The Journal of Systems and Software 86 (2013) 1831–1849

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

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

A mixed-method approach for the empirical evaluation of the issue-based variability modeling

Anil Kumar Thurimella[a,b,*], Bernd Brügge[b]

[a] Harman International, Germany
[b] Applied Software Engineering, Technische Universität München, Germany

ARTICLE INFO

Article history:
Received 30 May 2011
Received in revised form
13 December 2012
Accepted 14 January 2013
Available online 16 March 2013

Keywords:
Rationale management
Software product lines
Variability
Requirements engineering
Empirical software engineering
Mixed-methods

ABSTRACT

Background: Variability management is the fundamental part of software product line engineering, which deals with customization and reuse of artifacts for developing a family of systems. Rationale approaches structure decision-making by managing the tacit-knowledge behind decisions. This paper reports a quasi-experiment for evaluating a rationale enriched collaborative variability management methodology called issue-based variability modeling.
Objective: We studied the interaction of stakeholders with issue-based modeling to evaluate its applicability in requirements engineering teams. Furthermore, we evaluated the reuse of rationale while instantiating and changing variability.
Approach: We enriched a quasi-experimental design with a variety of methods found in case study research. A sample of 258 students was employed with data collection and analysis based on a mix of qualitative and quantitative methods. Our study was performed in two phases: the first phase focused on variability identification and instantiation, while the second phase included tasks on variability evolution.
Results: We obtained strong empirical evidence on reuse patterns for rationale during instantiation and evolution of variability. The tabular representations used by rationale modeling are learnable and usable in teams of diverse backgrounds.

1. Introduction

Software product line engineering (*SPLE*) (Pohl et al., 2005) supports the development of a family of systems by customizing artifacts from a set of core assets. Clements and Northrop (2006) define a *software product line* (SPL) as a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market-segment or mission which are developed from a common set of core assets in a prescribed way. SPLs provide several advantages over customized software development such as improved reuse, shorter time-to market, improved cost savings and decreased defect rates.

SPLE includes two high-level processes called domain engineering and application engineering. These processes perform identical activities (such as requirements engineering, design and testing) for different purposes. The *domain engineering* process focuses on the production and maintenance of assets. An *asset* is any artifact used in the development of the systems. *Application engineering* focuses on the production of individual systems for specific market segments by instantiating and extending assets.

Variability management covers the identification of variability, instantiation of variability for individual products of a SPL and the evolution of variability, which covers the change of variability itself. The management, documentation and communication of variability are accomplished with variability models.

Artifacts are identified and changed based on the decisions of stakeholders. *Rationale management* (Dutoit et al., 2006) is the branch of science that deals with decision-making using rhetorical models that represent the reasons and justifications behind these decisions. An example of a rhetorical model is QOC (questions, options and criteria) (MacLean et al., 1991) which aids stakeholders in making decisions. QOC has been used in the context of requirements engineering (Dutoit et al., 2006) as well as distributed software development (Wolf, 2007).

Product line requirements engineering involves decision-making for variability management. *Issue-based variability modeling* (*IVM*) (Thurimella and Bruegge, 2012) is a methodology that extends traditional variability management (Pohl et al., 2005) based on QOC. In particular, IVM guides stakeholders through the elicitation, instantiation and evolution of variability. For example, application engineers can use a QOC-based collaborative discussion to configure assets for a new product by instantiating variability.

* Corresponding author at: Harman International, Germany.
Tel.: +49 89 358 70 292; fax: +49 89 358 70 170.
E-mail addresses: anil.thurimella@harman.com, anil.thurimella@gmail.com (A.K. Thurimella).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.01.038

Using QOC prescriptively[1] captures rationale behind variability decisions. The captured rationale is integrated into the variability models and is then shared between all stakeholders. Later, the rationale of the variability decision can be reused to instantiate and change variability in a different context. Variability instantiation and evolution are recurrent activities. For example, a variability model is instantiated again and again for developing new products. To systematize the reuse of rationale, in our previous contribution we have proposed the rationale reuse patterns for the instantiation and evolution of variability.

A recent *Systematic Literature Review* (SLR) from Chen and Babar (2010) (covering 625 papers on variability management) concluded that the use of empirical software engineering techniques for variability management is quite disappointing. For example, only 97 papers included an evaluation (ranging from an example, a discussion to a scientific study). Based on this review, Chen and Babar suggested the empirical evaluation of new variability management techniques (e.g. IVM). In another review, Zhang et al. (2010) identified that the state-of-the-art of empirical studies in the context of the distributed software engineering is also unsatisfactory. Burge (2008) mentions that there is a little or no empirical evaluation in the rationale management community. All the above described literature reviews motivate the empirical evaluation of IVM.

