From Intentions to Decisions: Understanding Stakeholders’ Objectives in Software Product Line Configuration

Mahdi Noorian\textsuperscript{1}, Ebrahim Bagheri\textsuperscript{2}, Weichang Du\textsuperscript{1}
University of New Brunswick, Fredericton, Canada\textsuperscript{1}
Ryerson University, Toronto, Canada\textsuperscript{2}
m.noorian@unb.ca, bagheri@ryerson.ca, wdu@unb.ca

Abstract—Software Product Line (SPL) engineering promotes the systematic and large-scale reuse of design and implementation artifacts. Feature models are one of the main artefact of SPL approach which essentially characterize the similar and variant functional and operational specifications of the product family. Given the complexity of the variabilities represented by feature models, it is often hard for the stakeholders to analyze a feature model and identify the features that are most important for their purpose. So, given large-scale software product families, one of the important questions is how and what features should be selected for the target software product from the product family. To address this problem, we adopt concepts from the domain of goal-oriented requirement engineering and base feature selection decisions on software stakeholders’ intentions and expectations. In this work, we propose a framework to automatically map stakeholders’ objectives, which can be captured in the form of goal models, on feature models through the application of semantic analysis methods. Our proposed approach not only provides the means to systematically interrelate feature models and goal models but also helps software practitioners in moving from the stakeholders’ goals and expectations towards domain model feature selection decisions in such a way that a more desirable final product for the stakeholders is developed.

I. INTRODUCTION

A software product family provides the means for capturing the commonalities of all possible products of a given domain and also addresses variability by covering a comprehensive set of dissimilarities between the products [16]. There are basically two lifecycles in software product lines, namely domain engineering and application engineering. While, domain engineering is concerned with the process of understanding the target domain and developing a comprehensive formal representation of the concepts in that domain, application engineering takes the products of the domain engineering process and develops an appropriate application instance by carefully choosing and instantiating the right elements of the formal representation of the domain model. The process of selecting the right set of features and development of a new product from a software product line during application engineering is referred to as the configuration process [8].

Software product line configuration is an important step in application engineering, which can be seen as a constraint satisfaction process throughout which a product is being developed that needs to possess the highest number of desired features and properties and the least number of excessive and undesirable ones from the perspective of stakeholders [4]. The software product line research community has developed effective methods for configuring product line models such as feature models. The basic assumption of these methods and tools is that the set of initial desirable features is already known to the stakeholders or at least to the software product designers [5]. However in reality regardless of the configuration process itself, the selection of the initial set of desirable features is both important and very difficult to do for the stakeholders and product designers. The selection of these features depends on the restrictions placed by and objectives of the stakeholders and the requirements of the target deployment environment. Therefore, an important research challenge is to develop methods or processes that can help identify the set of desirable features to fulfill the stakeholders’ needs.

In reality, stakeholders are more comfortable in explaining their expectations from a software product in terms of their objectives, intentions, and goals rather than very formally explaining the functionality that they expect [1], [9]. Therefore, in the realm of software product line engineering where software products are defined by their features, the gap between the stakeholders’ intention space and the product family feature space needs to be bridged. Kang [11] has mentioned that feature elements represent system functionality in a domain and the authors in [20] indicated that task elements in goal models represent the operation or function that can be defined for satisfying parent goals. Therefore, it is possible to integrate feature and goal models by identifying conceptually related pairs of task and feature elements.

