Stratified Analytic Hierarchy Process: Prioritization and Selection of Software Features

Ebrahim Bagheri[1,3], Mohsen Asadi[1,2], Dragan Gasevic[1], Samaneh Soltani[1]

[1]Athabasca University, Canada
[2]Simon Fraser University, Canada
[3]National Research Council Canada
{ebagheri, dragang}@athabascau.ca, masadi@sfu.ca, soltanisa@gmail.com

Abstract. Product line engineering allows for the rapid development of variants of a domain specific application by using a common set of reusable assets often known as *core assets*. Variability modeling is a critical issue in product line engineering, where the use of feature modeling is one of most commonly used formalisms. To support an effective and automated derivation of concrete products for a product family, staged configuration has been proposed in the research literature. In this paper, we propose the integration of well-known requirements engineering principles into stage configuration. Being inspired by the well-established Preview requirements engineering framework, we initially propose an extension of feature models with capabilities for capturing business oriented requirements. This representation enables a more effective capturing of stakeholders' preferences over the business requirements and objectives (e.g.,. implementation costs or security) in the form of fuzzy linguistic variables (e.g., high, medium, and low). On top of this extension, we propose a novel method, the Stratified Analytic Hierarchy process, which first helps to rank and select the most relevant high level business objectives for the target stakeholders (e.g., security over implementation costs), and then helps to rank and select the most relevant features from the feature model to be used as the starting point in the staged configuration process. Besides a complete formalization of the process, we define the place of our proposal in existing software product line lifecycles as well as demonstrate the use of our proposal on the widely-used e-Shop case study. Finally, we report on the results of our user study, which indicates a high appreciation of the proposed method by the participating industrial software developers. The tool support for S-AHP is also introduced.

1 Introduction

A key aspect of software product line engineering is capturing the common characteristics of a set of software-intensive applications in a specific problem domain [1]. Product line engineering allows for the rapid development of variants of a domain specific application by using a common set of reusable assets often known as *core assets*. Such an approach supports the management of commonality as well as variability in the software development process [2][3]. Feature modeling is an important conceptual tool that offers modeling support for software product lines. It provides for addressing commonality and variability both formally and graphically, describing interdependencies of the product family attributes (features) and expressing the permis-

sible variants and configurations of the product family.

One of the important issues in any of the feature modeling methodologies is the selection of the best and at the same time allowable combination of features that would satisfy the mission of the target application and the stakeholders' requirements. Many researchers in the software product line domain have proposed techniques to deal with these challenges and produce appropriate software product line configurations [4][5][6]. In order to handle a large number of comparisons during the configuration process, in many cases, a configuration is gradually developed in several stages breaking down a large amount of required feature comparisons into a set of consecutive stages. In each stage, a subset of preferred features are selected and finalized and the unnecessary features are discarded yielding a new feature model whose set of possible configurations are a subset of the original feature model. This feature model is referred to as a *specialization* of the feature model and the staged refinement process constitutes *staged configuration* [6].

For an effective staged configuration process there is a need to systematically manage different business-oriented requirements, which will drive the process of staged configuration. This can be addressed by extending feature modeling languages with attributes reflecting business-specific concerns (e.g., security and implementation costs). In this paper, we propose such an extension to cardinality-based feature models in Section 2. While this expressivity might appear quite useful, when it is combined with feature models, it can sometimes be a double-edged sword. That is, such comprehensiveness makes the selection of the best set of features for a particular application difficult. Therefore, it is important to understand the relative importance of features and business-specific concerns of a product family from various perspectives. The difference in the importance and priority of features in dissimilar domains would allow for the creation of application specific preference ordering between them.

It is clear that for an advanced staged configuration process, in addition to the use of feature models, there is a need to both capture and leverage business-oriented requirements more effectively. Such a support method must enjoy three important characteristics to be of interest to the professional software engineers: *i) It must be easy, straightforward and fast to use*: Complicated methods that take too many factors into account and are hence slow often fail to become prevalent in industrial settings; *ii) It should provide correct, reliable and provable/repeatable outcomes*: Provability is an important factor in any software engineering process, since the underlying reasons for certain decisions need to be traceable back to their original causes and roots for future decision making purposes; iii) *It needs to be able to effectively communicate with and capture the intentions of the process stakeholders*: This is a very important success factor as stakeholders' satisfaction plays a key role in the continuity of the project.

In this paper, we introduce a new method called the Stratified Analytic Hierarchy Process (S-AHP) for prioritizing (ranking) and filtering the features of a product family to enhance and expedite the feature selection and product configuration process (Section 3). S-AHP is basically concerned with finding the most appropriate set of features that need to be included in the final software product, given the stakeholders' requirements and their important goals and objectives. The output of S-AHP will be the input of typical feature modeling configuration algorithms that specialize a feature model based on the stakeholders' requests and relevant integrity constraints. S-AHP is based on a pair-wise comparison scheme, which although is simple to use and

straight-forward to comprehend for software practitioners, it can at the same time significantly reduce the number of required comparisons from an otherwise computationally explosive number of possibilities. Furthermore, it provides means for tracing between stakeholders' objectives and the feature selection decisions. To support for the systematic use of the proposed method, we show how the proposed method can be incorporated into the current SPL lifecycles and outline the tooling support that we have provided for our method (Section 4). Further, we describe a detailed case study illustrating the proposed method (Section 5). Before concluding the paper, the results of a user study that we conducted to evaluate S-AHP will be provided (Section 6) and relevant related work will be reviewed (Section 7).

