Journal of Logical and Algebraic Methods in Programming 85 (2016) 146–172

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

On the weaving process of aspect-oriented product family algebra

Qinglei Zhang [*], Ridha Khedri

Department of Computing and Software, McMaster University, 1280 Main Street West, Hamilton, Ontario, Canada

ARTICLE INFO

Article history:
Received 16 July 2014
Received in revised form 16 July 2015
Accepted 26 August 2015
Available online 1 September 2015

Keywords:
Product line
Product family
Aspect-orientation
Weaving process
Feature-modeling
Product family algebra

ABSTRACT

It is widely reported that product family engineering contributes to improving productivity, increasing software quality, reducing cost and labor needs, and decreasing the time to market. With the growing complexity of product families, more sophisticated techniques are required to develop, maintain and evolve the product families. More specifically, we focus on the problem space of product families engineering and address the problems in modeling large-scale feature models. Recent research on feature models adopts the principle of separation of concerns at the feature-modeling level. However, current modularization approaches still have limitation for handling *crosscutting concerns* and *unanticipated changes* at the feature-modeling level. We use a language called Aspect-Oriented Product Family Algebra (AO-PFA) to tackle those challenges. To fully attain the benefits of the aspect-oriented paradigm at the feature-modeling level, this paper presents the formalization of the weaving process for AO-PFA. Moreover, since the weaving process is associated with the *word problem*, which is in general undecidable, we prove that the weaving process of AO-PFA is convergent, leads to unambiguous weaving results, and that its rewriting system is terminating and confluent.

1. Introduction

Practice experience of different organizations has revealed that it is advantageous to develop a set of related products from core assets instead of developing them one by one independently [1]. Such a set of related products is referred to as a product family [2]. In the literature, there is a wide usage of the terms *software product line* and *product line*, which denote a family of products that share common features, managed set of features satisfying specific needs of a particular market segment, and that are developed from a common set of core assets. Therefore, a product line is a product family. Since we are not discussing any managerial issues related to products or marketing of product lines, we look at product lines as product families without any other considerations.

Feature models are widely used in product family engineering to capture the commonality and variability of product families in terms of features. Using feature models can help to further guide the following domain design, implementation, and product configuration. Feature-modeling techniques have been well accepted and applied in both academic and industrial projects [3]. However, with the increasing complexity of software product families in practice, more sophisticated feature-modeling techniques are required to address the problems in modeling large-scale feature models. It reveals that a

[*] Corresponding author.
E-mail addresses: zhangq33@mcmaster.ca (Q. Zhang), khedri@mcmaster.ca (R. Khedri).
URL: http://www.cas.mcmaster.ca/~khedri/ (R. Khedri).

http://dx.doi.org/10.1016/j.jlamp.2015.08.004
2352-2208/© 2015 Elsevier Inc. All rights reserved.

large feature model cannot be understood and analyzed if they are treated as a monolithic entity [1]. Recent research on feature models focuses on the idea of adapting the principle of separation of concerns [4] to the feature-modeling level [5,6].

In other words, a large feature model is developed by composing multiple feature models such that each feature model corresponds to the modularization of a particular concern. However, current modularization approaches at feature-modeling level still have limitation for handling *crosscutting concerns*. For example, suppose that we have an authentication feature that is in charge of identifying the caller agent that remotely communicates with a sub-system in a product family. After a while, we find that the feature interaction of the authentication feature with other features enables an intruder to take control of the system. The remedy of the detected defect in the product family leads to the introduction of new variability in the family or to the amendment of the existing variability by confining it to some products but not others. This evolution scenario due to the security issue may not be anticipated at the time of the feature-modeling stage. Consequently, it can be a tedious and costly work to recognize and make such unanticipated changes in feature models. The difficulties increase if the unanticipated changes happen when a serious defect is detected after the product family has been put on the market. The question then becomes how to supersede the current feature model of a family by a new one in an effective and efficient way [6].

