Information and Software Technology 000 (2016) 1–5

Contents lists available at ScienceDirect

Information and Software Technology

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

Emerging themes in agile software development: Introduction to the special section on continuous value delivery

Torgeir Dingsøyr[a,b,*], Casper Lassenius[c]

[a] SINTEF, Trondheim, Norway
[b] Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway
[c] Department of Computer Science and Engineering, Aalto University, Helsinki, Finland

ARTICLE INFO

Article history:
Received 19 April 2016
Revised 26 April 2016
Accepted 27 April 2016
Available online xxx

Keywords:
Agile software development
Software process improvement
Value-based software engineering
Requirements engineering
Continuous deployment
Lean startup
Scrum
Extreme programming

ABSTRACT

The relationship between customers and suppliers remains a challenge in agile software development. Two trends seek to improve this relationship, the increased focus on value and the move towards continuous deployment. In this special section on continuous value delivery, we describe these emerging research themes and show the increasing interest in these topics over time. Further, we discuss implications for future research.

1. Introduction

Since the inception of agile development methods in the late 1990s, there have been a stream of topics of interest amongst practitioners and the research community. Early research on agile development focused on extreme programming practices such as test-first development [1,2] and pair programming [3,4], on whole methods such as extreme programming [5], Scrum and Lean software development. We have seen an increase in study quality after a number of special issues and special sections on agile development, a larger number of studies published in journals, and a larger amount of studies connecting empirical findings to theories that are taken from more mature research fields [6].

In this special section, we focus in particular on two recent trends in research on agile software development: First, the transition from a focus on agile methods on team level with emphasis on team performance (illustrated by the focus on pair programming and test first development), to a broader organizational understanding where more focus is put on value of the developed product. Second, the transition from iterative development with initial recommendations on 30 day iterations in Scrum to continuous deployment of new features. We describe these two trends as a focus on *continuous value delivery*. This is a challenging topic. In one of the few reliable scientific surveys we have on usage of agile methods [7], many respondents indicate that customer/supplier relationships is a one of the main challenges, yet many see improved customer understanding as an effect of adopting agile development methods. Furthermore, many report using iterations and practices such as continuous integration, which is a prerequisite for continuous delivery. The top reasons for adopting agile methods are to increase productivity, increase product and service quality and to reduce development cycle times and time-to-market.

But is there anything new in the search for continuous value delivery? In Beck's first book on extreme programming [8], he states that we "need to make our software economically more valuable by spending money more slowly, earning revenue more quickly and increasing the probably productive lifespan of our project" (page 11), and the practice of continuous integration was suggested already then. Also, some have claimed that even the practices in extreme programming is "old wine in new bottles" and have been established practices for a long time [9]. We argue that the ideas of continuous value delivery are old, but that the possibilities have increased with maturing technology. Further, as we will see, the ideas have developed since the initiation of agile methods.

* Corresponding author at: SINTEF, Trondheim, Norway.
E-mail address: torgeird@sintef.no, xp2015specialissue@gmail.com (T. Dingsøyr).

http://dx.doi.org/10.1016/j.infsof.2016.04.018

Please cite this article as: T. Dingsøyr, C. Lassenius, Emerging themes in agile software development: Introduction to the special section on continuous value delivery, Information and Software Technology (2016), http://dx.doi.org/10.1016/j.infsof.2016.04.018

In the following, we introduce three articles, which have been extended and revised for this special section. The articles are chosen from the XP2015 conference [10]. Finally, we highlight what we see as main implications for research on agile software development given these trends.

2. What is value?

Many of the recent improvement trends that have influenced software development practice have a focus on business value. The agile manifesto focuses on customer collaboration and working software, and a principle behind the manifesto is to satisfy the customer through early delivery. Lean production puts emphasis on value through reducing costs [11], through eliminating "waste", where waste can be waiting time or large inventories (see [12] for a complete list). Proponents of lean production claim that waste can be reduced by applying techniques such as value stream mapping or just in time production. The recent trend of lean start-ups [13] takes a similar position on value, making the argument that waste can be reduced through early learning about customer value.

The improvement trends are not very specific on how they define value. An obvious reason is that different environments might have very different interpretations of what gives business value to them. The general use of the word value ranges from "usefulness or importance" and "relative worth, utility, or importance" to "the monetary worth of something" [14]. When value is determined by usefulness or even monetary worth, at least it suggests that value of software is assigned by stakeholders outside of the development team. Proponents of agile development and lean startup would argue that a development team needs to learn what external stakeholders value during a development project, while traditional approaches would argue for understanding the view of value up-front.

