Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges

Artem Marchenko
Nokia, Hatanpäänkatu 1, FIN-33100 Tampere, Finland
artem.marchenko@nokia.com

Pekka Abrahamsson
VTT Technical Research Centre of Finland, P.O.Box 1100, FIN-90571 Oulu, Finland
Pekka.Abrahamsson@vtt.fi

Abstract

Agile methods continue to gain popularity. In particular, the Scrum method appears to be on the verge of becoming a de-facto standard in the industry, leading the so called Agile movement. While there are success stories and recommendations, there is little scientifically valid evidence of the challenges in the adoption of Agile methods in general, and Scrum in particular. Little, if anything, is empirically known about the application and adoption of Scrum in a multi-team and multi-project situation. The authors carried out an ethnographically informed longitudinal case study in industrial settings and closely followed how the Scrum method was adopted in a 20-person department, working in a simultaneous multi-project R&D environment. Altogether 10 challenges pertinent to the case of multi-team multi-project Scrum adoption were identified in the study. The authors contend that these results carry great relevance for other industrial teams. Future research avenues arising from the study are indicated.

1. Introduction

Software development methodologies are constantly evolving due to the contemporary dynamic business environments, fast development of the technologies and persistent increase of end user demands. During the last years, Agile software development methods have gained popularity and are increasingly important to a significant number of software development organizations. [1] The area of Agile software development methodologies has been researched to some extent. [2]

Scrum is an Agile software development process that focuses on project management practices**.** Lately it has gained considerable popularity in large companies. For example, the literature shows that Scrum has been adopted by large companies such as Yahoo! [3], Microsoft [4], Intel [5] and Nokia [6].

In the current literature, the adoption of Scrum is usually situation-specific focusing on a single team and a single project. [7] However, in the large companies the opposite can frequently be the case, with several teams in each department supporting several projects simultaneously. [8] Surprisingly, despite the growing popularity the Scrum, the process has not been examined in the existing studies – less than 5% of the existing scientifically valid evidence on Agile software development addresses Scrum. [2]

This study aims at contributing to this knowledge gap in the field of Agile software development in general and the process of adoption of Scrum in particular. The focus of the research is the adoption of Scrum in a multi-team and multi-project environment.

We examined the adoption of the Scrum process in a three-team multi-project environment, where one of the authors led the adoption process and took the role of one of the Scrum Masters. The research question was: *How can Scrum be used in a multi-team and multi-project environment and what challenges, if any, emerge from the empirical qualitative evidence?* In line with this, we carried out an ethnographically-inspired case study applying certain elements of the grounded theory approach, and compiled 32pages, or 18075-words, of field notes in a diary kept during the 8 months of observation.

The principal results of the study are an increased understanding of the application of Scrum in the setting, as well as a set of challenges to be aware of during the adoption of Scrum in a multi-team multi-project environment. We also describe the evolution of the challenges and of the attempts to solve them.

The results bear direct pragmatic implications and

978-0-7695-3321-6/08 $25.00 © 2008 IEEE
DOI 10.1109/Agile.2008.77

we contend that the identified challenges have relevance for similar organizations and settings planning the adoption of Scrum. The research community can use the results as they are scientifically grounded on the ethnographically-inspired research approach, and they identify future research avenues.

The rest of the paper is organized as follows. In Section 2 an overview of the related work is presented. In Section 3 the research environment, method and research questions are described. In Section 4 our findings are reported. Section 5 presents a discussion on the findings and Section 6 provides the conclusions.

2. Related work

In recent years, Agile software development methods have gained increasing popularity. However, there are few empirical studies on the topic. The most recent and in-depth systematic review of 1,996 articles and studies in the area identified only 36 empirical studies of acceptable rigor, credibility and relevance. [2]

While some success stories [9] and adoption recommendations [10] have been published, the process of adopting Agile methods and especially Scrum has been given little attention. Begel and Nagappan [4] also claim that there has been limited empirical evidence on the usage/perception of Agile software development practices. In particular there is little focus on the problems to be expected, when adopting Scrum in multi-team and multi-project environments.

According to Schwaber and Beedle [11] there is a list of practices to follow in order to use Scrum. Schwaber and Beedle [12] divide these practices into seven categories: The Scrum Master, Product Backlog, Scrum Teams, Daily Scrum Meetings, Sprint Planning Meeting, Sprint, Sprint Review. Later Schwaber [13] added Sprint Retrospective to the list. In the existing literature we identified a number of challenges related to the adoption of the Agile processes in general, and Scrum in particular. We divided these findings into the above eight categories.

Schwaber and Beedle's [12] [13] categories should not be seen as a scientific research framework but they serve as a coherent structure for reporting the results. An alternate avenue to have taken would have been the selection of an adoption theory to guide the analysis. However, the majority of the adoption literature (e.g. Technology Adoption Model by Davis [14]) is mostly concerned with the identification of determinants and variables as variance theories often are (cf. [15]). In an area where very little research exists, an exploratory strategy is more appropriate.

