Patterns for Identifying and Structuring Features from Textual Descriptions: An Exploratory Study

Nili Itzik and Iris Reinhartz-Berger

Department of Information Systems, University of Haifa, Israel
nitzik@campus.haifa.ac.il, iris@is.haifa.ac.il

Abstract. Software Product Line Engineering (SPLE) supports developing and managing families of similar software products, termed Software Product Lines (SPLs). An essential SPLE activity is *variability modeling* which aims at representing the differences among the SPL's members. This is commonly done with feature diagrams – graph structures specifying the user visible characteristics of SPL's members and the dependencies among them.
Despite the attention that feature diagrams attract, the identification of features and structuring them into feature diagrams remain challenging. In this study, we utilized Natural Language Processing (NLP) techniques in order to explore different patterns for identifying and structuring features from textual descriptions. Such a catalog of patterns is important for both manually-created and automatically-generated feature diagrams.

Keywords: Variability Analysis, Feature Diagrams, Natural Language Processing, Empirical Evaluation

1 Introduction

Software Product Line Engineering (SPLE) supports developing and managing similar software products (SPLs) [14]. SPLE has been proven to be successful in reduction of development cost, time-to-market and improvement of product's quality [12]. Variability modeling is a crucial activity for identifying and documenting the precise differences among the SPL's members for effective and efficient development and management of the entire SPL. Feature diagrams [3], which are commonly the outcomes of the variability modeling activity, are graph (or tree) structures that describe "features" of a SPL and the relationships and dependencies among them [9]. A "feature" can be defined as "a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems" [9].

Variability models in general and feature diagrams in particular are created either manually by humans or automatically utilizing methods such as [2], [4], [15], [18]. Being manually or automatically created, the identification and structuring of features are important for the comprehensibility of the models. In this study we explored different semantic patterns to create feature diagrams from textual descriptions. The

descriptions were short and focused on the (visible) behaviors of the SPLs' members. We further grounded these patterns utilizing Natural Language Processing (NLP) techniques and proposed guidelines for their use. Thus, the contribution of the work is two-fold. First, the patterns provide guidelines for modelers who create feature diagrams as to how to identify and structure features. Second, the patterns may be the basis for flexible automatic feature extraction processes.

The rest of the paper is structured as follows. Section 2 briefly reviews related work. Section 3 describes the study's settings and execution, while Section 4 presents the patterns extraction process and outcomes. Section 5 discusses the findings and refers to limitations. Finally, Section 6 concludes and suggests future directions.

2 Related Work

As noted, feature diagrams can be created manually or automatically from text. Only a few studies suggest guidelines for (manually) creating feature diagrams. In [10] guidelines for identifying features and classifying them according to the types of information they represent are suggested. The four classes of features are: capability, operating environment, domain technology, and implementation technique. Organization of the features into diagrams is then done by analyzing the relations between these classes of features. In [11] further guidelines have been suggested for domain planning, feature identification, feature organization, and feature refinement. For example, it is recommended not to "organize features to represent functional dependencies," but to "capture and represent commonalities and differences."

Studies that use textual descriptions for automatically (or semi-automatically) identifying and extracting features relay on syntactic patterns, utilizing different parts-of-speech (POS). Examples of such studies are [2], [4], [5], [7], [13], [18]. They mainly use nouns and verbs for this purpose. Some of them suggest structuring the features using different clustering algorithms. In [8], an automatic Semantic and Ontological Variability Analysis (SOVA) method is suggested to construct feature diagrams based on behavioral similarity and variability. The input textual descriptions are parsed according to the semantic roles of the phrases in the sentences and the behavioral elements are extracted utilizing an ontological model. Then the semantic similarity between the behavioral elements is used for creating feature diagrams.

There are also studies that generate features diagrams from feature configurations using refactoring techniques, e.g., [1] and [16]. In these studies the inputs are some structured or formal representations (such as, propositional formulas or feature lists and dependencies). Identification of features is not needed, as the features are already given. Organization of features into diagrams is done by analyzing implication graphs, which are directed graphs with features as vertices and edges that represent dependencies between features.

To summarize, existing studies provide some general guidelines for creating feature diagrams. Others extract features using specific, pre-defined syntactic or grammatical patterns. The organization of features into diagrams in those studies is done using transformation rules or clustering algorithms, without referring to the extracted

features characteristics (e.g., their POS). In our study, we explored the various semantic patterns that can represent features, as well as the relations among them.

3 Study's Settings and Execution

In order to explore the various ways modelers extract features from textual descriptions and organize them into feature diagrams, we developed a questionnaire with eight short paragraphs. Each paragraph described a SPL including information on the applications' behaviors and the allowed variability in the SPL. The task was to present for each description a feature diagram that resembles the description[1].

