Information and Software Technology 77 (2016) 56–60

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 10 May 2016

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

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,

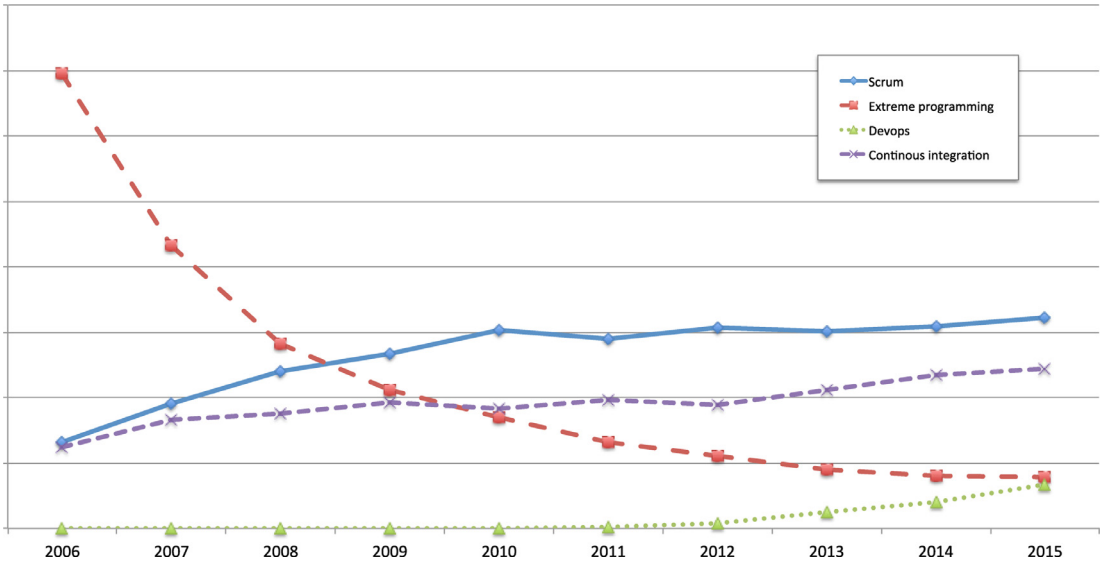

Fig. 1. Relative interest over time on themes "Scrum", "extreme programming", "DevOps" and "continuous integration" based on searches in Google Trends, showing results for category "computers and electronics/programming".

when trying to adopt continuous delivery, organizations have faced numerous challenges [27,28].

Continuous deployment takes the final step in automation, and that each change is built, tested and deployed to production automatically. Thus, in contrast to continuous delivery, there are no manual steps or decisions between a commit by a developer and production deployment. The motivation for automating the deployment to production is to gain faster feedback from production use to fix defects that would be otherwise too expensive to detect. Research on continuous deployment is still in its infancy, despite the industrial relevance of the topic [29].

Interestingly, but not surprisingly, the topics of continuous experimentation and continuous deployment seem similar to other agile topics in the sense that they are industry rather than research driven. The state-of-the art is driven by industry and consultants, and research is lagging behind in synthesizing and systematizing knowledge and helping to validate or dismiss the many claims made by proponents for various tools and techniques. However, as the article on the current state of experimentation in product development in this special section ("*Raising the Odds of Success: The Current State of Experimentation in Product Development*" by Lindgren and Münch) shows, there is a considerable potential in exploiting these ideas in many companies, and in particular there are challenges with changing the organizational culture, accelerating the development cycle speed and also in identifying measures for customer value.

4. Implications for future research

We argued for an increasing interest in continuous value delivery as a research topic. This trend has been described by leading scholars in the software engineering field such as Fitzgerald and Stol [12] focusing on the trend towards continuous development and Bosch [30] focusing on the importance of learning. But is the trend shown in practitioner or researcher interest so far?

In Figs. 1 and 2 we show development practice trends the last ten years. Fig. 1 is based on Internet searches[1] and indicates relative interest amongst developers on topics. Fig. 2 shows relative interest amongst researchers.[2] We have plotted interest in two established topics in agile software development, namely the methods extreme programming (XP) and Scrum. We see that Scrum has received by far the most interest, and the interest is increasing over time (the drop in 2015 amongst researchers is probably due to late indexing of articles in the database). The high interest in Scrum amongst researchers might be due to the general popularity of Scrum as a development method. It could also be that Scrum is described as the context of studies, not necessarily that there is such a high interest in studying Scrum itself. Furthermore, we have plotted the interest in emerging topics, which we argue is under the umbrella continuous value delivery, namely DevOps and the practice of "continuous integration".