2.1. The Scrum Master

The Scrum Master is a new management role introduced by Scrum. The Scrum Master is supposed to facilitate the process and to help people resolve problems (including psychological ones) while enforcing the process rules.

Silva and Doss [16] report that during the massive adoption at Capital One they found it challenging to maintain the quality of Agile coaches.

Begel and Nagappan [4] report that a survey performed on the global level in Microsoft, where Scrum is the most popular Agile methodology, revealed that "too many meetings" were considered to be the second biggest problem in the Agile methodologies. One of the problem roots is claimed to be the inefficiency of the meetings, especially when poorly run by a Scrum Master who is not focused enough to run the meeting quickly.

2.2. Product backlog

Product backlog is an evolving list of technical and business functionality that needs to be developed. Product backlog is supposed to be under the sole control of the Product Owner. The Product Owner has to make a clear list of priorities and show the project perspective to the team.

Moore et. al. [17] found that when implementing Scrum in a formalized environment the most challenging part was the development of a good product backlog. One tends to underestimate the time and effort involved in putting the backlog in place and developing the user stories for each backlog item.

2.3. Scrum teams

A Scrum team should be a cross-functional and self-organizing group of individuals working on a project. A Scrum team is responsible for meeting the sprint goal, however, it is supposed to be autonomous and have control over the exact process.

Sridhar et. al. [18] identify the cultural change from isolated specialist work to collaborative cross-functional style of work as being one of the key issues, when adopting an Agile methodology.

Greene [5] notes that, in the situation where the key skill is the domain knowledge, the need for very specialized domain knowledge makes it complex to ensure sufficient cross-training. Greene also states that, when applying the socially intensive Agile programming methods, the team members' communicational preferences play a significant role.

Schatz and Abdelshafi [19] identify a career perspective challenge. After switching to Agile

methodologies, people can become unsure about the career growth in the new environment since the traditional career ladder is not really made for cross-functional specialists.

For most Yahoo! Music teams, Agile methods required more discipline than before - keeping it simple required a sustained effort. According to Cloke [3] switching to Agile demands a significant shift in the thinking of the team.

At Google an ongoing issue is the QA involvement. The reason is that the QA Engineers support several projects. Some of the projects are not Agile, i.e. require little attention during development, but a lot of it at the end. This makes it challenging for the QA engineers to spend time each day on the project to give the Engineers immediate feedback. [20]

At Yahoo! Music they experienced handovers with the functional departments as very difficult to resolve within the sprint boundaries. The sometimes awkward co-existence of Agile and traditional product lifecycles continues to be a challenge at Yahoo! [3]

In Microsoft the scalability to large projects is considered to be the biggest problem for the adoption of Agile methodologies. The problem of inter-team coordination was found to be the 5th problem. [4]

Mahanti [21] claims that Nokia noticed that XP works best with small, independent, and co-located teams. According to Mahanti, Nokia found that hybrid approaches to software development were a more favorable option. He also refers to Motorola where it was found that Agile teams had difficulties interfacing with teams using traditional practices.

2.4. Daily Scrum meetings

A daily Scrum meeting is a short, stand-up, typically 15-minute meeting, during which the team members explain to each other what they accomplished since the last meeting, what they are going to do by the next meeting and what obstacles are on the way.

Cloke [3] reports that in Yahoo! Music team daily standup meetings run "by the book" started as disappointingly sterile so the team had to extend the meeting in order to drill out the useful information needed to reach the state of collaboration.

Greene [5] reports that in one software development team at Intel they had to de-emphasize explicitly the "what did you do yesterday?" daily question, as people felt it was a general status meeting question and tended to lose too much time on it.

2.5. Sprint planning meeting

During the sprint planning meeting the team together with the Scrum Master and the Product Owner plans the functionality to be built during the coming sprint, and how the team is going to complete it.

Striebeck [20] reports that, in Google, Product Managers often refuse to make prioritization decisions because of the engineering driven culture. Often, when a Product Manager was asked for prioritization of a feature, he turned to his Technical lead and simply asked "What do you want to do?" Regularly, the tech leads do not see the need to make such decisions during the planning meetings as they know that they can make them at a later point.

At Yahoo! Music measuring the team velocity was a challenge that stroke back at the release planning time, when the team was naturally hesitant to estimate the complexity of the backlog items, as it had a limited grasp of actual performance at the task level. [3]

2.6. Sprint

Sprint is a period of time, when a team is focusing on meeting the sprint commitments. During this period of time the team is supposed to have full authority over its actions and no external influence from the Product Owner, or anybody else, is allowed.

Cohn and Ford [22] report that a surprising number of developers view using Agile processes as an attempt to micromanage. Since approaches like Scrum and XP accelerate project cycles, developers interact with their managers more often but for shorter periods of time.

Begel and Nagappan [4] find similar fears in Microsoft where the fear of micromanagement is viewed as one of the reasons why daily standup meetings can be ineffective.