2 Extending Cardinality-based Feature Models

The cardinality-based notation for feature modeling integrates four existing extensions of the FODA notation: feature cardinalities, group cardinalities, feature diagram references, and attributes [2]. Cardinality-based feature models are based on a meta-model that defines their abstract syntax [6]. In Figure 1, we depict a modified version of its metamodel where some important concepts for the purpose of our work have been included and two new concepts are added to the meta-model. As it can be seen in the extended meta-model, each feature can be annotated with one or more *concerns*. We have adopted the concept of concerns from the Preview framework in the multiple-viewpoint requirements elicitation domain [7]. The Preview framework has improved the requirements elicitation process by the explicit identification of the importance of organizational needs and priorities via the recognizing of *concerns*. The role of concerns in Preview is to concentrate on factors that are central to the success of a system under development. Similarly we employ concerns within the feature selection process to understand and evaluate the factors that are central to the final product's success. Therefore, in order to enhance the selection of features for a particular application, during the feature modeling process we should take the concerns of the business requirements and high-level objectives of the stakeholders and their relative importance and priorities into account. Similar to Preview, used for requirements analysis of a single system, we consider concerns as the high-level strategic objectives of the application domain and product family stakeholders. Hence, they can be used to ensure consistency and alignment between the vital goals pursued by the design of the product family and the product family features.

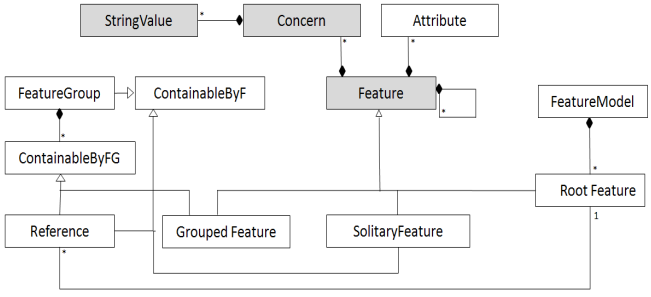


Fig. 1. A modified meta-model for the cardinality-based feature models.

Concerns should be expressed at a high-level of abstraction to avoid overlap with some of the actual features of the product family. Examples of concerns can include but are not limited to *implementation cost, time, risk, volatility, customer importance,* and *penalty*. Furthermore, each concern can be described by a textual description tag, referred to in this paper as a *qualifier tag*. For instance, options available for cost could be *expensive*, *affordable*, and *cheap*. In this way, each feature will be assigned several concerns based on the discretion of the modeler. Each concern can be further described by a tag qualifying that concern. A good choice for qualifier tags would be the use of fuzzy linguistic variables such as high, medium, and low. This will give the modeler the option to qualify assigned tags for different concerns in a similar way.

3 Business Centered Staged Configuration

In this section, we introduce our proposed method (S-AHP) that satisfies the requirements mentioned in the introduction for an efficient feature selection process and considers the stakeholders' concerns, goals and strategic objectives during its course. Our proposal is based on the Analytic Hierarchy Process (AHP) [8], which is a well-established framework in the domain of decision theory.

3.1 Analytic Hierarchy Process

AHP [8] is a rather easy-to-use pair-wise comparison method that computes relative rankings of various phenomena based on the judgments of its users. Compared with methods that are based on absolute value assignment such as scale-based (1-to-10) rankings and voting schemes [1], AHP is less susceptible to judgmental errors and inaccuracy [9]. Furthermore, AHP provides complete justification for its ranking results based on the provided comparisons. AHP undertakes a pair-wise comparison process between the objectives of a study. AHP takes a matrix in which each row of the matrix contains observations of the stakeholders' judgments on the relative importance of a phenomenon compared to some other phenomena. Given this matrix, AHP computes the relative importance and ranking of the available phenomena. The steps to perform AHP are summarized as follows:

1. Let $M\ (n, n)$ be an input square matrix and $m_{i,*}$ refers to the i^{th} row of M and $m_{*,j}$ refers to the j^{th} column of M. Each cell in the matrix such as $m_{i,j}$ represents the relative comparison value between the phenomena i and j;
2. The columns of M are normalized in such a way that each cell is divided by the sum of the values in its column;
3. The eigenvalues of the normalized matrix are calculated;
4. The relative importance and rank of $m_{i,*}$ is computed as $R(m_{i,*}) = (\Sigma_{0<j<|n|} M[i, j])/n$.

