The Journal of Systems and Software 127 (2017) 78–90

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Designing and applying an approach to software architecting in agile projects in education

S. Angelov*, P. de Beer

Software Engineering, Fontys University of Applied Sciences, Postbus 347, 5600 AH Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Received 12 February 2016
Revised 13 January 2017
Accepted 31 January 2017
Available online 1 February 2017

Keywords:
Software architecture
Agile
Scrum
Teaching
Software engineering education
Students
Project
Course

ABSTRACT

Software architecting activities are not discussed in most agile software development methods. That is why, the combination of software architecting and agile methods has been in the focus of numerous publications. However, there is little literature on how to approach software architecting in agile projects in education. In this paper, we present our approach to the introduction of software architecting activities in an agile project course. The approach is based on literature sources and is tailored to fit our educational goals and context. The approach has been applied in two consecutive executions of the course. We observe improved understanding on the value of architecting activities and appreciation among students on the combination of architecting activities and agile development. We applied the approach predominantly in cases with an architecturally savvy Product Owner. Further research is required to understand how the approach performs in scenarios with architecturally unsavvy Product Owners and if it needs to be adapted for these scenarios. We also conclude that more research is needed on the challenges that architects face in agile projects in order to better prepare students for practice.

1. Introduction

Software architecting has received ample attention in academia and industry and many educational institutions address software architectures in their curricula, (e.g., Garlan et al., 1992; Lago and Van Vliet, 2005; Mannisto et al., 2008). However, teaching software architectures remains a difficult task. It requires a realistic context, teamwork, sufficient complexity, and substantial coaching (Galster and Angelov, 2016).

With the increasing popularity of agile software development methods, the software architecting community has faced a new challenge. Most agile methods "pay very little attention to common architectural design activities" (Babar, 2014). For example, Kent Beck sees architectures as emerging and evolving in the daily design (Beck, 1999). Kruchten et al. predict that "Software architecture will be recognized as a key foundation to agile software development" (Kruchten et al., 2006). The question of how much architectural effort is needed in agile projects was rated as "the second-equal most burning question facing agile practitioners" (Freudenberg and Sharp, 2010). Researchers and practitioners have embraced this challenge and published approaches, visions, experiences, surveys on how software architecting can be (or is) approached in agile development methods. A general consensus seems to exist around the value of paying explicit attention to software architectures in agile projects. However, there are different approaches to architecting in agile projects. Jan Bosch states that the community is switching to a "user-driven architecture work" from the traditional "quality based architecting" (Mirakhorli and Cleland-Huang, 2013). Following the agile principles, teams should focus upfront on the requirements delivering value to the client rather than on the architecturally significant requirements, accepting that an initial architecture may not be the optimal architecture. He sees the price of constant refactoring acceptable given the benefits achieved in terms of agility and customer satisfaction. Waterman et al. draw a similar conclusion in their study (Waterman et al., 2013). Friedrichsen analyses the idea of emergent architecture and concludes that it is a well-suited approach with respect to elaborating detailed architectures, but it is not adequately supporting critical architecting activities like making of architectural choices, elaboration of high-level designs, and architecture validation (Friedrichsen, 2014).

The coupling between agile methods and software architectures in education has not yet been sufficiently addressed in literature. Cleland-Huang et al. present an approach to treating software architectures in agile projects in education (Cleland-Huang et al., 2014a). The approach focuses on the architecture design phase in agile projects and specifically on the role of the stakeholders but does not address the actual dynamics of an agile project.

* Corresponding author.
E-mail addresses: s.angelov@fontys.nl (S. Angelov), p.debeer@fontys.nl (P. de Beer).

http://dx.doi.org/10.1016/j.jss.2017.01.029
0164-1212/

In our curriculum, we have a course in which groups of students develop a software system using an agile (Scrum-based) method. One of the skills that they need to demonstrate in this course is system design. In the past, we have required a software architecture design and its documentation but did not provide any guidance on how this had to be done in an agile project. The students approached this activity either as an "end-of-project" documentation activity (in most cases) or as a "Big Design Up Front" activity (less common). The former behavior was driven by the agile principle of working software over comprehensive documentation, while the latter behavior was influenced by previously acquired knowledge on software architecting in waterfall projects. Groups exhibiting the former behavior were unmotivated to deepen on the software architecture aspects, focusing primarily on the development process. The architecture was hastily and superficially discussed in the teams at the beginning of the projects and documented at the end of the projects to pass the course without realizing the purpose and benefits of architecting in an agile project. The lack of a clear approach to software architecting in agile projects resulted in invalid and unreliable grading of the software architecting skills of students.