In this work, we are interested in analyzing the early stages of the software product application engineering process, i.e., we investigate how the most suitable set of software product line features can be selected for the product configuration process by examining the stakeholders’ needs and requirements gathered through a standard requirement engineering process. We propose a (semi) automated framework, which systematically maps the stakeholders’ objective and goals (which is in the form of a goal model) to a domain feature model through shared domain Semantic Web ontologies. To this end, we first annotate feature and goal models’ elements with ontological concepts through an annotation process and then integrate these two models by identifying and connecting conceptually related
elements e.g., pair of feature and task elements through a mapping process. Note that, the information in feature and goal models come from domain documents such as interview transcripts, functional requirements documents, strategic planning documents, among others [12]. In the feature and goal modeling process, domain analysts use these documents as a source to derive the elements to design the feature and goal models. Based on this assumption, we posit the existence of either explicit or implicit relation between the domain documents and the elements of feature and goal models. In order to have precise annotations, we analyze such domain documents and associate feature and goal model elements with their relevant supporting texts (the part of the domain documents based on which an element is contextualized). In the annotation process, we benefit from text analysis methods to extract important domain concepts from the associated texts and automatically annotate the feature and goal model elements with relevant ontological domain concepts. In the mapping process, we utilize an explicit semantic similarity function to measure the semantic relatedness of elements in the feature and goal models and connect the conceptually related elements. Finally, through the established mapping links and using diagrammatic reasoning techniques it is possible to trace between the intention and feature spaces and therefore find the most relevant features of the target software product family in the context of the stakeholders’ goals and requirements.

II. BACKGROUND

A. Feature Models

Features are important distinguishing aspects or characteristics of a family of systems [16]. In a feature model, features are hierarchically organized by Structural Constraints which can be typically classified as: 1) Mandatory: a feature must be included in the product if the parent feature is included; 2) Optional: a feature may or may not be included in product if the parent feature is included; 3) Alternative feature group: one and only one of features from the feature group can be selected if the parent node is included; 4) Or feature group: one or more features from a feature group can be included in the product if their parent feature is included. In some case, the cross tree mutual interdependencies among the features exist, thus, additional constraints are often added to feature models and are referred to as Integrity Constraints. The two most widely used integrity constraints are: Includes - the presence of a given feature (set of features) requires the inclusion of another feature (set of features); and Excludes - the presence of a given feature (set of features) requires the elimination of another feature (set of features). Figure 1 depicts a small feature model for the tablet domain.

B. Goal Models

Goal models have been introduced and widely used, as a controlled approach for organizing and structuring stakeholders’ intentions in a graph-like representation [9]. In essence, goal models are fundamentally built over three important concepts, namely goals, softgoals, and tasks. Goals are objectives related to the functional aspects of the system. In contrast, softgoals refer to non-functional or quality attributes of the system. Furthermore, tasks are ways to operationalize stakeholders’ goals.

Goal models are often refined such that high-level goals are expressed through finer grained goals. This is achieved using decomposition links, i.e., each parent node is broken down into smaller child nodes whose disjunction or conjunction will satisfy the parent. In addition, since goals are operationalized through tasks, they are interrelated through contribution links. Tasks can also be refined using decomposition links. Moreover, impact links are used to show to what extent the developed tasks contribute to the satisfaction of a softgoal. Figure 2 shows a goal model representing some of the stakeholders’ expectations from the tablet domain.

III. A FRAMEWORK FOR FEATURE MODEL PRE-CONFIGURATION

The overview of our proposed feature model pre-configuration process is shown in Figure 3. Generally, the domain level processes are involved to create an overarching representation of the domain using feature models. Later, the application level process takes the developed formal models,
The application engineering process is as follows:

- **(D1)** Using an available standard domain analysis process the set of all possible features and their dependencies (structural and integrity constraints) will be identified. As a result of this process, a formal representation of the features of the target domain are developed, which is referred to as the feature space.

- **(D2)** During the feature modeling process, domain documents are used as information sources for extracting feature model elements. Based on this, our second step will create the extended version of the feature model in which the elements in the model are linked with the sections of the domain documents from which the elements are derived from. In other words, each model element will be accompanied by textual snippets from domain documents that justify the existence of the model element.

- **(D3)** Once the feature elements are labeled with their related domain texts, each element will be semantically annotated with domain ontological concepts. We employ an automated semantic text analysis method in order to analyze the elements’ associated texts and cross-reference the elements with a set of concepts that are identified from the texts.

The main focus of the devised processes in domain engineering is to develop a feature model whose elements are extended with semantic concepts. Furthermore, in application engineering the annotated feature model will be used as a basis of the mapping process. In the context of feature model configuration, the application engineering process is as follows:

- **(A1)** Communicates and understands the stakeholders’ needs and requirements by identifying and modeling their intentions and goals through an available standard requirement engineering process (the intention space).

