The Journal of Systems and Software 106 (2015) 132–149

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review

Noor Hasrina Bakar[a,b,*], Zarinah M. Kasirun[a], Norsaremah Salleh[c]

[a] Department of Software Engineering, Faculty of Computer Science & Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
[b] Department of ICT, Centre for Foundation Studies, International Islamic University Malaysia, 46350 Petaling Jaya Selangor, Malaysia
[c] Department of Computer Science, Kulliyyah of Information & Communication Technology, International Islamic University Malaysia, 53100 Jalan Gombak, Kuala Lumpur, Malaysia

ARTICLE INFO

Article history:
Received 17 April 2014
Revised 30 April 2015
Accepted 3 May 2015
Available online 9 May 2015

Keywords:
Feature extractions
Requirements reuse
Software product lines
Natural language requirements
Systematic literature review

ABSTRACT

Requirements for implemented system can be extracted and reused for a production of a new similar system. Extraction of common and variable features from requirements leverages the benefits of the software product lines engineering (SPLE). Although various approaches have been proposed in feature extractions from natural language (NL) requirements, no related literature review has been published to date for this topic. This paper provides a systematic literature review (SLR) of the state-of-the-art approaches in feature extractions from NL requirements for reuse in SPLE. We have included 13 studies in our synthesis of evidence and the results showed that hybrid natural language processing approaches were found to be in common for overall feature extraction process. A mixture of automated and semi-automated feature clustering approaches from data mining and information retrieval were also used to group common features, with only some approaches coming with support tools. However, most of the support tools proposed in the selected studies were not made available publicly and thus making it hard for practitioners' adoption. As for the evaluation, this SLR reveals that not all studies employed software metrics as ways to validate experiments and case studies. Finally, the quality assessment conducted confirms that practitioners' guidelines were absent in the selected studies.

1. Introduction

Software product lines engineering (SPLE) refers to software engineering methods, tools, and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production (Northrop and Clements, 2015). These shared software assets or sometimes referred to as core assets may include all artefacts in the product lines: requirements, architecture, codes, test plans, and more (Pohl et al., 2005). Meanwhile, requirements reuse (RR) is the process of reusing previously defined requirements for an earlier product and applying them to a new, similar product. Generally, RR can produce more benefits than only the design code reuse since it is done earlier in the software development (Clements and Northrop, 2002). When RR was planned systematically in the SPLE context, several studies (Eriksson et al., 2006; Monzon, 2008; Moros et al., 2013; Von Knethen et al., 2002) indicated positive improvement in software development: speed up time to market, increase team productivity, reduce development costs in the long run, and provide a better way of sustaining core assets' traceability and maintainability. Software requirements can be reused either in an ad hoc basis such as in clone and own applications, software maintenance, or when systematically planned in SPLE. However, many problems exist when dealing with ad hoc reuse of natural language (NL) requirements. The problems with manual requirements reuse includes arduous (Weston et al., 2009), costly (Niu and Easterbrook, 2008), error-prone (Ferrari et al., 2013), and labour-intensive (Boutkova and Houdek, 2011) process, especially when dealing with large requirements.

In the following subsections, we will briefly describe the terms that bring together features extraction and RR in the SPLE context: requirements versus features, core assets development in SPLE, and the contributions of our work in SPLE.

1.1. Requirements versus features

Firstly, it is important to understand the key distinction between software requirements and features. Software requirements describe the functionality of a software system to be developed. The definition

* Corresponding author at: Department of Software Engineering, Faculty of Computer Science & Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia. Tel.: +60 126927506
E-mail addresses: noor.hasrina@gmail.com, noorhasrina@iium.edu.my (N.H. Bakar), zarinahmk@um.edu.my (Z.M. Kasirun), norsaremah@iium.edu.my (N. Salleh).

http://dx.doi.org/10.1016/j.jss.2015.05.006
0164-1212/

of software *requirements* in accordance with IEEE Standard Glossary of Software Engineering Terminology, page 62 in IEEE Computer Society (1990) is given as:

(1) "A condition or capability needed by a user to solve a problem or achieve an objective.
(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
(3) A documented representation of a condition or capability as in 1 or 2."

The majority of requirements are written in NL (Denger et al., 2003). This is because text is commonly used to convey information to communicate stakeholders' needs (Niu and Easterbrook, 2008). Pohl et al. (2005) emphasised that in SPLE, software requirements are documented either by using NL or model-based. As an example, NL requirements do not only appear in the form of Software Requirements Specification (SRS) format. NL requirements can also be recorded in the forms of goals and features, product descriptions including product brochures, user manual, or scenarios. Model-based requirements can be recorded in the forms of functional data analysis such as data flow diagram, UML models such as class diagram, state dependent system behaviour and more, and they are usually supplemented by NL descriptions of features (Nicolás and Toval, 2009).

