”Can we do useful industrial software engineering research in the shadow of Lean and Agile?”

Kristian Wiklund, Sigrid Eldh, Daniel Sundmark, Kristina Lundqvist.

Preprint - Accepted to CESI 2013, ICSE, San Francisco, California

Can we do useful industrial software engineering research in the shadow of Lean and Agile?

Kristian Wiklund, Sigrid Eldh
Ericsson AB
SE-164 80 KISTA
Stockholm, Sweden
{kristian.wiklund,sigrid.eldh}@ericsson.com

Daniel Sundmark, Kristina Lundqvist
School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
{daniel.sundmark,kristina.lundqvist}@mdh.se

***Abstract*—The software industry is rapidly changing from traditional ways of working to lean and agile development methods using self-organized feature development teams that are performing a much larger part of the development process than before. Face to face communication will replace many of the design artifacts used for work-in-progress, such as defect reports and feature system design specifications. This type of data will cease to exist when the feature is developed or the problem is solved, and will not be readily available to researchers. As a consequence, software engineering research in industry will have to rely primarily on participatory and observational methods.**

***Index Terms*—empirical research;agile;lean**

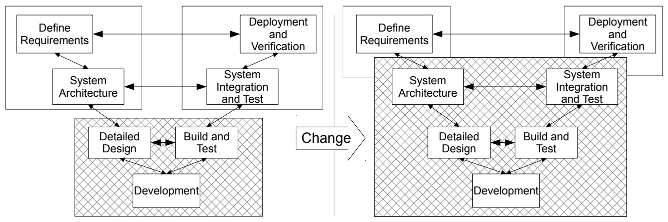

Fig. 1. Changing to feature-oriented end-to-end development

I. Introduction

During the recent decade, a large number of software companies have changed to agile development methods [1][2]. Transforming to agile methods for software development is usually considered by software developers to be a a great improvement over ”traditional” software development methods. Laanti *et al.* [2] report from a survey at Nokia that agile methods are perceived by the respondents as making work more fun, development more effective, and product quality higher. Less than 10% of the respondents wanted to return to their old ways of working, something that is in line with informal survey results reported to us. Petersen and Wohlin found in their case study [3] that learning and understanding was improved by an increase in face to face communication, and Parnell-Klabo [4] reports lead-time improvements in the order of 40% compared to the ”waterfall baseline”. In short, it is likely that agile and lean development methods are here to stay.

How will this change the field of empirical research in industry? As empirical researchers in the industry we are spoiled with access to a vast amount of information. There are databases filled to the brink with valuable information, such as project plans and their fulfilment, problem reports with resolutions and root causes, build logs, and change information showing changes to both source code and documents.

Lean thinking is in large parts about optimizing flow and reducing or removing queues [5], and according to the Scrum Alliance, ”the most efficient and effective method of conveying information to and within a development team is face-to-face conversation” [6].

One way of implementing lean and agile software development is by using cross-discipline, colocated, feature-oriented, development teams [7][4]. The team members work physically close to each other, and the team is responsible for the end-to-end development of a feature that brings value to the customer. End-to-end development is about doing as much of the work with the feature as possible within the team. This gives an opportunity to significantly reduce, and in many cases totally eliminate, the queues [7] in the development process, and to replace parts of the documentation with direct communication [3].

When a larger part of the work flow is handled internally in a single team, the need to communicate across team boundaries will be reduced. Information and actions that were previously managed across a larger organization will be completely enclosed in the team scope as shown in Figure 1. This means that there is an opportunity for decentralization and simplification [8]. The handling of information such as problem reports and process metrics is likely to take place on the team task boards, and cease to exist after sprint completion.

II. Changes to Information Availability

Encapsulating the information in the team and removing the queues is likely to eliminate the need for the information carrier used to hold the queue. Based on our observations, a software development team do not want to produce written information only to produce a trace of how someting was done or to enable metrics. The primary measurement of success is working software and they want to reach that point quickly. Why spend more time reporting a problem than on the fix itself, and why write a delta specification when you can can do the design on a whiteboard, then change the design and the documentation directly?

