Capturing Non-functional Properties through Model Interlinking

Mahdi Noorian[1], Ebrahim Bagheri[2], and Weichang Du[1]
University of New Brunswick, Fredericton, Canada[1]
Ryerson University, Toronto, Canada[2]
m.noorian@unb.ca, bagheri@ryerson.ca, wdu@unb.ca

***Abstract*—The Software Product Line (SPL) paradigm promotes systematic reuse-based software development and is founded on the idea of capturing commonalities and variabilities between software products of a target domain. Feature model as the main artefact of SPL development mostly captures functional and operational variability of a system. Researchers have argued that connecting intentional variability models such as goal models with feature variability models in a target domain can enrich feature models with valuable quality and non-functional information. Interrelating goal models and feature models has already been proposed in the literature for capturing non-functional properties in software product lines, however, the integration process is cumbersome and tedious. In this paper, we propose a (semi) automated approach that systematically integrates feature models and goal models through domain-specific conceptual models represented by ontologies. We first associate each element of the models with their raison-d'etre, i.e., sections of the domain documents that justify each model element. Then, using texts that are associated with the elements and through a semantics-enabled textual analysis process, the model elements will be semantically annotated with related ontological concepts. Finally, a mapping recommendation process is proposed that would connect feature model and goal model elements through measuring the semantic similarity of their annotated ontological concept. Our proposed approach not only provides the means to systematically interrelate feature models and goal models but also allows domain engineers to identify the mutual impact of features and non-functional properties.**

I. Introduction

Software Product Line (SPL) engineering is a systematic reuse-based software development approach which is founded on the idea of building software products using a set of core assets rather than developing individual software systems from scratch [5]. The software product line approach consists of two main development lifecycles, namely Domain Engineering and Application Engineering [12]. Domain engineering involves analyzing and modeling the target domain as a whole and producing a set of reusable core assets. Application Engineering involves developing a domain-specific software product using and through the customization of artefacts that are developed in the domain engineering phase.

Domain analysis as an important part of domain engineering is the process for analyzing and modeling a set of related software systems in a domain of interest. In the context of software product lines, system common and variant characteristics are organized in a tree-like representation named feature models [11]. Feature models mostly address functional aspects of a domain [8], [11], [13] and there is a need for systematic approaches to deal with non-functional properties [4]. As indicated in the literature, non-functional properties are an important factor for the success of software development projects [15]. Neglecting and not addressing NFPs adequately can cause a series of critical problems for the final software, which can impose extra time and cost for fixing software errors.

In recent years, the idea of employing goal-oriented approaches has gained more attention in the product family research community [2], [10], [19], [22]. The proposed approaches mainly focus on using goal models for feature model construction and product configuration processes. Some researchers argue that goal models are useful in capturing and modeling functional and non-functional aspects of the domain at the time when features of the intended system (system to-be) have not yet been identified and developed [1], [22]. In fact, goal models can provide a basis for representing intentional variability [14]. Integration of intentional variability model, which contains both domain's functional and non-functional aspects, with feature variability model, which usually addresses functional and operational aspects, can enrich feature models with domain related quality information. Accordingly, in [10] Jarzabek et. al proposed the F-SIG method to connect feature and goal models in order to bring quality attributes in the feature models. Although this approach provides a means for representing NFPs in feature models, since the process needs to be conducted manually, it demands more time dedication and perseverance from domain analysts specifically when the complexity and size of the models grow.

Kang [11] mentioned that feature elements represent system functionality in a domain and the authors in [21] indicate task elements in goal models represent the operation or function that can be defined for satisfying parent goals. Therefore, it could be possible to integrate feature and goal models by identifying conceptually related pair of task and feature elements. Furthermore, softgoals in a goal model represent domain non-functional aspects which are interconnected with task elements. Through the integration of the feature model and the goal model, one can identify the impact of a feature on a non-functional property. This can be achieved by finding the correlated task in a goal model which has influence on the softgoal. In other words, in a goal model the impact of functional aspects of a domain on the non-functional properties are captured through the links which are established between task and softgoal elements. Based on this, by identifying the feature and task pairs, which representing similar functionality, it can be inferred that the feature can has similar influence as its related task has on the domain softgoal.