Sridhar et. al. [18] identify the shift in the power balance between project managers, stakeholders and the team to be the biggest challenge, when adopting an Agile methodology. According to them, since much of the knowledge in Agile development is tacit and resides in the heads of the development team members, this can make the organization strongly dependent on the development teams. Such a situation may not be acceptable for many organizations.

2.7. Sprint review

A Sprint review meeting is held at the end of every sprint. During this meeting the team demonstrates to the Product Owner, and optionally to the customers, what it was able to accomplish during the iteration.

We did not find any empirical (i.e. anecdotal or scientific) evidence on the Scrum adoption challenges related to the sprint review. To our knowledge, the current literature reports no challenges in sprint reviews.

2.8. Sprint retrospective

Sprint retrospective is supposed to be a focused time dedicated to the adjustment of the software development process and its improvements. Therefore we put issues related to development process improvements into this category.

Packlick [23], reporting on XP adoption that started from Scrum-like practices in a large organization, notices that in their experience development teams tend to reach a plateau after implementing a subset of Agile practices. In general, teams improved for four months to a year and then slowly leveled out.

Schatz and Abdelshafi [19] also noticed the danger of slipping back to old practices. When the goal of establishing a balanced and consistent workload is met, the team, product owners and stakeholders can become "bored" with the process. Without continuous improvement being everyone's main focus, it becomes easy to lose sight of the Agile principles that brought success in the first place.

3. Research design

This study covers an 8-month-long period of time from the Scrum adoption started in April 2007 to December 2007.

3.1. Research setting

The case department is a part of Nokia responsible for speech recognition, speech synthesis and related areas. Together with associated people formally working in the other functional departments (i.e. UI designer, test manager, build manager) there are about 20 people. In the department these positions are full or almost full-time involved in its projects. The department ships most of its software deliveries directly to phone product programs. The majority of people are located in closely located two person offices. At the beginning of this research there was no common meeting place allocated for the department. During the observed period, teams managed to acquire a common room dedicated to the department.

The Software Engineers officially belong to two sub-departments. The Chief Engineer reports directly to the Department Head as shown in Figure 1. During the adoption process, these developers formed three Scrum teams, referred to below as E-team, T-team and S-team. The UI Designer, Build Manager and Test Manager never belonged to the Scrum teams and cooperated depending on the need.

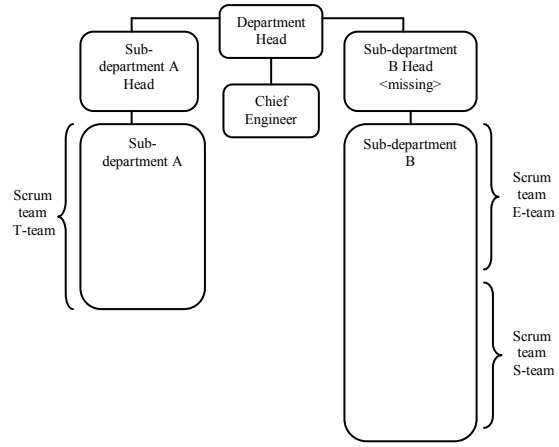

Figure 1. Department structure

During the study period one of the authors was in the department as Chief Software Engineer. He had some previous experience with Agile methods – about a year of trying some Agile practices within individual projects in the same department.

At the time of the adoption of Scrum, the department's Engineers had some experience with Agile methods. In 2005 one of the authors did some early trials of Scrum in a small single project team which later became a core of the sub-department A. The early trials brought positive results: the project was delivered ahead of schedule, the customer satisfaction level was above the average department level and the team was happy. In 2006 the author tried introducing Agile elements in a bigger and more complex project with the sub-department B members. In this project, a smaller number of Agile practices were introduced. Still the results achieved were noticeably better than the average ones and Project Manager reported that he was surprised there were so few problems in the final integration and localization that usually are the sources of difficult problems. These individual project incidents served as motivators to increase the level of Agile adoption in the department.

The Scrum adoption started in April 2007 with a single pilot team (E-team) consisting of 5 team members in sub-department B, one of the authors took the role of Scrum Master and the Head of sub-department A took the role of Product Owner. The rest of the department joined Scrum in August 2007. Everybody got at least one-day-long Scrum training in mid-May 2007, all the people performing as Scrum Masters took the Certified Scrum Master courses before starting in this position.

3.2. Research method

We performed an ethnographically-informed

empirical case study [24]. During the period of observation one of the authors was an active team member playing one of the Scrum Master roles and overseeing the whole adoption process. While performing the everyday actions, the author also reflected on the observations.

The results of the participant observation were recorded into a research diary that was kept daily in the beginning of the observation period and daily to weekly towards the end of the study period. When the author was not able to observe the actions directly, a summary of what happened during the absence period was filled after discussions with the team members.

