The Journal of Systems and Software 111 (2016) 157–184

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

A systematic mapping study on the combination of software architecture and agile development

Chen Yang[a,b], Peng Liang[a,*], Paris Avgeriou[b]

[a] State Key Lab of Software Engineering, School of Computer, Wuhan University, 430072 Wuhan, China
[b] Department of Mathematics and Computing Science, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands

ARTICLE INFO

Article history:
Received 18 February 2015
Revised 16 September 2015
Accepted 20 September 2015

Keywords:
Software architecture
Agile development
Architecting approach

ABSTRACT

Context: Combining software architecture and agile development has received significant attention in recent years. However, there exists no comprehensive overview of the state of research on the architecture-agility combination.
Objective: This work aims to analyze the combination of architecture and agile methods for the purpose of exploration and analysis with respect to architecting activities and approaches, agile methods and practices, costs, benefits, challenges, factors, tools, and lessons learned concerning the combination.
Method: A systematic mapping study (SMS) was conducted, covering the literature on the architecture-agility combination published between February 2001 and January 2014.
Results: Fifty-four studies were finally included in this SMS. Some of the highlights: (1) a significant difference exists in the proportion of various architecting activities, agile methods, and agile practices employed in the combination. (2) none of the architecting approaches has been widely used in the combination. (3) there is a lack of description and analysis regarding the costs and failure stories of the combination. (4) twenty challenges, twenty-nine factors, and twenty-five lessons learned were identified.
Conclusions: The results of this SMS help the software engineering community to reflect on the past thirteen years of research and practice on the architecture-agility combination with a number of implications.

1. Introduction

Software Architecture (SA)[1] represents "*the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution*" (ISO, 2011). SA acts as a high-level design and as a means of performing complicated trade-offs (e.g., quality requirements) between stakeholders of software-intensive systems (Bass et al., 2012). Critics of traditional architecting processes argue that they tend to employ a Big Design Up-Front (BDUF) approach leading to excessive documentation and implementation of possibly unneeded features, which introduce additional development effort (Abrahamsson et al., 2010). As an alternative to BDUF, agile development is proposed, which mainly focuses on adapting to change and delivering products of high quality through simple work-processes (Dingsøyr et al., 2010).

Freudenberg and Sharp listed the top 10 burning research questions in the agile development community collected from about 300 practitioners at the XP 2010 conference, and the question "*Architecture and agile - how much design is enough for different classes of problem?*" is ranked in the second position (Freudenberg and Sharp, 2010). Many approaches, techniques, processes, and tools have been proposed and developed to support either the use of SA or the use of agile methods in software development. However, ways to combine them is a challenging issue, which has been heavily debated over the past years (Abrahamsson et al., 2010). To understand how SA and agile development can be used together, we conducted a systematic mapping study (SMS) to collect primary studies on **using software architecture in agile development as well as using agile methods in architecture-centric development**. This study aims at identifying available evidence on various aspects of this topic (for details see Section 3.1), and spotting gaps in the application of architecture in agile development and the other way round, i.e., the application of agile methods and practices used within architecture-centric development.

An SMS aims at mapping the evidence at a high level for a specific topic, and is particularly suitable when the studied topic is very broad (Kitchenham and Charters, 2007). Systematic literature review (SLR) is another form of secondary study, which provides "*a means of identifying, evaluating, and interpreting all available research relevant to a particular research question*" (Kitchenham and Charters, 2007). We

* Corresponding author. Tel.: +86 27 68776137; fax: +86 27 68776027.
E-mail address: liangp@whu.edu.cn (P. Liang).
[1] For readability and clarity, we list all the abbreviations used in this paper in Appendix B for reference.

http://dx.doi.org/10.1016/j.jss.2015.09.028
0164-1212/© 2015 Elsevier Inc. All rights reserved.

decided to conduct an SMS because the studied topics (software architecture and agile development) cover very broad areas. However, we did not just simply perform a mapping (classifying primary studies into categories), but also synthesized data from studies (by using grounded theory (Strauss and Corbin, 1998)).

The rest of this paper is structured as follows. Section 2 provides the context of this SMS, i.e., architecting activities and agile practices. Section 3 details the mapping study process with the research questions. Section 4 presents the results of the research questions. Section 5 further discusses the study results with their implications for researchers and practitioners. Section 6 presents the threats to validity. Section 7 concludes this work.

2. Context

The contextual elements of this SMS include architecture-centric development and agile development. We discuss the topic of architecture-centric development through architecting activities and agile development through agile practices. We summarize eleven architecting activities in Section 2.1, which are collected from existing literature, and briefly review the practices of agile development in Section 2.2.

