root of the feature

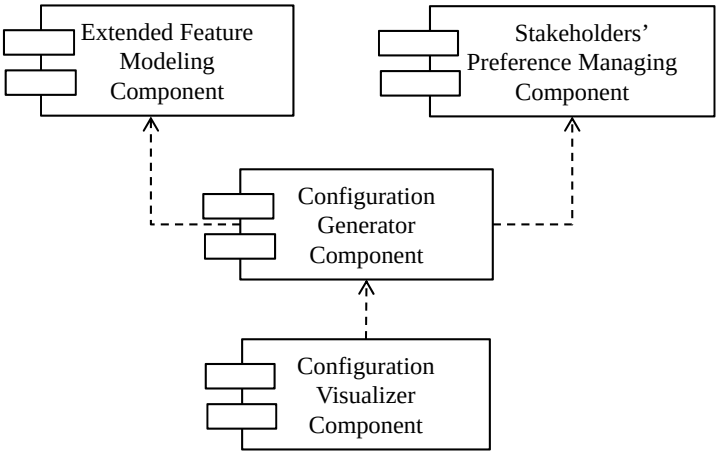


Figure 16: High level architecture of the *Vis-fmp*

model are translated into initial tasks of the planning problem.

The SHOP2 planner starts with the first task in the initial tasks, if the task is a *non-primitive* task, the planner chooses an appropriate method and instantiates it to decompose the task into sub-tasks; if the task is a *primitive* task, the planner chooses an applicable operator and instantiates it to perform an action. A solution plan is found if all constraints are satisfied, otherwise the planner backtracks and chooses the other methods. Consequently, the planner tries all possible paths in the feature model tree and returns the best features (defined as operators in the planning domain) in the feature model. Then, based on the selected atomic features in the produced plan, intermediate features are selected using a propagation algorithm in which all intermediate features are selected if they have a selected sub-feature(s).

5. Tooling Support

Tools play an important role in both domain and application engineering phases. Having an effective tooling support facilitates software developer tasks in those phases. Although, researchers have already contributed in developing tooling support for defining, managing, and configuring feature models [45][46], there is still the need for product configuration tooling support [47]. Moreover, one common capability lacking in each of these tools is the use of interaction and visualization techniques with respect to the feature model configuration based on the non-functional properties. Hence, in our work we aimed at developing a tool which:

1. provides a facility to integrate the feature model and non-functional properties;
2. facilitates and enhances the feature model configuration process by :
   (a) automating the feature model configuration process based on non-functional requirements;
   (b) employing visualization and interaction techniques, which provides software developers with clear and understandable view of the generated configuration.

For providing tooling support, we extended a well-known feature model plug-in (fmp) [46]. The fmp tool is an eclipse plug-in which supports editing and configuring feature models.

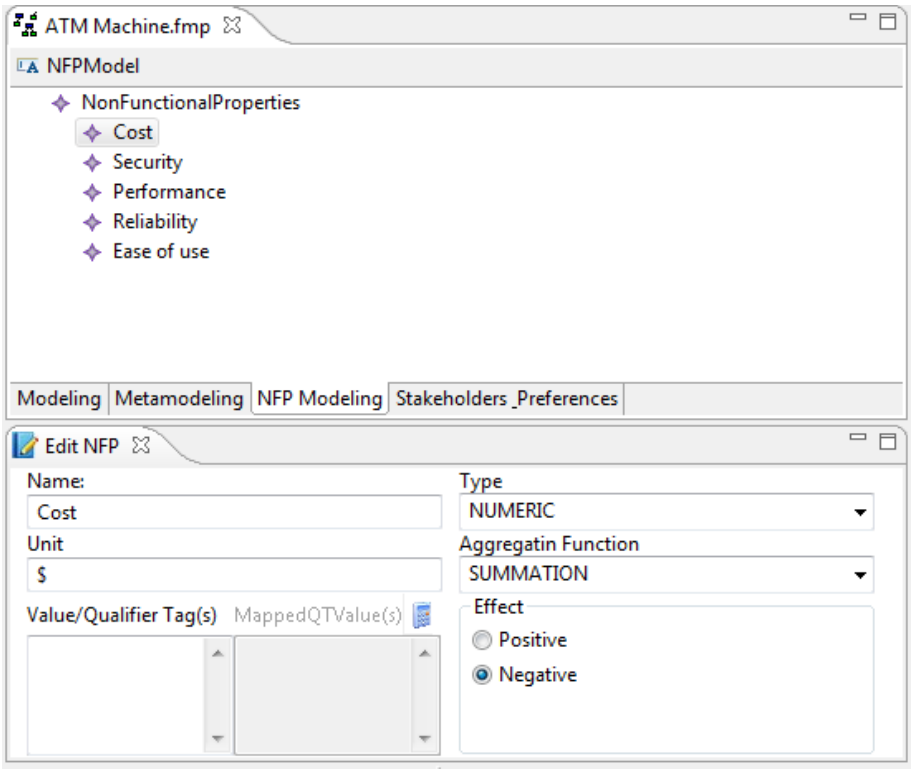


Figure 17: NFP model definition view

It provides very basic visualization and (i.e. it used indented list to show feature model) interaction techniques, however, it provides a good facilities to define a feature model and manage the variabilities. Moreover, the tool is accessible for academic researchers. Hence, we consider fmp as a basis and extend it to support more functionalities such as integrating feature model with non-functional properties and automating the configuration process.

Figure 16 illustrates the architecture of the tool and its components. In the following sections we discuss different components of the tool in more detail including: 1) Extended feature modeling component, 2) stakeholders' preferences managing component, 3) configuration generator component, and 4) configuration visualizer component.

5.1. Extended Feature Modeling Component

In order to generate the feature model configuration based on non-functional requirements and provide information about the non-functional properties of the final product, the effect of each feature on the NFPs should be specified. Accordingly, the total value of NFPs for the final product can be calculated based on the effect of selected features on the NFPs. For example, if the cost of each feature in a feature model is specified, the total cost of the product is calculated by summation of the selected features' cost. To this end, we extend the feature model in such a way that software developers can define non-functional properties as well as functionalities in the feature model. In the developed tool, as shown in Figure 17 a user can define the non-functional property model(i.e., all possible non-functional properties in the domain and their characteristics such as type, unit, aggregation function, default values, and so on). After defining the NFP model,

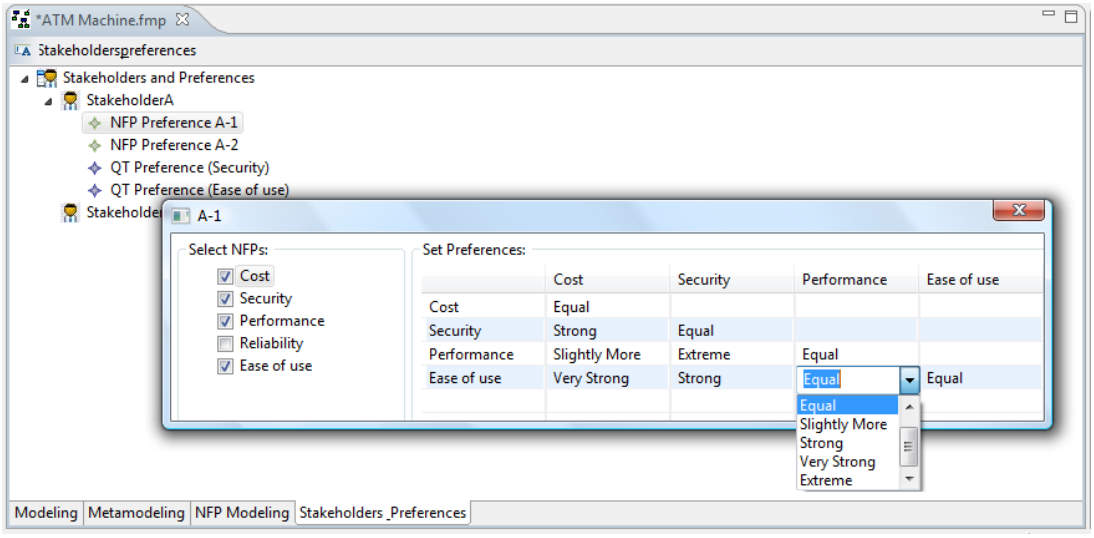


Figure 18: A snapshot of the view for managing stakeholders and their preferences

features can be annotated with corresponding NFPs and the value of each properties is specified based on the features' implementation.

