Information and Software Technology 000 (2016) 1–18

Contents lists available at ScienceDirect

Information and Software Technology

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

Software product lines traceability: A systematic mapping study

Tassio Vale[a,b,c,*], Eduardo Santana de Almeida[b,c], Vander Alves[d], Uirá Kulesza[e], Nan Niu[f], Ricardo de Lima[d]

[a] Center of Exact Sciences and Technology, Federal University of Recôncavo da Bahia, Cruz das Almas, BA, Brazil
[b] Computer Science Department, Federal University of Bahia, Salvador, BA, Brazil
[c] RiSE - Reuse in Software Engineering, Salvador, BA, Brazil
[d] Computer Science Department, University of Brasilia, Brasilia, DF, Brazil
[e] Computer Science Department, Federal University of Rio Grande do Norte, Natal, RN, Brazil
[f] Department of EECS, University of Cincinnati, Cincinnati, OH, United States

ARTICLE INFO

Article history:
Received 8 March 2016
Revised 10 December 2016
Accepted 13 December 2016
Available online xxx

Keywords:
Systematic mapping study
Software product lines
Software and systems traceability
Software reuse

ABSTRACT

Context: Traceability in Software Product Lines (SPL) is the ability to interrelate software engineering artifacts through required links to answer specific questions related to the families of products and underlying development processes. Despite the existence of studies to map out available evidence on traceability for single systems development, there is a lack of understanding on common strategies, activities, artifacts, and research gaps for SPL traceability.

Objective: This paper analyzes 62 studies dating from 2001 to 2015 and discusses seven aspects of SPL traceability: main goals, strategies, application domains, research intensity, research challenges, rigor, and industrial relevance. In addition to the analysis, this paper also synthesizes the available evidence, identifies open issues and points out areas calling for further research.

Method: To gather evidence, we defined a mapping study process adapted from existing guidelines. Driven by a set of research questions, this process comprises three major phases: planning, conducting, and documenting the review.

Results: This work provides a structured understanding of SPL traceability, indicating areas for further research. The lack of evidence regarding the application of research methods indicates the need for more rigorous SPL traceability studies with better description of context, study design, and limitations. For practitioners, although most identified studies have low industrial relevance, a few of them have high relevance and thus could provide some decision making support for application of SPL traceability in practice.

Conclusions: This work concludes that SPL traceability is maturing and pinpoints areas where further investigation should be performed. As future work, we intend to improve the comparison between traceability proposals for SPL and single-system development.

1. Introduction

Software Product Line (SPL) engineering is a paradigm to develop variant-rich systems at lower costs, in shorter time, and with higher quality [1]. To achieve such benefits, two development phases take place: domain engineering, focusing on generating assets that systematically incorporate domain commonalities and variabilities; and application engineering, which instantiates domain engineering assets into specific software products to address customers' needs.

One important aspect of software engineering is software traceability. In the SPL area, it refers to the interrelation of software artifacts through required links to answer specific questions related to the families of products and the underlying development processes.

Tracing in SPLs becomes even more complex when compared to single system development, since there are additional issues emerging from variability when applying systematic traceability. For instance, Jirapanthong and Zisman [2] discuss the following difficulties for achieving traceability in SPL: (i) the large num-

* Corresponding author at: Center of Exact Sciences and Technology, Federal University of Recôncavo da Bahia, Cruz das Almas, BA, Brazil.
E-mail address: tassio.vale@ufrb.edu.br (T. Vale).

http://dx.doi.org/10.1016/j.infsof.2016.12.004
0950-5849/

ber and heterogeneity of documents when compared to traditional software development; (ii) the need to have a basic understanding of variability consequences under all development phases; and (iii) the need to establish relationships between product members (of the family) and the product line architecture, or relationships between the product members themselves.

Since the last decade, researchers have been investigating traceability and a considerable number of proposals are available in the literature. Despite the existence of studies to map out available evidence on traceability for single systems development [3], there is a lack of understanding on common strategies, activities, artifacts, and research gaps for SPL traceability. Therefore, a literature review is important to investigate the state-of-the-art and identify research topics that researchers and practitioners should systematically address.

Accordingly, this systematic mapping study aims to synthesize existing research related to systematic traceability in the SPL field, which we refer to as SPL traceability. In particular, we identify common practices, research trends, open issues, and research topics for improvement. In addition, we categorize the primary studies' evidence according to a well-defined scheme and emerging knowledge structures. Further, the study balances the relevance of research by analyzing the potential industrial impact in terms of rigor and industrial relevance.

