Towards a Framework for Feature Deduplication during Software Product Lines Evolution

Amal Khtira
(Supervised by Prof. Bouchra El Asri)

IMS Team, SIME Laboratory, ENSIAS, Mohammed V University
Rabat, Morocco
amalkhtira@gmail.com

Abstract. Software product lines are long-living systems that evolve continuously over time to satisfy the new requirements of customers. This evolution consists of adding or modifying features in the core platform of the product line or in derived products. As a result of this change, many model defects can occur, such as inconsistency and duplication. In this paper, we describe our work which proposes a framework to manage the software product line evolutions. The aim of the framework is to formalize the representation of the software product line models and the specifications of the new evolutions. Then, a set of algorithms are provided which enable the detection of feature duplication.

Keywords: Software Product Line; Evolution; Feature Duplication.

1 Introduction

The Software Product Line Engineering (SPLE) [1] is an approach that aims at creating individual software applications based on a core platform, while reducing the time-to-market and the cost of development. Many SPLE-related issues have been addressed both by researchers and practitioners, such as variability management, product derivation, reusability, etc. The focus of our work will be on SPL evolution.

The evolution of a SPL involves both changes in the domain model of the product line and the application models of derived products. This evolution consists of adding new features or modifying or deleting existing ones. As a result of these changes, many model defects can arise. In the literature, many papers have dealt with defects such as the incompleteness and inconsistency of features [3], [4], [5] and the non-conformance of constraints [6]. Other papers have dealt with duplication in the code level [7], but few have addressed the problem of features duplication. Our study is different because it aims at finding a solution to the problem of duplication in the feature level, which helps avoid wasting time and effort in implementing duplicate functionalities. Thus, we propose a framework that focuses especially on this specific issue.

Our approach allows, among others, to formalize the representation of the feature models related to software product lines and of the specifications of the new evolutions. Based on the unification of these inputs, a set of algorithms are proposed to enable an efficient detection of features duplication. An automated tool will be developed and its accuracy will be verified incrementally using a case study until we achieve satisfying results.

The rest of the paper is organized as follows. Section 2 positions our approach with related works. In Section 3, we define the research questions and the research goal. In Section 4, we describe the methodology used to carry out our study, namely the DSRM process model. Section 5 explains our approach aiming at detecting duplication when evolving software product lines and presents the progress of our work. Finally, Section 6 concludes the paper.

2 Related Work

A plethora of papers have dealt with evolution in software product lines. This evolution concerns either the feature level, the architecture level or the code level. In our approach, we focus especially on features evolution. When evolving the product line or its derived products, some defects can be introduced to the existing models. Several papers in the literature have addressed model defects. For example, Guo and Wang [12] propose to limit the consistency maintenance to the part of the feature model that is affected by the requested change instead of the whole feature model. Romero et al. [5] introduced SPLEmma, a generic evolution framework that enables the validation of controlled SPL evolution by following a Model Driven Engineering approach. This study focused, among others, on SPL consistency during evolution. Mazo [13] defines different verification criteria of the product line model and classifies them into four categories: expressiveness criteria, consistency criteria, error-prone criteria and redundancy-free criteria.

Since the model defects are introduced most of the time from specifications, many studies have dealt with the detection of defects in specifications. For instance, Lami et al. [14] present a methodology and a tool called QuARS (Quality Analyzer for Requirement Specifications) which performs an initial parsing of the specifications in order to detect automatically specific linguistic defects, namely inconsistency, incompleteness and ambiguity. Kamalrudin et al. [15] use the automated tracing tool Marama in order to give the possibility to users to capture their requirements and automatically generate the Essential Use Cases (EUC). This tool supports the inconsistency checking between the textual requirements, the abstract interactions and the EUCs. Holtmann et al. [16] proposed an approach that uses an extended CNL (controlled natural language) from the automotive industry. The CNL requirements are first translated into an ASG (Abstract Syntax Graph) typed by a requirements metamodel. Then, structural patterns are used to allow an automated correction of some requirements errors and the validation of requirements due to new evolutions. A system called CIRCE was introduced by Ambriola and Gervasi [17]. The system processes natural language requirements to build semi-formal models in an almost automatic fashion, then checks the consistency of these models and produces functional metric reports. Zowghi [4] provides an evolutionary framework that deals with inconsistency and incompleteness in a way that ensures the correctness of specifications.

