The Journal of Systems and Software 139 (2018) 211–237

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Controversy Corner

Change impact analysis for evolving configuration decisions in product line use case models

Ines Hajri[a], Arda Goknil[a,*], Lionel C. Briand[a], Thierry Stephany[b]

[a] SnT Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg
[b] International Electronics & Engineering (IEE), Contern, Luxembourg

ARTICLE INFO

Article history:
Received 19 June 2017
Revised 22 December 2017
Accepted 5 February 2018
Available online 15 February 2018

Keywords:
Change impact analysis
Product line engineering
Use case driven development
Use case configurator
Evolving decisions
Incremental reconfiguration

ABSTRACT

Product Line Engineering is becoming a key practice in many software development environments where complex systems are developed for multiple customers with varying needs. In many business contexts, use cases are the main artifacts for communicating requirements among stakeholders. In such contexts, Product Line (PL) use cases capture variable and common requirements while use case-driven configuration generates Product Specific (PS) use cases for each new customer in a product family. In this paper, we propose, apply, and assess a change impact analysis approach for evolving configuration decisions in PL use case models. Our approach includes: (1) automated support to identify the impact of decision changes on prior and subsequent decisions in PL use case diagrams and (2) automated incremental regeneration of PS use case models from PL use case models and evolving configuration decisions. Our tool support is integrated with IBM Doors. Our approach has been evaluated in an industrial case study, which provides evidence that it is practical and beneficial to analyze the impact of decision changes and to incrementally regenerate PS use case models in industrial settings.

1. Introduction

Product Line Engineering (PLE) is becoming crucial in many domains such as automotive and avionics where software systems are getting more complex and developed for multiple customers with varying needs. In such domains, many development contexts are use case-driven and this strongly influences their requirements engineering and system testing practices (Nebut et al., 2006a, 2006b; Wang et al., 2015a, 2015b).

For example, IEE S.A. (in the following "IEE") (IEE, International Electronics & Engineering), a leading supplier of embedded software and hardware systems in the automotive domain, follows a use case-driven development process to develop automotive sensing systems for multiple major car manufacturers worldwide. To develop a new product in a new project, IEE analysts elicit requirements as use case models from the initial customer. For each new customer of the product, IEE analysts clone the current models and identify differences to produce new use cases. With such practice, analysts loose track of commonalities and variabilities across products and they, together with the customer, need to evaluate the entire use cases. This practice is fully manual, error-prone and time-consuming, which leads to ad-hoc change management for requirements artifacts, e.g., use case diagrams and specifications, in the context of product lines. Therefore, product line modeling and configuration techniques are needed to automate the reuse of use case models in a product family.

The need for supporting PLE in the context of use case-driven development has already been acknowledged and many product line use case modeling and configuration approaches have been proposed in the literature (e.g., Eriksson et al., 2005; Eriksson et al., 2004; Fantechi et al., 2004a; Fantechi et al., 2004b; Czarnecki and Antkiewicz, 2005; Alférez et al., 2009). Most of the existing approaches rely on feature modeling, including establishing and maintaining traces between features and use case models (Sepulveda et al., 2016; Santos et al., 2015). The analysts should capture variability information as features, and establish traces between feature and use case models to model variability in use cases. For each new product in a product family, features should be selected to make configuration decisions and automatically generate use case models. In practice, many software development companies find such additional traceability and modeling effort to be impractical. In addition, requirements evolution results in changes in configuration decisions and variability information, e.g., a selected variant use case being unselected for a product. It is critical for the analysts to identify in advance the impact of such evolution for better decision-making during the configuration process.

* Corresponding author.
E-mail addresses: ines.hajri@uni.lu (I. Hajri), arda.goknil@uni.lu, goknil@svv.lu (A. Goknil), lionel.briand@uni.lu (L.C. Briand), thierry.stephany@iee.lu (T. Stephany).

https://doi.org/10.1016/j.jss.2018.02.021
0164-1212/© 2018 Elsevier Inc. All rights reserved.

For instance, impacted decisions, i.e., subsequent decisions to be made and prior decisions cancelled or contradicting when a decision changes, need to be identified to reconfigure the generated use case models.

To the best of our knowledge, there is no existing approach that explicitly supports automated change management of product line use cases for evolving configuration decisions. There are approaches (Thüm et al., 2009; Bürdek et al., 2015; Pleuss et al., 2012; Heider et al., 2012b; Paskevicius et al., 2012) that study the evolution of feature models in terms of identifying the impact of feature changes on other features, but they do not address the change impact on configuration decisions or on generated use cases.

In addition, existing configurators (e.g., Eriksson et al., 2005; Fantechi et al., 2004b; Czarnecki and Antkiewicz, 2005) do not support incremental reconfiguration of use case models, a capability that is essential in practice. For a variety of reasons, analysts manually assign traces from the configured use case models to other software and hardware specifications as well as to the customers' requirements documents for external systems (Ramesh and Jarke, 2001). Evolving configuration decisions result in the reconfiguration of Product Specific (PS) use case models. When the use case models are reconfigured for all decisions, including unimpacted decisions, manually assigned traces are lost. The analysts need to reassign all the traces after each reconfiguration. It is therefore vital to enable the incremental reconfiguration of use case models focusing only on changed decisions and their side-effects. With such support, the analysts could then reassign traces only for the parts of the reconfigured models impacted by decision changes.