The diary contains observations of the team actions, summaries of the talks to team members, the Product Owner and interested stakeholders (e.g. Department Manager), as well as the author's impressions of the reasons behind the actions observed. The observer's thoughts were always clearly separated from the participants' observations. An anonymized extract of a diary entry is provided below:

November 19, Monday
<s-team> retrospective day
- ScM took more active role in the retrospective (some team members, especially A-guy earlier explicitly asked ScM to make more decisions) and focus the retro on the sprint planning procedure. The main problem was everybody's uncertainty about what to do during the sprint planning. e.g. ScM kept asking the team "are you committed?" or "how big is it?" and everybody was silent, especially when they didn't really know the answer. A-guy then expressed that the team should have stated that more clearly.
- The proposal (both by the team members and ScM) was to create a more specific sprint planning procedure with clear moments when all the team members have to answer with one of the several options (yes, no, more discussion needed right now, research task needed). ScM later recorded the procedure to wiki as the first concrete'n'agreed working agreement.

During the observation, special consideration was given to the challenges in the Scrum adoption process. As reflective practitioners [25] we applied the grounded theory approach [26, 27]. In practice, this means that the researchers did not expect any particular challenges to appear or to happen. The research protocol followed required an accurate recording of whatever the team itself considered to be challenging or painful. We found that the sprint retrospectives were the most useful for the participant observations.

Our study aims at answering the following research question: *How can Scrum be used in a multi-team and multi-project environment and what challenges, if any, emerge from the empirical qualitative evidence?*

3.3. Data collection and analysis

During the eight-month-long period of observation an 18,075-word diary was compiled.

On examining the diary, a set of issues was identified as being challenging or painful by the team members. These challenges are reported in the empirical results section as a short summary and a chronologically arranged set of extractions from the related diary sections, edited slightly for clarity. It should be noted, that all the quotations are of the actual participants from the development process (team members, Scrum Master and Product Owner).

Finally the set of challenges was divided into categories related to the Scrum process concepts presented in Section 2. To validate our findings we used the member-checking approach and had a few of the team members study the findings and their evolution. This subsequently led to a few adjustments in the material presented.

4. Empirical results

In this section we present a set of challenges observed in the course of the adoption process. We categorize our findings in the same structure as the one presented in Section 2 – according to the Schwaber and Beedle [12] [13] view of what is needed in order to implement Scrum. It should be noted that the challenges identified do not necessarily go hand in hand with the categories as they were not so comprehensive. Each adoption challenge is presented systematically using the timeline of 8 months (when applicable) to explain how the challenge evolved over the time period. We also include an explanation of how the issue was subsequently handled. Furthermore, when a particular challenge can belong to several categories it was put into the most relevant one.

4.1. The Scrum Master

The role of Scrum Master was found to be a crucial one during the adoption phase. In particular, it was found challenging to find a balance between the enforcing and caring roles.

4.1.1. Challenge 1: Placing an overemphasis on the Scrum process and practices. At times it felt like the Scrum Master was pushing people too much towards what he wanted to see. Sometimes encouraging people to self-organize was viewed as the opposite: the Scrum Master forced team members to "self"-organize. We found that being too committed to the process is more

likely to be harmful, than beneficial.

From the very beginning, during the daily stand-ups, the developers addressed the Scrum Master more than the other team members. It turned out that the daily standup felt more like a typical status meeting to them.

The Scrum Master tended to be “too concerned” about solving the people’s problems, instead of allowing them to solve the problems themselves

In May, there was a tendency to follow the Scrum rules of freezing the content for a sprint. It escalated almost into a conflict with one of the team members. “Scrum rules are too strict and against common sense”. As a result of a number of similar conflicts in May, the team decided not to let the Scrum Master and Product Owner be present in their retrospectives (the idea came from the common one day Scrum training). As a result, the team found that the retrospectives became more comfortable. However, as a side consequence, the retrospectives became less organized and solved fewer issues, they also took a considerable amount of time to get started and only those people who were the most outgoing expressed their concerns. Also, the retrospectives produced less specific actions to be carried out. In one team member’s opinion, the reason for solving fewer issues was that “Scrum was already good and adjusted to the situation and there were simply no more issues to be resolved”.

In June one of the personal conflicts with the Scrum Master was caused by the fact that some people on the team felt upset because of the Scrum Master’s eagerness to promote Scrum. One of the team members claimed that "It might have worked in the US, but not in Finland, where modest people are valued". The Scrum Master was perceived as the one being involved everywhere and overly emphasizing himself. The Scrum Master was advised to listen to people (instead of the Scrum books) more. He was also recommended to ensure that his gestures would say the same as his words: e.g. in some situations the Scrum Master claimed not to be an expert in some area, but his gestures expressed the opposite.

In June one of the team members reflected about the beginning of the Scrum adoption. According to him, when the Scrum was introduced, the process was emphasized, and people had to adapt, not vice versa. Retrospectives seemed to help in resolving this issue.

After these June conversations, the Scrum Master took care to be less aggressive and listen more. The results were positive.

Still, in August at least one team member complained that Scrum was too inflexible. It felt like Scrum made everything feel more shared, leaving little space for the individual work. "We are not in a kindergarten and everybody fulfills their tasks. But it seems that the management believes this is a kindergarten, since they support Scrum so strongly".