Previously, IVM was evaluated comparatively using an experimental survey with 34 software professionals (Thurimella and Bruegge, 2007), which showed that using rationale for variability improves the representation, configurability and changeability of variability. We also found evidence on reuse patterns for rationale, but it was anecdotal, because we studied only a single team. To demonstrate the evidence of the rationale reuse patterns, we wanted to do a more thorough empirical study.

However, it turned out to be difficult to validate all aspects of IVM in one empirical study, because concrete projects usually impose specific design restrictions. For example, in our experimental survey we were not able to consider the interaction of participants in various teams. Therefore, we designed another study to evaluate aspects of IVM that were not considered in the survey described in (Thurimella and Bruegge, 2007). In addition, to ensure continuity with this survey, we planned to obtain extended empirical evidence on the reuse of rationale. In particular, we aimed for the following:

A1. Evaluate the effect of the motivation of participants and the team size on the quality of output of IVM. As observed by many researchers, motivation is a key factor for allowing knowledge-sharing in teams (e.g. Palmisano, 2008; Munroe et al., 2004; Munroe and Luck, 2004). Team sizes in collaborative environments vary considerably. Babar and Kithenham (2007) described the importance of studying the impact of team size on the quality of output.

A2. IVM improves the collaboration in software engineering teams. As our research focuses on requirements engineering, we studied the applicability of IVM in requirements engineering teams consisting of people from various backgrounds. This is important because stakeholders with various backgrounds (business, technical, marketing, law, etc.) are involved in requirements engineering.

A3. The ability to reuse rationale in the future is of the main benefits of capturing rationale. Our study also aimed to validate the rationale reuse patterns of IVM during instantiation and evolution of variability. Validating these patterns can encourage domain and application engineers to use them as much as possible.

A4. A4: Identification of open issues and future enhancements to aid researches interested in rationale and variability. This is critical because Babar et al. (2010) have shown a vital need for capturing and sharing variability decisions.

The aims A1–A3 have conceptual interdependencies. For example, to evaluate A1, we varied the team size to study its effect on the quality of output. Having different sized teams helped to address A2, in particular when looking at teams with people from different backgrounds (relationship between A1 and A2) and to evaluate the rationale reuse patterns across multiple teams of various sizes (relationship between A1 and A3).

In a rationale-based decision-making process such as IVM it is difficult to control the communication between participants, their motivation and quality of documentation available on legacy decisions. A *case study* (Yin, 2003) is generally used when the extent of control over contemporary events is small. Runeson and Host (2009) suggest that software engineering can benefit from case studies because they are more flexible to design than controlled experiments (e.g. Wohlin et al., 2000). Furthermore, a case study is meant to be used in a real life context. For example, Dhungana et al. (2011) reported a case study about the evaluation of a variability modeling tool across four organizations.

Ideally, our evaluation should have taken place in an industrial environment with requirements engineering stakeholders originating from various backgrounds (e.g. business, management, technology, law, technical documentation, etc.). However, due to resource constraints, we had to evaluate IVM in a student project environment at a university. We therefore made sure, that our sample included students from various educational backgrounds, so that we could treat the student sample as a representative for requirements engineering stakeholders. Previously, Hoest et al. (2000) suggested that student samples may be used as representatives for professional samples.

Evaluation with students often involves actively creating an environment in which the study takes place. A replica of the work life situation is created and the students are told what to do and with whom to collaborate. Experimentation techniques are suitable for such student project environments (laboratories, lectures, etc.), where approaches (e.g. IVM) are evaluated by creating teams and modeling tasks for students. On the other hand, case studies are good at studying interrelated aspects that are difficult to separate and control (e.g. rationale-based decision-making). To make use of methods from case-based research in a student project environment, we embed case-based methods in a high-level quasi-experimental design, to contribute a *quasi-experiment enriched with case-based methods*. A quasi-experiment (Laitenberger and Rombach, 2003) is an experimentation technique in which the subjects are not randomly assigned to groups.

1.1. Research panorama

To evaluate our aims A1–A4, we used a sample of 258 students from the Technical University of Munich in a quasi-experiment

[1] Based on their usage rationale approaches are descriptive or prescriptive (Dutoit et al., 2006).
Descriptive approaches are aimed at describing the reasons behind the decisions that stakeholders (e.g. requirements engineers, designers) have made. Descriptive approaches might reuse records of rationale for other purposes (such as implementation, bringing new members of the development team up-to-date, or reuse of designed artifacts), but they make no attempt to alter stakeholders' reasoning. For example, QOC can be used without influencing the identification of variability. However, after identifying variability the stakeholder has to spend additional time and document the rationale information using the QOC model.
Prescriptive approaches on the other hand enhance the process by improving the reasoning of stakeholders. They typically attempt to remedy perceived deficiencies by making the reasoning more correct, consistent and thorough. IVM which uses QOC methods for variability management is an example of a prescriptive approach. IVM's main goal is to manage variations; rationale is captured as a byproduct. As stakeholders may not have the time to capture the rationale after performing their decision, we use the prescriptive approach.