5.2. Stakeholders' Preferences Managing Component

In the configuration process, the application engineer captures the stakeholders' functional requirements including the required feature set, the preferences (i.e., the relative importance of non-functional properties), and the constraints over NFPs. By applying AHP, fuzzy cognitive map, and the utility function, which are implemented in the developed tool, the ranks of NFPs and features are computed, respectively. The developed tool has facilities to define stakeholders and managing their preferences (See Figure 18).

5.3. Configuration Generator Component

This component transfers a feature model and requirements into planning domain and problem. Next, in order to generate an optimal configuration, an efficient planner (i.e. SHOP2) is employed, which uses a search-control strategy called ordered task decomposition to perform reasoning on the HTN planning domain [24].

The SHOP2 planner – a domain independent HTN planner – is automatically executed with its required inputs (i.e., HTN planning domain and HTN planning problem). The output of the planner is a set of plans (i.e., sequences of operators for the given tasks), which satisfy the initial conditions and have optimal value. The optimal configuration is achieved by selecting the features corresponding to the operators in the optimal plan.

5.4. Configuration Visualizer Component

After generating the configurations using the SHOP2 planner, in addition to having an ability to export the results, the results are shown to the stakeholders in a visual view that highlights the selected features and shows the corresponding NFPs along with features. Using the visual view, the stakeholders can navigate the configuration and perform changes in the configuration. Figure 19 shows a snapshot of a generated configuration in the visual view. The view not only assists software developers to comprehend the

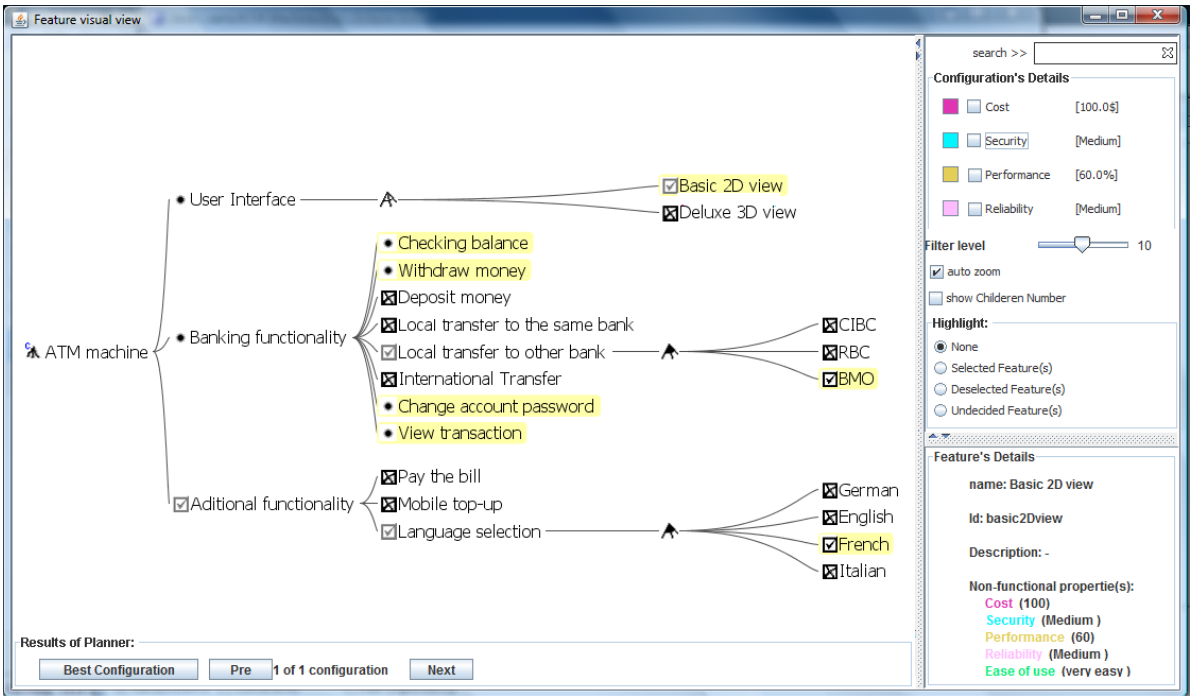


Figure 19: The result of planner in the visual view

generated configurations better, but it lets users to manipulate the configuration and create necessary changes. In the remainder of this section we discuss more about the visualization and interaction techniques used in the tool.

To implement the tool, we used prefuse [48], a Java graphic library with different tree and graph views. We used 2D tree-views as an initial representation and extended it for our purposes (i.e., visualizing feature model models, NFPs, and configurations). In the used tree, as a layout of this visualization, nodes represent the features of the feature model and links show the composition and variability relations between features.

In information visualization, color is used to group the items and show more information. The literature reviews showed that proper use of color can enhance and clarify a presentation [49][50]. We used color to show the NFPs related to each feature along with the feature. To this end, we assigned predefined colors to each NFPs. However, software developers can interact with the tool and determine desired colors for the NFPs. Having indicated colors corresponding to the NFPs (by the tool or software developers), we apply the coloring on the features of the feature model for showing the NFPs with which they were annotated. Moreover, when software developers select a NFP, Vis-fmp unfolds the features and expands the tree to show all features having the selected NFP. The visual view makes it easy for software developers to see which NFPs are covered with the specific features; which features contain specific NFPs; or how the coverage of NFPs is in the feature model.

In addition to the representation of feature models and their associated NFPs, other visualization and interaction tasks are provided in the visual view including:

1. *Overview*: This technique includes zoomed out views of each data type to see the entire collection along with an attached detail view [51]. In our case, user can see the

zoomed out view of a configured feature model with the detail of that configuration. The detail view contains the total value of NFPs which are calculated based on the selected features in a configured product.

2. *Zooming*: User can zoom in or out on the feature model in order to focus on the specific part of it or view the entire feature model. Moreover, auto-zooming is used to fit the tree to the screen after expanding.
3. *Filtering*: Vis-fmp provides two facilities for applying filtering on feature model configuration tree. 1) Filter-level – by setting a filter-level in the feature model tree, 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, Vis-fmp 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 require further unfolding, the tool shows the number of children of each feature along with their name.
4. *Highlight*: This technique is employed to highlight the features based on their states (i.e., selected, deselected, or undecided features).
5. *Detail on demand*: In addition to configuration detail , detail of each feature is shown including the general information about the feature and annotated NFPs and their values.
6. *Interactive Configuration*: Software developer can interact with this view by clicking on the features and changing their states (selected, deselected, or undecided). By each change the system updates the detail view of a configuration.