By performing the above steps, AHP computes a rank for each row, where each row corresponds to some phenomenon. Suppose AHP is employed to rank the n features present in a given feature model. AHP will take a square matrix with size of n as a starting point. Each cell in the matrix will contain a relative importance of a feature regarding the other features. Therefore, in order to compute the relative ranking of

each feature, AHP will need to perform $n \times (n-1)/2$ comparisons. Since complicated feature models have a large size and some features are not comparable with the others, it would not be suitable to use AHP directly for feature ranking. In our proposed S-AHP method, we employ concerns attached to the features to reduce the number of required comparisons and to make the comparison of heterogeneous features feasible.

3.2 Formalizing the Configuration Process

Generally, the feature model and its annotated information (i.e. concerns and concerns tag values) in our context can be formalized as a triple $<F, C, V>$ where

- F is the initial feature model;
- C is the set of concerns relevant for the feature model (e.g. implementation cost and security);
- V is the set of concern tag values (e.g. high, low, medium)

Based on this triple, we define the feature configuration process as follow:

$$< F, C, V > \xrightarrow[\text{Feature specialization stage}]{\text{Concern prioritization stage}} < F', C', V' > \xrightarrow{\text{feature ranking stage}} < F', C', V' >$$
$$\xrightarrow{\text{Feature specialization stage}} < F'', C', V' >$$

The specialization process starts given the initial feature model, the concern set, and the set of tag values relevant to each concern. First, the concern prioritization stage is performed by executing AHP to identify the desirable set of business concerns (C′) from the stakeholders' perspective. This stage gathers the relative importance of each concern in comparison to the other concerns from the involved stakeholders. This would allow us to filter out the least significant concerns and their corresponding tag values from V and a new set V' is created. Moreover, features whose concern set are a subset of the removed concerns, and their deletion does not violate the feature model constraints are removed from the feature model, which entails the creation of a new feature model F'. Then, the relative importance of the concern tags is calculated. This process is done by employing the AHP algorithm with the matrix containing the values of relative importance for each concern tag. Finally, the calculated rank for each of the tag values are applied to the relevant features. In cases when a feature has been annotated with more than one concern, the representative rank value of that feature is selected based on the users' opinion. Finally, the feature specialization step is done iteratively to select or remove features from the feature model (F') based on their ranks and constraints.

3.3 Feature Configuration Stages

As it can be seen from the process explained above, ranking of features is performed in two layers, which create a stratified (layered) process: i) the identified concerns are prioritized and the less important concerns are filtered out; and ii) the qualifier tags of the remaining selected concerns are compared and prioritized. The structure and formulation of S-AHP is such that it significantly reduces the number of comparisons required for ordering the features of a given feature model. This section describes the stages performed to produce a valid feature model configuration with S-AHP.

3.3.1 Concern Prioritization Stage. This stage aims at comparing and ranking the list of concerns for a specific target application. All defined concerns on the feature model and their relative importance for the target application are considered, and the pair-wise comparison process (AHP) is undertaken to produce a ranked list of concerns. It can be realized as follows:

1. Assume C is set of concerns. A square matrix $P_{|C|\times|C|}$= *{P[i, j] =* α *| 1* $\leq$ *i, j* $\leq$ *|C| and* α *is relative importance of concern i to concern j}* is created. Traditionally within AHP, the values 1, 3, 5, 7, and 9 have been used to represent the degree of importance showing *equal value, slightly more value, strong valued, very strong value* and *extreme value*, respectively; Each cell in the P such as $p_{i,j}$ contains one of the values from 1, 3, 5, 7 and 9 and shows how significant concern *i* is with respect to concern *j;*
2. The steps of AHP are performed on matrix *P* where $P_{i,*}$ (i.e., row *i* of matrix *P*) shows the relative importance of concern *i* with respect to all other concerns;
3. The priority of each concern is calculated and a relative ranking for the list of concerns is developed.

Assuming that we have four concerns called con1 to con4, the first step creates a 4×4 matrix whose entries show the relative importance of the concerns from both the stakeholders' perspective and technical value (See Table 1-a). For instance, P [1, 3] =7 would show that concern 1 is slightly more important than concern 3 and hence, we would also infer that P [3, 1] =1/7.

Table 1. Example concern prioritization steps

a) relative importance of concerns

	Con1	**Con2**	**Con3**	**Con4**
Con1	1	3	7	5
Con2	1/3	1	3	3
Con3	1/7	1/3	1	1/3
Con4	1/5	1/3	3	1

b) normalized values

	Con1	**Con2**	**Con3**	**Con4**	**Sum**
Con1	0.6	0.65	0.5	0.53	2.28
Con2	0.18	0.21	0.21	0.32	0.92
Con3	0.08	0.06	0.07	0.03	0.24
Con4	0.12	0.06	0.21	0.1	0.49

Once the matrix is created, AHP is executed to estimate the eigenvalues of the matrix (Table 1-b). Since we have four concerns, the division of the last column (Sum) will result in the relative significance of each concern: [0.58, 0.23, 0.06, 0.13]. This shows that the ordering between the four concerns is con1>con2>con4>con3. The stakeholders will have the chance to filter out the lesser important concerns based on the ranking outcome.