In this paper, we introduce a (semi) automated framework, which systematically integrates feature models and goal models through shared domain ontologies. 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 recommendation process. Note that, the information in feature and goal models come from domain documents such as interview transcripts, functional requirements documents, strategic planning documents, and etc [12]. In the feature and goal modeling process, domain analysts use these documents as a source to derive the elements and 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 annotation, 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 method to extract important domain concepts from the associated texts and automatically annotate the feature and goal model elements with relevant ontological domain concepts. Afterwards, in the mapping recommendation 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. We should point out that, the objective of our work is not to take over the feature and goal models integration process from the domain analysts, but rather we intend to provide the facilities that allow domain analysts to perform the integration process through a decision support process. The remainder of this paper is as follows: The next section reviews some preliminaries. Then, the propose method for integrating feature and goal models is presented in Section III . In Section IV, we support our method with an illustrative example. Related works are discussed in Section V and the paper is concluded in Section VI.

II. Background

A. Feature Modeling

Features are important distinguishing aspects or characteristics of a family of systems [16]. They are widely used for depicting the shared structure and behavior of a set of similar systems. To form a product family, all the various features of a set of similar/related systems are composed into a feature model. 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 shows a sample feature model for the tablet domain.

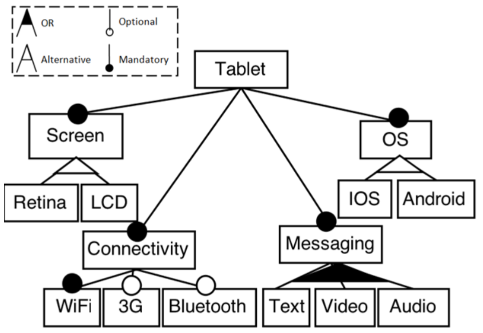

Fig. 1: Sample feature model for tablet domain.

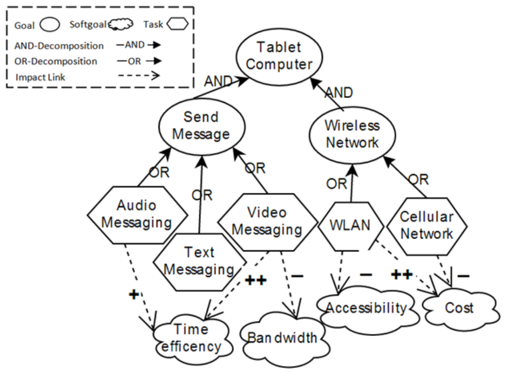

Fig. 2: Sample goal model for tablet computer.

B. Goal Modeling

Goal oriented methods provide a framework for capturing and managing early stage system requirements and are generally built over three important concepts namely goals, softgoals, and tasks [21]. Goals can be defined as a desired result for system under development that stakeholders plan to achieve. In contrast, softgoals refer to non-functional properties of system. In addition, tasks are the methods that can operationalize goals. In goal oriented methods, all elements (goals, softgoals, and tasks) are organized and represented in a tree-like structure called goal models. In a goal model, the elements can be refined such that high level elements are expressed through the finer grained elements using AND-decomposition and OR-decomposition. Moreover, impact links are defined to identify to what extent an individual task can contribute to satisfaction of the softgoals. Impact links can be annotated with: "+" helps, "++" makes, "-" hurts, or "- - " breaks [15]. Figure 2 shows a sample goal model for the tablet domain.

III. The Proposed Method

The overview of our approach is shown in Figure 3. Essentially, our work focuses on introducing a process which

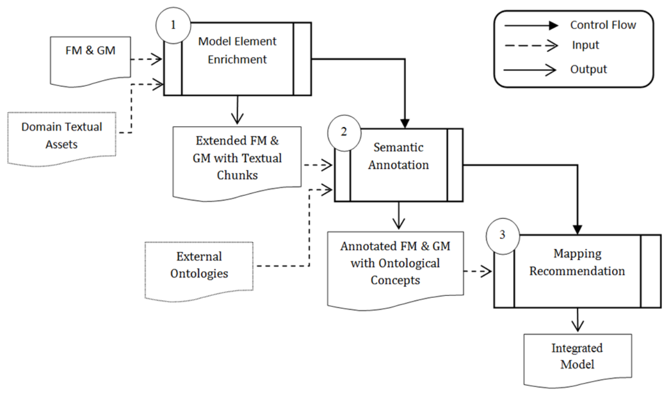

Fig. 3: Overview of our approach.

facilitates the tasks of the domain analysts for addressing non-functional properties during the domain engineering lifecycle. Within our proposed approach, we achieve this by systematically extending feature models with the relevant goal models targeting same domain of interest.

