The Journal of Systems and Software 82 (2009) 435–447

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Managing requirements specifications for product lines – An approach and industry case study

Magnus Eriksson[a,b,*], Jürgen Börstler[b], Kjell Borg[a]

[a] BAE Systems Hägglunds AB, SE-891 82 Örnsköldsvik, Sweden
[b] Department of Computing Science, Umeå University, SE-901 87 Umeå, Sweden

A R T I C L E   I N F O

Article history:
Received 22 November 2007
Received in revised form 22 July 2008
Accepted 22 July 2008
Available online 12 August 2008

Keywords:
Natural-language requirements specification
Software product line
Feature model
Variability management

A B S T R A C T

Software product line development has emerged as a leading approach for software reuse. This paper describes an approach to manage natural-language requirements specifications in a software product line context. Variability in such product line specifications is modeled and managed using a feature model. The proposed approach has been introduced in the Swedish defense industry. We present a multiple-case study covering two different product lines with in total eight product instances. These were compared to experiences from previous projects in the organization employing clone-and-own reuse. We conclude that the proposed product line approach performs better than clone-and-own reuse of requirements specifications in this particular industrial context.

1. Introduction

Software-intensive defense systems (e.g., vehicles) keep growing ever more complex. They consist of tightly integrated mechanical, electrical/electronic and software components, which are usually developed by different, often quite independent, organizational units. Typically, only few units are manufactured of each system, and they are always customized for each specific customer's needs. However, there are still many commonalities between systems. Furthermore, such systems have a very long life span, often 30 years or longer. High levels of reuse and high component quality are therefore critical factors in a market segment like this.

A promising approach to address these issues is known as software product line (SPL) development (Weiss and Lai, 1999; Czarnecki and Eisenecker, 2000; Clements and Northrop, 2002; Pohl et al., 2005; Sugumaran et al., 2006; van der Linden et al., 2007). SPL development has emerged as one of the leading approaches to software reuse. The basic idea of the SPL approach is to identify the commonalities in a family of products and share them in an organized way.

The main contribution of this paper is twofold. First, our tool-supported product line requirements management approach is presented. Second, an empirical study is presented where this approach is applied and evaluated in two large-scale defense projects. The SPL approach described here builds on PLUSS, our previously developed product line use case modeling approach (Eriksson et al., 2005a,b, 2006b). The extended version of PLUSS provides means for developing and maintaining natural-language requirements in a product line context. The two main reasons for extending PLUSS to also include natural-language requirements was: (a) use cases alone can not describe the whole requirements space of a non-trivial system (see, e.g., Eriksson et al., 2008); (b) natural-language requirements specifications are widely used in industry (see, e.g., Neill and Laplante, 2003).

The remainder of this paper is structured as follows: Section 2 provides a brief introduction to software product lines. Section 3 briefly reviews related work on requirements reuse. Section 4 describes an extension to the PLUSS (product line use case modeling for systems and software engineering) approach, to enable management of natural-language requirements specifications for product lines. Section 5 presents extensions to the commercial requirements management tool Telelogic DOORS developed to support the approach. Section 6 presents an industrial case study in which the proposed approach was applied and evaluated in the domain of software-intensive defense systems. Section 7 summarizes the paper and discusses some future work.

2. Software product lines

SPL development requires an organizational mind shift. When moving from single system development to product lines, several

* Corresponding author. Address: Department of Computing Science, Umeå University, SE-901 87 Umeå, Sweden.
E-mail addresses: magnus.eriksson@baesystems.se, magnuse@cs.umu.se (M. Eriksson), jubo@cs.umu.se (J. Börstler), kjell.borg@baesystems.se (K. Borg).

0164-1212/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2008.07.046

related products must be envisioned together to develop an architecture/design that can fulfill the requirements for an entire family of products.

Development in a SPL organization can be divided into two main activities (Weiss and Lai, 1999); *domain engineering* and *application engineering* (see Fig. 1):

- The purpose of the *domain engineering* activity is to develop the product line's reusable core assets. The overall goal of core asset development is to provide a production capability for products (Northrop, 2002). As illustrated in Fig. 1, some of the key artifacts of a product line are its requirements, its architecture and its components. However, the reusable core assets also include many other artifacts; for example test cases, budgets, schedules, specifications, etc. (Clements and Northrop, 2002). All of these different types of artifacts are developed to be easily (re)useable within the product line.
- The purpose of the *application engineering* activity is to generate new products utilizing the assets developed by domain engineering. The main input to this activity, besides reusable core assets, is requirements for the new product. During the application requirements activity, product requirements are analyzed and related to the existing product line requirements. The resulting product requirements will typically include both (instantiations of) existing product line requirements and new product specific requirements.

