Information and Software Technology 69 (2016) 16–36

Contents lists available at ScienceDirect

Information and Software Technology

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

Requirements modeling languages for software product lines: A systematic literature review☆

Samuel Sepúlveda[a,]*, Ania Cravero[a], Cristina Cachero[b]

[a] Departamento de Ciencias de la Computación e Informática, Centro de Estudios en Ingeniería de Software, Universidad de La Frontera, Casilla 54-D, Temuco, Chile
[b] Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, España

A R T I C L E   I N F O

Article history:
Received 20 February 2015
Revised 26 August 2015
Accepted 27 August 2015
Available online 7 September 2015

Keywords:
Requirements engineering
Modeling languages
Software product lines
Systematic literature review

A B S T R A C T

Context: Software product lines (SPLs) have reached a considerable level of adoption in the software industry, having demonstrated their cost-effectiveness for developing higher quality products with lower costs. For this reason, in the last years the requirements engineering community has devoted much effort to the development of a myriad of requirements modelling languages for SPLs.

Objective: In this paper, we review and synthesize the current state of research of requirements modelling languages used in SPLs with respect to their degree of empirical validation, origin and context of use, level of expressiveness, maturity, and industry adoption.

Method: We have conducted a systematic literature review with six research questions that cover the main objective. It includes 54 studies, published from 2000 to 2013.

Results: The mean level of maturity of the modelling languages is 2.59 over 5, with 46% of them falling within level 2 or below -no implemented abstract syntax reported-. They show a level of expressiveness of 0.7 over 1.0. Some constructs (feature, mandatory, optional, alternative, exclude and require) are present in all the languages, while others (cardinality, attribute, constraint and label) are less common. Only 6% of the languages have been empirically validated, 41% report some kind of industry adoption and 71% of the languages are independent from any development process. Last but not least, 57% of the languages have been proposed by the academia, while 43% have been the result of a joint effort between academia and industry.

Conclusions: Research on requirements modeling languages for SPLs has generated a myriad of languages that differ in the set of constructs provided to express SPL requirements. Their general lack of empirical validation and adoption in industry, together with their differences in maturity, draws the picture of a discipline that still needs to evolve.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

The ever-increasing pace at which software customers demand new products and services has forced the software industry to devise new approaches that increase the productivity of their processes and the quality of their products. Software product lines (SPLs) are among these approaches, and can be defined as a family of systems that share a common set of core technical assets, with preplanned extensions and variations to address the needs of specific customers or market segments [1]. The purported benefits of SPLs include a reduction in the cycles of product development, productivity increases by an order of magnitude, cost reductions and a substantial improvement in the quality of products [2,3].

The aim of this paper is to account for and synthesize the current state of research reported in the literature with respect to the degree of empirical validation, origin and context of use, level of expressiveness, maturity, and industry adoption of existing requirements modeling languages used in SPLs. For some of these aspects the motivation is to update the results of previous SLRs to cover the period between 2009 and 2013, and check for improvement tendencies in the field. In particular, that is the case of industry adoption and degree of empirical validation of the languages, which, as will be discussed in Section 5, have been repeatedly pointed out as important deficiencies of the field.

☆ This document is a collaborative effort.
* Corresponding author. Tel.: +56 452744216.
E-mail addresses: samuel.sepulveda@ceisufro.cl (S. Sepúlveda), ania.cravero@ufrontera.cl (A. Cravero), ccachero@dlsi.ua.es (C. Cachero).

http://dx.doi.org/10.1016/j.infsof.2015.08.007
0950-5849/© 2015 Elsevier B.V. All rights reserved.

Also, we include other aspects that had been ignored in these previous studies, namely the level of expressiveness and maturity of the different modeling languages. The reason for this inclusion is that there are several challenges associated with the use of these languages that are related with these two aspects [4–7]. One of these challenges refers to the lack of a proper conceptual foundation, which causes significant differences in the set of concepts included in each language. In this paper, we have analyzed the pervasiveness of this problem in the field through both the analysis of the level of expressiveness of each language and its maturity level. Another challenge refers to a limited tool support. This aspect has also been addressed through the analysis of the maturity level of the different languages.

This review may be of interest to industry professionals who wish to get an updated view of the extent to which these modeling languages have been applied and validated, and which are their shortcomings and strengths in terms of expressive power and maturity. This knowledge will put them in a better position to assess the potential benefits and risks associated with adopting each modeling language. Also, it is of interest to researchers looking for gaps in research or planning to embark on additional studies on requirements modeling languages for SPLs.

