A Case Study on Benefits and Side-Effects of Agile Practices in Large-Scale Requirements Engineering

Bjarnason, Elizabeth; Wnuk, Krzysztof; Regnell, Björn

Published in:
[Host publication title missing]

2011

A Case Study on Benefits and Side-Effects of Agile Practices in Large-Scale Requirements Engineering

Elizabeth Bjarnason, Krzysztof Wnuk, Björn Regnell
Department of Computer Science, Lund University,
Lund, Sweden
{elizabeth.bjarnason | krzysztof.wnuk | bjorn.regnell}@cs.lth.se

ABSTRACT

In the software industry, there is a strong shift from traditional phase-based development towards agile methods and practices. This paper reports on a case study aimed at investigating if, and how, agile Requirements Engineering (RE) can remedy the challenges of traditional RE, and what new challenges agile RE may pose. The results from an initial case study with 9 practitioners from a large software development company, which is transitioning towards agile-inspired processes, show that agile practices address some RE challenges such as communication gaps and overscoping, but also cause new challenges, such as striking a good balance between agility and stability, and ensuring sufficient competence in cross-functional development teams.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specification – *methodologies (e.g., object-oriented, structured)*

General Terms

Management, Documentation, Human Factors.

Keywords

Requirements engineering, Agile, Empirical study, Case study.

1. INTRODUCTION

Requirements Engineering (RE) for agile software development (e.g. eXtreme Programming [1], Scrum [14]) is different from traditional Requirements Engineering. Traditionally, requirements are managed by RE specialists, in a phase separated in time from design and development, and documented in specific requirements artefacts. In contrast, in agile Requirements Engineering the detailed requirements are defined gradually in interaction between the customer (or customer representative) and the development team. Agile RE is often less formal and therefore not always explicitly denoted 'RE', and the requirements are not always documented. The RE for agile software development versus traditional (phase-based) software development has been compared (e.g. [7].) There are also a number of experience reports on the differences (e.g. [5], [13]) and the challenges of transitioning to agile methods (e.g. [13].) Based on a large empirical study of companies that use agile methods [10], six agile RE practices have been identified including their benefits and challenges.

In order to further increase the understanding of agile practices, we empirically investigated the following research questions: (RQ1) how do agile RE practices address the challenges of traditional RE, and (RQ2) what new challenges do agile RE practices incur. We have performed a case study at a large-scale software development company that is transferring from a traditional to an agile RE process. The study investigates challenges of traditional RE, their causes and consequences, and how the newly introduced agile RE practices affect this situation. Results for two of the challenges (for traditional RE) have been reported, namely *Overscoping* [2] and *Communication gaps* [3]. The other RE challenges (covered by the full study) are *Keeping the SRS updated*, *Development Work Monitored from Requirements* and *Manual Selection of Requirements for Products*. In this paper, we report on the findings around the impact of the agile RE practices.

The remainder of this paper is structured as follows: Section 2 provides background information about the context of our industrial case study. Section 3 describes the methodology used in this study. Section 4 contains the results from the interviews, while we in Section 5 interpret and conclude the results of the study, and describe future research.

2. THE CASE COMPANY

Our results are based on empirical data from industrial projects at a large company that is using a product line approach [9]. The company has around 5000 employees and develops embedded systems for a global market. A typical project has a lead time of up to 2 years and develops around 60-80 new features, corresponding to approximately 700-1000 detailed requirements. Each feature is developed by a cross-functional team including around 2-10 developers. In combination with the legacy functionality, which amounts to a very complex and large set of requirements at various abstraction levels in the order of magnitude of 20,000 entities, it is an example of the *Very-Large Scale Requirements Engineering* context [11].

To meet the challenges of high requirements volatility in very-large scale software development, the case company is introducing a new development process that is partly influenced by the agile method Scrum [14]. The responsibility for

Agile RE'11, July 26, 2011, Lancaster, UK.

requirements management has been transferred from the (previous) requirements unit, partly into the business unit and partly into the software development unit. The old stage-gate model with several increments is replaced by a continuous development model with a toll-gate structure for the software releases of the software product line (to allow coordination with hardware and product projects.) Five RE-related agile practices are being introduced at the company, namely:

- *One Continuous Scope Flow*. The scope for all software releases is continuously planned and managed via one priority-based list (comparable to a product backlog.) The business unit gathers and prioritizes features from a business perspective. The software unit estimates the cost and potential delivery date for each feature, based on priority and available software resource capacity.
- *Cross-Functional Development Teams* that include a customer representative assigned by the business unit (comparable to customer proxy.) These teams have the full responsibility for defining the detailed requirements, implementing and testing a feature (from the common priority-based list) within the given boundaries of time and resources.
- *Integrated Requirements Engineering*. The requirements engineering tasks are integrated with the other development activities, i.e. the detailing and formal documentation of requirements is done at the same time as design and development of the feature and within the same (development) team together with its customer representative (proxy).
- *Gradual & Iterative Detailing of Requirements*. The requirements are first defined at the high level (features in the priority-based list) and then iteratively refined, by the development team, into more detailed requirements as the design and implementation work progresses.
- *User Stories & Acceptance Criteria* [6] are used to formally document the requirements agreed for development. The acceptance criteria are then covered by test cases.

At the time of the study all of these methods were defined in the company's internal development process. The methods *One Continuous Scope Flow* and *Cross-Functional Development Teams* were fully implemented and applied in the projects, while the *Gradual & Iterative Detailing of Requirements* was partly implemented and the usage of *User Stories & Acceptance Criteria* and *Integrated Requirements Engineering* were in the process of being implemented (training and tool support was being planned.)

The size and complexity of the software development remains the same as with the previous process. The impact on project lead times is to be evaluated.

3. RESEARCH METHODOLOGY

The research was conducted using a qualitative research approach [8] that aims at understanding complex phenomena in the context where they exist. Semi-structured interviews with a high degree of discussion between the interviewer and the interviewee [12] were performed in order to study the impact of the agile practices. The experience of one of the authors (who has worked at the case company) has served as input in shaping the interview instrument (available on-line [4].) It covers (1) a number of challenges of traditional RE, and their potential causes and consequences, and (2) a number of agile RE practices, their impact on the previous RE challenges and which new challenges they may pose. The data from the transcripts was analyzed by using content analysis [8] based on the interview instrument [4]. The transcribed chunks of text were placed within the relevant sections (corresponding to challenges and agile practices.) These were numbered and relationships were captured by noting dependencies to and from each category in specific columns.

Nine practitioners who have worked for the case company for 5-8 years were interviewed. Their roles (in the old process) cover the full project life cycle from requirements definition through development to the end product. They currently work with early software technology studies (2 persons), software project management (2 persons), software quality management (3 persons), software process management (1 person), and software development (1 person.)

Limitations. The set of agile practices covered by the study is limited to the practices adopted by the case company and the set of challenges is limited to the challenges experienced at the case company. Further, the results may not be valid for cases with characteristics too dissimilar from the case in this study.

4. RESULTS

This section reports on the responders' view of the agile RE practices; which RE challenges they address, and which (new) challenges they incur. Table 1 contains an overview of the results. Our interpretation and conclusions can be found in Section 5.

4.1 One Continuous Scope Flow

The challenge of *overscoping* (aka over allocation) is seen to be addressed by this practice (which was implemented at the time of our study.) The practice unites the inflow of requirements to the development unit into one list of features that is continuously re-prioritized and agreed with development according to the amount of available capacity. In combination with a short preparation period for each feature (where it is prepared for development) this practice is experienced to reduce the amount of *wasted effort* and subsequent loss of *staff motivation* when features are dropped and removed from the project scope. It was also mentioned that this practice has improved the *communication* around planning, resulting in more efficient and co-ordinated planning between the business unit and the development units.

In general, the view is that there is (still) a tendency towards overscoping; the business unit requests unrealistically large amounts of features, which are *weakly prioritized* (most features are critical) and the development unit commits *without planning for agility*, i.e. for handling changes later on. In addition, the development teams *underestimate the effort* required for development. However, the practice has brought transparency and visibility to the scoping process. As stated by one interviewee, 'We still have overscoping in all projects. But, it is more controlled now and easier to remove things without having done too much work.' Another consequence of this practice, mentioned by the interviewees, is that the *system is not complete until late* in the life cycle, with the risk of not uncovering system-level issues until late in the project.

Table 1. Summary of the view of the interviewees on each agile RE practice (number of mentioning interviewees given, 9 in total); traditional RE challenges that the practice addresses & challenges the practice incurs.