To improve our course, we design and introduce an approach to software architecting. In this paper, we present the approach. An approach to software architecting in agile projects needs to resolve the traditional problems of teaching software architectures (e.g., realistic context, introducing teamwork, sufficient complexity, substantial coaching). In addition, it has to intertwine architecting and agile activities in a way that does not contradict the agile principles and practices. Therefore, we design our approach using publications on software architecting in agile development and on teaching software architectures, considering at the same time our educational context. We apply our approach in two consecutive course instances. This paper is a continuation of our work from Angelov and de Beer (2015). In this paper, we extend our findings by providing results from the application of the approach in two consecutive course executions and improvements to the approach based on our findings. Furthermore, we discuss the challenges that require further research by the software architecting research and education communities.

The paper is structured as follows. In Section 2, we present our context, problem, and research approach. In Section 3, we discuss our literature findings on the interplay between software architectures and agile methods. In Section 4, we present our approach. In Section 5, we discuss the application of the approach and lessons learned. We end the paper with conclusions.

2. Context, problem, and research approach

In this section, we describe our context and define our problem. We discuss literature on teaching software architectures and show that it does not provide a solution to our problem. Then, we present our research approach.

2.1. Course context

We analyze our context using the *situational factors* defined by Fink: general and specific context, nature of the subject, characteristics of the learners (Fink, 2013):

- *General context*: Fontys University of Applied Sciences is the largest Dutch university of applied sciences. It offers bachelor and master programs. Being an applied university, its focus is on teaching practice oriented knowledge and skills. In particular, it offers a software engineering, bachelor program. Upon completion of their studies, students from this program are skilled software engineers. The four-year program consists of 4 semesters of courses, followed by an internship semester, another courses-based semester, a free-choice minor study, and a final internship (graduation project). Within each courses-based semester, there is a "project-course" called PTSn (where "n" indicates the number of the semester: PTS2 – second semester, PTS3 – third semester, etc.). Project-courses apply knowledge from other courses for the development of a software system. The students are introduced to software architectures in their third semester in a course called GSO3. In PTS3, the students develop a software system using a waterfall development method. They perform requirements elicitation and documentation, architecture elicitation and documentation, implementation, and testing activities. The students elaborate an architecture document consisting of functional and non-functional requirements, use case, class, sequence, component, and deployment diagrams.
- *Specific context*: The course with acronym PTS4 is the focus of this work. PTS4 is 6 ECTS credits (European Credit Transfer and Accumulation System). The students work in groups of 5–6 students for 20 weeks, one day a week. The students are allowed to form the groups themselves, which typically leads to balanced groups in terms of skills, interests, and motivation. Our curriculum implies the application of architecting activities in PTS4.
- *Nature of the subject*: PTS4 focuses on applying agile practices in a Java-based project. It follows mainly the Scrum method, augmented with practices from other agile methods like Lean (eliminate waste) and Crystal Clear (osmotic communication, personal safety). All top 20 agile practices listed in Yang et al. (2016) are followed except for the "system metaphor". The family of agile methods are thoroughly studied in another course (BS41) given in parallel to PTS4. On a weekly basis, the students are visited by their Product Owner (PO) and Tutor (typically both roles are performed by one teacher). Scrum Masters are students from the teams. A commercial tool for agile project management is used by all groups. The project is divided into 4 iterations (sprints) of 4 weeks.
- *Characteristics of the learners*:
  - *Capabilities*: Students of applied universities are predominantly interested in applying knowledge and skills. They are typically less inclined on reflecting on theoretical aspects of a problem. However, there are also students with capabilities for deeper reflections.
  - *Motivation*: A student needs to be motivated during the course for achieving best learning results. The motivation of our students is influenced by:
    - realism: overlap between knowledge taught and activities performed in courses and real-life practices;
    - *purposefulness*: having a clear purpose for the knowledge taught and activities performed in courses;
    - *clarity*: unambiguous, easy to understand knowledge taught and activities performed in courses;
    - *degree of challenge*: knowledge and activities that are not too easy or too difficult.
  - *Maturity*: Typically students are about 20 years old, in the early stages of their professional development. They are relatively inexperienced in design and architecting activities.

2.2. Problem definition and desired situation

In the old setup of PTS4, we used a case called "PhotoStore". The students need to develop an application for ordering of customized photos that can be optionally printed on a product (e.g., on a T-shirt). The case is realistic, technically challenging, and having many features to build. The case lacks novelty and is therefore of a somewhat trivial architecture nature: "most projects are