An analysis of the literature shows that the majority of studies deal with inconsistency, while the problem of feature duplication has not been thoroughly treated. In addition, these studies focus either on the detection of defects in feature models or in specifications, but do not address the comparison between the new features and the existing ones to avoid the introduction of defects into the SPL.

3 Research Questions and Research Goal

Feature Duplication is among the defects that can be introduced into the model during software product lines evolution. According to [18], this defect occurs due to many reasons, such as mistakes in the design, the non-synchronization between the different people working on the project, the rapid implementation of requirements without referring to the existing models, etc. In the purpose of solving this problem, we need to answer the following research questions:

– How can we define feature duplication?
– How to detect feature duplication when evolving software product lines?
– How can we avoid the introduction of duplication in the SPL?

When evolving a software product line, the duplication of features must be verified in three levels: in the feature models (domain model and application models), in the specification of the new evolution, and between the feature models and the specification. Thus, other specific research sub-questions have to be answered:

– How can we formalize the representation of the feature models and the natural language specifications in order to facilitate the deduplication process?
– How can we detect feature duplication between the new specifications and the existing feature models?
– How to avoid the introduction of duplicate features from specifications to the existing models of software product lines?

Based on the research questions, the goal of our work is to construct:

"A framework that aims at formalizing and unifying the representation of the SPL feature models and the specifications of new evolutions, detecting duplicate features, and generating duplication-free specifications. To enable an automatic deduplication, a tool will be developed based on the proposed framework."

4 Research Methodology

In our study, we adopt a design science approach in IS. The purpose of design science as stated by Hevner et al. [19] is to build and evaluate IT artifacts designed to solve identified business problems. In order to structure our work, we use the Design Science Research Methodology (DSRM) process model proposed by Peffers et al. [20]. It is a sequential process based on six main activities: Problem identification and motivation, Definition of objectives for the solution, Design and development, Demonstration, Evaluation and Communication. Figure 1 illustrates the customized steps of the adopted process.

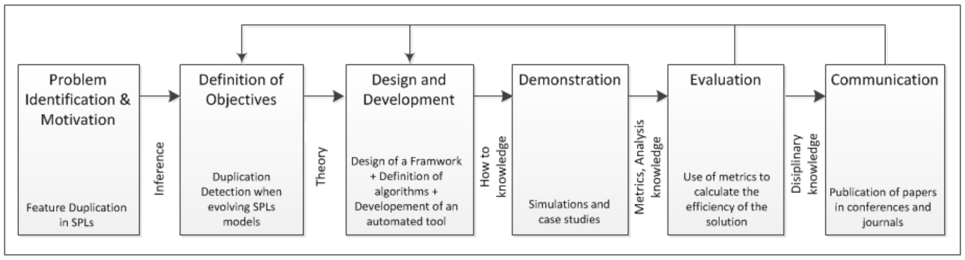
Problem Identification & Motivation
Feature Duplication in SPLs
Inference
Definition of Objectives
Duplication Detection when evolving SPLs models
Theory
Design and Development
Design of a Framwork + Definition of algorithms + Developement of an automated tool
How to knowledge
Demonstration
Simulations and case studies
Metrics, Analysis knowledge
Evaluation
Use of metrics to calculate the efficiency of the solution
Disiplinary knowledge
Communication
Publication of papers in conferences and journals

Figure 1. The DSRM process model applied to our research (adapted from [20]).

In the rest of this section, we describe the details of each of the process steps.

4.1 Problem Identification and Motivation

The introduction of new features into the domain and application models of a software product line can be the source of many model defects (e. g. inconsistency, incorrectness, incompleteness, redundancy). A review of the literature has shown that these defects have been treated by several studies, while little attention has been given to the problem of feature duplication.