- **(A2)** During the goal modeling process, application engineers use domain related documents to derive goal, softgoal, and task elements. Similar to step (D2), in this step, the derived elements will be extended with their related supporting texts for more clarification.

- **(A3)** Similar to step D3, here we benefit from semantic text analysis methods in order to extract ontological concepts from the associated supporting text and annotate task elements in the goal model accordingly.

- **(A4)** In this step, the main concern is to identify the possible correspondences between the task elements of the intention space with feature elements of the feature space and link them through appropriate mapping links. In the D3 and A3 steps, both feature elements and task elements have already been semantically annotated with concepts from a shared domain ontology. Hence, through the mapping process the pair of semantically related task and feature elements can be identified. Here, pairwise comparisons will be performed on the elements’ representative concepts in order to measure the conceptual relatedness of each pair of elements. This process highlights a set of possible mappings between the models.

- **(A5)** Finally, diagrammatic reasoning techniques [10], [17] are employed to trace between the intention and feature spaces and therefore find the most relevant features of the target software product family in the context of the stakeholders’ goals and requirements. These set of selected features can be used as appropriate input for feature model configuration techniques that given a feature model and a set of input features are able to create a fully configured software product model.

In the following sections, the details behind each of these steps are introduced.

### A. Feature and Task Element Enrichment

In real world practice, domain engineers refer to a set of textual assets such as interview transcripts, functional requirements documents, strategic planning documents, as their information source in order to design and develop a feature and/or goal model. These documents are used as a source of information to extract the model elements like features and define relations among the derived elements [12].

During the design process, the rationale for extracting elements is documented by building traceability links between the derived elements and their supporting texts in the domain documents. These explicit links help to keep track of the reasons behind selecting the developed model elements, which augment the maintainability of the model. Specifically such augmented models will be more understandable and maintainable when they needs to be updated with new identified requirements. Based on this, we consider that there is a traceability link between the identified elements, e.g., features or tasks, and the sources of information that the element is derived from.

In the current literature, the typical approach for mapping tasks and features has been to look at the syntactical similarity or synonymity between feature and task element names. Our approach takes this one step further by trying to find correspondences between feature and task elements by looking at their textual sources. In fact, we benefit from the traceability link between task/features and their textual sources, developed in the element enrichment step.

The result of the feature/task element enrichment is an extended version of feature and goal models in which each individual feature and task elements in the models is connected to its appropriate supporting text. For instance, the “IOS” feature can be extended with “IOS is an OS developed and distributed by Apple Inc.” supporting text. In the following section, we will explain how the associated supporting text can be used in a semantic annotation process for identifying the relevant ontological concepts of each element.

### B. Feature and Task Element Annotation

The main task in this step is to annotate feature and goal model elements with appropriate semantic concepts. For
this purpose, we use each element’s supporting texts that are added in feature/task element enrichment step (D2 and A2). We employ a text semantic analysis method to automatically extract semantic concepts from the texts and annotate the feature/task elements with ontological concepts.

Various types of semantic annotation systems exist, which can be used in our text semantic analysis process. The underlying techniques of such systems rely on a combined use of Natural Language Processing (NLP) and the large scale knowledge bases like DBpedia [6]. In our work, we benefit from Denote [7], a semantic annotation system, which employs each element’s supporting text as input. For a given element’s supporting text, the system automatically detects and identifies the relevant phrases (spots) that are available in the supporting text and relates each of them to appropriate ontological concepts. Accordingly, the element can be annotated with the identified semantic concepts.

C. Mapping Process

The mapping process provides the opportunity to connect feature space with goal space through identifying a set of feature and task elements that are represented by related ontological concepts. The underlying challenge of mapping can be simply viewed as the problem of selecting the feature and the task that are annotated with a set of most similar concepts. The mapping process can automatically identify the mappings in two stages: first, it calculates the relatedness between all feasible combinations of tasks and features. Second, using Integer Linear Programming (ILP) [13], the best mappings are identified.