For practitioners, we see that there is a decline in the interest in extreme programming, while the interest in Scrum is increasing over time. There is an increase in interest on continuous integration from 2006 to 2015, and a sharp increase on DevOps in the last years. For researchers, we see a sharp decline in interest on extreme programming, a steady increase in interest on Scrum and continuous integration and a more sharp increase in interest on continuous integration. A striking difference between practitioners and researcher is the relative higher interest in Scrum amongst researchers. Another difference is the high interest in continuous integration amongst practitioners, while this topic is more or less on the same level as DevOps and extreme programming amongst researchers.

The final article in this special section focuses on giving voice to practitioners in discussions on future research. The article "*The challenges that challenge: Engaging with agile practitioners' concerns*" by Gregory et al. provides a thorough review of existing discussions on research directions, and draws on a rich material from practitioners in order to discuss future research directions. They identify the trends we have discussed in this introduction, such as the increasing focus on organisations, including a tighter collaboration between business and technical staff, as well as a general focus on demonstrating product value from agile methods. They also identify a number of other research areas, such as the emerging focus not on agile adoption as most organizations at least claim to do agile development, but on sustaining agility in projects and organisations.

[1] Searches in Google analysed by Google Trends.
[2] Measured by the number of articles on topics in the Scopus database.

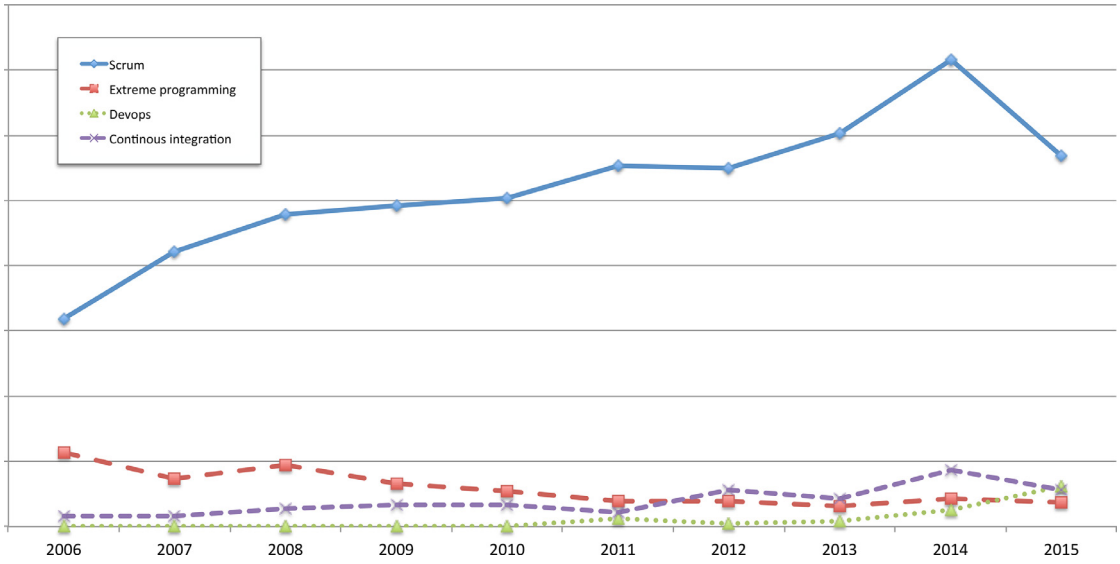

Fig. 2. Relative interest over time on themes "Scrum", "extreme programming", "DevOps" and "continuous integration" based on searches for publications in the Scopus scientific database.

Acknowledgments

We would like to thank the program committee members for the XP2015 conference for their effort in reviewing conference papers, nominating papers and finally for reviewing selected articles for this special section. We would also like to thank Claes Wohlin for guidance regarding the special section. The work with this section has partially been supported by the project Agile 2.0 funded by the Research council of Norway through grant 236759 and by the companies Kantega, Kongsberg Defence & Aerospace, Sopra Steria, and Sticos, and by TEKES, the Finnish Funding Agency for Innovation, as part of the Need for Speed (N4S) programme.