As a result, this work provides a structured understanding of SPL traceability and identifies possibilities for future research. In addition, the unveiled lack of evidence regarding the application of research methods indicates the need for more rigorous SPL traceability studies with better description of context, study design, and limitations. It also reinforces that more data about existing methods should be collected. Furthermore, for practitioners, although most identified studies have low industrial relevance, a few of them have high relevance and thus could provide some decision making support for application of SPL traceability in practice. Nonetheless, future evaluation of SPL traceability should be carried out by more well-described studies conducted with industrial-strength applications having practitioners as subjects.

The remainder of this paper is organized as follows: Section 2 presents the background on SPL and software traceability research areas; Section 3 describes the mapping study process that guides our research; Section 4 presents the results from the analysis and synthesis activities; Section 5 discusses the main findings of this work; Section 6 presents threats to the validity of this work; Section 7 argues on the related work; and Section 8 presents the concluding remarks.

2. Background

2.1. Software product lines

Software product line engineering addresses the development of software system families for a specific market segment [4]. A system family is a set of programs that share common functionalities and maintain specific functionalities that vary according to specific systems [5].

Several benefits are expected through the adoption of SPL engineering [4], such as: cost reduction, the improvement of product quality and development productivity, and shortened time to market. A software product line is usually modeled, designed and implemented in terms of common and variable features [6], which are system properties or functionalities relevant to stakeholders. They are used to capture commonalities or discriminate among systems in SPLs.

The SPL development is typically organized in [1]: (i) domain engineering – that focuses on the development of common artifacts, variation points and variants of the SPL products at the requirements, architecture, code and testing levels; and (ii) application engineering – that reuses the artifacts produced during domain engineering to instantiate different SPL products.

Considering the high upfront investment associated to the development of a SPL from scratch, alternative SPL adoption strategies have been proposed and used in practice. Krueger [7] discusses two alternative adoption strategies to the traditional SPL development from scratch using domain and application engineering: extractive strategy – that advocates the SPL development through the extraction and isolation of common and variable features from existing systems to derive an initial version of the SPL; and reactive strategy – motivates the incremental development of SPLs where initially only a few products are considered; when new requirements and/or products need to be addressed, the common and variable artifacts of the SPL are incrementally extended.

2.2. Software and systems traceability

Traceability in software engineering has been addressed since the NATO conference, in 1968. Over the years, commercial tools were developed and applied in several domains (aviation, healthcare, etc.) increasing the research activity in this area, moving from manual to semi-automated and automated traceability strategies [3]. Despite the growing interest and consequently the higher number of proposals, this area still remains challenging.

Software and systems traceability is the ability to interrelate software engineering artifacts, maintaining the required links over time, and such information can answer questions about both the software product and its development process [8]. Systematic traceability eases communication between stakeholders and allows the dissemination of feedback to architects and designers about the current state of the development. More specifically, the resulting trace information can be used for impact analysis for estimating change effort, ensuring sufficient test coverage, supporting safety case or some other third party certification, identifying potential candidates for reuse, tracking project progress, or reconstructing earlier decisions to avoid rework.

Current research addresses a set of the traceability issues for all software traceability process phases. For the planning and management phases, they focus on understanding stakeholders' needs, developing techniques to support traceability that fits specific needs of a project. The creation and maintenance of traces explore trace creation, trace maintenance, and trace integrity. Finally, the use of traces aims to access, query and visualize trace data [8].

Realizing traceability involves establishing and using traces, which are strictly related to trace artifacts and trace links [3]. *Trace artifact* represents any piece of data amenable to tracing, which could vary in terms of granularity (e.g. architectural module, source code fragment, etc.). The element that enables tracing is the *trace link*, an association forged between two trace artifacts. Such links have an intrinsic directionality, emerging from the source artifact to the target artifact (e.g., the architectural module guides the data flow output of the source code fragment). Depending on the implementation, trace links could be *bi-directional* by including the reverse direction of the association. According to [3], we can refer to *trace* as the "complete triplet of trace elements that enable the juxtaposition of two pieces of data: the source artifact, the target artifact and the trace link". This terminology is used throughout this paper.

Trace links can be used to analyze how changes applied to existing artifacts might affect other artifacts at the same or different levels of abstraction. Several trace link classifications have been proposed by the community. For instance, Lindval and Sandah [9] define the concept of vertical and horizontal traceability to distinguish between trace links that refer to the same model or to different models, respectively.

Please cite this article as: T. Vale et al., Software product lines traceability: A systematic mapping study, Information and Software Technology (2016), http://dx.doi.org/10.1016/j.infsof.2016.12.004