The proposed process for incorporating non-functional properties in a feature model takes three main steps:

1) *Model element enrichment*: In the feature and goal modeling process, domain documents are usually used as information source for extracting feature and goal models' elements. Based on this fact, our first step will create the extended version of feature and goal models in which the elements in the models are linked with the sections of domain documents that the elements are derived from. In the other words, each model element will be accompanied by textual snippets from domain documents that justify the existence of the model element.

2) *Semantic annotation*: Once the feature and goal model elements are labeled with their related supporting 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.

3) *Mapping recommendation*: In the third step, both feature and goal model elements are semantically annotated with concepts from shared domain ontologies. Mapping recommendation process identifies and recommends the pair of elements in the feature and goal models that are semantically related. Here, pairwise comparisons will be performed on the elements' representative concepts in order to measure the conceptual relatedness of each pair elements. This process highlights a set of possible mappings between the models. Furthermore, domain analysts can review the mapping suggestions and decide on their suitability, i.e., accept or reject the recommendation, and then finalize the integration process by connecting the accepted elements via mapping links. In the following sections, the details behind each of these three steps are introduced.

A. Model Element Enrichment

In the real world practice, domain analysts usually 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 will be 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 will be documented by building the 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 the model will be more understandable and changeable at the time when it needs to be updated with new identified requirements. Based on this, we can consider that there is a link between the identified elements, e.g. features or goals, and the source of information that the element is derived from.

The general approach for mapping goals and features has been to look at the syntactical similarity or synonimity between feature and goal element names. Our approach takes this one step further by trying to find correspondences between feature and goal model elements by looking at their textual sources. In fact, the first step of our approach can be considered as preprocessing step for the semantic annotation process.

B. Semantic Annotation

The main goal of our proposed approach is connecting the feature model space with the goal model space through identifying and relating the elements that are semantically related, e.g., the pair of feature and tasks that represent the same functionality. In the feature and goal models, elements can be further described using a set of semantic tags. These semantic tags are in the form of references to concepts within external ontologies. The associated ontological concepts can be considered as a representative, which indicates the inherent intended meaning of a element. Based on this it can be claimed that, for each pair of elements, the more the element's associated concepts are similar the more probable that the elements are semantically close to each other.

The main task in this step is to annotate the feature and goal model elements with appropriate semantic concepts. For this purpose, we use element's supporting texts that are added in model element enrichment step and through the text semantic analysis method; we automatically extract semantic concepts from the texts and annotate the model elements with ontological concepts. As a result of the annotation process, a meaningful profile will be produced for each element in the feature and goal models. Each profile allows each element to be clearly described using its related concepts.

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

Typical semantic annotation system rely on three main phases, i.e., *spotting*, *disambiguation* and *pruning* [6]. The main goal of the spotting phase is to parse the input supporting text and identify the appropriate spot (which can be a single word or phrase). The main objective of the disambiguation phase is twofold: 1) first cross-reference each spot identified in the input text with one appropriate ontological concept and 2) then among all concepts that are linked to a spot, select the one that reflects the best semantic association. Finally, in the pruning phase, the candidate annotations which are not related to input text context will be deleted.

Our work is independent of the choice of the semantic annotation system as long as the selected system addresses the following capabilities: accepting short supporting text as an input text, using well accepted general ontologies like DBpedia as the underlying knowledge base, and providing real time annotation result efficiently within a reasonable time frame.

C. Mapping Recommendation

Once the feature and goal model elements are annotated with the concepts within a shared ontology, the models can be explored to find the pair of conceptually related feature and task elements. The mapping recommendation process provides this opportunity to connect feature space with goal space through identifying a set of feature and task elements that are represented by related concepts.

The underlying challenge of mapping process can be simply viewed as the problem of selecting the feature and the task that are annotated with a set of similar concepts. Given A_F (which is the concept sets for feature set F in the feature model) and A_T (which is the concept sets for task set T in the goal model); the mapping process calculates the relatedness between all feasible combinations of tasks and features.

As indicated by Strube et al. in [18], the measure for computing semantic similarity between two ontological concepts 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. For the purpose of calculating and quantifying the conceptual distance between two ontological concepts, we employ the distance metric proposed in [20], [3] which is based on the combined measures that we introduced before.

Lets assume that feature $f \in F$ is annotated with a set of concepts $A_f\langle c_1, c_2 ... c_n\rangle \in A_F$ and, task $t \in T$ is annotated with $A_t\langle c'_1, c'_2 ... c'_m\rangle \in A_T$. 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.