Agile RE practices:	One cont scope flow	Cross-functional teams	Integrated RE	Gradual detailing	User stories & ATC
Addressed RE Challenges					
Communication gaps	1	9	4	5	2
Overscoping	6	4		4	
Keeping SRS Updated		4		4	2
Dev work not monitored fr reqs		2		1	
Unclear requirement coverage		1			
Customer expectations not met		2			1
Low motivation for reqs work	1	1			1
Quality issues		4			
Waste	2	1		4	1
Low reqs quality		1			1
Unreliable SRS				1	
Unstable SRS		1			
Challenges of the agile practices					
Planning for agility	3			3	
Weak reqs prioritization	3				
Weak effort estimates	1				
Quality issues	1				
System completed late	1				
Capturing innovation	1				
Lack of documented reqs		1			
Customer-proxy role		2			
Ensuring competence (RE, VV)		5			
Motivating teams for reqs work		3			
Weak requirements at start				2	

4.2 Cross-Functional Development Teams

This practice (which was implemented at the time of the study) was experienced to address *communication gaps*, though several interviewees also mentioned that communication within cross-functional teams is a challenge. For example, several interviewees mentioned that the customer representative (or proxy) is not always sufficiently available and involved in the development team. *Overscoping* is also seen to be addressed by this practice; since the team can focus on the most important and relevant requirements for their feature. In addition to implementing a feature, a cross-functional team is responsible for defining and documenting the requirements for a feature. This is believed to address both the challenge of *keeping the SRS updated*, as well as, the challenge of *development work monitored from requirements.* It was also mentioned that this practice increases the *clarity of requirement coverage* and degree to which *customer expectations are met*; by working closely together unclarities in the requirements can be resolved early on. This results in *requirements of higher quality* (e.g. clearer, unambiguous) and subsequent *higher software quality* (less errors), as well as, *less waste* due to rework since issues are resolved already while discussing requirements. One respondent contrasted this view by pointing out that independent testing leads to increased quality, i.e. where there is a communication gap and competition between developers and testers more issues are found and reported.

Challenges experienced in applying this practice include difficulties with (already mentioned) *customer-proxy role*, *including innovative ideas* from the developers, ensuring *sufficient test competence* within the team, as well as, getting the development teams to *document requirements*.

4.3 Integrated RE Process

Communication gaps are also seen to be addressed by integrating the requirements engineering work into the software development process; both by describing the development process in one (unified) development process, and by integrating the requirements work with other development work. (At the time of the interviews this practice was described in the process, but only partly implemented.) The awareness of each others' roles is believed to increase when they are all defined in the same process description. In addition, by bringing the requirements engineering tasks closer to the development work, several interviewees have experienced that the business and the engineering roles actively discuss the requirements and so gain an increased common understanding, which increases their ability to find solutions that satisfy both business and engineering aspects. One interviewee said, 'Working together on the requirements, you understand each other better and solve problems as you go along.'

4.4 Gradual Detailing of Requirements

When detailing the requirements gradually and iteratively, as the development progresses, it is natural to expect that the requirements are actively worked with throughout development. (At the time of the interviews, this practice was under implementation.) Gradual detailing of requirements was mentioned as addressing the challenges of *monitoring development from a requirements perspective* and *communication gaps* within development, as well as, between business role(s) and development team. This, in turn, was believed to lead to a more feasible scope, i.e. less *overscoping*. Finalizing and documenting the detailed requirements only when they are needed for implementation means that the requirements are (by then) more stable and less likely to change. This was mentioned by several of the interviewees as addressing the challenge of *keeping the SRS updated* and, by one interviewee, as resulting in a *more reliable requirements specification* (closer to what is actually implemented.) It was also mentioned that this reduces the (*wasted*) *effort* that would be required to handle the updates for those changes (that now occur before the detailing takes place.)

A couple of the interviewees had experienced a lack of a *clear requirements picture* at the beginning of development. This had resulted in significant requirements changes during the development with subsequent rework and frustration within the development team.

4.5 User Stories & Acceptance Criteria

Defining requirements with user stories [6] was mentioned by a couple of interviewees as a way to facilitate the *communication* between business and engineering roles, and (by expressing the viewpoint of the users) increase the probability of capturing and *meeting the customers' expectations*. (This practice was agreed, but not implemented at the time of the interviews.) One interviewee also believed that the user story technique increases the *quality of the requirements*; by ensuring that the user context is captured, the actual requirements are more clearly communicated. Documenting detailed requirements as acceptance test criteria was believed to increases the motivation of the developers to work with requirements since, as expressed by one interviewee, 'code is more fun to write than requirements.' (The case company uses automatic test cases.) It was also mentioned that this practice is believed to address the challenge of *keeping the SRS updated* (by generating it from the acceptance test cases.)