2.1. Architecting activities

The architecting process is comprised of a number of specific architecting activities (covering the entire architectural lifecycle) and a number of general architecting activities (supporting the specific activities). The specific activities are listed and described below:

- **Architectural Analysis (AA)** is aimed at defining the problems an architecture must solve. The outcome of this activity is a set of architecturally significant requirements (ASRs) (Hofmeister et al., 2007).
- **Architectural Synthesis (AS)** proposes candidate architecture solutions to address the ASRs collected in AA, thus this activity moves from the problem to the solution space (Hofmeister et al., 2007).
- **Architectural Evaluation (AE)** ensures that the architectural design decisions made are the right ones, and the candidate architectural solutions proposed in AS are measured against the ASRs collected in AA (Hofmeister et al., 2007).
- **Architectural Implementation (AI)** realizes the architecture by creating a detailed design (Tang et al., 2010).
- **Architectural Maintenance and Evolution (AME):** Architectural maintenance is to change an architecture for the purpose of fixing faults (ISO, 2006, ISO, 2011) and architectural evolution is to respond to new requirements at architectural level (Postma et al., 2004). In this SMS, we simply considered architectural maintenance and architectural evolution as one activity, in which an architecture is changed either to fix faults or to implement new requirements.

An architecting process is composed of the five specific activities mentioned above (Hofmeister et al., 2007; Tang et al., 2010). There are also general architecting activities (e.g., Architectural Description) identified in (Li et al., 2013) that are meant to support the specific activities. For example, Architectural Description (ADp) can be used to specify and document the candidate architecture solutions during Architectural Synthesis (AS). The general activities are described as follows:

- **Architectural Recovery (AR)** is used to extract the current architecture of a system from the system's implementation (Medvidovic and Jakobac, 2006).
- **Architectural Description (ADp)** is used to describe the architecture with a set of architectural elements (e.g., architecture views). This activity can help stakeholders (e.g., architects) to understand the system, and improve the communication and cooperation among stakeholders (ISO, 2011).
- **Architectural Understanding (AU)** is used to comprehend the architectural elements (e.g., architectural decisions) and their relationships in an architecture design (Li et al., 2013).
- **Architectural Impact Analysis (AIA)** is used to identify the architectural elements, which are affected by a change scenario (Bengtsson et al., 2004). The analysis results include the components in architecture that are affected directly, as well as the indirect effects of changes to the architecture (Bengtsson et al., 2004).
- **Architectural Reuse (ARu)** aims at reusing existing architectural design elements, such as architecture frameworks, decisions, and patterns in the architecture of a new system (IEEE, 2010).
- **Architectural Refactoring (ARf)** aims at improving the architectural structure of a system without changing its external behavior (Babar et al., 2013; Fowler et al., 1999).

2.2. Agile practices

Agile development aims at stripping away, as much as possible, the effort-intensive activities in software development (Erickson et al., 2005), and focuses on quick response to various changes of a project (Erickson et al., 2005). Agile practices are in general practices used to support agile development. An example of such an agile practice is iterative and incremental development, which focuses on small releases and a planning strategy based on a release plan and an iteration plan (Augustine, 2005). Agile practices in principle adhere to the values proposed in the agile manifesto (16).

To the best of our knowledge, there is no work that systematically summarizes, analyzes, and classifies all existing agile practices. In literature we found more than a hundred such practices. We collected the top 20 of these agile practices (listed in Table 1) according to the number of articles that discuss them (as shown in the "Sources" column).

This list of agile practices as well as the related literature presented above is not comprehensive; there are potentially many other papers that discuss similar or other agile practices. Furthermore, the aforementioned agile practices may be partially overlapping as listed in the parentheses of Table 1. For example, "Direct Interaction with Customer" in (Begel and Nagappan, 2007) is the same as "On-Site Customer" in (Silva et al., 2014). Considering this, we include "agile practice" as a data item to be extracted in selected studies (see Section 3.2.3) as shown in Table 5. Note that we only focus on the agile practices related to software architecture in this SMS.

3. Mapping study process

3.1. Research questions

The goal of this SMS, formulated using Goal-Question-Metric approach (Basili et al., 1994), is to **analyze** the combination of software architecture and agile methods **for the purpose of** exploration and analysis **with respect to** architecting activities and approaches, agile methods and practices, costs, benefits, challenges, factors, tools, and lessons learned **from the point of view of** researchers and practitioners **in the context of** software development.

We decomposed the goal into nine research questions (RQs) shown in Table 2. The answers of these RQs can be readily linked to the objective of this mapping study: an understanding of how architecting can be used in agile development (RQ1, RQ2), which agile methods and practices can be used with architecture (RQ3, RQ4), the costs, benefits, challenges, and the available tools of the architecture-agility combination (RQ5, RQ6, RQ8), the factors which may have an impact on the combination (RQ7), and the lessons learned from the combination (RQ9).