Stage 1: Calculating semantic relatedness

Let’s assume that feature \( f \in F \) is annotated with a set of concepts \( A_f = \{\langle f, c_1 \rangle, \langle f, c_2 \rangle, \ldots, \langle f, c_n \rangle \} \) and, task \( t \in T \) is annotated with \( A_t = \{\langle t, c'_1 \rangle, \langle t, c'_2 \rangle, \ldots, \langle t, c'_m \rangle \} \). In order to be able to calculate the semantic relatedness between two elements, we will need to measure and quantify the semantic relatedness between their concept sets.

\[
\text{Relatedness}(f, t) = \text{Relatedness}(A_f, A_t). \tag{1}
\]

Equation (2) calculates the semantic relatedness between two elements \( f \) and \( t \) based on their associated concept sets. The similarity between each pair of concepts \( c \) and \( c' \) is calculated through \( SLM(c, c') \) function.

\[
\text{Relatedness}(A_f, A_t) = \sum_{i=1}^{\left| A_f \right|} \left( \max_{j=1}^{\left| A_t \right|} SLM(c_i, c'_j) \right). \tag{2}
\]

As indicated by Strube et al. in [19], the measure for computing semantic similarity between two ontological concepts \( c \) and \( c' \) can be categorized as: 1) path based measures, 2) information content based measures, and 3) text overlap based measures. To explain, path based measures compute the relatedness of two concepts by calculating the number of edges along the path between two concepts within the taxonomy hierarchy. Information content based measures compute the relatedness based on the extent in which two concepts share information. Text overlap based measures calculate relatedness based on the overlap exist between the texts associated to each concept. Here, \( SLM \) function calculates and quantifies the conceptual similarity between two ontological concepts based on the combined measures that are introduced in [19]. Interested readers are encouraged to see this paper for details of the \( SLM \) function.

Stage 2: Find the mapping elements

We adopt the Integer Linear Programming (ILP) [13] method to find the most relevant feature element for each task element. We assume that, at most there is only one relevant feature for each task. This problem can be formulated as one variation of the classic assignment problem [13]. For a given feature set \( F = \{f_1, f_2, \ldots, f_n\} \) and task set \( T = \{t_1, t_2, \ldots, t_m\} \), the mathematical model is as follows:

\[
\text{Maximize} \sum_{i=1}^{n} \sum_{j=1}^{m} \text{Relatedness}(i, j) \cdot x_{ij} \tag{3}
\]

\[
\sum_{i=1}^{n} x_{ij} \leq 1 \quad j = 1, 2, \ldots, m \tag{4}
\]

\[
\sum_{j=1}^{m} x_{ij} \leq 1 \quad i = 1, 2, \ldots, n \tag{5}
\]

Equation (3) represents the objective function, which is the summation of \( x_{ij} \) as the decision variable where, \( x_{ij} = 1 \) if feature \( i \) can be mapped on task \( j \) in the optimal solution and 0 otherwise. The \( \text{Relatedness}(i, j) \) represents the similarity degree of feature \( i \) in compare to task \( j \) which can be calculated based on Equation (1). Equation (4) shows the first set of constraints, which ensure that every feature is mapped on only one task and Equation (5) as a second set of constraints ensure that every task is mapped on a feature. As a result of the mapping process, each task \( t \in T \) will be mapped onto a semantically related feature element \( f \in F \).

D. Feature Selection

In the mapping process step, the stakeholders’ objectives (intention space) is connected to the feature model (decision space) through a set of mapping links. In fact, considering the intention space can be helpful in selecting the most appropriate set of features for the specific purpose of the stakeholders. Here, the decision related to the selection of the best features can be translated into which one of the available features are related to the important goals of the stakeholders. Simply stated, the features that are connected through some path to the highly desirable goals of the stakeholders are considered to be more attractive to be included the final product.