The participants in this study were 11 information systems students at the University of Haifa, Israel. Those students participated in an advanced software engineering course that was devoted to SPLE. The descriptions referred to application domains expected to be familiar to students: e-shop, library management, photo sharing, and text editing. Overall we received 78 feature diagrams from the eleven participants, as a few participants answered the questionnaire partially.

4 Outcomes: Feature Patterns and their Relations

Looking at the obtained feature diagrams, we observed that the participants used various linguistic and semantic parts of the original descriptions when naming the features. Therefore, we decided to utilize NLP techniques to analyze the results. Particularly, we decided to use the Semantic Role Labeling (SRL) technique [6], which refers to the semantic roles of phrases in particular sentences and goes beyond POS. Next we elaborate on SRL, the patterns we identified, and the relations found between the extracted patterns when organizing the features into diagrams.

4.1 Semantic Role Labeling

SRL [6] associates constituents of a phrase with their semantic roles in the phrase. Those semantic roles identify the relationships that a syntactic constituent has with a predicate. Typical semantic arguments include: (1) **Agent (A0)** – Who performs the action?; (2) **Object (A1)** – On what object is the action performed?; (3) **Instrument (A2)** – How is the action performed? Identification of adjunctive arguments, termed **modifiers**, is further supported in SRL, for example: Temporal (AM-TMP) – When is the action performed? or Adverbial (AM-ADV) – In what conditions is it performed?

The benefits of SRL for analyzing variability of functional requirements have already explored in [15][2]. As an example consider the following sentence:

---

[1] The questionnaire can be found at http://mis.hevra.haifa.ac.il/~iris/research/SOVA/featureExtQue.pdf.

[2] We used the English version of SRL. As the text descriptions were given in Hebrew – the mother tongue of the participants, we had to translate them to English. Two researchers verified the translation and especially its accuracy with respect to POS.

The users subscribe to the site and update their profiles, using an online interface.

Two verb predicates are identified in this sentence: 'subscribe' and 'update'. Accordingly, the extracted roles (marked in square brackets) are:

The $\text{users}_{[\text{Agent}]}$ $\text{subscribe}_{[\text{action}]}$ to the $\text{site}_{[\text{object}]}$ using an online $\text{interface}_{[\text{AM-ADV}]}$

The $\text{users}_{[\text{Agent}]}$ $\text{update}_{[\text{action}]}$ their $\text{profile}_{[\text{object}]}$ using an online $\text{interface}_{[\text{AM-ADV}]}$

The features extracted from this sentence are expected to include "update profile" and "subscribe to site." However, these features could appear in different contexts. Fig. 1 demonstrates three such contexts for the feature "update profile": the agent who performs the action (a), the object on which the action is performed (b), and the action itself (c).

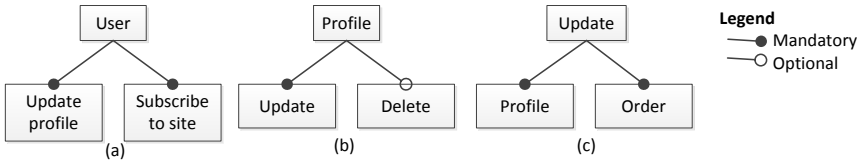

Fig. 1. Possible contexts to the feature "update profile"

4.2 Identified Feature Patterns

To identify the feature patterns we mapped the features that were specified by the participants to the different descriptions to the outcomes of the SRL technique on those descriptions. The extracted patterns are listed in Table 1. Since instruments and modifiers played similar roles in our descriptions – they both describe actions – currently we do not distinguish between patterns based on those roles. Furthermore, the other roles, namely, agents, actions, and objects, appear in (almost) any sentence, while modifiers and instruments interchangeably appear, if at all.

As can be seen the commonly used patterns in our study described partial functionality (i.e., combination of actions and the objects on which they are performed, e.g., "update profile") and objects (e.g., profile). These are followed by "descriptive" and "actions" patterns, which utilize different modifiers/instruments and the sentences' predicates, respectively. We further observed stakeholders-related patterns, namely, patterns involving the agent role, and different combinations of functionality-related roles, e.g., Action+Modifier/Instrument and Action+Object+Modifier/Instrument.