3.3.2 Feature Ranking Stage. This steps aims at ranking the relative importance of the qualifier tags of the remaining concerns and uses them to rank the available features within the feature model. This will provide a final ranking over the most important concerns and their most significant qualifier tags. Given such a ranking, each feature can be prioritized based on the significance of its annotated concerns. So, the idea is that features that have more important concerns attached to them are more important than the others.

The process for ranking qualifier tags is similar to ranking concerns where qualifier tags sit on the rows and columns of the developed matrix, instead of concerns.

1. Let *V´= {t | t is possible values of c and c*$\in$ *C´}*;
2. A Matrix $M_{|V1\times|V1}$= *{M [i, j] =*β *| 1* $\leq$ *i, j* $\leq$ *|V1| and* β *is relative importance of the tag concern i to tag concern j}* is developed;

3. AHP is executed over the matrix M where $M_{i,*}$ shows the relative importance of qualifier tag i with respect to all other tag concerns;
4. Rankings of features are calculated by applying a predefined function (i.e. minimum, maximum, or mean) on the qualifier tag value ranks for each feature. The predefined function is a function that is used to select a rank for a feature when a feature has more than one concern. This function is defined by the domain engineer for the features based on stakeholders' input.

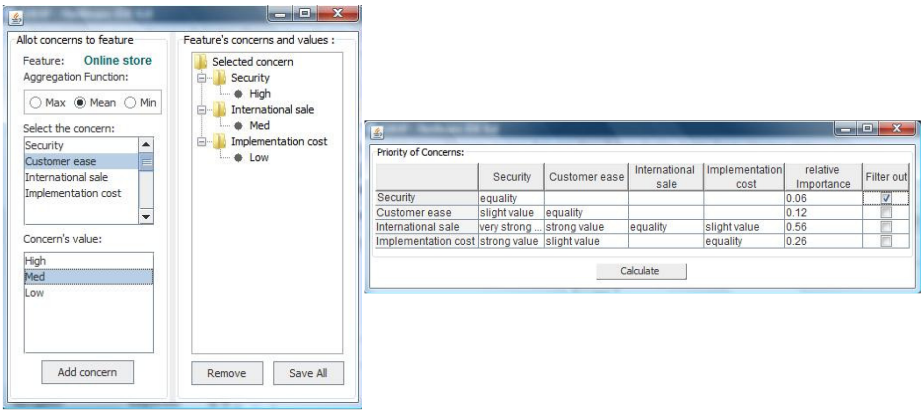
One important point to note is that some features might be annotated with more than one concern, e.g., a feature can be described as having a high performance (with the relative importance of 0.54) and low security (with the relative importance of 0.1). This annotation means that the feature will provide high system performance; a concern liked by the stakeholders (0.54), but at the same time may cause security problems, which is not liked by the stakeholders (0.1). The decision on which value to assign to the feature would depend on the modelers' strategy. A conservative modeler would fear for a security breach as a result of the inclusion of this feature and would hence take a minimization strategy and would select the lower value as the representative importance for this feature (0.1), but a modeler designing the application for an already secure environment, might adopt a more avant-garde strategy and assign higher importance (0.54) to the feature. It is also possible to take the average value as the representative (0.32).

3.3.3 Feature Model Specialization Stage. After ranking the available features based on their relative significance to the target application stakeholders, we start conducting feature specialization steps iteratively. Each specialization step uses feature ranks to select the most appropriate features for that specific application. During the different iterations of the feature model specialization stage, selection or filtration of features can be performed based on the developed rankings in the previous stage (i.e., the feature ranking stage). For this stage, we adopt the six steps introduced in [6] to limit the configuration space and adapt some of the steps of the adopted process. The steps defined in [6] are as follow: i) *Refining feature cardinalities***:** it eliminates or decreases the cardinality of features; ii) *Refining group cardinalities***:** it decreases the interval of possible grouped features that can be chosen within a group; iii) *Removing a grouped feature from a feature group:* this step removes one of a group sub-features with all of its decedents. In our process, this step removes the features which have the least importance calculated by the feature ranking stage. iv) *Assigning an attribute value:* values are assigned to uninitialized attributes during this step; v) *Cloning a solitary sub-feature*: this step provides the possibility of cloning a feature as well as its entire sub-tree.

Feature model specialization is performed iteratively. It gradually moves the specialized feature model towards its final configuration. The main contribution of our S-AHP process is that it provides suitable rankings for the available features that can be selected by the stakeholders in each stage. This ranking of features is based on the business objectives and high-level goals of the stakeholders and therefore facilitates the feature selection process.