This approach to feature selection can be formally defined through two types of diagrammatic reasoning approaches performed over goal models [15]. The first approach is called the backward label propagation algorithm proposed by Sebastiani et al. [17]. This algorithm accepts as input the desired degrees of (dis)satisfaction of a set of high level goals, and propagates...
these degrees throughout the goal model over the lower level goals and tasks. So, using the backward propagation algorithm, the stakeholders can select a set of high level goals as highly desirable, which will then be propagated through the goal diagram until the utility of all of the related lower level goals and tasks have been calculated. Now, since features are accessible for the goal model through the task-feature mappings, the desirability of each feature can also be calculated based on its relation to the goal model’s goals and tasks. In this way, we are able to calculate the degree of utility and desirability of the feature model elements based on the stakeholders’ intentions.

The second approach is referred to as the forward label propagation algorithm [10], which is introduced by Giorgini et al. In contrast to the latter approach, this algorithm starts from lower level goals and works its way to the top goals; therefore, it is able to estimate to what extent high level goals are satisfied given the satisfiability of the lower level goals and tasks. The interpretation of this algorithm for our context would be to select a specific feature and see how it relates to the stakeholder goals and to what extent it is able to satisfy the most important concerns of the stakeholders.

By benefiting from the backward and forward label propagation algorithms on goal models and the mapping between goal model and feature model elements, we are able to move from the intentions of the stakeholders to actual decisions with regards to the selection of the appropriate product features. Hence, we are able to find the features that significantly contribute to the satisfaction of stakeholders’ intentions and objectives, and choose them as the most desirable ones.

IV. A DESCRIPTIVE CASE STUDY

Here, we intended to show how stakeholders’ objectives can be translated into desirable software product line features using our proposed framework. We benefit from the tablet feature model (represented in Figure 1) and a related goal model (represented in Figure 2) as a case study. We first explain the processes in domain engineering and then discuss the processes that need to be conducted in application engineering.

D1. Domain feature modeling: In this step, using a standard domain analysis method a feature model is developed. Given we already have the tablet feature model, we skip this process.

D2. Feature enrichment: Here, the leaf features in the feature model will be extended with the related textual snippets which come from the sections of domain documents that the elements are derived from. The supporting texts that are selected for each element support and describe the context where the element is defined. In fact, during the feature model development these textual snippets should already be linked to each model element. Table I represents the supporting texts that are associated to each elements’ associated supporting text as input and identifies the meaningful spots from the text. Furthermore, using DBpedia, each spot will be linked to a proper concept URI. Accordingly, each element will be cross-referenced with a set of concept URI that are identified from the supporting text. Table II represents the spots and the related concept URIs that are associated to “WiFi” and “3G” features. For other feature elements we can perform a similar process in order to extract concept URIs from their supporting texts. Due to space limitation, we only present the concepts extracted for “WiFi” and “3G” features.

D3. Feature annotation: Once the feature elements are extended with their appropriate supporting texts, each feature element will be annotated with domain ontological concepts. We benefit from a semantic annotation system named Denote for extracting semantic concepts. Denote gets

### Table I

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Supporting Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reina</td>
<td>LCD is a flat panel display with high pixel density.</td>
</tr>
<tr>
<td>WiFi</td>
<td>WLAN is a wireless technology that enables wireless communication.</td>
</tr>
<tr>
<td>3G</td>
<td>_accessor via cellular network.</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>bluetooth is a wireless technology that enables wireless communication.</td>
</tr>
<tr>
<td>MMS</td>
<td>MMS is a standalone tool for sending messages that include video and photos.</td>
</tr>
<tr>
<td>SMS</td>
<td>SMS is a messaging service component for sending text and mobile communication systems.</td>
</tr>
<tr>
<td>HSDPA</td>
<td>HSDPA is a development and distributed by Apple Inc.</td>
</tr>
<tr>
<td>Android</td>
<td>Android is an OS that is developed by Google.</td>
</tr>
<tr>
<td>3G</td>
<td>3G is mobile telecommunications technology, which provides high-speed internet access.</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Spot</th>
<th>Concept URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiFi</td>
<td><a href="http://dbpedia.org/page/WiFi">http://dbpedia.org/page/WiFi</a></td>
<td></td>
</tr>
<tr>
<td>Tablet</td>
<td><a href="http://dbpedia.org/page/Tablet">http://dbpedia.org/page/Tablet</a></td>
<td></td>
</tr>
<tr>
<td>Wireless</td>
<td><a href="http://dbpedia.org/page/Wireless">http://dbpedia.org/page/Wireless</a></td>
<td></td>
</tr>
<tr>
<td>Internet</td>
<td><a href="http://dbpedia.org/page/Internet">http://dbpedia.org/page/Internet</a></td>
<td></td>
</tr>
<tr>
<td>Cellular</td>
<td><a href="http://dbpedia.org/page/Cellular">http://dbpedia.org/page/Cellular</a></td>
<td></td>
</tr>
</tbody>
</table>

