Logic-Based Analysis and Verification of Software Product Line Variant Requirement Model

Shamim H Ripon, Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh
Sk. Jahir Hossain, Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh
Moshiur Mahamud Piash, Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh

ABSTRACT

Software Product Line (SPL) provides the facility to systematically reuse of software improving the efficiency of software development regarding time, cost and quality. The main idea of SPL is to identify the common core functionality that can be implemented once and reused afterwards. A variant model has also to be developed to manage the variants of the SPL. Usually, a domain model consisting of the common and variant requirements is developed during domain engineering phase to alleviate the reuse opportunity. The authors present a product line model comprising of a variant part for the management of variant and a decision table to depict the customization of decision regarding each variant. Feature diagrams are widely used to model SPL variants. Both feature diagram and our variant model, which is based on tabular method, lacks logically sound formal representation and hence, not amenable to formal verification. Formal representation and verification of SPL has gained much interest in recent years. This chapter presents a logical representation of the variant model by using first order logic. With this representation, the table based variant model as well as the graphical feature diagram can now be verified logically. Besides applying first-order-logic to model the features, the authors also present an approach to model and analyze SPL model by using semantic web approach using OWL-DL. The OWL-DL representation also facilitates the search and maintenance of feature models and support knowledge sharing within a reusable engineering context. Reasoning tools are used to verify the consistency of the feature configuration for both logic-based and semantic web-based approaches.

Keywords: Domain Model, Feature Diagrams, OWL-DL, Software Product Line (SPL), Web-Based Approaches

1. INTRODUCTION

Designing, developing and maintaining a good software system is a challenging task still in this 21st century. The approach of reusing existing good solutions for developing any new application is now one of the central focuses of software engineers. Building software systems from previously developed components saves cost and time of redundant work, and improves the
A new software development paradigm, software product line (Clements & Northrop, 2001), is emerging to produce multiple systems by reusing the common assets across the systems in the product line. However, the idea of product line is not new. In 1976 Parnas (Parnas, 2001) proposed modularization criteria and information hiding for handling product line.

A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way (Clements & Northrop, 2001). Core assets are the basis for software product line. The core assets often include the architecture, reusable software components, domain models, requirements statements, documentation and specifications, performance model, etc. Different product line members may differ in functional and non-functional requirements, design decisions, run-time architecture and interoperability (component structure, component invocation, synchronization, and data communication), platform, etc. The product line approach integrates two basic processes: the abstraction of the commonalities and variabilities of the products considered (development for reuse) and the derivation of product variants from these abstractions (development with reuse) (Hein, Schlick, & Vinga-Martins, 2000).

The main idea of software product line is to explicitly identify all the requirements which are common to all members of the family as well as those that are different, and then arrange them in a model. This implies a huge model which will help the stakeholders to be able to trace any design choices and variability decisions as well. Finally, the derivation of the product is done by selecting the required variants from the model and configuring them according to product requirements.

Common requirements among all family members are easy to handle as they simply can be integrated into the family architecture and are part of every family member. But problem arises from the variant requirements among family members. Variants are usually modeled using feature diagram, inheritance, templates and other techniques. In comparison to analysis of a single system, modeling variants adds an extra level of complexity to the domain analysis. In any product line model, the same variant has occurrences in different domain model views. Different variants have dependencies on each other. Tracing multiple occurrences in different model views of any variant and understanding the mutual dependencies among variants are major challenges during domain modeling. While each step in modeling variant may be simple but problem arises when the volume of information grows. As a result, the impact of variant becomes ineffective on domain model. Therefore, product customization from the product line model becomes unclear and it undermines the very purpose of domain model.

The objective of this chapter is to provide an approach to modeling variants in the domain model of a product line. This model carries all the variant related information like specifications, interdependencies, origins of variants, etc. In developing product line, the variants are to be managed in domain engineering phase, which scopes the product line and develops the means to rapidly produce the members of the family. It serves two distinct but related purposes; firstly, it can record decisions about the product as a whole including identifying the variants for each member, and secondly, it can support application engineering by providing proper information and mechanism for the required variants during product generation (Fig. 1).

We present a variability model to draw the common and variant features of a product line. In the variability model, a tabular based approach (Ripon, 2012) is used in conjunction with feature model to portray the reusable features of product line. The tabular mechanism also provides a decision table that support product customization. However, the variability model lacks logic based formal definition. A formal verification of such variability model ensures a sound and consistent variability model facilitating a verified product customization mechanism. We present a logic verification approach for variant requirements.
Fractal Geometry and Computer Science
[www.igi-global.com/chapter/fractal-geometry-computer-science/28386?camid=4v1a](www.igi-global.com/chapter/fractal-geometry-computer-science/28386?camid=4v1a)

Polarization and Non-Positive Social Influence: A Hopfield Model of Emergent Structure
[www.igi-global.com/article/polarization-non-positive-social-influence/69960?camid=4v1a](www.igi-global.com/article/polarization-non-positive-social-influence/69960?camid=4v1a)