In September an S-team member was annoyed about the Scrum Master asking the three questions in the daily meeting. The team member claimed these questions to be simply “stupid”, since he was already busy with the task hanging on the board. In September the same team wanted the Scrum Master to report more about what he was doing in order to make the situation fairer.

4.1.2. Challenge 2: The Scrum Master caring only about the individuals and interactions (and ignoring the process). Caring too much and being too cautious can cause the team to lose the feeling of discipline essential in software engineering. Individuals and interactions should be more important, than processes and tools, but the evidence indicates that there is some value in the process as well, especially when the Scrum Master is not an experienced facilitator.

Inspired by the literature [13], from the very beginning the Scrum Master tried to keep the teams as self-organized as possible. The Scrum Master approached the team members with questions, raised the issues and tried to facilitate problem solving, but for a long time tried not to propose any solutions.

In June, one of the team members commented to the Scrum Master that, while listening to people and not pushing his own ideas is important, the Scrum Master should not be “too nice”. For example, sometimes during the daily standup the team might go off-track into a side conversation. In this case the Scrum Master should keep the discussion on-track. The same team member insisted that it was the Scrum Master’s responsibility to resolve conflicts in the team. “You are the Scrum Master and should look after the process".

In August one team member noted that if the team was always allowed to change its commitment during the iteration, the concept of a sprint lost its meaning.

In September the E-team Scrum Master was the one who placed the tasks on the information radiator (i.e. the wall). He shared the concern that the team did not do things the right way if he put the tasks on the wall instead of the team members.

In September the same E-team retrospectives became less useful than before. In the E-team, the Scrum Master was a sympathetic person who wanted to let the team decide on the retrospective process. However, the meeting lost its structure. While it was still used for resolving the issues and talking about the process, the amount of root causes analyzed and actions taken dropped significantly.

In October the author discussed with the E-team Scrum Master the fact that during the several last retrospectives the team identified no possible

improvements. The E-team Scrum Master agreed that their sprints did not look that *Scrumish* anymore, and tasks were late on the task board. He explained that without someone taking a personal interest in Agile it became quite easy to slowly drift back to the "old ways". In the next retrospective the E-team raised this issue and some corrective actions were identified.

In November the S-team was disappointed (and one team member was even angry) about the fact that the Scrum Master was not leading the sprint planning meeting enough. The team consisted of not very talkative people who just started working together, and without a clear leader there was a lot of silence at the meetings, while people had some troubles, e.g. with understanding the particular Product Owner's request.

Later in the November retrospective this issue was discussed and the Scrum Master was asked to be more decisive in the situations when the team was unsure what to do. The Scrum Master helped the team to craft a few procedures the team members should follow during the sprint planning. These procedures helped to eliminate the moments of uncertainty.

In December during the retrospective process, which was driven by the Scrum Master (in response to an earlier team request), the team came up with several self-disciplining actions. For example, "If it is important, then there should be a card" [on the tracking board] and "Writing tests is part of the everyday stuff as much as writing the code".

4.2. Product backlog

In the observed case we noticed that it was difficult for the Product Owner and organizational roles close to him to make decisions about the project priorities and present an attractive product vision.

4.2.1. Challenge 3: A lack of clear management expectations and actions. The need for a management vision was a constant request throughout the Scrum journey. The lack of a management vision was reality long before the Scrum adoption, but the Scrum introduction made it all the more obvious and thus more painful.

From the very first adoption weeks and building of the initial product backlog, it was evident that there was no single clear priority list and the amount and desired direction of the long-term research was not clear. The realization of this enabled change and during the adoption journey the understanding of a high level vision was constantly growing.

In August after the whole department switched to Scrum, another team found a permanent problem because the product backlog items were not in a priority order – during the sprint planning their Product Owner often wanted the team to do some item from the middle of the list. It took a couple of months of trial and effort to have the top of the product backlog prioritized for the sprint planning. At that point there was no single product backlog, but instead several lists were dynamically assembled into an actual product backlog for the sprint planning.

In September with the introduction of the whole department backlog and the whole department it became clear that the department was indeed trying to support and develop many projects simultaneously. This led to splitting the backlog into more manageable sub-categories. It also became clear that sometimes the Product Owner was prioritizing some project of little benefit for the whole department, but of high benefit for him personally.

It is interesting to note that in September one of the teams was disappointed with the Product Owner accepting "automatically" everything the team was reporting (about the completeness of the product backlog item). The team was clearly expecting product backlog management to take a more active role in dealing with the product backlog item deliveries.

4.2.2. Challenge 4: Too much maintenance and bug fixing undermining the team productivity and morale. Challenges related to the team being overloaded with a heavy amount of maintenance and bug fixing are not Scrum specific, however, they surfaced in the adoption process of Scrum. After the practices of short iterations and velocity measurement were introduced, it became clear just how much effort went into maintenance. The need to make an explicit decision on whether bug fixing or new functionality was more important was not always comfortable either for the Product Owner or the team.