References

[1] H. Erdogmus, M. Morisio, M. Torchiano, On the effectiveness of the test-first approach to programming, IEEE Trans. Softw. Eng. 31 (2005) 226–237.
[2] D. Janzen, H. Saiedian, Test-driven development: concepts, taxonomy, and future direction, Computer 38 (Sep 2005) pp. 43–+.
[3] V. Balijepally, R. Mahapatra, S. Nerur, K.H. Price, Are two heads better than one for software development? The productivity paradox of pair programming, MIS Q. 33 (Mar 2009) 91–118.
[4] J.E. Hannay, T. Dybå, E. Arisholm, D.I.K. Sjøberg, The effectiveness of pair programming: a meta-analysis, Inf. Softw. Technol. 51 (Jul 2009) 1110–1122.
[5] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (2008) 833–859.
[6] T. Dingsøyr, S. Nerur, V. Balijepally, N.B. Moe, A decade of agile methodologies: towards explaining agile software development, J. Syst. Softw. 85 (2012) 1213–1221.
[7] P. Rodriguez, J. Markkula, M. Oivo, K. Turula, Survey on agile and lean usage in finnish software industry, in: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Lund, Sweden, 2012.
[8] K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, 2000.
[9] M.-R. Hilkka, T. Tuure, M. Rossi, Is extreme programming just old wine in new bottles: a comparison of two cases, J. Database Manag. 16 (2005) 41.
[10] C. Lassenius, T. Dingsøyr, M. Paasivaara, in: Agile Processes, in Software Engineering, and Extreme Programming: 16th International Conference, XP 2015, Proceedings vol. 212, Helsinki, Finland, May 25-29, 2015, Springer, 2015.
[11] K. Conboy, Agility from first principles: reconstructing the concept of agility in information systems development, Inf Syst Res 20 (2009) 329–354.
[12] B. Fitzgerald, K.-J. Stol, Continuous software engineering: a roadmap and agenda, J Syst Softw (2015), doi:10.1016/j.jss.2015.06.063.
[13] E. Ries, The lean startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Books, 2011.
[14] Webster's, Encyclopedic Unabridged Dictionary of the English Language, Gramercy Books, New York, 1989.
[15] PMI, A Guide to the Project Management Body of Knowledge, 5th edition, Project Management Institue, 2013.
[16] C. Bentley, Prince2: A Practical Handbook, Routledge, 2010.
[17] B. Boehm, Value-based software engineering: reinventing, ACM SIGSOFT Softw. Eng. Notes 28 (2003) 3.
[18] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, P. Grünbacher, Value-Based Software Engineering, Springer Science & Business Media, 2006.
[19] M. Shepperd, Cost prediction and software project management, in: Software Project Management in a Changing World, Springer, 2014, pp. 51–71.
[20] K. Strand, K. Karlsen, Agile Contracting and Execution, PROMIS, 2014.
[21] F. Fagerholm, A.S. Guinea, H. Mäenpää, J. Münch, Building blocks for continuous experimentation, in: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, Hyderabad, India, ACM, 2014, pp. 26–35.
[22] Fowler, M. (2006). Continuous Integration.
[23] Eck, A., Uebernickel, F., and Brenner, W., "Fit for continuous integration: how organizations assimilate an agile practice," 2014.
[24] D. Ståhl, J. Bosch, Automated software integration flows in industry: a multiple-case study, in: Companion Proceedings of the 36th International Conference on Software Engineering, New York, NY, USA, 2014, pp. 54–63.
[25] Fowler, M. (2013). Continuous Delivery.
[26] J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, Addison-Wesley Professional, 2010.
[27] G.G. Claps, R.B. Svensson, A. Aurum, On the journey to continuous deployment: technical and social challenges along the way, Inf. Softw. Technol. 57 (2015) 21–31.
[28] S. Neely, S. Stolt, Continuous delivery? Easy! just change everything (Well, maybe it is not that easy), in: Agile Conference (AGILE), 2013, 2013, pp. 121–128.
[29] P. Rodríguez, et al., Continuous deployment of software intensive products and services: a systematic mapping study, J. Syst. Softw. (2016).
[30] J. Bosch, Speed, data, and ecosystems: the future of software engineering, IEEE Softw. 33 (2016) 82–88.

Torgeir Dingsøyr focuses on software process improvement and knowledge management as chief scientist at the SINTEF research foundation. In particular, he has studied agile software development through a number of case studies, co-authored the systematic review of empirical studies, co-edited the book *Agile Software Development: Current Research and Future Directions*, and co-edited the special issue on Agile Methods in the Journal of Systems and Software. He wrote his doctoral thesis on *Knowledge Management in Medium-Sized Software Consulting Companies* at the Department of Computer and Information Science, Norwegian University of Science and Technology, where he is an adjunct professor.

Casper Lassenius is an associate professor of software engineering at Aalto University. His research interests include agile and lean software development in small and large contexts, global software engineering, software measurement, and industrial software testing. He has PhD and M.Sc degrees from Helsinki University of Technology.