6. Evaluation

To assess our technique and the corresponding tool *Vis-fmp*, we formulated the following research questions:

- **RQ1 (Scalability)**: Can the approach configure feature models, in a reasonable time, based on functional and non-functional requirements and preferences?
- **RQ2 (Effectiveness)**: How effective is the approach in producing a feature model configuration? **RQ2-1**: Does the approach generate reliable results for application engineers? **RQ2-2**: What is the automation degree of the approach?

Our proposed approach mainly consists of two parts: computing feature ranks and the feature model configuration using the HTN planning technique. The former consists of analytical hierarchy process, fuzzy cognitive map, and utility functions whose execution time is polynomial. Therefore, we only concentrate on exploring the execution time of the planning technique for finding an optimal plan.

6.1. RQ1 (Scalability)

The purpose of this research question is to evaluate whether a feature model configuration can be performed in a reasonable amount of time. Hence, we conducted several experiments to investigate this research question.

6.1.1. Objects of Study

To evaluate the configuration approach, we adapted Betty FM Generator[2] which enables the random generation of highly-customized feature models [52], to generate feature models with different characteristics (e.g., number of features, probability of mandatory and optional features, probability of OR and XOR groups, and percentage of integrity constraints). We set the characteristics of generated feature models as: 50%, 25%, and 25% for the probability of being features in AND, OR, and XOR groups, respectively. Moreover, 50% of features in AND groups are optional features. The branching factor is also set to 10. These characteristics are backed up by most of surveyed feature models [14][53] to reflect the characteristics of real feature models.

According to the results of an investigation on non-functional properties done by Mairiza et al. [54], the number of relevant non-functional properties for different application domains is at most 11. Moreover, Sommerville and Sawyer [55] highlighted that the effective number of non-functional properties is around six. Considering these two studies, we defined 10 non-functional properties; six quantitative; and four qualitative non-functional properties. Five qualifier tags were defined for each non-functional property.

6.1.2. Experimental Setup

For each feature model, we applied our tool to configure the feature model based on randomly generated preferences and constraints. The evaluation was performed on a computer with an Intel Core DUO 2.2 GHZ CPU, 4GB of RAM, Windows Vista, Java Runtime Environment v5.0, SHOP2 v2.8, and SBCL (Steel Bank Common Lisp) v1.0.55.

To configure the feature models, we considered three independent variables including *number of features*, *number of constraints*, and *integrity constraints*; and *time* as a dependent variable. For each feature model in the study, features were annotated with quantitative non-functional properties and their values were produced by a random function with normal distribution. From a practical point of view, only some of the features may have an impact on a qualitative NFP like security. Hence, to reflect this point, features were randomly annotated with 0 to 4 qualitative non-functional properties with normal distributions, that is, most of the features had one or two non-functional properties. Based on our analysis on SPLOT repository [3], which shows an average of 18% of integrity constraints for real feature models [56], we considered two distributions of integrity constraints: 10% and 20%. Finally, for the constraints over NFPs, four cases were considered: no constraint and constraints over 2, 4, and 6 NFPs to reflect the effect of various constraints over NFPs at run time.

6.1.3. Experimental Results

Figure 20 illustrates the average time for configuring feature models with different numbers of features and percentages of integrity constraints. Our experiments reveal that, for feature models with less than 200 features, the planner returns results in a feasible time (around 16 second). The number of possible configurations exponentially grows by increasing the number of features. That is, the search space for finding an optimal plan by SHOP2 is boosted by increasing the size of feature models.

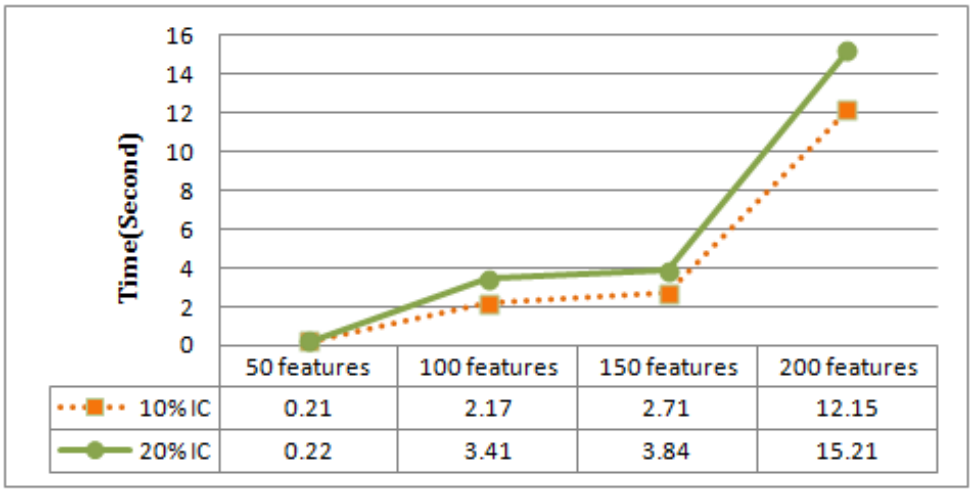
[2]Betty Feature Model Generator Version 1.1, http://www.isa.us.es/betty

[3]SPLOT - Software Product Line On-line Tools, http://www.splot-research.org

Generating optimal configurations based on stakeholders' preferences and constraints is NP-hard [5, 13, 11]. Hence, HTN planners similar to CSP solvers have problems in finding an optimal solution for large-scale problems. Although the SHOP2 planner applies some heuristics for improving the search time for finding an optimal plan (See ref. [26]), due to explicit representations of states in the memory, SHOP2 runs into memory problems for large domains. However, our experiments showed that for feature models with less than 200 features, SHOP2 returns an optimal plan in a feasible time.

We also investigated the effect of constraints over NFPs on the running time of the configuration technique proposed in Section 4. To this end, we run the tool with feature models containing 100 features and 20% integrity constraints. The results are shown in Figure 21. In all of these situations, the process of generating optimal configurations was successful.

According to the experiment results, all three independent variables (i.e., *number of features*, *number of constraints*, and *integrity constraints*) have an impact on running time. As shown in Figure 20, for the fixed number of integrity constraints, increasing the number of features raises the time for finding an optimal configuration, as increasing the number of features expands the search space for finding an optimal plan. Also, with the same number of features, increasing integrity constraints from 10% to 20% causes an increase in the time for finding a plan, as the planner needs to check more pair combinations of tasks and operators for optimizing a final plan.



	50 features	100 features	150 features	200 features
10% IC	0.21	2.17	2.71	12.15
20% IC	0.22	3.41	3.84	15.21

Figure 20: Running time of the configuration technique for feature models with different numbers of features and integrity constraints.

Finally, as illustrated in Figure 21, the number of constraints over NFPs has an impact on increasing the time for configuration, even if the number of features and integrity constraints is fixed. According to the results of the experiment, the impact of constraints over NFPs is higher than the impact of integrity constraints.

Since generating an optimal feature model configuration based on stakeholders' preferences and constraints is NP-hard, it may need a long time to produce a configuration for large feature models. Moreover, the manual configuration process is very hard and time-consuming task for application engineers. Consequently, the reported time in the Figure 20 and 21 can be considered as an acceptable time from the perspective of user.