These changes are likely great time-savers for the team [4], and the impact on quality is likely positive. Misinterpretation of requirements and functionality have been reported to be a large source for errors [9], and the risk for miscommunication is reduced if the developers interact directly instead of via documents. Turning to project management, planning is done with sticky notes on a sprint-to-sprint basis, and teams typically track their progress as a "burn-down chart" [10] on their task board. This information vanishes at the end of a sprint, what remains is usually an adjustment to the estimated work capacity in the next sprint.

III. Impact on Research

Lean and agile methods, in particular if the development teams are encapsulating a large part of the process, have the potential to decentralize information and to reduce the time information is available. Self-organization is an important part of agile methods [11], and depending on the freedom given by line management, this could result in eliminating a large part of the previously observable design artifacts and meta-data.

This could have severe impact on the availability of data for industrial software engineering research. We will no longer have the comfort of very large databases of information reaching years and years back through history. This not only makes it harder to research new methods and improvements, but also prevents us from evaluating the results of the improvements that caused the loss of data, for example, how do we know if we work more efficient in the design phase if we have nothing to compare to the baseline data? An even worse situation may occur if there indeed is "[a] lack of research into cause and effect" in some research areas as reported by Höfer and Tichy [12], and we as researchers do not realize that the change in availability has occured and continue to use the data in the same ways as before.

We can still do end-to-end measurements, from sales to customer, but the lack of detailed observability in that type of measurements and the lead time to obtain them in large-scale industrial development could make the information useless for research and process improvement.

Line managers and project managers in organizations changing to agile have to change and operate closer to the engineers. With decentralized information and daily team stand-ups, participatory management [13] is needed. The same will be required from software engineering researchers.

Data analysis in its own right can likely not be a major source of knowledge from industrial research in the future and obtaining this type of supporting information for other types of studies will be harder.

Instead, a much more participatory approach is needed. Seeking information through direct methods [14], such as interviews, participatory observation and walking around to collect data observed over time on task boards will be necessary. We also expect that the lean and agile revolution will bring new opportunities for research as well, the built-in willingness to change and the possibility for rapid feedback through observation makes the setting perfect for experimentation and action research.

References

[1] *Need for speed: More IT companies switch to Agile code development*. 2012. URL: http://articles.economictimes.indiatimes.com/2012-08-06/news/33065621%5C_1%5C_thoughtworks-software-development-iterative.

[2] Maarit Laanti, Outi Salo, and Pekka Abrahamsson. "Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation". In: *Information and Software Technology* 53.3 (Mar. 2011), pp. 276–290.

[3] Kai Petersen and Claes Wohlin. "A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case". In: *Journal of Systems and Software* 82.9 (Sept. 2009), pp. 1479–1490.

[4] Emma Parnell-Klabo. "Introducing Lean Principles with Agile Practices at a Fortune 500 Company". In: *AGILE 2006 (AGILE'06)*. IEEE, 2006, pp. 232–242.

[5] James P Womack and Daniel T Jones. *Lean Thinking*. Free Press, 2003.

[6] *Scrum Alliance Code of Ethics*. 2013. URL: http://www.scrumalliance.org/pages/code%5C_of%5C_ethics.

[7] Craig Larman and Bas Vodde. *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Addison-Wesley, 2009, p. 154.

[8] Lisa Crispin. *Limbo Lower Now : An Agile Approach to Defect Management*. 2010. URL: http://lisacrispin.com/presentations/.

[9] Victor R. Basili and Barry T. Perricone. "Software errors and complexity: an empirical investigation0". In: *Communications of the ACM* 27.1 (Jan. 1984), pp. 42–52.

[10] Henrik Kniberg. *Scrum and XP from the Trenches*. InfoQ Enterprise Software Development Series, 2007.

[11] Mikael Lindvall et al. "Empirical Findings in Agile Methods An Experience Base for Software Engineering". In: *Extreme Programming and Agile Methods—XP/Agile Universe*. 2002, pp. 197–207.

[12] Andreas Höfer and Walter F. Tichy. "Status of empirical research in software engineering". In: *Int. Conf. Empirical software engineering issues: critical assessment and future directions*. June 2006, pp. 10–19.

[13] Jason Yip. *Lean Software Development: Seeing the IT and Software Development Gemba*. 2011. URL: http://www.shmula.com/lean-software-development-gemba/8837/.

[14] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14.2 (Dec. 2008), pp. 131–164.