Table 1. The extracted feature patterns

Pattern Name	Patterns Content	Example (modifier type)	# occurrences
Partial functionality	Action+Object	"make order"	314
Objects	Object	"price"	246
Descriptive	Modifier/ Instrument	"via credit card" (AM-MNR)	230
Actions	Action	"purchase"	163
Stakeholders	Agent	"supplier"	59
Described actions	Action+ Modifier/ Instrument	"view by popularity" (Instrument)	39
Described partial functionality	Action+ Object+ Modifier/ Instrument	"prices automatically updated" (AM-MNR)	9
Described objects	Object+ Modifier/ Instrument	"operations on file" (AM-LOC)	9
Actions by stakeholders	Agent+Action	"users register"	5
Described stakeholders	Agent+ Modifier/ Instrument	"retrieval interface from supplier's site"	4
Full functionality	Agent+ Action+ Object	"Suppliers publish items"	3

4.3 Relations of Feature Patterns

We further tried to examine the relations between the features' patterns when organizing the features into diagrams. To this end, we examined for each one of the five top found patterns what patterns their descendants follow, and particularly their direct child features. Table 2 summarizes those findings.

- A child of a feature following the **"partial functionality"** pattern commonly follows "descriptive", "objects", or "partial functionality" patterns. This means that the child refers to other aspects of the functionality (as in "descriptive" and "objects" patterns) or refines the parent (as in "partial functionality" pattern).
- A child of a feature following the **"objects"** pattern commonly follows "objects" to refine the parent. However, it can also follow "descriptive", "partial functionality", or even "actions" to describe or specify the possible uses of the objects.
- A child of a feature following the **"descriptive"** pattern commonly follows "descriptive", "actions", or "partial functionality" patterns. In many cases the "descriptive" pattern appears in the leaves of the diagram.
- A child of a feature following the **"actions"** pattern follows "descriptive", "objects", "actions", or "partial functionality" patterns.
- A child of a feature following the "**stakeholders**" pattern mainly follows the "partial functionality" pattern that describes the actions which are performed by the stakeholders and the objects on which they are performed. Other children's patterns were also observed in this case, most notably, "actions" and "objects".

Table 2. Relations between parent's and child's patterns

Parent's pattern → Child's pattern ↓	Partial functionality	Objects	Descriptive	Actions	Stakeholders
Partial functionality	35	50	22	14	74
Objects	58	62	11	20	26
Descriptive	72	53	26	35	8
Actions	9	18	23	16	25
Stakeholders	5			2	7
Described actions	11	2	1	9	2
Described objects		1	2	1	
Described partial functionality		3	4	2	
Full functionality	2				
Actions by stakeholders	1				

5 Discussion and Limitations

We identify a number of interesting results that worth further discussion. First, we found that in many cases the features describe functionality-related aspects. These findings are in-line with the definitions of features that many of them highlight the functional part of the features.

Second, the top found patterns refer to either functionality or structure. In [8], we have already reported that most feature diagrams in S.P.L.O.T[3] – an academic repository of feature diagrams – followed to some extent a structural or a functional perspective. A structural perspective corresponds to our "objects" and "described objects" patterns. A functional perspective is highly related to our "actions"-involving patterns. We further found that actions were usually accompanied with the objects on which they are performed (corresponding to our "partial functionality" pattern).

Third, our findings can be directly transformed into guidelines for modeling feature diagrams from textual descriptions:

1. Extract actions (potentially with their associated objects), objects, modifiers, and agents from the textual descriptions. Examine whether they can serve as features, namely, their variability is of interest.
2. Try to refine each of the extracted features with the same pattern (used to extract the feature) or with the other top found patterns.
3. Examine the parts of the descriptions not covered by the previous steps. Try to use the other patterns to fully model variability.

Finally, our patterns cover the ways existing methods extract features from textual descriptions. This originates from the fact that our patterns are semantic, as opposed

[3] S.P.L.O.T Software Product Lines Online Tools. http://www.splot-research.org/

to the syntactic and grammatical patterns used by the existing approaches. The same sequence of POS can be mapped to different semantic roles. For example, adjective+noun can be used for describing the agent (e.g., registered user) or the object (e.g., small items). As such, the suggested catalog goes beyond the syntactical structure of the sentence. It can further be used by those methods to enable more flexible generation of feature diagrams and may set the ground for systematic methods to identify and structure SPL features.

The validity of our study is subject to several threats. First, the knowledge and skills of our participants may be questioned. However, they were students in an advanced software engineering course who had the required background in SPLE and feature modeling. The use of students in different software engineering research areas is acceptable as it was shown that students have a good understanding of the way industry behaves [17]. Second, the relatively low number of participants may challenge the ability to generalize the results. Thus, each participant was required to model several feature diagrams, resulting with 78 diagrams overall. Although these diagrams are not independent, it enabled us analyzing more cases. Another threat is the possibility that the way the descriptions were phrased influenced the feature extraction process. Thus, we used eight different descriptions. We did not use a fixed, predefined way to phrase those descriptions. Finally, the questionnaire used in this study was written in Hebrew and translated to English in order to apply the SRL technique. This may affect the pattern extraction process. To overcome this threat, a researcher not involved in the current study additionally verified the translation in general and with respect to POS in particular. Following her feedback, a few corrections were made prior to execution of the study.