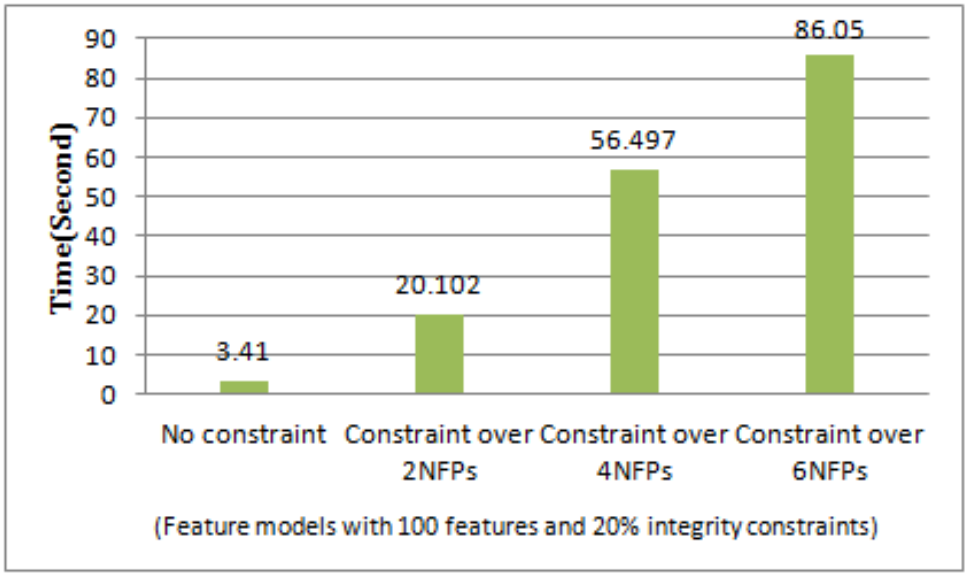


Figure 21: Running time of configuration technique based on different number of constraints over NFPs.

6.2. RQ2 (Effectiveness)

This research question aims to investigate if application engineers can trust the results returned by the planner and if the approach is beneficial for stakeholders to facilitate their tasks.

6.2.1. RQ2-1 (Reliability of Results)

Regarding the reliability of the results, our approach is based on transforming feature models into the HTN formalisms and applying SHOP2 to find an optimal configuration. According to the framework proposed in [57], we can investigate if this representation can be considered as a good surrogate for representing feature models and preferences. As shown, we can represent feature model relations and constraints using existing constructs in HTN such as method, tasks, and operator. Additionally, operators provide properties to define a feature rank. The constraints over non-functional properties and integrity constraints can be defined as preconditions of the operators and methods. Hence, we can consider HTN to be a surrogate for feature models. With respect to the ontological commitment, HTN views the world as a set of actions, tasks, constraints between them; this makes it suitable for representing both feature models and constraints. With respect to the planner, SHOP2 has been extensively applied in many projects [24] in government laboratories (e.g., evacuation planning), industry (e.g., evaluating of enemy threats), and academia (e.g., automated composition of Web services and goal model reasoning [19]), which is an indication of the usefulness of the returned plans and their corresponding configurations.

6.2.2. RQ2-2(Automation and Improvement)

With respect to facilitating the stakeholders tasks during the feature model configuration, we emphasize on the effectiveness of our approach for automating configuration

process, improving the process of handling preferences and value related requirements dependencies, and enhancing the interaction with generated configuration models.

Regarding the automation level, our approach requires only very few manual interventions. The main tasks of the application engineers are: 1) specifying the relative importance of non-functional properties; 2) creating the mapping function for qualitative non-functional properties; 3) indicating the interdependencies between non-functional properties; and 4) specifying the atomic features that must be included based on the functional requirements. Computing ranks of non-functional properties based on preferences and requirements dependencies and finding optimal solution are fully automated in our approach.

Additionally, application engineers have a large number of features to choose from during the configuration process. The existing approaches [10, 11, 13] only select features based on stakeholders' priorities over requirements. However, due to the fact that non-functional requirements depend on each other and affect the other non-functional requirements, this can not be the only basis for selecting features [20]. Hence, interdependencies between non-functional requirements must be considered in the configuration process. On the other hand, value related dependencies increase the complexity of the feature selection. We facilitate the selection of features based on priorities and dependencies by formulating the knowledge about the dependencies in terms of fuzzy cognitive maps. Also, combination of AHP, fuzzy cognitive map, utility theory provides a systematic approach for computing features ranks based on stakeholders requirements.

Finally, by applying visualization and interaction techniques for representing configuration results(i.e. the configuration model and the total values of non-functional properties), we improve the comprehension and changeability of the feature model configuration. In order to investigate the tool, we performed the controlled experiments, and visualization was found to be beneficial for configuration understandability and changeability tasks. The details of the experiment are reported in the technical report[7].

6.3. Threats to Validity

The validity of results of an experimental evaluation is always subject to different threats including internal and external threats. In this section, we explore the major threats to validity of our results.

The major concern in the internal validity is the confounding variables (i.e., variables which are not considered as independent variables and might have impact on the dependent variable). We prevented these kinds of threats by controlling confounding variables. When we were investigating the impact of independent variables related to feature models (e.g., size of feature models, and integrity constraints) on the running time, we controlled impacts of the non-functional constraints(see figure 20). Similarly, we controlled the feature models characteristics (see figure 21) when we investigating the impact of constraints over non-functional properties. Additionally, we considered a fixed distribution of the variability relations in feature models as we only explored the size of feature models and integrity constraints.

External validity investigates if the results of the experiments are generalizable. We identify two main factors influencing the external validity of our experiments: the characteristics of feature models and the number of non-functional properties. With respect to the size of feature models, our investigation over SPLOT repository confirmed that

the sizes of generated models are aligned with the size of existing models in the SPLOT. However, the limitation on the model size is due to planner (i.e. SHOP2) which we used. In this regard, we aim at using another HTN planner (called HTNPLAN-P [25]) which applies new heuristics and can support larger planning space problems. The distributions of integrity constraints and variability relations in the feature models backed up by our analysis over SPLOT repository and existing studies [14][53] on the characteristics of real feature models. Finally, regarding the number of non-functional properties, the existing studies by Mairiza et al. [54] and Sommerville and Sawyer [55] showed that the number of relevant non-functional properties for different application domains is at most 11 and the effective number of non-functional properties is around six.

7. Related Work on Feature Model Configuration

The configuration process has been a center of interest in software product line research community for a long time. Many research papers have been proposed to guide application engineers for configuring feature models as well as automate parts of the configuration process and optimize non-functional properties in the product derivation. In this section, first we mention the existing feature configuration approaches and then compare our approach with them according to a set of criteria. Additionally, we listed a set of similar techniques in requirements engineering domain where decision making algorithms were used.

7.1. Feature Model Configuration Approaches

The first attempt, important for our research, has been done by Czarnecki et al. [9] who introduced a stage configuration process. In that work, a number of specialization steps are introduced to remove variability from feature models. In each specialization step, some parts of the feature model variability are resolved. These steps include: refining feature cardinality, refining group cardinality, removing a sub-feature from a feature group, selecting a feature from a feature group, assigning an attribute value, and cloning a solitary sub-feature. In order to help application engineers to derive a valid configuration using this approach, tooling support and validation approaches are provided by Czanrecki et al. [9]

Second group of related works focuses on automating configuration process and considers functional and non-functional requirements of the target application when configuring the feature model. Benavides et al. [5] developed an automated reasoning technique over extended feature models (i.e., feature models with extra-functional features). Using their extension, they were able to assign extra-functionalities such as price range or time range to features. The purpose of their technique is to find a product of a model based on given constraints. Their technique is based on mapping feature models to Constraint Satisfaction Problems (CSP)s and use of CSP solvers [5][10].