Finally, in order to support the proposed configuration process, we have created a prototype extending the Feature Model Plug-in (*fmp)* [13] with required functionality for managing the concerns and supporting our configuration process. *fmp* is a widely used plug-in for feature modeling and configuration. A screenshot of the tool's dialog

for editing concerns and their qualifier tags is given in Figure 2a, while the part of the *fmp* extension for filtering out the most important concerns is shown in Figure 2b. Similarly, the tool allows users to effectively perform the other steps of S-AHP.



a) b)

Fig. 2. S-AHP extension of the Feature Plug-in: a) Editing concerns and associating with features; b) Concerns prioritization

4 Process Changes in Software Product Line Methodology

Software product line methodologies have commonly defined two lifecycles, namely *Domain Engineering* and *Application Engineering* [10]. In domain engineering, the common assets, family reference architecture and the variability models are developed. Afterwards, in the application engineering lifecycle, the common assets are reused and variability models are configured to produce a product family. By applying our approach for modeling the concerns within the feature models, some slight changes need to be made on the processes within the software product line lifecycles. In order to provide a clear understanding of these changes, we mention the phases in each lifecycle and highlight our changes and extensions in each of these phases. Due to the limited space for this paper, Table 2 only summarizes the activities added by S-AHP to the well-known software product line lifecycles.

Table 2. Process changes in software product line process by S-AHP

Life-cycle	Phase	Traditional Activities	Added activities by S-AHP
Domain Engineering	Product line scoping	• Product portfolio scoping • Domain scoping • Asset scoping	• Examine strategic managements' viewpoints • Identify major concerns
	Domain Requirements Engineering	• Family requirements elicitation • Family requirements analysis • Family requirements specification, • Family requirements validation and verification	• Refine concerns (break down to sub-concerns) • Define concerns tag values
	Variability Modeling (Feature Modeling)	• Identifying features • Identify feature relation • Model features	• Annotate features with concerns • Assign tag values to features concerns, if it is determined.

			• Define aggregation function when there are more than one concern in each feature
	Domain Design	• Design reference architecture • Detail design of assets	• Update feature concerns and their values based on design information
	Domain Realization and Testing	• Make/buy/mine/commission • Test	• Update feature concerns and their values based on implementation information
Application Engineering	Application Requirement Engineering	• Application requirements elicitation • Application requirements analysis • Application requirements specification • Application requirements validation and verification • Reuse family requirement model	• Indicate relative importance of concerns for the specific product
	Application Design	• Binding variability based on application requirements • Verify and Validate specialized feature model • Automatically create application	• Ranking the concerns • Ranking tag values • Ranking features • Bind variables based on the provided ranks
	Application realization and testing	• Complete application • Test Application • Deploy application	N/A

5 Case Study

The online store is a commonly used example in the literature. This example is employed to represent the behavior of feature modeling frameworks [12]. Figure 3 depicts a small feature model designed to depict some of the aspects of such an online store. In this model, the features have been annotated with four concerns, namely security, customer ease, international sale, and implementation cost. The concerns are qualified using high, medium and low values; therefore, the modelers are able to express their belief with regards to the features in the context of these concerns. The concerns and their qualifier tags are shown in the lower part of the leaf features and the legend explaining their interpretation is placed in the top left part of Figure 3. For instance, using cash as a payment method has been considered to be highly secure, terrible for international sales and inexpensive from an implementation perspective.

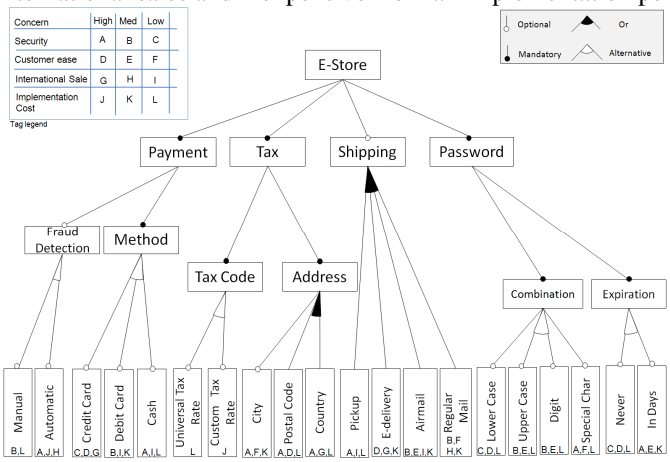


Fig. 3. A small feature model for an online store

Furthermore, we can assume that the feature model is accompanied by the following four dependency constraints: 1) cash payment implies pickup shipping; 2) credit card payment implies automatic fraud detection; 3) never changing the password implies the inclusion of special characters in the password; and 4) manual fraud detection excludes credit card payment. These constraints will be automatically applied if any of their antecedents are included in the feature model through the specialization process. Now, supposing that an actual online store needs to be developed based on the requirements, needs and goals of its target audience, S-AHP can be used to select the best set of features. In case AHP is used, since there are 20 open features in the model, 190 comparisons need to be made. As will be shown S-AHP requires a much less number of comparisons. It should be noted that in a real-world large-scale feature model, the number of features and concerns are much higher. In the first step, the concerns are ranked through a pair-wise comparison process as shown in Table 3.