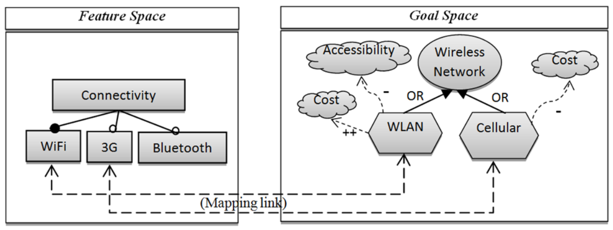

Fig. 4: Abstract feature model and goal model.

The mapping recommendation process works as a decision support platform and assists domain engineers by recommending the conceptually related task and feature elements. For each feature $f \in F$, mapping recommendation process lists the potential tasks which feature can be mapped on them. The tasks are ranked based on their calculated relatedness scores, which indicate the similarity degree of the feature to the each individual task. Furthermore, a domain engineer can review the mapping suggestions and finalize the process by accepting the tasks that are suitable enough to be mapped on the feature. As a result of this process, the feature model will be extended with the non-functional properties of the domain. In fact, in the goal model the association between tasks and softgoals are implicitly expressed via impact links. Therefore, task elements are already associated with domain non-functional properties. By connecting the feature elements to their semantically related task elements, we create an indirect link from features to non-functional properties through the tasks.

IV. Illustrative Example

Given the proposed framework that is described in the previous sections, let us explain the feature and goal models integration process using an illustrative example. Assume that a domain analyst wants to model and address non-functional properties of a domain in a feature model, e.g., feature model in Figure 4. The provided goal model (right box in Figure 4) can be used as the basis in which non-functional properties of the domain are represented as softgoals (in reality, feature and goal models are much bigger than this short example). As seen in the process shown in Figure 3 the integration can be conducted in three steps. In the first step (model element enrichment), the feature and goal model elements will be enriched with their domain supporting text. Table 1 shows sample supporting texts for “3G” and “WiFi” features, and “WLAN” and “Cellular” tasks. A Domain analyst can help to find the appropriate supporting text for each element from their related domain text document. In reality, this should already be available from when goal and feature models were initially developed.

In the second step, using textual semantic analysis the ontological concepts will be identified and assigned to each element. As an example, Table 2 represents the spots and their associated concepts that are extracted from the “WiFi” feature’s supporting text: *“WiFi technology allows tablets connect to internet using wireless interface”*. In fact, the “WiFi” feature is now restricted with the semantic meaning attached to the identified concepts URI. This process needs to be performed

Table 1: Sample supporting text

Element Name	*Supporting Text*
WiFi	WiFi technology allows tablets connect to internet using wireless interface.
3G	The tablet supports 3G which is the third generation of mobile telecommunications technology. 3G provides the mobile Internet access via cellular network.
WLAN	A wireless local area network links two or more devices using wireless interfaces. Most modern WLAN are marketed under the Wi-Fi brand name.
Cellular	The tablet computer can connect to mobile network in order to transmit voice and data using cellular network.

Table 2: Concept set for "WiFi" feature

Spot	Concept URI
WiFi	http://dbpedia.org/page/Wi-Fi
Tablets	http://dbpedia.org/page/Tablet_computer
Internet	http://dbpedia.org/page/Internet
Wireless	http://dbpedia.org/page/Wireless

for other elements in the feature and goal model. Due to space limitation, the focus of this example is to find the mapping tasks for "WiFi" feature. It should be noted that, the semantic annotation process is fully automated.

In the third step, each feature element's associated concepts will be compared with the concepts that are associated to task elements in a goal model and a list of candidate tasks that can be mapped on to the target feature will be suggested. In this example, mapping recommendation process suggests "WLAN" and "Cellular network" as possible mapping tasks, and then ranks them based on their calculated relatedness scores ($0 \leq score \leq 1$). The relatedness scores are computed by comparing the "WiFi" feature's concepts set (presented in Table 2) with the concepts that are associated to "WLAN" and "Cellular network" tasks (as presented in Table 3). Our approach provides domain engineers with a means to choose the best mapping (either "WLAN" or "Cellular network" tasks) for the "WiFi" feature by considering the calculated relatedness scores. In this case, "WLAN" with the score value of 0.9 is more probable to be best choice for mapping compared to "Cellular network" with the score value of 0.6. Therefore, "WLAN" will be chosen as a most similar task thus it can be connected to "WiFi" feature via a mapping link.