To tackle the above challenges in large-scale feature models, we adopt the aspect-oriented paradigm at feature-modeling level [7]. The idea of using the aspect-oriented paradigm to the whole life-cycle of the product family engineering was proposed years ago [8]. However, techniques to adopt the aspect-oriented paradigm at the earlier analysis and design stages are only articulated recently. On the other hand, while the domain engineering is not aspect free [9], adopting the aspect-oriented paradigm at the feature-modeling level are quite limited in the literature. Adopting the aspect-oriented paradigm at the feature-modeling level fills the research gap in the literature, and is one of the essential steps to achieve a systematic aspect-oriented development for product families. The benefit of adopting the aspect-oriented paradigm at the feature modeling level are twofold. In particular, it is advantageous to improve the modifiability of large feature models and to alleviate the complexity in handling crosscutting concerns.

Handling unanticipated changes in feature models. Quantification and *obliviousness* are identified as the two characters of the aspect-oriented paradigm [10]. Quantification means the ability to make quantified statements over the underlying code, while obliviousness indicates that the original code need not be prepared for being extended with new aspects. These two characteristics of the aspect-oriented paradigm provide a way to propagate unanticipated changes in feature models by composing aspects. The composition mechanism of aspects enables all generic changing operations on feature models: 1) a set of feature combinations can be introduced to a feature model at one or multiple places, and 2) a set of feature combinations can be modified or removed from a feature model at one or multiple places. With the aspect-oriented paradigm, the *pointcut* is responsible for identifying where to make changes in the original feature models, while the *advice* captures a set of features in arbitrary combinations.

Handling crosscutting concerns in feature models. Besides improving the modifiability of large feature models, adopting the aspect-oriented paradigm at the feature-modeling level also aims to handle the crossing concerns throughout the whole life-cycle of product family engineering. While many features can be developed independently, some features are crosscutting features that inherently have dependencies with others and cannot be mapped to a single basic abstraction. One of the most efficient strategy to handle crosscutting features in the literature [11] is to separate those crosscutting features and then to compose them when necessary [12]. To alleviate the complexity in handling crosscutting concerns at the implementation level, it is advantageous to trace crosscutting concerns from the early analysis and design stages. Aspects at those analysis and design levels are sometimes referred to as *early aspects*. A more comprehensive discussion on the main benefits of dealing with early aspects is given in [9,13].

In the context of the aspect-oriented paradigm, the process to compose an aspect with the base specification is called the weaving process. The language Aspect-Oriented Product Family Algebra (AO-PFA) [7] that we are discussing the properties of its weaving system is able to articulate two types of specifications: base specifications and aspect specifications. Aspects are modification to be made to the syntactical representation of a feature model (i.e., a Product Family Algebra (PFA) specification). After weaving aspects to the base specification, the resulting specification has the same type as the base specification and therefore can be analyzed using original tools of the base specification. To increase the maintainability of feature models, a number of analysis operations are proposed in the literature to address the detection of anomalies in feature models (e.g., undesirable properties such as dead feature, conditionally dead features, false optional features, wrong cardinalities and redundancies). While the paradigm of aspect-orientation provides the flexibility to make changes on feature models, the composition of aspect and base specifications is asymmetric and complex. An aspect inherently has broad impacts on many terms and constraints of the base specification. One of the most serious problems in aspect-oriented modeling is the potential of taking a valid model and hindering its validity when weaving an aspect to it. In [14], the authors discuss these problems and present a formal verification technique of aspectual composition in the context of feature-modeling that is based on PFA. A set of validity criteria for aspects are defined with regard to their corresponding base specifications. The verification is carried prior to the weaving of the aspects to their base specifications. We refer the reader to [14] for more details on this issue.

To fully attain the benefits of the aspect-oriented paradigm at the feature-modeling level, we still have to handle the weaving process properly and automatically. In this paper, we focus on the formalization of the weaving process for AO-PFA. In particular, one of the key issues is to establish the formal representations for both base specifications and aspects.