In the case environment, the situation was amplified by the fact that the software developed was supposed to work on many platforms while there was no reasonably large automated test suite. Regularly the teams were getting bug reports reproducible on the specific platform only (or on hardware only). The teams spent a reasonable amount of time on just getting the version of the platform, where a bug had been found.

In April, the E-team decided to explicitly allocate time for maintenance and side-line tasks (e.g. not critical algorithm improvements) during the sprint planning.

In May, the Scrum Master and Product Owner were surprised to find out how easily the extra work slipped into a sprint, because of the Department Head's requests and sudden bugs.

In August after a particularly long and difficult bug fixing the Product Owner told the team: "If you think

you could do something in order to prevent bugs in the future, please, do it". It did lead to a more careful attitude towards testing the newly created material, but did not lead to the introduction of test driven development or to automating more test cases.

In autumn, the S-team identified the need for the new development as one of the top requests from a couple of retrospectives in a row. The reason was that their Product Owner highly prioritized one of his favorite and bug-filled projects. The low quality of that project suddenly became everybody's pet gripe instead of being some people's once-in-a-while issue.

4.3. Scrum teams

In the observed case the most important team-related challenges related to building a cross-functional team from initially highly specialized people.

4.3.1. Challenge 5: Fitting Scrum and short iterations into research intensive teamwork. This concern was expressed on the very first day of adoption. The department had a reasonable amount of long-term research activities performed mostly by the specialized individuals. These activities were sometimes related to the ongoing projects, sometimes not. The need to timebox the activities that were difficult to evaluate and were quite different from the activities of the rest of the team remained a challenge for some time. Eventually the specialists started performing more of the external expert roles. One of them collaborated with the team only during the sprint planning and reviews, the other one used to come to daily stand-ups two times a week only. The research-related part of the work was often handled on the basis of the individual agreement with the Product Owner.

For these almost remote experts, the practice of timeboxing did not allow them stray too far from the team activities. Those wanting were able to follow each other's progress and when possible, the collaboration did happen more easily.

4.3.3. Challenge 6: Overspecialism undermining collaboration. The E-team had several specialized team members that made it difficult for the team itself to have common reference points during the estimation and planning sessions and for the other team members to collaborate with the specialists.

Already in April one of the specialists felt uncomfortable about demonstrating her work, because not many people could recognize any improvements.

During the sprint plannings in May, it was clear that people working on the related things liked Scrum and people working on individual items did not. It was more evident that a couple of the team members were doing things only slightly related to the other people's work. In the meetings these team members were silent most of the time.

For specialists the sprint planning meetings were feeling quite silly. They were specialists, working on a single full sprint task, did not need any help and could not help anybody else. Some of the specialist work was out of the product and sprint backlogs and was tracked between the Product Owner and specialists.

At the end of May, one of the specialists refused to participate in the estimation/planning process at all. She listened but did not play planning poker and "didn't care" about the number on the items associated with her. The reason she gave was that in a specialized team it was difficult to understand what the other specialists were actually going to do.

In June one of the team members noted during the daily standup that Pair Programming and TDD (which he tried together with the Scrum Master) made it easier for him to get started in a specific and somewhat specialized area.

In the June retrospectives the specialist that decided to participate in the team activities the least was looking disconnected from the others - for the whole sprint he was working on "his" large task and was not taking part in the daily stand-ups. However, he presented his results and expressed the hope that after the next sprint the other people would be able to help him with the "general software" issues. In reality it happened much later.

In June one of the team members told the Scrum Master that it might be quite difficult to use Scrum in the team, because there were several sub-teams that had multiple different goals.

In July one of the specialists tried estimating and planning her work together with the team, but it was difficult for her. She lacked the common reference points - her tasks and area of expertise were quite different from the other team members

In August another team member expressed the opinion that they were a specialized team and sometimes they were just used to the particular work being done by a particular person even though at times some other people could actually help though with the lower speed.

By November-December, most of the individualistic/specialized team members of the pilot E-team were making progress visible to the rest of the team (by putting the corresponding cards on the task board) and the team also became more capable of helping each other.

4.3.4. Challenge 7: Overindividualism. This issue is often related to overspecialism, but is separate. Some people (usually excellent performers) do not value

team collaboration and prefer individual work to the extent of sabotaging the whole team activities

Already in May the team decided to allow one specialist to participate in only two daily stand-ups a week and another team member not to come unless he had something to share.

From May to June one of the team members was working on a particular task in parallel with some other team members "in order to choose the best implementation later". While the idea of figuring out the best implementation was definitely valuable, the individual style of work was producing a solution poorly integratable with the others' work.

In August the conflict between the abovementioned person and the Scrum Master (more) and the same person and the team (less) came to its culmination. This team member observed the rules (during planning, estimation, daily stand-ups), when team members asked him, but not when the Scrum Master asked. When he missed a daily standup another team member said "I think everybody should be present, but maybe it's only my opinion". In that particular case the individualist had a respectable reason not to come, but the team already assumed it was his negligence of them or the common rules. In one daily standup it was physically visible how this same individual positioned himself as a solo: the whole team was standing in front of the card board in a half-circle while he stood to the side.