The main objectives of a software product line are the reduction of time-to-market, the reduction of cost, and the improvement of product quality. The introduction of duplication in a SPL prevents from meeting these objectives, because it causes a waste of time, money and effort by implementing the same functionalities many times. In addition, duplicate features can change independently from each other, which may cause inconsistencies in the model. For example, a feature can be deleted or modified while its copy in another place remains the same, which leads to a contradiction. Moreover, duplication in the feature level impacts the quality of the product by causing the famous problem of code cloning, resulting in the recurring-bug problem and the increase of the maintenance effort [21]. A solution is thus necessary to detect duplicate features in the first step of an evolution, which is requirements analysis, which helps avoid their inclusion into the existing models from the very beginning.

4.2 Definition of Objectives for the Solution

The objective of our solution is to detect feature duplication between the existing feature models and the new specifications related to a software product line evolution. To achieve this, our artifact has two main concerns. First, the artifact must allow the formalization of the feature models and the specifications in order to facilitate the verification of defects. The second concern of the artifact is to detect and remove duplicate features by providing a set of algorithms.

4.3 Design and Development

This step consists of designing and building the artifact. Hence, we define in details the basic framework of our approach, which should meet the objectives set during the previous stage. The first action is thus to identify a method and select tools to formalize the representation of the framework inputs. The second action consists of defining a set of algorithms to detect duplication in the level of specifications, in the level of feature models, then between the specifications and the SPL models. Since manual verification has proved to be time-consuming and error prone, a tool is to be developed based on the framework in order to automatize the two actions.

4.4 Demonstration

To demonstrate the efficacy of our solution, we will use a case study from the CRM (Customer Relationship Management) field. Indeed, a CRM project has to follow continuously the market change at the lowest possible cost and satisfy new requirements of customers on tight deadlines. Consequently, an optimization of the requirements implementation is necessary, which requires an efficient verification of the model defects, especially duplication. Thus, we take the feature model of the CRM and the textual specifications of a new evolution as inputs of the automated tool. In the first place, the two inputs have to be formalized and unified. Then, the algorithms of deduplication are applied to detect and remove the duplicate features.

4.5 Evaluation

After the development of the artifact, an iterative evaluation is necessary to determine how effective it is. This evaluation is carried out using some metrics such us *the number of detected duplications in a specification*, or *the percentage of duplicate features between a specification and a feature model*. To decide whether the results generated by the artifact are satisfying or not, we define the required values of the proposed metrics in agreement with the customer.

4.6 Communication

The identified problem and the proposed artifact are communicated to researchers through several publications in conferences and journals. Hitherto, we published a first paper in the proceeding of the ICSEA 2014 Conference [8], in which we defined duplication and proposed a first design of the framework and the formalization of the basic concepts of our solution. An extended version of this paper is under review [9]. Two other papers on the same subject are under publication [10], [11]. In [10], we deal with the duplication detection in the specifications of new evolutions, while in [11] we address the duplication between the specifications and the existing feature models. As this work progresses, we intend to publish other papers to communicate the new results.

5 Proposed Approach and Work Progress

To deal with the problem of duplication in software product lines, we propose an approach based on a two-process framework. The first process consists of formalizing and unifying the representation of the SPL models and the specifications of an evolution. The second process involves the detection and removal of duplicated features caused by the new evolution. Figure 2 represents the proposed framework.

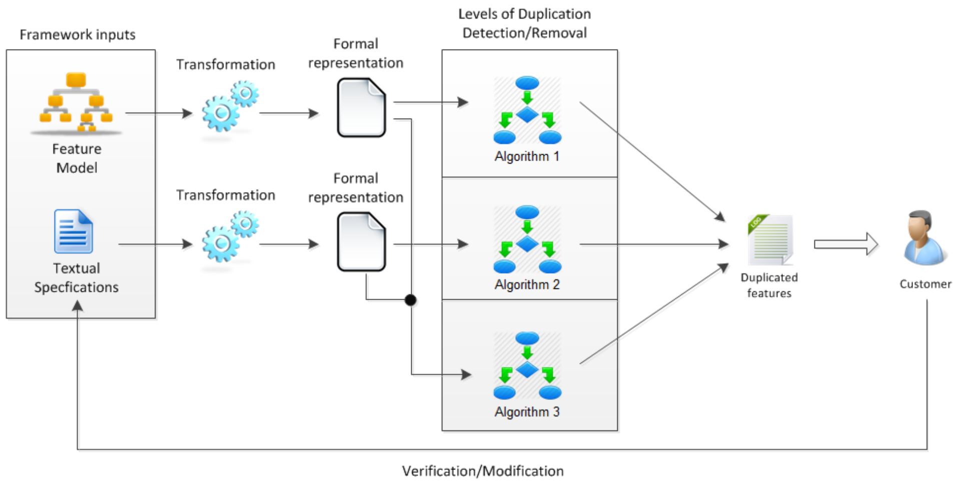

Figure 2. The Overview of the Framework.