Batory et al. [40] integrated feature models with grammars and propositional formulas. Moreover, they used Logic Truth Maintenance Systems (LTMS) and satisfiability solver (SAT), in feature models. The SAT techniques are used for checking the violation of the feature model constraints during the configuration process. Therefore, the SAT techniques can be used as peripheral means during configuration process.

Siegmund et al. [13] have developed a technique called SPL conqueror which extends feature models with non-functional properties and applies CSP to find an optimal configuration based on user defined objective functions. In their technique, a number of preprocessing steps are taken to reduce the search space for optimal configuration. Their approach differs from Benavidas' approach on types of non-functional properties they manage. Siegmund et al. considered both qualitative and quantitative non-functional properties while Benavidas et al. concentrate on quantitative non-functional properties.

White et al. [11] used a Filtered Cartesian Flattening (FCF) method to select optimal feature sets according to the resource constraints. In their method, they map the feature selection problem to a multi-dimensional multi-choice knapsack problem (MMKP) [11]. By applying the existing MMKP approximation algorithms, they provide partially optimal feature configurations in polynomial time. Filtered Cartesian Flattening consists of different steps including: 1) producing a number of independent MMKP sets (i.e. cutting the FM diagram); 2) converting feature model parts to XOR relations because MMKP just supports XOR relation; 3) flattening with Filtered Cartesian Products; 4) handling Cross-tree Constraints; 5) MMKP Approximation. Although their approach is successful for large feature models, they have a limitation on the number of resources and the amount of resources consumed by features. Moreover, for performing their approach, multi-core processors and parallel computing are required.

White et al. [15] also formalize the stage configuration and introduce a Multi-step Software Configuration probLEm solver (MUSCLE) in which they provide a formal model for multi-step configuration and map it to CSPs. Hence, CSP solvers were used to determine the path from the start of the configuration to the desired final configuration. They consider non-functional properties such as cost constraints between two configurations and formalize them as CSP constraints. Their approach is only applicable for multi-stage configuration and focuses on creating new configurations from an already derived product configuration.

Mendonca et al. [58] proposed a translation of basic feature models into propositional logics and used Binary Decision Diagrams (BDD) as the reasoning system. Their approach concentrates on validating feature models and does not offer a facility for automated configuration. It can be used in a multi-stage configuration process to validate the results of every specialization of a feature model (called interactive configuration). An interactive configuration only checks the structural constraints of feature models and does not consider preferences and non-functional requirements of stakeholders. A tool is implemented to support software developers in validation.

Guo et al. also addressed the challenge of optimizing feature model configuration in their work [14]. They proposed an approach in which Genetic Algorithms are employed to optimize Feature Selection (GAFES). In the GAFES algorithm the first population is created by randomly generating a string with n binary digits (which represents a chromosome so that a value of 1 and 0 for a digit indicates the selected and deselected feature, respectively). Next, the arbitrary feature selection is transformed into a valid feature selection by applying FesTransform (Feature selection Transform) algorithm which is a key component of their approach. The results of their empirical study show that their proposed algorithm can achieve an average of $86 - 90\%$ optimality for feature selection.

Zhang et al. [59] proposed an AHP based method for quality aware configuration of

feature models. They considered quality attributes as features in feature models and defined interdependency relation between functional features and non-functional features. Having formed the groups of functional features (also called contributors) for each quality attribute based on interdependency relations, they employed Analytical Hierarchy Process (AHP) to calculate the relative importance of contributors of each quality attribute. The overall impact of a quality attribute is achieved by forming distinct subsets of the feature (contributor) set based on feature model relations and aggregating the values of valid selections from these subsets. Later the values of quality attributes are utilized for modifying the configuration and filtering out features in the feature model.

7.2. Comparing the Approaches

To systematically compare the proposed approach with other existing approaches, we devise a number of criteria that need to be supported by configuration techniques in order to be effective for application engineering. To define the criteria for systematic comparisons, similar to [60], we applied bottom-up and top-down approaches. Following the bottom-up approach, we identified various important aspects of feature model configuration in the description of existing related works and added them to the criteria set. Following the top-down approach, we used an existing survey on configuration of software product lines. We do not claim that this criteria set is complete, but it provides appropriate aspects to compare our work with related works. These criteria include: 1) Managing functional and non-functional requirements; 2) Modeling stakeholders' preferences; 3) Optimization; 4) Considering stakeholder constraints; 5) Providing tooling support; 6) Automating configuration process; 7) Ensuring the feature model constraints; 8) Effective representation of results to stakeholders; 9) Time efficiency. In the following subsections, we review existing configuration techniques and compare them with respect to the above criteria. Table 5 summarizes the results of the comparison of the approaches based on the criteria identified earlier.

Managing functional and non-functional properties. Stage configuration [9] and the work in [58] provide no guideline for configuration based on non-functional properties. FCF [11], MUSCLE [15], and the technique from [5] support selection of features only based on quantitative non-functional requirements. Our approach and SPL conqueror [13] guarantee the selection of features based on functional and non-functional properties. Zhang et al. [59] configure feature models based on functional and qualitative non-functional requirements. Furthermore, only SPL conqueror [13] and our approach consider both qualitative and quantitative non-functional properties. Finally, our approach is the only technique which handles the interdependencies between non-functional properties.

Modeling stakeholders' preferences. CSP based approaches [13][15], FCF [11], and GAFES [14] model stakeholders' preferences in terms of user defined objective functions. Considering the diversity of non-functional properties, it is not easy for stakeholders to define an objective function, which reflects their preferences. However, our approach provides a systematic and easy technique to capture stakeholders' preference in terms of relative importance and defines the objective function using these inputs. To capture stakeholders' preferences we utilize the AHP algorithm which is used in many applications; and it is simple enough, so that stakeholders can use it without any formal training [61]. Similar to our work, Zhang et al. [59] capture preferences in terms of relative importance

and apply AHP algorithm to compute overall impact on quality attributes. However, the number of pairwise comparison in their approach depends on the number of features while in our approach it depends on the number of non-functional properties. Stage configuration does not provide any support for stakeholders' preferences.

Considering stakeholders' constraints. Stakeholders may define constraints based on the resources that are available to them and level of non-functionality. These constraints need to be considered and only a configuration which satisfies the stakeholders' constraints and optimizes the preferences must be produced. FCF [11], GAFES [14], CSP based approaches [5][13][15], and Zhang et al. [59] support constraints on the stakeholders' resources. On the other hand, our approach handles constraints over both qualitative and quantitative non-functional properties.

Optimization and time efficiency. Generating optimal configurations based on the stakeholders' preferences and constraints is NP-hard. All CSP approaches [5][13][15] and our approach ensure optimality of the solution, but they require high computation time. To compare the running time of our approach with CSP approaches, the running time of CSP based and planning based approaches depends on the solvers which are available in the domains and heuristics that these solvers use for finding solution. According to the size of feature models solved by these techniques, shown in table 5, the HTN based planner performs better than the CSP solver used in FAMA [5]. Additionally, using the planning based approach, we are able of handling qualitative constraints which is not possible in the CSP based approaches. The FCF [11] and GAFES [14] provide partially optimal solutions in a polynomial time. Zhang et al. [59] modify existing configurations to satisfy the quality requirements of stakeholders, but do not produce an optimal configuration. Stage configuration and the work in [58] do not support optimization of the stakeholders' requirements.