Using the mapping recommendation process we identified the semantically related feature and task elements. According to the quality information that is encoded in the goal model, we want to recognize the domain related non-functional properties for the feature model elements. Note that, in a goal model the non-functional properties of the domain are represented as a softgoals and softgoals are linked to task elements via impact links. For instance, as shown in Figure 3, "WLAN" task is linked to "Accessibility" softgoal with positive impact. Also, "WLAN" task and "Cost" softgoal are linked with negative impact. Considering the fact that, we know the feature/task relations (which are identified through mapping process) and the task/softgoals relations (which are expressed in a goal model), the relation between features and non-functional properties can be identified. As an example, through the mapping process the link is established between "WiFi" feature and "WLAN" task. On the other hand, "WLAN" task has negative impact on the "Accessibility" and positive impact on the "Cost" softgoals. Therefore, it can be inferred that "WiFi" feature also has the same impact, as "WLAN" task has on "Accessibility" and "Cost" softgoals.

Table 3: The list of recommended task elements for "WiFi" feature.

Task	Concept URI	Score
WLAN	http://dbpedia.org/page/Wireless http://dbpedia.org/page/Wireless_LAN http://dbpedia.org/page/Wi-Fi	0.9
Cellular network	http://dbpedia.org/page/Cellular_network http://dbpedia.org/page/Transmission_(telecommunications) http://dbpedia.org/page/Tablet_computer	0.6

The spotlight of this work is that we consider stakeholders' both functional and non-functional properties in domain analysis process. Adopting the goal models which inherently address non-functional properties of the domain and systematically integrating them with feature model provides this opportunity to bring the feature model and its related quality aspects under one umbrella. Having comprehensive model that address both functional and non-functional properties in domain of interest can be used throughout of product line development such as reference architecture design and product configuration. Specifically, in configuration process, using the integrated model an application engineer can make more informed decision in feature selection process with respect to the associated non-functional properties. He can simply trace back the selected feature into the goal model space, and observe the impact of his decision over the related non-functional properties (softgoals); hence the application engineer would be able to develop quality-centric software products.

V. Related Work

Software product lines have been gaining interest in both industry and academia due to their usefulness for facilitating reuse and enhancing efficiency and production time. Due to the importance of non-functional properties, researches have recently started to explore the development of methods for capturing and modeling the non-functional properties in the domain analysis phase, which is the main phase in domain engineering lifecycles [4], [9], [10], [17], [23].

Benavides et al. in [4] introduce an approach for modeling NFPs in feature models. The model is named "extended feature model" where features in feature models are annotated with some quality attributes such as availability, cost, latency, and bandwidth. In addition, for each attribute the domain of possible value are defined which can be discrete or continuous. This model can be used to automatically configure products using CSP (Constraint Satisfaction Problems) solvers.

In another work, Sinnema et al. [17] propose a framework (COVAMOF) for modeling variability in software product lines. COVAMOF supports variation in several levels ranging from features to code. Variations are organized hierarchically and various types of interdependency among variation points are modeled and treated in this framework.

In [23], the authors develop a Bayesian Belief Network (BBN) approach to predict and assess quality of software products. In this approach, BBN is used to capture and identify the impact of variants over the set of quality attributes. In the BBN, the nodes represent features and quality attribute variables. The directed edges represent the impact relation between two nodes. Conditional probability is assigned to each node in BBN in order to quantify the conceptual relationships. This probability numbers show the domain expert's belief, the extent that a given feature impacts quality attributes. Using the BBN network, an application engineer can understand how selecting one feature can impact a specific quality attribute thus making more informed decisions in the configuration stage. In the abovementioned approaches, capturing and managing the non-functional properties of the domain are highly dependent on domain analysts' knowledge and experience.

Most closely to our work, Jarzabek et al. [10] have considered goal models as a foundation to represent NFPs in a feature model. In their research, they develop F-SIG (Feature-Softgoal Interdependency Graph) that is founded on FODA (domain analysis method) and NFR Framework (goal-oriented analysis method). In F-SIG, FODA is extended with NFR Framework model in order to benefit from the potential of goal-oriented analysis in SPL. F-SIG nodes represent the domain features and quality attributes. The interdependencies between concepts in F-SIG are represented by directed edges. Using F-SIG, application developers can analyze and evaluate how the selection of particular variant can influence the quality attributes. The work proposed by the authors mostly need a domain analyst's intervention in terms of identifying the interdependencies between functional and non-functional properties, whereas our approach provides a framework that assist domain analysts to recognize the domain related NFPs for each feature.