Such an up-front understanding is eminent in traditional project management. The most popular frameworks for project management, the project management body of knowledge [15] and the PRINCE2 framework [16] both focus on the business value of projects. The project management body of knowledge defines business value as both tangible and intangible elements. Tangible elements include equipment and monetary assets while intangible elements include "good will", brand recognition or public benefits. The central idea in PRINCE2 is to achieve benefits with projects, and the benefits are defined prior to project initiation in a "business case". The business case is under continuous justification and lists the benefits that are to be achieved.

Also in software engineering, there has been a history of discussing value. Boehm introduced the term "value-based software engineering" in 2003 [17,18], arguing that many practices in the field are done in a "value-neutral" setting where requirements are treated as equally important and that accounts of "earned value" in development projects are focusing on costs and schedule and not business value. Boehm suggested to integrate value considerations into principles and practices, suggesting research on a number of topics including value-based requirements engineering, value-based planning and control. In his article [17], he discusses how software development can be made more value-based, for example through conducting more thorough analysis of the benefits to be achieved by new software, elicitation of value propositions that stakeholders hold, and conducting business case analyses on software projects.

We argue that these ideas now have been taken up more broadly through the trends of agile software development and lean software development with an even sharper focus on value.

Predicting the value of software is probably at least as challenging as predicting the cost of software [19]. Based on experience from a large development project in Norway, the company Promis has suggested to estimate value in the form of "benefit points" [20]. The idea is to get a similar estimate of value to an epic (set of user stories), as agile development teams often make an estimate of the development cost in "story points". The "benefit points" are also relative to an epic with "known" value to the customer organization, and then these figures can be helpful in deciding about priority in a product backlog. The method involves translating overall goals of a project or program into how much can be achieved through implementation of an epic.

To summarize, we see an increased focus on value in improvement trends relevant for software development. This focus has led to suggestions on how to operationalize calculations on business value such as from Promis, and also on techniques to advance understanding of customer needs. A particularly interesting area of research is using agile techniques in achieving early feedback from users and customers. The article in this special section on agile requirements engineering and use of test cases as requirements ("*Multi-Case Study of Agile Requirements Engineering and the Use of Test Cases as Requirements*" by Bjarnason *et al.*) draws on a rich empirical material to show a variety of practices, and discuss benefits and challenges when using test cases to elicit, validate, verify and manage requirements.

3. Continuous deployment and continuous experimentation

As the theoretical approaches to model and assess value up-front have proven to be challenging, there is a current trend towards using empirical means to understand value. Empirically understanding customer value relies on the idea *of continuous experimentation*, an approach in which potentially valuable features are delivered to customers, and data is collected to understand the value of the delivered functionality. In this emerging approach, different versions of the software might be delivered to different user groups, making it possible to understand experienced customer value and how different feature sets or implementations affect product usage. While relying on other practices, including *continuous integration* and *continuous deployment*, continuous experimentation also requires additional infrastructure to support experiment planning execution and analysis [21].

At this moment, research on continuous experimentation is starting to appear, but as more and more companies move towards continuous value delivery, its practical importance is likely to be very significant, and companies' ability to quickly use data about customer behavior in innovative ways likely to be a major contributor to their competitiveness. As the academic research on continuous experimentation is in its early stages, there is much opportunity for ambitious research on the topic in the near future.

Continuous integration (CI) is a software development practice where software is integrated continuously during development [22]. CI requires at least daily integration and that each integration is verified by automated build and tests. As a basic building block of a working agile implementation, there exists a growing set of case studies, and experience reports on CI discussing both challenges and benefits related to the practice, see e.g. [23,24]. And while there is a lack of synthesizing research, it seems basic issues like what the characteristics of a CI process should be still needs clarification. E.g., Ståhl and Bosch [24] studied CI in industry and found that the practices were not really continuous: "activities are carried out much more infrequently than some observers might consider to qualify as being continuous".

Building upon continuous integration, continuous delivery aims at constantly keeping the software in a releasable state [25,26]. This is achieved through optimization, automatization and utilization of the build, deploy, test and release process [26]. The proposed benefits of continuous delivery include increased visibility, faster feedback and empowerment of stakeholders [26]. However,

Please cite this article as: T. Dingsøyr, C. Lassenius, Emerging themes in agile software development: Introduction to the special section on continuous value delivery, Information and Software Technology (2016), http://dx.doi.org/10.1016/j.infsof.2016.04.018