Later this issue was raised in the retrospective in the absence of the Scrum Master. Almost nothing has been recorded as the official retrospective summary. However, the issue seemed to be discussed seriously. The solution was more-or-less that strict application of the common rules should be more attentive to the concrete circumstances. So the situation somehow fell between the lines, but it seemed the individual discussed would also try to participate more in the process - not-necessarily in terms of the number of daily Scrums attended, but more in terms of a real involvement (e.g. not looking skeptical during the sprint planning).

After this retrospective the atmosphere in the team became more productive and collaborative. The Scrum Master felt that an invisible barrier fell down – the team had still progress to be made in making estimates, updating each other, etc, but it somehow felt that now these were work issues to be resolved or considered, and not to be ignored.

In September a similar issue arose in the T-team. One of the team members was more ignorant of Scrum than the others. However, we do not know how this issue was resolved if it was resolved at all.

As mentioned in Section 4.3.3 by November-December most of the individualistic/specialized team members of the pilot E-team were making their work visible to the rest of the team.

4.4. Daily Scrum meetings

We did not observe any daily Scrum meeting related to challenges other than the ones described above in Sections 4.3.3 and 4.3.4.

4.5. Sprint planning meeting

In the observed case, the largest challenge related to the sprint planning was balancing the workload, i.e. the teams tried to avoid overcommitting themselves.

4.5.1. Challenge 8: Committing to too much. Committing to too much work for the sprint was one of the common problems during the adoption journey, especially in the beginning.

In May the Scrum Master and Product Owner were surprised to find out how easily extra work was able to slip into a sprint, because of the higher management requests and sudden bugs. It escalated to almost a conflict ("Scrum rules are too strict and against common sense").

In the June sprint, there was a particularly difficult situation with one of the team members desperately trying to complete one product backlog item up to the point of doing overtime (not officially permitted). His situation was known to the team, but due to the high level of specialization others could provide little help.

During the vacation period in July, there was a six-week-long sprint that was a total disaster in terms of the amount of material the team committed to build compared to what was actually built.

The results of the sprint and the previous experience were discussed and resulted in a clear decision to have "no more 6-week sprints, even during the vacation season - it's too difficult to plan for that long period of time" and the undertaking to commit only to what the team was really sure it could do and possibly some items taken into sprint as extra i.e. "we don't commit to doing them, but if we happen to have free time, these ones will be taken up". This rule was not always closely followed but it was easy to see that in the end of the sprint the team and Product Owner were more satisfied when this rule was followed more.

4.6. Sprint

In the observed case it was difficult to track the progress during the sprint and to react to the tracking results being the biggest sprint concept related challenge. Only minor problems were observed with management trying to interfere mid-sprint.

4.6.1. Challenge 9: Difficulty in tracking progress and in using the results of the tracking. Burndown charts and tracking work in-sprint are supposed to help the team coordinate efforts and meet or adjust to sprint commitments, when needed. Unfortunately the results of the careful tracking take time to become visible; some people become bored with the need to play with the childish cards and tracking process. Also tracking brings less benefit, when applied to a specialized team. If the team members are hardly able to help each other, then why track?

From the beginning of the adoption journey in May, the specialized team members did not like splitting the product backlog items into tasks and had a single task for the whole sprint. Some specialized work was deliberately tracked out of the product backlog. Specialists tended to ignore the technical means supposed to aid the collaboration. For example, they tended not to write, update or move task cards during the daily stand-ups.

In June there was a discussion about what kind of task deserved a task card and for what (the reason to have a card per bug was to talk over a pile of bug fix task cards during the retrospective).

In June the issue of task estimation and updates being difficult and not too meaningful was raised again. Some team members wanted to update the time spent on the task, not the time still needed for the completion.

In mid-June the E-team came up with an improvement the daily standup. To make it easier for the Scrum Master to follow up the changes (e.g. when copying them to electronic form), when something was changed in the task, its card was flipped upside down as shown in Figure 2.

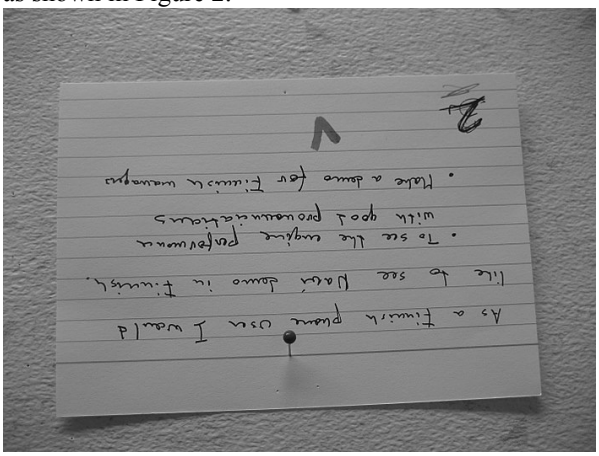

Figure 2. Task card flipped upside down for easier tracking

In August several people were happy about the idea to have a number-of-tasks based on a burndown chart. After the emotional investment in the idea it motivated them to create tasks and follow the idea.