During the domain engineering of a software product line, the common and variant features of all the specific applications are captured. To document and model variability, many approaches have been proposed. For instance, [2] introduced the orthogonal variability model which defines variability in a separate way. Salinesi et al. [22] used a constraint-based product line language. Other approaches proposed to model variability using UML models or feature models (FODA [23]). In our study, we opt for the FODA method used by the Feature-oriented software development (FOSD) paradigm [24] whose objective is to generate automatically software products based on the feature models. Hence, tools such as FeatureIDE [25] have been proposed to formalize the representation of feature models and enable the automatic selection of features of derived products. This tool will be used during the first process of our framework.

During the evolution of a derived product, the requirements are most of the time expressed in the form of natural language specifications. This form of presentation makes it difficult to detect the different defects that can occur (Duplication in our case). To deal with this problem, the solution is to transform natural language specifications into formal or semi-formal specifications. For this, we adopt a Natural Language Processing (NLP) approach. NLP is a technology of computer science whose objective is to process sentences in a natural language such as English and to build output based on the rules of a target language understandable by the machine. In our study, the purpose is to transform specifications into the same format of the SPL feature models by using syntax and semantic parsers. The syntax parser analyzes the specifications and generates the syntactic tree based on the English grammar, while the semantic parser extracts the meaning of the sentences. The operation of parsing will be performed using the OpenNLP library [26], which is a machine learning based toolkit for the processing of natural language text.

The second process of the framework consists of applying a set of algorithms of search and comparison to detect duplications in the processed specifications, feature models and between these two inputs. To help define the algorithms, we need to express mathematically the different concepts of the framework.

So far, we have identified the processes of the framework and its basic concepts [8], [9]. We have started the definition of the algorithms of duplication detection in the specifications [10], and between the specifications and the feature models [11]. In future work, we intend to implement our approach by designing an automated tool that takes as inputs the domain feature model of a SPL, the application feature model of a derived product and the specification of an evolution. The output generated by this tool is the list of duplicate features in these inputs and those caused by the evolution. This output will be sent to the customer to verify his initial needs and change them if necessary.

6 Conclusion

This paper contains an overview of our thesis dealing with software product line evolution. After a review of the existing approaches concerning the detection of model defects when evolving SPLs, we decided to focus on the resolution of a specific problem, which is feature duplication. The objective of this study is to construct a framework that helps detect and remove duplicate features introduced by new evolutions. An automated tool is to be developed to avoid the complexity of manual verification. The evaluation of the artifact will be performed by applying it to a case study from the CRM field.