Table 3. Relative importance of concerns as well as the final concerns ranks

	Security	Customer ease	International sale	Implementation cost	Normalized Sum	Importance
Security	1	1/3	1/7	1/5	0.22	0.06
Customer ease	3	1	1/5	1/3	0.4802	0.12
International sale	7	5	1	3	2.2305	0.56
Implementation cost	5	3	1/3	1	1.0505	0.26
Sum	16	9.33	1.67	4.5	3.9812	1

We can consequently calculate the relative importance of the concerns with each other, which shows the following order (Table 3): International sale>Implementation cost>customer ease>security. It can be inferred that the stakeholders are interested in focusing on the international sale aspect and implementation costs of the product while customizing the feature model. It is possible to filter the less important concerns and proceed to the second stage. Based on the filtration, the new matrix (Table 4) is developed by inquiring the stakeholders about their preferences on the selected concern's qualifier tags (see Figure 3 for letter interpretations).

Table 4. Tag values ranks calculated by S-AHP

	G	**H**	**I**	**J**	**K**	**L**	**Importance**
G	1	5	7	7	5	3	**0.4**
H	1/5	1	5	5	3	1	**0.15**
I	1/7	1/5	1	3	3	1/7	**0.01**
J	1/7	1/5	1/3	1	1/5	1/7	**0.04**
K	1/5	1/3	1/3	5	1	1/5	**0.1**
L	1/3	1	7	7	5	1	**0.3**

It is clear from the developed ranking that the stakeholders are interested in having high international sale in their online store. In addition, they are not interested in spending a lot of money on technical implementation issue; therefore, they should select those features which would result in high international sale with low implementation cost. Based on this information, a ranking of features can be simply developed. As an example and based on averaging the qualifier tag values of the concerns attached to each feature, the manual fraud detection method (0.3) would be more desirable than its automatic (0.1) counterpart, and the credit card-based payment method

(0.4) is also more attractive than the other methods (cash: 0.15; debit: 0.05). So, the modelers would select manual fraud detection and credit card based payment features; however, this selection would be in conflict with one of the dependency rules (manual fraud detection excludes credit card payment). In this case, the modelers will probably decide to remove the manual fraud detection feature since it is less important than the credit card-based payment method (0.4>0.1) and also include automatic fraud detection to satisfy the dependency rule (credit card payment implies automatic fraud detection). This decision can conclude the first stage of the configuration process, which results in the selection of the appropriate features for payment method and fraud detection; hence, peer features to the credit card payment method and the automatic fraud detection that have not been selected in this stage will be removed from the online store feature model.

The first specialization in the staged configuration of the online store feature model can be seen in Figure 4. The feature model can be further specialized through more rounds of refinement and feature selection based on the developed ranking, so that the most desirable configuration of the feature model is achieved. An important point here is the significant difference between the required number of comparisons in AHP and S-AHP. For this rather small feature model, AHP would require 190 comparisons, while S-AHP needs 6 comparisons in its first layer and 15 comparisons in the second layer, making it 21 comparisons in total. It is clear that as the size of the feature model grows larger, the efficiency and utility of S-AHP will become even more noticeable.

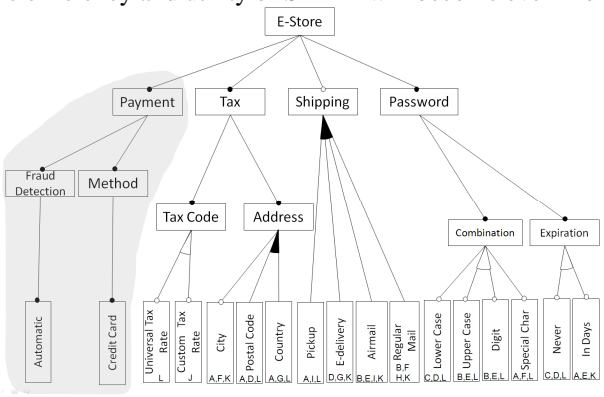


Fig. 4. A specialization of the online store feature model. The shaded area shows the result of the specialization performed in the first stage

6 User Evaluation

It is also important to see how S-AHP is perceived by software designers and practitioners to be applied to real world problems. For this purpose, we have evaluated the usefulness of S-AHP by providing S-AHP to software practitioners and asking for their feedback on how helpful they find it for the tasks that they usually perform. Eleven software practitioners were invited to participate in the study whose range of software development experience was from 6 to 15 years, with an average of 9.5 years. The participants expressed that they were familiar with various application de-

sign and planning methodologies including SSADM, OOP, and RUP, and were involved in major industrial design projects spanning the areas of geographical information systems, accounting, mobile applications, customer relation management and others. The participants were asked to provide their evaluation of the S-AHP method from their perspective for its usefulness, efficiency, easiness for use, and practicality. The participants were asked eight questions by using a seven-level Likert scale where 1 being strongly unfavorable to the concept and 7 being strongly favorable to the concept. The following eight questions were asked. We provide the average value that S-AHP received for each of these questions along with the question: 1) It is simple to use S-AHP: ***6.0***; 2) I can effectively complete my work using S-AHP: ***5.0***; 3) I am able to complete my work quickly using this method: ***5.1*** ; 4) I am able to efficiently complete my work using this method: ***5.3***; 5) It was easy to learn to use S-AHP: ***6.1***; 6) I believe I became productive quickly using S-AHP: ***4.8***; 7) I believe S-AHP increase the chances of selecting the most suitable/important software features: ***5.8***; 8) Overall, I am satisfied with this method: ***5.8.*** The participating software practitioners in the study felt that S-AHP is effective in helping them more conveniently choose the best set of software features, and that its simplicity makes it practical for real-world software development environments. In addition, it was pointed out that the proper selection of the most relevant set of concerns for each project has effect on the final set of selected features; therefore, close attention needs to be made in order to select the best concerns. In the future reports, we will provide a more detailed statistical analysis of the collected data and the collected qualitative insights.