In our previous work (Hajri et al., 2015), we proposed and assessed the Product Line Use case modeling Method (PUM) to support variability modeling in Product Line (PL) use case diagrams and specifications, intentionally avoiding any reliance on feature models and thus avoiding unnecessary modeling and traceability overhead. PUM adopts the existing PL extensions of use case diagrams in the work of Halmans and Pohl (Halmans and Pohl, 2003). In order to model variability in use case specifications, we introduced new product line extensions for the Restricted Use Case Modeling method (RUCM) (Yue et al., 2013). We developed a use case-driven configuration approach (Hajri et al., 2016a, 2016b) based on PUM. Our configuration approach supports guiding stakeholders in making configuration decisions (e.g., checking consistency of a decision with prior decisions) and automatically generating PS use case models from the PL models and configuration decisions. It is supported by a tool, *PUMConf (Product line Use case Model Configurator)* (Hajri et al., 2016b).

In this paper, we propose, apply and assess a change impact analysis approach, based on our use case-driven modeling and configuration techniques, to support the evolution of configuration decisions. We do not address here evolving PL use case models, which need to be treated in a separate approach. Change impact analysis provides a sound basis to decide whether a change is adequate, and to identify which decisions should be changed as a consequence (Passos et al., 2013). In our context, we aim to automate the identification of decisions impacted by changes in configuration decisions on PL use case models. Our approach supports three activities. First, the analyst proposes a change but does not apply it to the corresponding configuration decision. Second, the impact of the proposed change on other configuration decisions for the PL use case diagram are automatically identified. In the PL use case diagram, variant use cases and variation points are connected to each other with some dependencies, i.e., *require, conflict* and *include*. In the case of a changed diagram decision contradicting prior and/or subsequent diagram decisions, such as a subsequent decision resulting in selecting variant use cases violating some dependency constraints because of the new/changed decision, we automatically detect and report them. To this end, we improved our consistency checking algorithm (Hajri et al., 2016a), which enables reasoning on subsequent decisions as part of our impact analysis approach. The analyst is informed about the change impact on decisions for the PL use case diagram. One crucial and innovative aspect is that our approach identifies not only the impacted decisions but also the cause of the impact, e.g., violation of dependency constraints, changing decision restrictions, and contradicting decision restrictions. In practice, the reason of the impact is important to help the analyst identify what further changes to make on impacted decisions. Using the output of our impact analysis, the analyst should decide whether the proposed change is to be applied to the corresponding decision. Third, the PS use case models are incrementally regenerated only for the impacted decisions after the analyst makes all the required changes. To do so, we implemented a model differencing pipeline which identifies decision changes to be used in the reconfiguration of PS models. There are two sets of decisions: (i) the set of previously made decisions used to initially generate the PS use case models and (ii) the set of decisions including decisions changed after the initial generation of the PS models. Our approach compares the two sets to determine for which decisions we need to incrementally regenerate the PS models. To support these activities, we extended PUMConf.

This paper is an extension of our work published in REFSQ 2017 (Hajri et al., 2017b). The published work reported on the incremental reconfiguration of PS use case models. In the current paper, we introduce the automated impact analysis of decision changes on other decisions and we provide the details of the proposed tool support, which is made publicly available. We also improve the evaluation of our entire approach with a questionnaire study and some structured interviews with experienced engineers at IEE. To summarize, the contributions of this paper are:

- A change impact analysis approach that informs the analysts about the causes of change impacts on configuration decisions in order to improve the decision-making process and to incrementally reconfigure the generated PS use case models for the impacted decisions only;
- A publicly available tool integrated as a plug-in in IBM DOORS, which automatically identifies the impact of configuration decision changes and incrementally regenerates the PS use case models;
- An industrial case study demonstrating the applicability and benefits of our change impact analysis approach.

This paper is structured as follows. Section 2 provides the background on PUM and PUMConf on which this paper builds the proposed change impact analysis approach. Section 3 introduces the industrial context of our case study to illustrate the practical motivations for our approach. In Section 4, we provide an overview of the approach. Sections 5 and 6 provide the details of its core technical parts. In Section 7, we present our tool while Section 8 reports on an industrial case study, involving an embedded system called Smart Trunk Opener (STO). Section 9 discusses the related work. In Section 10, we conclude the paper.

2. Background

In this section we present the background regarding the elicitation of PL use case models (see Section 2.1), and our configuration approach (see Section 2.2).

In the rest of the paper, we use Smart Trunk Opener (STO) as a case study, to motivate, illustrate and assess our approach. STO is a real-time automotive embedded system developed by IEE. It provides automatic, hands-free access to a vehicle's trunk, in combination with a keyless entry system. In possession of the vehicle's