VI. Conclusion

The focus of the work presented in this paper is to address non-functional properties in domain analysis for SPL, which is the first step in the domain engineering phase. We propose a semi-automatic approach which intends to systematically integrate feature models, which represent functional aspects of a domain, and goal models, which represent non-functional properties. Through this integration, the quality aspects of the domain can be captured from the early stage of SPL development. Our proposed framework consists of three main steps: 1) model element enrichment (relate the model elements to corresponding domain textual documentation); 2) semantic annotation (semantically annotate model elements with standard ontological concepts), and 3) mapping recommendation (extend feature model with domain non-functional properties via softgoals). As a future work, we aim to use various types of ontologies in the textual analysis process in order to have more accurate semantic annotation process. Especially, we intend to develop NFP ontologies and use it in our proposed approach. Moreover, we intend to develop a toolset in order to support the activities that are presented in our proposed framework.

References

[1] A.I. Anton. Goal-based requirements analysis. In *Requirements Engineering, 1996., Proceedings of the Second International Conference on*, pages 136–144. IEEE, 1996.

[2] M. Asadi, E. Bagheri, D. Gašević, M. Hatala, and B. Mohabbati. Goal-driven software product line engineering. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 691–698. ACM, 2011.

[3] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *IJCAI*, volume 3, pages 805–810, 2003.

[4] D. Benavides, P. Trinidad, and A. Ruiz-Cortés. Automated reasoning on feature models. In *Advanced Information Systems Engineering*, pages 381–390. Springer, 2005.

[5] P. Clements and L. Northrop. Software products lines: Practices and patterns. 2001.

[6] Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. A framework for benchmarking entity-annotation systems. In *Proceedings of the 22nd international conference on World Wide Web*, pages 249–260. International World Wide Web Conferences Steering Committee, 2013.

[7] J. Cuzzola, Z. Jeremic, E. Bagheri, D. Gasevic, J. Jovanovic, and R. Bashash. Semantic tagging with linked open data. In *Canadian Semantic Web Symposium*, 2013.

[8] Krzysztof Czarnecki and Ulrich W. Eisenecker. *Generative programming - methods, tools and applications*. Addison-Wesley, 2000.

[9] B. González-Baixauli, M.A. Laguna, and Y. Crespo. Product line requirements based on goals, features and use cases. In *International Workshop on Requirements Reuse in System Family Engineering (IWRE-QFAM)*, pages 4–7, 2004.

[10] S. Jarzabek, B. Yang, and S. Yoeun. Addressing quality attributes in domain analysis for product lines. *IEE Proceedings-Software*, 153(2):61–73, 2006.

[11] K.C. Kang, S.G. Cohen, J.A. Hess, W.E. Novak, and A.S. Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, DTIC Document, 1990.

[12] K.C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh. Form: A feature-; oriented reuse method with domain-; specific reference architectures. *Annals of Software Engineering*, 5(1):143–168, 1998.

[13] K.C. Kang, J. Lee, and P. Donohoe. Feature-oriented product line engineering. *Software, IEEE*, 19(4):58–65, 2002.

[14] S. Liaskos, A. Lapouchnian, Y. Yu, E. Yu, and J. Mylopoulos. On goal-based variability acquisition and analysis. In *Requirements Engineering, 14th IEEE International Conference*, pages 79–88. IEEE, 2006.

[15] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *Software Engineering, IEEE Transactions on*, 18(6):483–497, 1992.

[16] K. Pohl, G. Böckle, and F. Linden. Software product line engineering: Foundations, principles, and techniques. 2005.

[17] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch. Covamof: A framework for modeling variability in software product families. *Software Product Lines*, pages 25–27, 2004.

[18] Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *AAAI*, volume 6, pages 1419–1424, 2006.

[19] K. Uno, S. Hayashi, and M. Saeki. Constructing feature models using goal-oriented analysis. In *Quality Software, 2009. QSIC'09. 9th International Conference on*, pages 412–417. IEEE, 2009.

[20] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[21] E.S.K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 226–235. IEEE, 1997.

[22] Y. Yu, J.C.S. do Prado Leite, A. Lapouchnian, and J. Mylopoulos. Configuring features with stakeholder goals. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 645–649. ACM, 2008.

[23] H. Zhang, S. Jarzabek, and B. Yang. Quality prediction and assessment for product lines. In *Advanced Information Systems Engineering*, pages 1031–1031. Springer, 2003.