7 Related Works

The research community has put much emphasis on developing methods for the syntactical validity checking of model configurations. These methods mainly focus on forming grammatical correspondences for the graphical representation of feature models and perform automated syntactical analysis (e.g. using AI configuration techniques such as SAT or CSP solvers) based on the model dependency rules and constraints [12][14]. However, considering the strategic objectives of the stakeholders and the specific domain requirements can create a semantic validation process that will ensure that the requisites of the target audience is met [15]. Such semantic validation process can complement syntactical consistency checking methods in order to create a valid and at the same time useful final configuration of a feature model.

Some researchers have proposed to annotate features of a model with priority information, which will be later used to select high priority features at specialization time [2]. The idea of prioritization of features is interesting, but it does not seem to be suitable for the context of feature models and product families. This is because feature priorities change based on the context where the feature model is being specialized and configured; therefore, priority values for features should be developed during specialization, which is supported by the S-AHP method. As an example, a password control feature for an application would take high priority on a network installation, while it would have less priority in a standalone unique installation. Therefore, single priority values are not sufficient and methods that support dynamic priority assignment are required.

Closely related to the idea of feature selection, the requirement engineering community has significantly contributed to the development of requirement prioritization techniques for requirement selection [16][17]. In contrast to feature modeling techniques, requirement specifications are commonly defined in an unstructured (free-text) or semi-structured forms; therefore, semantic validation of requirements by a human requirement engineer is more viable than automatic syntactic correctness checking. In this context, one suggestion for prioritizing requirements has been the use of requirement priority groups as a way to nest similar requirements together and create an internal rank for requirements in each group. This approach is not so suitable for feature models, since the structure of the feature model tree is of high importance while the decision about the priority and specialization of the feature model is being made; therefore, creating priority groups that would become feature tree structure oblivious might result in irrelevant feature model configurations.

Similarly, hierarchical Analytic Hierarchy Process (AHP) [8] has been proposed in the requirement engineering literature to create a structure for interrelated requirements. In this method, only requirements at the same layer are compared with each other and hence reduce the total number of required comparisons in contrast to AHP; however, this method is not so suitable for feature models since the features in the interim layers of the feature tree are usually not included in the feature comparison process, which removes the need for a hierarchical structure; therefore, this method does not actually reduce the number of required comparisons for a feature model. From a different perspective, the Bubble sort technique has been used order the requirement statements. Bubble sort is in essence very similar to AHP with the slight difference that requirement comparisons are made to determine which requirement has a higher priority, but not to what extent. It is clear that Bubble sort suffers from similar issues to AHP, e.g., the large number of required comparisons. There have been proposals to reduce the number of required comparisons in comparison-based techniques, which are generally referred to as incomplete pair-wise comparison methods [9]. These techniques are based on some local and/or global stopping rule, which determines when further comparison will not reveal more useful information with regards to the prioritization of the options. Such techniques can be beneficial if used along with techniques such as AHP, S-AHP, Bubble sort and others. Additionally, (hierarchical) cumulative voting has been used to prioritize requirements where top vote-getter requirements are prioritized higher than the others. One of the drawbacks of that approach is that as the number of requirements (options) increases, it becomes very hard for the stakeholders (voters) to select the best voting tactic, which would reveal their preferences about the highest priority requirements [18].

Other techniques such as our own experience with the formulation of multi-attribute utility theory for architectural tradeoff decision making have shown that these methods are too complicated and are therefore hard to understand by the stakeholders and the practitioners [1, 2]. This observation necessitates the development of an easy–to-use and straightforward yet efficient method for feature selection.

8 Concluding Remarks

Proper feature selection and feature model customization requires the systematic consideration of the stakeholders' needs and objectives as well as the constraints and dependency rules of the feature model. To this end, we have introduced a useful set of methodical activities to support feature selection, called S-AHP. The aim of S-AHP is to communicate with the stakeholders and understand their priorities with regards to business objectives and high-level goals and to use this information to find the most important features of a feature model for the stakeholders. Here, we outline how S-AHP satisfies the requirements of an efficient feature selection method based on the challenges and characteristics introduced earlier in the article:

1. S-AHP is able to tame the relatively large number of comparisons needed for developing the ranking between the features. This is achieved by a layered pair-wise comparison of the feature annotations, namely the concerns and their qualifiers. We showed that for a relatively small example a reduction from 190 to 21 comparisons was reached.
2. It overcomes feature incompatibility and the comparison problem by introducing the concept of concerns. Since features are annotated with multiple concerns they can be indirectly compared and ranked. This is due to the fact that concerns are comparable, and their priorities can be effectively propagated to their corresponding features.
3. This method prevents the appearance of different comparison scales by clearly employing and defining default comparison scales (1, 3, 5, 7, 9). A strategy to use these values can serve as a consistent comparison scale to be employed by the stakeholders and the practitioners.
4. The activities within S-AHP are very easy to perform and are based on a simple pair-wise comparison method. The relative simplicity of this approach is advantageous because it does not add complexity to the complicated feature selection process. Furthermore, it can be quickly and inexpensively implemented in a spreadsheet program such as Microsoft Excel without the need to purchase commercial decision support tools.
5. S-AHP is also able to effectively communicate with the stakeholders at a higher level. This is achieved thanks to the concerns and their qualifier tags. With the employment of concerns, the stakeholders do not need to be aware of the structure or meaning of the feature models, but would be able to understand whether the customized feature model satisfies their requirements and objectives.

In summary, S-AHP is a simple yet effective approach for ranking and filtering various features of a product family, which expedites the product specialization and staged configuration of a feature model. Its main advantages are its simplicity, practicality and its involvement with the target audience and product stakeholders that allows for the alignment of the final product with the strategic objectives and goals of the stakeholders. Furthermore, due its simplicity the development of tool support for this approach is quite easy and very inexpensive, and even many freely available tools for AHP can be easily used in the context of S-AHP. Currently, S-AHP only supports for a uni-dimensional pair-wise comparison of the concerns where the comparison of two concerns is represented by a single value from taken from the comparison scale.

As future work, we are interested in investigating whether multi-dimensional extensions of S-AHP would be beneficial that would simultaneously incorporate matters such as cost and value in the comparisons in order to perform tradeoff decision making.

9 References

[1] P. Clements and L. Northrop, *Software product lines: practices and patterns*, Addison-Wesley Longman Publishing, 2001.

[2] K. Czarnecki, S. Helsen, and U. Eisenecker, "Formalizing cardinality-based feature models and their specialization," *Soft. Proc. Improv. and Practice*, vol. 10, 2005, pp. 7–29.

[3] P. Heymans, P. Schobbens, J. Trigaux, Y. Bontemps, R. Matulevicius, and A. Classen, "Evaluating formal properties of feature diagram languages," *IET Soft.*, 2(3), 2008, p. 281.

[4] J. White, B. Dougherty, D.C. Schmidt, and D. Benavides, "Automated Reasoning for Multi-step Software Product-line Configuration Problems."

[5] M, Boskovic, E. Bagheri, D. Gasevic, B. Mohabbati, N. Kavinai, & M. Hatala, "Automated Staged Configuration with Semantic Web Technologies," International Journal of Software Engineering and Knowledge Engineering (in press)

[6] K. Czarnecki, S. Helsen, and U. Eisenecker, Staged Configuration Through Specialization and Multi-Level, Dep. of Electrical and Computer Eng,, University of Waterloo, 2004.

[7] I. Sommerville and P. Sawyer, "Viewpoints: principles, problems and a practical approach to requirements engineering," *Annals of Software Engineering*, vol. 3, 1997, pp. 101–130.

[8] T.L. Saaty, *The Analytic Hierarchy Process*, New York: McGraw-Hill, 1980.

[9] J. Karlsson, S. Olsson, and K. Ryan, "Improving Practical Support for Large-scale Requirement Prioritising," *Requerments Engineering*, vol. 2, 1997.

[10] F.J. Linden,, K. Schmid, and E. Rommes, "Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering," Springer Berlin Heidelberg, 2007.

[11] F. Linden, K. Phol, G. Bockle, E. Sikore, and B. Gunter, *Software Product Line Engineering: Foundations, Principles, and Techniques*, Springer Berlin Heidelberg, 2005.

[12] D. Batory, "Feature models, grammars, and propositional formulas," *Lecture notes in computer science*, vol. 3714, 2005, p. 7.

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

[14] D. Batory, D. Benavides, and A. Ruiz-Cortes, "Automated analysis of feature models: Challenges ahead," *Communications of the ACM*, vol. 49, 2006, p. 47.

[15] E. Bagheri and A.A. Ghorbani, "The analysis and management of non-canonical requirement specifications through a belief integration game," *Knowledge and Information Systems*, vol. 22, 2009, pp. 27-64.

[16] A. Perini, F. Ricca, and A. Susi, "Tool-supported requirements prioritization: Comparing the AHP and CBRank methods," *Inform. and Soft. Tech.*, vol. 51, 2009, pp. 1021–1032.

[17] A. Aurum and C. Wohlin, *Eng. and Managing Software Requirements*, Springer, 2005.

[18] P. Berander & P. Jönsson, "Hierarchical Cumulative Voting (HCV) – Prioritization of Requirements in Hierarchies," *Int'l J. Soft. Engi. & Know. Eng.*, vol. 16, 2006, pp. 819-849.