References

1. P. Clements and L. Northop, Software Product Lines - Practices and Patterns, Boston: Addison-Wesley, 2002.
2. K. Pohl, G. Böckle, and F. Van Der Linden, Software Product Line Engineering Foundations, Principles, and Techniques, Berlin, Germany: Springer-Verlag, 2005.
3. A. Reder and A. Egyed, "Determining the cause of a design model inconsistency," IEEE Transactions on Software Engineering, vol. 39, no. 11, pp. 15311548, 2013.
4. D. Zowghi and V. Gervasi, "On the interplay between consistency, completeness, and correctness in requirements evolution," Information and Software Technology, vol. 46, no. 11, pp. 763-779, 2004.
5. D. Romero, S. Urli, C. Quinton, M. Blay-Fornarino, P. Collet, L. Duchien, and S. Mosser, "SPLEMMA: a generic framework for controlled-evolution of software product lines," in Proc. 17th International Software Product Line Conference co-located workshops, ACM, 2013, pp. 59-66.
6. R. Mazo, R. E. Lopez-Herrejon, C. Salinesi, D. Diaz and A. Egyed, "Conformance checking with constraint logic programming: The case of feature models," in Proc. COMPSAC'11, IEEE, 2011, pp. 456-465.
7. S. Schulze, "Analysis and removal of code clones in software product lines," Doctoral dissertation, Magdeburg, Universitt, Diss., 2013.
8. A. Khtira, A. Benlarabi, B. El Asri, "Towards Duplication-Free Feature Models when Evolving Software Product Lines," in Proc. 9th International Conference on Software Engineering Advances (ICSEA'14), Oct. 2014, pp. 107-113.
9. A. Khtira, A. Benlarabi, B. El Asri, "A Framework for Ensuring Duplication-Free Feature Models when Evolving Software Product Lines," *Submited for publication to "International Journal On Advances in Software", under review.*
10. A. Khtira, A. Benlarabi, B. El Asri, "Detecting Feature Duplication in Natural Language Specifications when Evolving Software Product Lines," *Accepted in the 10th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'15), under publication.*
11. A. Khtira, A. Benlarabi, B. El Asri, "An Approach to Detect Duplication in Software Product Lines Using Natural Language Processing," *Accepted in the Mediterranean Conference on Information and Communication Technologies (MEDICT'15), under publication.*
12. J. Guo and Y.Wang, "Towards consistent evolution of feature models," In. Software Product Lines: Going Beyond, Springer Berlin Heidelberg, 2010, pp. 451-455.
13. R. Mazo, "A generic approach for automated verification of product line models," Ph.D. thesis, Pantheon-Sorbonne University, 2011.
14. G. Lami, S. Gnesi, F. Fabbrini, M. Fusani, and G. Trentanni, "An automatic tool for the analysis of natural language requirements," Informe tcnico, CNR Information Science and Technology Institute, Pisa, Italia, Sept. 2004.
15. M. Kamalrudin, J. Grundy, and J. Hosking, "Managing consistency between textual requirements, abstract interactions and Essential Use Cases," in Proc. COMPSAC'10, IEEE, July 2010, pp. 327-336.
16. J. Holtmann, J. Meyer, and M. von Detten, "Automatic validation and correction of formalized, textual requirements," in Proc. 4th International Conference on Software

Testing, Verification and Validation Workshops (ICSTW), IEEE, Mar. 2011, pp. 486-495.

17. V. Ambriola and V. Gervasi, "Processing Natural Language Requirements," in Proc. 12th IEEE Conference on Automated Software Engineering (ASE'97), IEEE Computer Society Press, Nov. 1997, pp. 36-45.
18. A. Hunt and D. Thomas, The pragmatic programmer: from journeyman to master, Addison-Wesley Professional, 2000.
19. A. R. Hevner, S. T. March, J. Park, and R. Sudha, "Design science in information systems research," MIS quarterly, vol. 28, no. 1, pp. 75-105, 2004.
20. K. Peffers, T. Tuunanen, A. M. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," Journal of management information systems, vol. 24, no. 3, pp. 45-77, 2007.
21. L. Aversano, L. Cerulo, and M. Di Penta, "How clones are maintained: An empirical study," 11th European Conference on Software Maintenance and Reengineering (CSMR'07), IEEE, 2007.
22. C. Salinesi, R. Mazo, O. Djebbi, D. Diaz, and A. Lora-Michiels, "Constraints: the Core of Product Line Engineering," In. RCIS11, IEEE, Guadeloupe- French West Indies, France, May 19-21, 2011, pp. 1-10.
23. K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Technical Report CMU/SEI-90-TR-21, Carnegie Mellon University, Software Engineering Institute, Nov. 1990.
24. S. Apel and C. Kästner, "An Overview of Feature-Oriented Software Development," Journal of Object Technology (JOT), vol. 8, pp. 49-84, 2009.
25. C. Kästner, T. Thüm, G. Saake, J. Feigenspan, T. Leich, F. Wielgorz, and S. Apel, "FeatureIDE: A Tool Framework for Feature-Oriented Software Development," in Proc. The 31st International Conference on Software Engineering, 2009, pp. 611-614.
26. The Apache Software Foundation, "OpenNLP," opennlp.apache.org.