Once the tablet feature model is annotated with ontological concepts, in the application engineering phase, this feature model can be used as a basis for the mapping and feature selection process. The application level processes can be conducted as follows:

A1. Capturing stakeholders’ requirement: Here, we will benefit from the goal model that is represented in Figure 2. We assume that the stakeholders’ objectives are captured and modeled based on a standard requirement engineering process.

A2. Task enrichment: Similar to step D2, in this step each task element in the goal model will be extended with supporting textual snippets. Table III represents the supporting textual snippets that are associated to the task elements.

### Table III

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Supporting Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text messaging</td>
<td>Text messaging is the act of sending electronic messages between mobile phones.</td>
</tr>
<tr>
<td>Video and photo messaging</td>
<td>Using WLAN technology to send video and photo content.</td>
</tr>
<tr>
<td>3G</td>
<td>3G is a mobile telecommunications technology, which provides high-speed internet access.</td>
</tr>
<tr>
<td>Cellular</td>
<td>Cellular technology enables cellular networks to communicate voice and data.</td>
</tr>
</tbody>
</table>

### Table IV

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Concept URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLAN</td>
<td><a href="http://dbpedia.org/page/Wireless-LAN">http://dbpedia.org/page/Wireless-LAN</a></td>
</tr>
<tr>
<td>Cellular</td>
<td><a href="http://dbpedia.org/page/Cellular">http://dbpedia.org/page/Cellular</a></td>
</tr>
</tbody>
</table>

675
A3. Task annotation: Similar to step D3, here we benefit from Denote in order to identify the ontological concepts from the task elements’ supporting text. As represented in Table IV, the “WLAN” and “Cellular” tasks are annotated with related concept URIs. Due to space limitation, we only present the identified concept URIs for “WLAN” and “cellular” task elements.

A4. Mapping process: The main concern in the mapping process is to connect the goal space to the feature space. To do this, the mapping process automatically identifies pairs of task and feature elements that are semantically related. The mapping process is preformed in two stages: 1) calculating semantic relatedness and 2) finding the mapping elements. First, the semantic relatedness between all possible combination feature and task pairs are calculated. For each feature and task pair, semantic relatedness is computed. Table V shows the calculated semantic relatedness for each feature and task pair. For instance, the computed semantic relatedness score between “Text messaging” task ($T_1$) and “Retina” feature ($F_1$) is 48%.

In the second stage, we benefit from ILP and calculate the mappings based on the available semantic relatedness scores. The integer linear programming formulation for this problem is represented in Table VI. The decision variables are developed according to combination of $F_i$ and $T_j$ letters, e.g., $F_1T_1$, $F_1T_2$, and etc. The objective function $P$ in this problem is the summation of $F_iT_j$ multiplied by the calculated semantic related score between $F_i$ and $T_j$, e.g., $(0.48)F_1T_1$. In addition, based on Equations 4 and 5, constraints are developed to ensure that a feature can only be mapped onto one task and also each task can only be mapped onto one feature.

The result for each decision variable $F_iT_j$ would be 1 or 0 (the value 1 indicates the $F_i$ potentially can be mapped on $T_j$ and value 0 indicates otherwise). For better representation, we show the obtained results in Table VII. The cells with value 1 (gray cells) indicate the optimal mapping between the corresponding feature $F_i$ and task $T_j$ elements. For instance, the value for $F_1T_1$ is 0 which indicate that the corresponding

"Text messaging” task ($T_1$) cannot be mapped onto “Retina” feature ($F_1$). While the value assigned to $F_1T_1$ variable is 1, which indicates that the “Cellular” task ($T_4$) can be mapped onto the “3G” feature ($F_4$).