Meanwhile, software feature is defined as a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems (Kang et al., 1990). In most cases, requirements tend to be lengthy in nature, while features represent services that a system must provide to fulfil customers' needs, most of the time in a shorter or precise manner. Software features tend to be more focused and granular as compared to software requirements.

1.2. Core assets development in SPLE

Fundamentally, in SPLE, core assets (including requirements) can be developed through three approaches: **proactive, reactive,** or **extractive** (Krueger, 1992). In the proactive approach, assets are developed prior to software development. In the reactive approach, common and variable artefacts are iteratively developed during the software development. Reuse in the context of extractive tends to be in between the proactive and reactive (Krueger, 2002). To ease the transition from single systems to software mass customisation, Krueger proposed the extractive adoption model as a means to reuse existing products for SPLE (Krueger, 2001). With the extractive approach, core assets are no longer created from scratch, but extracted from the existing repository and reused in developing similar system. The extractive approach is particularly very effective with organisations that have accumulated development experience and artefacts in a domain and intended to quickly shift from conventional software development to SPLE (Frakes and Kang, 2005). Niu and Easterbrook (2008) highlighted the basic tenets of extractive approach of software product lines (SPL) that include maximal reuse and reactive development, particularly for small and medium-sized enterprises.

1.3. Contributions of this work in SPLE

Up to date, various research works have been produced in SPLE focusing on the product line architecture, domain analysis tools (Lisboa et al., 2010), variability management (Chen and Ali Babar, 2011; Metzger and Pohl, 2014), detailed design, and code reuse (Faulk, 2001). However, there are few works that looked at the extractions of features from the requirements in SPLE (Niu and Easterbrook, 2008; Alves et al., 2008; Kumaki et al., 2012; Davril et al., 2013). Therefore, more parties can benefit from the formulation of feature extractions from NL requirements when various forms of input (not only SRS) are taken into consideration. In particular, we are interested in how current approaches that are used to extract features from NL requirements can support the reuse of requirements in SPL. Additionally, we are also looking at the implications for further research in this area. None of the related reviews presented in Section 2 adequately covers these issues. Fig. 1 illustrates the scope of our SLR contribution in regard to other related works in SPLE.

SPLE is a paradigm to develop software applications (software intensive systems and software products) using platforms and mass customisation (Pohl et al., 2005). Meyer and Lehnerd (1997) defined software platforms as a set of software subsystems and interfaces that form a common structure from which a set of derivative products can be efficiently developed and produced. The subsystems within a platform contain artefacts beyond source-codes which include requirements, architectures, test plans, and other items from the development process.

SPLE is distinct from the development of a single system, in which it involves two life cycles: domain engineering (DE) and application engineering (AE) (Pohl et al., 2005). In DE, the reusable assets (including requirements) are built. This is an entire process of reusing software assets for the production of a new similar system, with variation to meet customer demands. DE is responsible for defining and realising the commonality and the variability of software product line. On the other hand, AE is the process where the applications of the product lines are built by reusing the domain and exploiting the product line variability (Pohl et al., 2005). The most important part in Fig. 1 is the domain analysis (DA), where a specific set of common and variable features from the existing requirement documents to be reused for developing similar product is identified. DA is the key method for realising systematic software reuse (Frakes and Kang, 2005). It can provide a generic description of the requirements (either in model-based or natural language form) for that class of systems and a set of approaches for their implementation (Kang et al., 1990).

The process of reusing requirements takes place within the DA process and it is a part of general requirements engineering. Reuse of software artefacts is the key aspect of SPLE. This is different to non-SPL based methodology in Software Engineering where requirements are gathered through elicitations of stakeholders' needs with or without using the existing documentation for similar systems. In normal RE, reuse of requirements is not planned systematically and always occurs in an ad hoc manner. Pohl describes Domain Design as a subprocess within DE that refines the variability into design variability, defining the reference architecture/platform (Pohl et al., 2005). Essentially, as a result, the outcome from all sub-processes within the DE phase should be the representation of most (if not all) possible application for a given domain. Related literature reviews around the DA area were numbered in Fig. 1 and its summary is presented in Section 2.

Meanwhile, the second lifecycle, AE is concerned with the configuration of a product line into one concrete product based on the preferences and requirements of stakeholders produced in DE. Usually, the domain model produced within DE will now be used in AE. In AE, instance software products are often derived through the consultation with domain stakeholders that have specific requirements in mind (Bagheri and Ensan, 2013). Selection of desirable features that is now readily available should be gradually performed with ample interaction with the stakeholders, as described by Czarnecki et al. (2004) as staged configuration.

Various literature reviews have been published in the area of DE and AE (as numbered in Fig. 1); however, none of the reviews reported the approaches used to select features from NL requirements for reuse in SPLE. This SLR was performed in order to obtain a better comprehension of the current state-of-the-art in feature extraction approaches from NL requirements for reuse in SPLE.