We have conducted a systematic literature review (SLR) to identify and assess the set of relevant papers that could help to answer our research questions (RQs) [8]. Unlike a peer review, an SLR is a rigorous methodological review of research results, the aim of which is not only to provide all the relevant evidence – based on a set of *a priori* established criteria – to answer an RQ, but also to support the development of evidence-based directives for professionals [9].

The remainder of this paper is organized as follows. Section 2 shows some concepts related to SPLs and variability. Section 3 explains the SLR methodology. Section 4 provides a discussion about our main findings and results. Section 5 outlines the main related work. Finally, Section 6 presents the conclusions and further lines of research.

2. Background

SPLs can provide significant gains in quality and productivity through systematic reuse of software conceptual structures [10] and the provision of mechanism to define system adaptation and configuration options [11]. For systems such as embedded safety- or mission-critical ones, much of the development effort goes into understanding, specifying, and validating the requirements. If developers can re-use rather than re-do requirements for families of similar systems, they can improve productivity while significantly reducing the opportunity of requirements errors [10].

One of the key concepts in SPLs is variability, which provides the required flexibility for product differentiation and diversification [12]. Variability is defined as the ability of SPLs to be exchanged, customized, configured or extended to be of use in a specific context, and this is introduced through alternative definitions of reusable devices, included in the family of software products, that give rise to different products [13]. For example, a company developing Smart Home Systems (a system that observes and controls properties of a house) may need to be able to adapt them to each customer's home. Also, they need to be able to leave configuration choices to the customer and specificities to address different market segments. These variability needs makes this kind of system a typical example of SPL [14].

Variability management is considered one of the key aspects that distinguish SPLs from other software development approaches. It involves extremely complex and challenging tasks that must be supported by appropriate methods, techniques, and tools [12]. To date, several representations of variability, together with mechanisms that facilitate its systematic exploitation, have been reported in the literature, and some efforts have been made to unify the concepts and relations of the field [15].

In order to model such variability, methods can be broadly categorized in two groups: the feature-based group, which comprises methods that model the common and variable features of a product family (an analysis-oriented perspective), and the architecture-based group, which includes methods that describe variability in terms of components, interfaces, connectors and so on (a more design-oriented perspective) [49]. At the requirements engineering (RE) stage the focus is on the first group, that is, languages based on the modeling of features.

In the RE context, the term *feature* is a *problem-oriented* concept that refers to *a prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems* [11]. Back to our example of a Smart Home System SPL, in this kind of system it is common to find a feature to model security issues. This feature will be further decomposed in sub-features depending on the different security mechanisms that the SPL is able to provide (*e.g.* burglar alarm, police call) [11].

Most of the complexity of feature languages lies in the interoperation of *features*, some depending on each other while others being mutually disruptive [11]. The most popular way to express such interoperation is by means of a feature model (FM) [4]. A FM has a tree-like structure, with its root node representing the software product family. The characteristics of the family are organized down the tree. The FM leaves represent individual or configured components that can be assembled to generate a particular application [16]. The FM was first presented as part of the Feature-oriented domain analysis (FODA) method [17], although this model is now present, with slight variations, in virtually all the SPL methods that rely on a visual representation of product characteristics.

3. Research methodology

As previously mentioned, this study has been carried out according to the SLR methodology described by Kitchenham and Charters [9].

In this paper, we use the term SLR in its broad sense. The reason is that our contribution sits in between that of a mapping study and that of a conventional SLR, which are the two more common types of secondary studies in the discipline [18]. While this study cannot be categorized as a conventional SLR (whose objective is the compilation of all relevant quantitative data regarding an RQ and the execution of some sort of meta-analysis), it does not fit well under the category of mapping studies either. Mapping studies, according to Kitchenham et al. [19], aim to "identify all research related to a specific topic rather than addressing the specific questions that conventional SLRs address". Similarly, Petersen et al. [20] indicate that a systematic map is a method to build a classification scheme and structure a Software Engineering (SE) field of interest. The analysis of results focuses on frequencies of publications for categories within the scheme. Thereby, the coverage of the research field can be determined. In this paper, we have identified the existing research related with our topic of interest (requirements engineering languages for SPLs). We have classified the existing research and we have analyzed trends regarding the pace of publications and the evolution in the use of different notations along time. Additionally, we have analyzed the individual characteristics of the existing requirements modeling languages in SPLs, and this has driven the inclusion of RQs that involve in-depth analyses of each language.

Next, in Section 3.1 we define the SLR protocol. Then, in Section 3.2 we describe the study selection and the data extraction process whose outcome is the final list of papers included in our SLR. Fig. 1 depicts the whole process.

Download English Version:

https://daneshyari.com/en/article/550494

Download Persian Version:

https://daneshyari.com/article/550494

Daneshyari.com