Tooling support and automation. All the approaches, except FCF, provide tooling support. Stage configuration provides tooling support, but little automation for feature model configuration is provided. With respect to the usability, our approach and SPL conqueror [13] apply visualization techniques to present the configuration results to the stakeholders. Our tool shows the quality level of selected features along with them in the same view (Figure 19). Other tools provide basics views for representing configurations to the stakeholders.

Feature model integrity constraints. All approaches, except [5] ensure the satisfaction of integrity constraints (i.e., requires and excludes relations) during configuration.

In conclusion, our approach covers all criteria and the only limitation of the approach in comparison with other approaches is the completion time. Due to the NP-hard nature of the feature model configuration problem, among the existing feature model configuration approaches, some have limited scale for generating optimal feature model configurations (e.g., the approach in [5] and our approach) and some require special hardware (e.g., FCF by White et al. [11]) or return partial optimal configurations (e.g., GAFES by Guo et al. [14]). For very large feature models finding an optimal configuration is not feasible because the number of possible configurations has exponential growth. For example, for feature models with $1,000$ number of variation points, $2^{1,000}$ configurations must be evaluated. Although the AI approaches like planning techniques apply heuristics to improve completion time for large problem sizes, we still have restrictions over the size of feature

models.

Table 5: Comparative analysis of related works ((+) criterion met, (-) criterion not met, (+/-) criterion partially met).

Approach \ Criteria	FR/NFR	Preference	Optimization	Constraints	Automation	Integrity constraints	Tooling support	Time efficiency
Stage Configuration Czarnecki et al. [9]	+/-	-	-	-	-	+	+	- (no result was reported)
CSP - Benavides et al. [5][10]	+/-	-	+	+	+	-	+	- (up to 52 features, optimal results)
FCF White et al. [11]	+/-	+/-	+/-	+	+	+	-	+ (up to 10,000 features, results with 90% optimality)
SPL Conqueror Siegmund et al. [13]	+	+/-	+	+	+	+	+	+/- (not exactly mentioned the number of feature)
MUSCLE White et al. [15]	+/-	+/-	+	+	+	+	-	+/- (up to 500 features)
Mendonca et al. [58]	+/-	-	-	-	+	+	+	+ (up to 2,000 features)
GAFES Guo et al. [14]	+/-	+/-	+/-	+	+	+	+	+ (up to 10,000 features, results with 86-90% optimality)
Zhang et al. [59]	+	+	+/-	+	+/-	+	+	- (no result was reported)
Our approach	+	+	+	+	+	+	+	+/- (up to 200 features, optimal results)

7.3. Prioritization in Requirements Engineering

In the requirements engineering several techniques including the AHP [62, 63], bubble sort, Hierarchical Cumulative Voting(HCV) [64], and Satisfiability Modulo Theory [65] have been proposed to handle requirements prioritization. Among these prioritization techniques AHP is widely referenced and the result of an evaluation of six techniques by Karlsson et al. [33] revealed that AHP is the most promising technique in terms of providing trustworthy results. Liaskos et al. [62] and Vinay et al. [66] applied AHP to quantitatively assess contribution relationships in goal models based on stakeholder inputs. By the increase of requirements, the number of required pairwise comparisons is dramatically grown in AHP based approaches. To tackle this problem, Harker [63]

proposed a method incomplete AHP (IAHP) which minimizes the number of selected pairs and maintains a good trade-off between the precision of the final solution and the effort of the stakeholders.

Similar to AHP, Bubble sort and binary search approaches perform pairwise comparison to decide the rank of a requirement based on its relative position to the other requirements in the set. Bubble sort and binary search also suffer from high number of comparisons.

HCV [64] uses an absolute measure, represented in terms of votes, instead of relative measure to represent the preferences and requirements with top votes are prioritized higher by the approach. HCV computes the total priorities of a requirements set by adding up the ranks of all the requirements.

SMT based approach [65] applies pairwise comparison to capture relative importance of the requirements, then formulates the prioritization as a Max-SAT problem and uses a SMT solver to prioritize the requirements.

8. Conclusion & FutureWork

We target an open research question in software product lines: *how to select a suitable set of features from a feature model based on both the stakeholders' functional and non-functional requirements and preferences?* We investigated the notion of non-functional properties in software product lines and developed a non-functional model in which two main categories of non-functional properties are considered (i.e., quantitative and qualitative NFPs). Feature models were extended with the notion of NFPs. We mapped features and relations among features in the feature model into the components of the HTN planning domain (i.e., operator, task, and method). This transformation covers all relations in the feature model including: *AND*, *OR*, and *XOR* decompositions as well as *require* and *exclude* relations among features. We also formalized the configuration problem as an HTN planning problem and employed an existing HTN planner (SHOP2) to generate an optimal configuration based on the stakeholder preferences and constraints over non-functional properties.

The proposed feature model configuration approach has the following improvements in comparison with other existing approaches:

1) In addition to functional requirements, constraints over both qualitative and quantitative non-functional properties are taken into account; 2) we capture the preferences of stakeholders in terms of relative importance between non-functional properties which is an easy way for stakeholders to specify; 3) we calculate the weight of NFPs by utilizing AHP and FCM; 4) we introduce the concept of artificial intelligence planning in the context of feature model configuration and provide the transformation rules to convert feature models into Hierarchical Task Network; 5) we developed a tool which automates feature model configuration process and provides several visualization and interaction techniques to facilitate the configuration process; 6) the experimental results revealed that our approach can be useful because: *i*) it provides an optimal configuration based on stakeholders preferences and constraints over non-functional properties; and *ii*) it has a good performance on feature models with less than 200 features.

Finally, the proposed approach improves our earlier work in [1] by considering more complex relations in the non-functional model (i.e. dependencies between non-functional

properties), and providing complete tooling support with a new sets of interaction and visualization techniques such as zooming, filtering, and interactive configuration.

This work can be extended in several directions:

- For larger feature models, the approach is computational demanding, similar to CSP-based approaches [5, 13]. Another HTN planner (called HTNPLAN-P [25]) applies new heuristics and can support larger planning space problems, so that the practical performance time can be much improved as shown in [25]. Since HTNPLAN-P is not yet publicly available, we could not test our work on larger feature models. In the future work, HTNPLAN-P can be employed for finding optimal plans which will improve scalability of our approach.

- Other planning techniques such as PDDL [67](i.e., Planning Domain Definition Language) based approach can be employed. A wide range of planning engines support PDDL; if the feature model can be successfully transformed into PDDL, we can utilize more powerful planning engines. Moreover, PDDL supports optimization so that the specific metrics (i.e., NFPs in our context) can be minimized or maximized during plan generation.

- According to our initial study, effective visualization techniques improve the configuration process and ease the task for application engineers. Therefore, more interaction and visualization techniques can be applied on the tool and an experimental study can be done in an industrial environment to examine the effects of the applied techniques. In the current approach, we assume that the importance of non-functional properties is the same for all of the features of a given feature model, which may not be the case in some scenarios. Therefore, a possible work can include the handling of local importance of non-functional properties for different parts of feature models.

References

[1] S. Soltani, M. Asadi, D. Gasevic, M. Hatala, E. Bagheri, Automated planning for feature model configuration based on functional and non-functional requirements, in: SPLC (1), 2012, pp. 56–65.

[2] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. S. Peterson, Feature-oriented domain analysis (foda) feasibility study, Tech. rep., Carnegie-Mellon University Software Engineering Institute (November 1990).