6 Summary and Future Work

Variability modeling is an important activity in Software Product Line Engineering (SPLE). Extraction of features and structuring them into diagrams are challenging, time-consuming, and error-prone. In this paper we present a catalog of patterns that can be used to extract features from textual descriptions. These patterns are based on semantic considerations (rather than syntactic and grammatical ones). We further discuss the relations between those patterns in order to assist in organizing the features hierarchically and creating feature diagrams. As far as we know, we are the first ones to create a catalog of semantic patterns and use it to create feature diagrams.

Further work is required to replicate the study with different experienced populations of participants and different textual descriptions (in terms of length and phrasing styles). The usefulness of the patterns for modeling variability and automatically generating feature diagrams needs to be explored as well. Finally, exploration and analysis of the indirect relations between patterns and refinement of patterns based on existing or additional roles may provide interesting findings that can help improve feature diagrams creation processes.

References

1. Acher, M., Baudry, B., Heymans, P., Cleve, A., & Hainaut, J. L. (2013). Support for reverse engineering and maintaining feature models. Workshop on Variability Modelling of Software-intensive Systems (VaMoS), Article #20.
2. Acher, M., Cleve, A., Perrouin, G., Heymans, P., Vanbeneden, C., Collet, P.,& Lahire, P. (2012). On extracting feature models from product descriptions. Workshop on Variability Modelling of Software-intensive Systems (VaMoS), pp. 45-54.
3. Chen, L., & Babar, M.A. (2011). A systematic review of evaluation of variability management approaches in software product lines. Information and Software Technology 53, pp. 344-362.
4. Davril, J. M., Delfosse, E., Hariri, N., Acher, M., Cleland-Huang, J., & Heymans, P. (2013). Feature model extraction from large collections of informal product descriptions. The 9th Joint Meeting on Foundations of Software Engineering, pp. 290-300.
5. Ferrari, A., Spagnolo, G. O., & Dell'Orletta, F. (2013). Mining commonalities and variabilities from natural language documents. Software Product Line Conference (SPLC'13), pp. 116-120.
6. Gildea, D. & Jurafsky, D. (2002). Automatic Labeling of Semantic Roles. Computational Linguistics 28 (3), pp. 245-288.
7. Hariri, N., Castro-Herrera, C., Mirakhorli, M., Cleland-Huang, J., & Mobasher, B. (2013). Supporting domain analysis through mining and recommending features from online product listings. IEEE Transactions on Software Engineering 39(12), pp. 1736-1752.
8. Itzik, N. & Reinhartz-Berger, I. (2014). Generating Feature Models from Requirements: Structural vs. Functional Perspectives. Software Product Line Conference (SPLC'14) – Volume 2: Workshops, Demonstrations, and Tools, pp. 44-51.
9. Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E. & Peterson, A. S. (1990), Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report, SEI.
10. Kang, K.C., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M. (1998). Form: A feature-oriented reuse method with domain-specific reference architectures. Annals of Software Engineering 5, pp. 143–168.
11. Lee, K., Kang, K.C., & Lee, J. (2002). Concepts and guidelines of feature modeling for product line software engineering. International Conference on Software Reuse, pp. 62–77.
12. McGregor, J.D., Muthig, D., Yoshimura, K., & Jensen, P. (2010). Guest Editors' Introduction: Successful Software Product Line Practices. IEEE Software 27(3), pp. 16-21.
13. Niu, N. and Easterbrook, S. (2008). Extracting and modeling product line functional requirements. Requirements Engineering conference (RE'08), pp. 155-164.
14. Pohl, K., Böckle, G., & van der Linden, F. (2005). Software Product-line Engineering: Foundations, Principles, and Techniques, Springer.
15. Reinhartz-Berger, I., Itzik, N., & Wand, Y. (2014). Analyzing Variability of Software Product Lines Using Semantic and Ontological Considerations. Conference on Advanced Information Systems Engineering (CAiSE'14), LNCS 8484, pp. 150-164.
16. She, S., Lotufo, R., Berger, T., Wasowski, A., & Czarnecki, K. (2011). Reverse engineering feature models. International Conference on Software Engineering, pp. 461-470.
17. Svahnberg, M., Aurum. A., & Wohlin, C. (2008). Using Students as Subjects – An Empirical Evaluation. ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 288-290.
18. Weston, N., Chitchyan, R., & Rashid, A. (2009). A framework for constructing semantically composable feature models from natural language requirements. Software Product Line Conference (SPLC'09), pp. 211-220.