A5. Feature selection: In order to identify the features that are relevant to the stakeholders’ goals, the links between the goal space and feature space can be employed. These links allow us to trace from goals and their operational tasks through to actually implementable features. Figure 4 shows how goals can be decision rationale for choosing features. As an example, “enable to connect to Internet” goal constitutes the “WLAN” and “Cellular” tasks. If we follow the linkage from this goal and its corresponding task to the feature space, we will find the matching features, which are the “WiFi” and “3G” features. Therefore, the justification can be made with regards to the decision to choose the “WiFi” or “3G” feature, which is because they are required to implement the “WLAN” and “Cellular” tasks that are needed to satisfy the “enable to connect to Internet” goal of the stakeholders. The process described here simply starts from stakeholders’ high level goals and works its way through the defined tasks and finds the corresponding features from the feature space; therefore, this approach is a form of backward label propagation algorithm.

V. RELATED WORK

Due to the importance of formally capturing and modeling requirements in SPL lifecycles, the idea of employing goal-oriented approaches has gained more attention in the product line research community. Therefore, some authors have tried to benefit from goal models as a early requirement engineering products and use them as a foundation to develop a variability model like the feature model [2], [18], [21], [22].

In [22], the authors describe how one can gradually enrich a standard goal model with a set of notations and then using the provided heuristic rules derive high variability design models such as feature models, statecharts, and component models. Furthermore, the authors in [21] present a tool that benefits from the transformation rules described in [22] in order derive feature models from goal models. Also, in [2], Antonio et al., propose an approach for constructing feature models using the goal model represented in i* goal models. They define a set
of high level processes for transforming i* elements and their relationships to a feature model. Siva et al [18] also followed a similar approach to Antonio et al in which the i* model is extended with cardinality information and through a set of heuristics rules the extended i* model is transformed to a feature model.

Moreover, there has been interest in trying to employ goal models in software system customization and configuration process [14], [15], [23]. In [15], Liaskos et al. take into account the user goals in the configuration process. They, first develop a goal model on top of the software configuration options and use them as a mediator between user and configuration options. Then, through the qualitative reasoning on the goal model, the appropriate configuration options can be selected that satisfy the user’s high level goals. Similar to the work presented in [15], Yu et al. in [23] propose a requirement-driven configuration process which consist of two steps: 1) establish a goal model for software system and connect it to system’s configuration items; and 2) configure a personalized software by collecting user’s goal and expectations. Furthermore, in [14] Lapouchnian et al. show how goal models can be used as a basis for designing automatic software, which are able to self-configure based on the changing operating environment.

In the light of the aforementioned significant achievements, we propose a semi-automated framework in the context of software product line for mapping goal models onto a domain feature model. In our problem, we consider the stakeholders’ intentions and goals as a key factors in the feature selection process in order to ensure that 1) a complete and comprehensive set of initial features from the set of available features is selected (that can be passed into automated feature model configuration processes), which is due to the fact that we can make sure that all stakeholders’ intentions, objectives and concerns can be covered and addressed by the selected features; 2) irrelevant superfluous features are not included in the selected features, since features that do not correspond with at least one of the stakeholders’ goals will not be considered; and 3) the rationale behind the feature selection process is clear for the stakeholders.

VI. CONCLUDING REMARKS

In this work, we promote the consideration of early requirements information in the form of stakeholders’ goals and objectives in the process of selecting the right features for a particular software product. The introduced approach can serve as a best practice guideline that provides a convenient way for capturing stakeholders’ intentions, mapping them as concrete requirements into software features of the whole product line family, and feeding the process of the software product line configuration with a more accurate set of stakeholders’ desired features. To support our approach, we have proposed an automated framework to help application engineers identify the most suitable set of features of a software product feature model to be included in the final software product. We employ semantic analysis methods in order to map stakeholders’ goals onto feature models and then by diagrammatic reasoning techniques one can trace between the intention and feature spaces for making actual product functionality selection decisions. For future work, we are interested in enhancing our framework by considering softgoals as the decision criteria in the feature selection process.

REFERENCES