[3] J. Bosch, Design and use of software architectures - adopting and evolving a product-line approach., Addison-Wesley, 2000.

[4] F. J. v. d. Linden, K. Schmid, E. Rommes, Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering, Springer-Verlag New York, Inc., 2007.

[5] D. Benavides, P. T. Martn-Arroyo, A. R. Corts, Automated reasoning on feature models., in: O. Pastor, J. F. e Cunha (Eds.), CAiSE, Vol. 3520 of Lecture Notes in Computer Science, Springer, 2005, pp. 491–503.

[6] E. Bagheri, M. Asadi, D. Gasevic, S. Soltani, Stratified analytic hierarchy process: Prioritization and selection of software features., in: J. Bosch, J. Lee (Eds.), SPLC, Vol. 6287 of Lecture Notes in Computer Science, Springer, 2010, pp. 300–315.

[7] S. Soltani, M. Asadi, D. Gasevic, M. Hatala, The effects of visualization and interaction techniques on feature model configuration, Tech. rep., School of Intractive Art and Technology, SFU, `https://files.semtech.athabascau.ca/public/TRs` (2011).

[8] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, J. N. Dag, An industrial survey of requirements interdependencies in software product release plannin, in: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, RE '01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 84–.

[9] K. Czarnecki, S. Helsen, U. W. Eisenecker, Staged configuration through specialization and multilevel configuration of feature models., Software Process: Improvement and Practice 10 (2) (2005) 143–169.

[10] D. Benavides, S. Segura, P. T. Martn-Arroyo, A. R. Corts, Using java csp solvers in the automated analyses of feature models., in: R. Lmmel, J. Saraiva, J. Visser (Eds.), GTTSE, Vol. 4143 of Lecture Notes in Computer Science, Springer, 2006, pp. 399–408.

[11] J. White, B. Dougherty, D. C. Schmidt, Selecting highly optimal architectural feature sets with filtered cartesian flattening, Journal of Systems and Software 82 (8) (2009) 1268–1284. doi:10.1016/j.jss.2009.02.011.

[12] E. Bagheri, T. D. Noia, D. Gasevic, A. Ragone, Formalizing interactive staged feature model configuration, Journal of Software: Evolution and Process 24 (4) (2012) 375–400. doi:10.1002/smr.534.
URL `http://dx.doi.org/10.1002/smr.534`

[13] N. Siegmund, M. Rosenmüller, M. Kuhlemann, C. Kästner, S. Apel, G. Saake, SPL Conqueror: Toward optimization of non-functional properties in software product lines, Software Quality Journal (2011) 1–31doi:10.1007/s11219-011-9152-9.

[14] J. Guo, J. White, G. Wang, J. Li, Y. Wang, A genetic algorithm for optimized feature selection with resource constraints in software product lines, Journal of Systems and Software 84 (12) (2011) 2208–2221. doi:10.1016/j.jss.2011.06.026.

[15] J. White, B. Dougherty, D. C. Schmidt, D. Benavides, Automated reasoning for multi-step feature model configuration problems, in: Proceedings of the 13th International Software Product Line Conference, SPLC '09, IEEE, 2009, pp. 11–20.

[16] L. Olsina, G. Lafuente, G. Rossi, Specifying quality characteristics and attributes for websites, in: Web Engineering, Software Engineering and Web Application Development, Springer-Verlag, London, UK, UK, 2001, pp. 266–278.

[17] T. Saaty, The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation, McGraw-Hill, 1980.

[18] B. Kosko, Fuzzy cognitive maps, International Journal of Man-Machine Studies 24 (1) (1986) 65 – 75. doi:10.1016/S0020-7373(86)80040-2.

[19] S. Liaskos, S. A. McIlraith, S. Sohrabi, J. Mylopoulos, Integrating preferences into goal models for requirements engineering, in: Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, RE '10, IEEE Computer Society, 2010, pp. 135–144. doi:10.1109/RE.2010.26.

[20] J. Karlsson, S. Olsson, K. Ryan, Improved practical support for large-scale requirements prioritising, Requirements Engineering 2 (1997) 51–60. doi:10.1007/BF02802897.

[21] R. Yu, G.-H. Tzeng, A soft computing method for multi-criteria decision making with dependence and feedback, Applied Mathematics and Computation 180 (1) (2006) 63 – 75. doi:10.1016/j.amc.2005.11.163.

[22] R. Taber, Knowledge processing with fuzzy cognitive maps, Expert Systems with Applications 2 (1) (1991) 83 – 87. doi:10.1016/0957-4174(91)90136-3.

[23] M. León, C. Rodriguez, M. M. García, R. Bello, K. Vanhoof, Fuzzy cognitive maps for modeling complex systems, in: Proceedings of the 9th Mexican international conference on Advances in artificial intelligence: Part I, MICAI'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 166–174.

[24] D. Nau, T.-C. Au, O. Ilghami, U. Kuter, H. Munoz-Avila, J. W. Murdock, D. Wu, F. Yaman, Applications of shop and shop2, IEEE Intelligent Systems 20 (2) (2005) 34–41. doi:10.1109/MIS.2005.20.

[25] S. Sohrabi, J. A. Baier, S. A. McIlraith, Htn planning with preferences, in: Proceedings of the 21st international jont conference on Artifical intelligence, IJCAI'09, Morgan Kaufmann Publishers Inc., 2009, pp. 1790–1797.

[26] D. Nau, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, F. Yaman, Shop2: An htn planning system, Journal of Artificial Intelligence Research 20 (2003) 379–404.

[27] I. Ognjanovic, D. Gašević, E. Bagheri, M. Asadi, Conditional preferences in software stakeholders' judgments, in: Proceedings of the 2011 ACM Symposium on Applied Computing, SAC '11, ACM, 2011, pp. 683–690. doi:10.1145/1982185.1982335.

[28] R. Berntsson Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, Quality requirements in industrial practice—an extended interview study at eleven companies, IEEE Trans. Softw. Eng. 38 (4) (2012) 923–935. doi:10.1109/TSE.2011.47.

[29] S. Sohrabi, J. A. Baier, S. A. McIlraith, Htn planning with preferences, in: Proceedings of the 21st international jont conference on Artifical intelligence, IJCAI'09, Morgan Kaufmann Publishers Inc., 2009, pp. 1790–1797.

[30] P. C. Fishburn, Additive utilities with incomplete product set: Applications to priorities and assignments, Operations Research 15 (1967) 537–542.

[31] D. W. Miller, M. K. Starr, Executive decisions and operations research [by] David W. Miller and Martin K. Starr, 2nd Edition, Prentice-Hall Englewood Cliffs, N.J.,, 1969.

[32] E. Triantaphyllou, B. Shu, S. Nieto Sanchez, T. Ray, Multi-Criteria Decision Making: An Operations Research Approach, Vol. 15, 1999, pp. 175–186.

[33] J. Karlsson, C. Wohlin, B. Regnell, An evaluation of methods for prioritizing software requirements, Information and Software Technology 39 (1415) (1998) 939 – 947. doi:http://dx.doi.org/10.1016/S0950-5849(97)00053-0.

[34] T. L. Saaty, How to make a decision: The analytic hierarchy process, European Journal of Operational Research 48 (1) (1990) 9–26.

[35] F. Roos-Frantz, D. Benavides, A. Ruiz-Cortés, A. Heuer, K. Lauenroth, Quality-aware analysis in product line engineering with the orthogonal variability model, Software Quality Control 20 (3-4) (2012) 519–565.

