Journal of King Saud University – Computer and Information Sciences (2016) **28**, 364–380

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

Change impact analysis for software product lines

Jihen Maâzoun [a,*], **Nadia Bouassida** [a], **Hanêne Ben-Abdallah** [b]

[a] *MIR@CL Laboratory, Sfax University, Tunisia*
[b] *Abdulaziz University, Jeddah, Saudi Arabia*

Received 5 June 2015; revised 9 January 2016; accepted 13 January 2016
Available online 30 March 2016

KEYWORDS

Software product line;
Feature model;
Model evolution;
Change impact management

Abstract A software product line (SPL) represents a family of products in a given application domain. Each SPL is constructed to provide for the derivation of new products by covering a wide range of features in its domain. Nevertheless, over time, some domain features may become obsolete with the apparition of new features while others may become refined. Accordingly, the SPL must be maintained to account for the domain evolution. Such evolution requires a means for managing the impact of changes on the SPL models, including the feature model and design. This paper presents an automated method that analyzes feature model evolution, traces their impact on the SPL design, and offers a set of recommendations to ensure the consistency of both models. The proposed method defines a set of new metrics adapted to SPL evolution to identify the effort needed to maintain the SPL models consistently and with a quality as good as the original models. The method and its tool are illustrated through an example of an SPL in the Text Editing domain. In addition, they are experimentally evaluated in terms of both the quality of the maintained SPL models and the precision of the impact change management.

Contents

* Corresponding author.
E-mail addresses: jihenmaazoun@gmail.com (J. Maâzoun), Nadia.Bouassida@isimsf.rnu.tn (N. Bouassida), HBenAbdallah@kau.edu.sa (H. Ben-Abdallah).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2016.01.005

1. Introduction

A software product line (SPL) (Clements and Northrop, 2001) represents a set of software-intensive systems that share a common set of features and software assets pertinent to a specific application domain. Besides the common product features in a domain, an SPL also describes variability points that can be used to derive new products in the SPL domain. Thanks to the predictive and organized reuse of its features and software assets, the SPL promises decreased time to market and improved software development productivity.

Even though an SPL covers several software variants, the inevitable evolution of these latter induces an evolution of the SPL itself. Product variants evolve to meet new requirements introduced by new technologies, new business goals, or modified customer preferences. Such product variants' evolution feeds-back several types of changes on their SPL, e.g., the emergence of a new feature, disappearance of an obsolete feature, structural re-organization of a feature, etc. Managing the impact of product variants' evolution on the SPL must have a means to analyze the effects of a product variant change on the SPL in terms of change operations to conduct on all of the assets describing the SPL.

An SPL is often described in terms of a problem space and a solution space (Seidl et al., 2012). The problem space captures high-level requirements usually in the form of feature models, whereas the solution space contains shared assets like source code, design and test artifacts. Given the tight correlation between both spaces, any change induced by the SPL evolution must be managed in a consistent way in all pertinent assets. Most of the works dealing with SPL evolution, e.g., Pleuss et al. (2012), Passos et al. (2013), Seidl et al. (2012), Neves et al. (2011), Laguna and Crespo (2013), and Xue (2011), focus on the evolution of feature model-oriented SPL, but they do not address the impact of a change on the consistency of the various assets of the SPL, e.g., the design and the products' code. In addition, none of the existing works analyzes the cost of a change in terms of the effort estimated to handle the change; such change impact analysis is important, for instance, to examine the value added by a change.

Because features are more structured and coarse-grained than requirements, they facilitate the understanding and traceability of an SPL evolution (Passos et al., 2013). In fact, Passos et al. (2013) argue that changes ought to be managed in a feature-oriented manner. We agree with this argument since we believe that features can be the blueprints where evolution can be managed and from where it can be traced back to the design, code and other assets. Hence, managing SPL evolution implies, first, managing change at the problem space level (i.e., the feature model) and, then, tracing these changes to the solution space (i.e., the design).

To achieve this feature-oriented SPL evolution strategy, two questions must be addressed: how to keep the consistency between the feature model and the remaining assets, particularly the design? and how to measure the effort needed in the management of each change impact? To be addressed, both questions require an explicit specification of the relationship between the SPL feature model and its design. To do so, we use our previously proposed approach which extracts the feature model from source code and specifies it using a UML profile (Maazoun et al., 2013). Unlike existing SPL feature model extraction approaches (e.g., Acher et al. (2013), Lozano (2011), Ziadi et al. (2012), Al-Msie'Deen et al. (2012), and Paskevicius et al. (2012)), ours integrates the semantic aspect of the product variants. Moreover, it describes the SPL design with a UML profile that represents the SPL variation points enriched with information extracted from the feature model. The enrichment provides for the traceability between the feature model and the design.