A key distinction between traditional single system development with reuse and product line development is that the reusable assets of a product line are explicitly designed to capture differences, so-called variability, between products in a product line. Variability affects all types of artifacts, from requirements to code (Thiel and Hein, 2002). In the context of product line requirements, several types of variability can exist. For example, each requirement (or group of requirements) can:

- Be mandatory or optional, for all or only some products in a product line.
- Depend on other requirements to make sense in a system.
- Be mutually exclusive to other requirements.
- Have variations in their details.

3. A brief overview of major requirements-reuse approaches

This section provides a brief overview of some major approaches for reuse of natural-language requirements. The reason we limit our survey to natural-language requirements is that natural-language requirements are the focus of the present approach and study. Not all of the presented approaches can bee seen as "true" product line engineering approaches, however, they were still included as they were considered of interest to practitioners in the area. A more comprehensive overview over requirements reuse in general can for example be found in (Lam et al., 1997). For a survey of feature modeling approaches, see (Schobbens et al., 2006); and for a brief overview of approaches related to product line use case modeling, see (Eriksson et al., 2006).

3.1. FORE – variability tree and template requirements

In the FORE (Family Of REquirements) approach (Lam, 1998) the main deliverable is the FORE document. The FORE document contains a decision model in the form of a "variability tree" and parametric template requirements. During product instantiation, this document is copied by the product developers, who create a product requirements document by removing all variability information. This is done by traversing the variability tree and selecting values for all parametric requirements.

Also the FORE approach permits modeling of how product requirements specifications may be composed from the product line requirements documentation. However, a fundamental problem inherent in this "clone-and-own" approach to requirements reuse is that each new clone starts a separate maintenance trajectory. When the FORE document is copied the link to the original document is lost, i.e., there are two identical but unrelated copies making change management very difficult. In other words, a "double-maintenance" problem is inflicted on the product in the product line. This is also a problem for other approaches that rely on copying and editing for product instantiation (see, e.g., Cybulski and Reed, 2004; Heumesser and Houdek, 2003; Cheong and Jarzabek, 1998; Faulk, 2001).

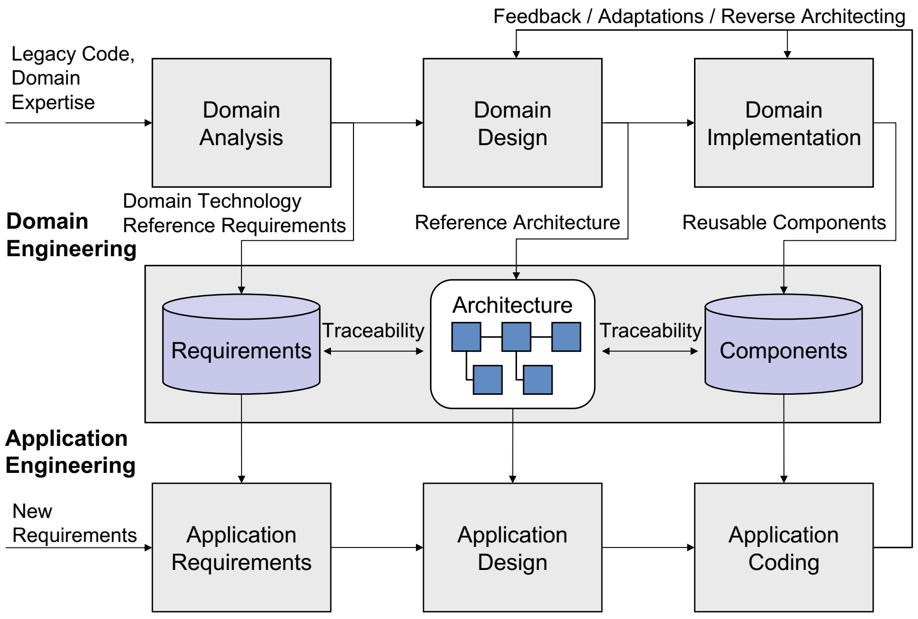

Fig. 1. Components and relationships of a reference process (PRAISE) for software product line development (van der Linden, 2002).