[36] H. Espinoza, H. Dubois, S. Gérard, J. Medina, D. C. Petriu, M. Woodside, Annotating uml models with non-functional properties for quantitative analysis, in: Proceedings of the 2005 international conference on Satellite Events at the MoDELS, MoDELS'05, Springer-Verlag, 2006, pp. 79–90. doi:10.1007/11663430_9.

[37] T. Yu, K.-J. Lin, Service selection algorithms for web services with end-to-end qos constraints, in: Proceedings of the IEEE International Conference on E-Commerce Technology, CEC '04, IEEE Computer Society, 2004, pp. 129–136.

[38] B. Mohabbati, D. Gasevic, M. Hatala, M. Asadi, E. Bagheri, M. Boskovic, A quality aggregation model for service-oriented software product lines based on variability and composition patterns, in: The 9th International Conference on Service Oriented Computing (ICSOC 2011), Springer, 2011, pp. 436–451.

[39] K. Czarnecki, E. Ulrich, Generative Programming: Methods, Tools, and Applications, Addison-Wesley, 2000.

[40] D. Batory, Feature models, grammars, and propositional formulas., in: J. H. Obbink, K. Pohl (Eds.), SPLC, Vol. 3714 of Lecture Notes in Computer Science, Springer, 2005, pp. 7–20.

[41] N. Siegmund, M. Rosenmuller, C. Kastner, P. G. Giarrusso, S. Apel, S. S. Kolesnikov, Scalable prediction of non-functional properties in software product lines, in: Proceedings of the 2011 15th International Software Product Line Conference, SPLC '11, IEEE Computer Society, 2011, pp. 160–169. doi:10.1109/SPLC.2011.20.

[42] B. Kosko, Hidden patterns in combined and adaptive knowledge networks, Int. J. Approx. Reasoning 2 (4) (1988) 377–393. doi:10.1016/0888-613X(88)90111-9.

[43] T. Yu, K.-J. Lin, Service selection algorithms for composing complex services with multiple qos constraints, in: Proceedings of the Third international conference on Service-Oriented Computing, ICSOC'05, Springer-Verlag, 2005, pp. 130–143. doi:10.1007/$11596141_1$1.

[44] F. Rosenberg, P. Celikovic, A. Michlmayr, P. Leitner, S. Dustdar, An end-to-end approach for qos-aware service composition, in: Proceedings of the 13th IEEE International Conference on Enterprise Distributed Object Computing, EDOC'09, IEEE Press, 2009, pp. 128–137.

[45] Variant management with pure:: variants., Tech. rep., Pure-systems GmbH (2003). URL `http://www.pure-systems.com`

[46] K. Czarnecki, P. Kim, Cardinality-Based Feature Modeling and Constraints: A Progress Report, in: Proceedings of the International Workshop on Software Factories at OOPSLA 2005, ACM, 2005.

[47] R. Rabiser, P. Grünbacher, D. Dhungana, Requirements for product derivation support: Results from a systematic literature review and an expert survey, Information and Software Technology 52 (3) (2010) 324–346. doi:10.1016/j.infsof.2009.11.001.

[48] J. Heer, S. K. Card, J. A. Landay, prefuse: a toolkit for interactive information visualization, in: Proceedings of the SIGCHI Conference on Human factors in computing systems, CHI '05, ACM, 2005, pp. 421–430. doi:10.1145/1054972.1055031.

[49] B. C., Color use guidelines for data representation, in: Proceedings of the Section on Statistical Graphics, American Statistical Association, 1999, pp. 55–60.

[50] M. Stone, Choosing colors for data visualization (2006).

[51] B. Shneiderman, The eyes have it: a task by data type taxonomy for information visualizations, Proceedings 1996 IEEE Symposium on Visual Languages 0 (UMCP-CSD CS-TR-3665) (1996) 336–343.

[52] S. Segura, J. A. Galindo, D. Benavides, J. A. Parejo, A. Ruiz-Cortés, Betty: benchmarking and testing on the automated analysis of feature models, in: Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems, VaMoS '12, ACM, 2012, pp. 63–71. doi:10.1145/2110147.2110155.

[53] T. Thum, D. Batory, C. Kastner, Reasoning about edits to feature models, in: Proceedings of the 31st International Conference on Software Engineering, ICSE '09, IEEE Computer Society, 2009, pp. 254–264. doi:10.1109/ICSE.2009.5070526.

[54] D. Mairiza, D. Zowghi, N. Nurmuliani, An investigation into the notion of non-functional requirements, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10, ACM, 2010, pp. 311–317. doi:http://doi.acm.org/10.1145/1774088.1774153.

[55] I. Sommerville, I. Sommerville, P. Sawyer, P. Sawyer, Viewpoints: principles, problems and a practical approach to requirements engineering, Annals of Software Engineering 3 (1997) 101–130.

[56] E. Bagheri, D. Gasevic, Assessing the maintainability of software product line feature models using structural metrics, Software Quality Journal 19 (3) (2011) 579–612.

[57] H. S. R. Davis, P. Szolovits, What is knowledge representation?, AI Magazine 14 (1) (1993) 17–33.

[58] M. Mendonca, A. Wasowski, K. Czarnecki, D. Cowan, Efficient compilation techniques for large scale feature models, in: Proceedings of the 7th international conference on Generative programming and component engineering, GPCE '08, ACM, 2008, pp. 13–22. doi:10.1145/1449913.1449918.

[59] G. Zhang, H. Ye, Y. Lin, Quality attribute modeling and quality aware product configuration in software product lines, Software Quality Journal (2013) 1–37doi:10.1007/s11219-013-9197-z.
URL http://dx.doi.org/10.1007/s11219-013-9197-z

[60] K. Schmid, R. Rabiser, P. Grünbacher, A comparison of decision modeling approaches in product lines, in: Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems, VaMoS '11, ACM, 2011, pp. 119–126. doi:10.1145/1944892.1944907.

[61] E. H. Forman, S. I. Gass, The Analytic Hierarchy Process: An Exposition, Operations Research 49 (4) (2001) 469–486. doi:10.2307/3088581.

[62] S. Liaskos, On eliciting contribution measures in goal models, in: Proceedings of the 2012 IEEE 20th International Requirements Engineering Conference (RE), RE '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 221–230. doi:10.1109/RE.2012.6345808.

[63] P. Harker, Incomplete pairwise comparisons in the analytic hierarchy process, Mathematical Modelling 9 (11) (1987) 837 – 848. doi:http://dx.doi.org/10.1016/0270-0255(87)90503-3.

[64] P. Berander, P. Jnsson, Hierarchical cumulative voting (hcv) - prioritization of requirements in hierarchies., International Journal of Software Engineering and Knowledge Engineering 16 (6) (2006) 819–850.

[65] F. Palma, A. Susi, P. Tonella, Using an smt solver for interactive requirements prioritization, in: Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, ESEC/FSE '11, ACM, New York, NY, USA, 2011, pp. 48–58. doi:10.1145/2025113.2025124.

[66] S. Vinay, S. Aithal, G. Sudhakara, A quantitative approach using goal-oriented requirements engineering methodology and analytic hierarchy process in selecting the best alternative (2012). doi:10.1007/978-81-322-0740-$5_5$4.

[67] M. Ghallab, C. K. Isi, S. Penberthy, D. E. Smith, Y. Sun, D. Weld, PDDL - The Planning Domain Definition Language, Tech. rep., CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control (1998).