5. CONCLUSIONS

Our results indicate that agile practices (at least partly) remedy several challenges and issues related to traditional RE in large-scale software development, though they also pose new challenges. By improving the communication between the business and the engineering roles (with cross-functional teams, gradual & iterative detailing of requirements, and user stories) the requirements can be identified, communicated and agreed upon more efficiently. Also, defining the project scope via one continuous & unified list of scope where the most prioritized features are worked on first (to some degree) addresses overscoping.

Transferring an organization to agile RE practices is in itself a challenge that requires major mind-set changes; the business unit must adapt to gradual commitment to project scope and to becoming more involved with development teams through-out the project life cycle; the development unit needs to become actively involved in the requirements detailing and management.

Our results also uncover some challenges with agile RE practices such as ensuring sufficient competence (customer representatives, requirements and testing), including innovative ideas from development, and striking a good balance between agility and stability both at project level (degree of commitment in relation to flexibility for late changes) and within a development team (accuracy in effort estimates in relation to level of requirements detailing.) For the methods that had not yet been fully implemented at the time of the study (*User Stories & Acceptance Criteria* and *Integrated RE*), our interviewees did not mention any (new) challenges with these practices. We assume this is due to lack of experience with applying the methods, and therefore pose interesting points to revisit.

Future research includes investigating the impact of different levels of agility, the long-term effects of agile methods in large-scale software development, as well as, widening the research to include more software companies.

6. ACKNOWLEDGMENTS

We would like to thank all anonymous interviewees for their invaluable contribution to this project. The project is partly funded by the Swedish Foundation for Strategic Research and VINNOVA (The Swedish Governmental Agency for Innovation Systems) within the EASE projects.

7. REFERENCES

[1] Beck, K. 1999. *Extreme Programming Explained*. Addison-Wesley, London.

[2] Bjarnason, E, Wnuk, K., Regnell, B. 2010. Overscoping: Reasons and Consequences – A Case Study in Decision Making in Software Product Management. In *Proceedings of 4th IEEE Int. Workshop on Software Product Management* (Sydney, Australia, 27th Sept 2010), IWSPM'10. 30-39, IEEE Press, New York, 30-39. DOI=10.1109/IWSPM.2010.5623866

[3] Bjarnason, E., Wnuk, K., Regnell, B. 2011. Requirements Are Slipping Through the Gaps - A Case Study on Causes & Effects of Communication Gaps in Large-Scale Software Development. Accepted for publication at 19th Int. Requirements Engineering Conference, Aug 29th-Sept 2nd, 2011.

[4] The interview guide for the Before aNd After (BNA) case study is available at http://serg.cs.lth.se/research/experiment_packages/bna/

[5] Bose, S., Kurhekar, M., Ghosal, J. 2008. *Agile Methodology in Requirements Engineering*. SETLabs Briefings Online.

[6] Cohen, M. 2004. *User Stories Applied. For Agile Software Development.* Addison-Wesley, New York.

[7] Paetsch, F., Eberlein, A., Maurer, F. 2003. Requirements Engineering and Agile Software Development. In *Proceedings of 12th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises,* pp 308-313, IEEE Comp Society Press, Washington. DOI=10.1109/ENABL.2003.1231428

[8] Patton Quinn, M. 2002. *Qualitative Research & Evaluation Methods.* 3rd edition. Sage Publication Ltd., London.

[9] Pohl, C., Böckle, G., van der Linden, F. J. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, New York

[10] Ramesh, B., Cao, L., Baskerville, R. 2010. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal,* vol 20, issue 5, 449-280. DOI=10.111/j.1365-2575.2007.00259.x

[11] Regnell, B., Berntsson-Svensson, R., Wnuk, K. 2008. Can We Beat the Complexity of Very Large-Scale Requirements Engineering? In *Proceedings of Int Conf on Requirements Engineering: Foundation for Software Quality*, Vol. 5025 of LNCS, Springer-Verlag, pp 123-128. DOI=10.1007/978-3-540-69062-7_11

[12] Robson, C. 2002. *Real World Research*. Blackwell Publishing, London.

[13] Savolainen, J., Kuusela, J., Vilavaara, A. 2010. Transition to Agile Development - Rediscovery of Important Requirements Engineering Practices. In *Proceedings of 18th Requirements Engineering Conference*, IEEE Comp Society Press, pp.289-294. New York. DOI=10.1109/RE.2010.41

[14] Schwaber, K., Beedle, M. 2002. *Agile Software Development with SCRUM*, Prentice Hall, New York