In September the E-team came up with an improvement for sprint backlog maintenance: not to estimate it in hours, but to have task cards of physically different sizes as shown in Figure 3. In this way a full card was a big task, a half-card was a normal task, and quarter-card was a small task. When calculating burndowns, the sizes were to be counted as 4, 2, and 1 respectively.

Figure 3. Task cards of different size. A part of the team table

In September a large number of excel sheets and backlogs started going out of control and even sprint backlogs were not really visible on the intranet.

In September all the teams were against the common product backlog for three teams, as it was too difficult to understand, when it was known in advance which team is going to work on which task.

4.6.2. Challenge 10: Management interfering too much. The team's self-organization is one of the Scrum corner stones. It does not match traditional ways of developing software and at times management tries to interfere. The department developers were lucky not to have too much management interference even before Scrum, although it could still happen that managers came with sudden important requests

When the adoption started the Product Owner sometimes had to be asked to stay silent during the daily Scrum, otherwise the team would not be able to fit into 15 minutes and would run in too many details.

In April the same Product Owner sometimes asked concrete team members to do a critical task without consulting the team. However, once the Product Owner became confident that the team was not going to ignore the critical issues he stopped interfering during the iteration except for the specialists described above who did a considerable amount of work "out of Scrum".

In May the team even developed a (not always followed) rule that most of the demos should be

accepted before the sprint review meeting.

All in all it was found that busy product owners were generally happy not to micromanage once they were confident that critical issues were going to be fixed relatively soon.

4.7. Sprint review

During the course of our observation we did not observe any challenges specific to the sprint review.

4.8. Sprint retrospective

During our observation we did not observe any challenges specific to the sprint retrospectives other than the ones already described above.

5. Discussion

Based on our analysis most of the issues identified can be traced back to the three basic challenges. These challenges form the principal lessons-learned from this study.

1. Transition from a specialist-based solo development to a cross-functional team development: learning to take as a unit of development a team rather than a single developer. It can require quite a significant social change from the solo-oriented software development into a cross-functional team able to commit and be accountable as a team. We observed a number of challenging situations related to the fact that the specialized team members were limited to help their peers or even had little interest in doing so because they had so many of their "own" tasks.

2. Change in the role of the management: The need for the management to decide about a high level vision and real priorities. The prioritization issue was deemed to be one of the most critical success elements in Scrum. While Scrum clearly raises the issue of the human capacity available for a project, the organization may not be willing to accept the fact that there are too many parallel projects running. In the previous development model, this issue was not clearly present.

3. Persistence and determination in the course of the continuous improvement. As Schatz and Abdelshafi [19] pointed out when the situation becomes somewhat better than the starting point, the team and stakeholders can become "bored" with the process. When it is not everyone's main focus, it becomes easy to lose sight of the Agile principles that brought success in the first place and to start slipping back to the old practices.

This study covers a single adoption case only and therefore further research is required for generalizations to be made. However, in this article some important lessons, which can be useful for other companies, were learned. Firstly, it is wise to pay special attention to the three basic challenges above whenever Scrum is to be adopted in a similar research intensive multi-team multi-project situation. Secondly, if the aim is to succeed in such an environment it is best to make advance preparations for ways of integrating the solo specialists into a cross-functional team, for ways of motivating the management to prioritize decisions on a regular basis and for ways of creating incentives (not necessarily monetary ones) to sustain continuous improvement after the initial adoption period.

6. Conclusion

To our knowledge the adoption of the Scrum process in a multi-team context with teams working simultaneously on several projects is poorly addressed in the existing literature. In this study we examined this type of situation to determine what types of challenges emerge doing the adoption of Scrum in such an environment. Currently, the case department has a positive impression of the Scrum process and it is to be used as the primary tool to develop software. However, the challenges we identified are applicable to other similar organizations adopting Scrum.

The results of our empirical observations add to the body of empirical evidence on the challenges in the adoption process of Scrum As an answer to our research question (*How can Scrum be used in a multi-team and multi-project environment and what challenges, if any, emerge from the empirical qualitative evidence?)* ten specific challenges were identified during the course of this case study. Furthermore, the case application of Scrum in research intensive team increases our understanding of the obstacles in different practical situations.

There are several limitations in the study that should be acknowledged. The observed time period was relatively short, i.e. only eight months. Yet, we consider it to be long enough to consider the adoption phase of the Scrum process. As only one person observed the teams in action, there is always a chance that due to the researcher's bias some challenges or factors may have gone unnoticed. This was mitigated by the discipline of keeping the research diary throughout the observation period and following a research protocol in doing so.

We believe that the results obtained can be directly used by the industry in similar situations. However, as the data presented by our study covers only a single adoption case only, there is room for more research in

the area to derive more general adoption challenges. The collection of qualitative data in a longitudinal study, such as this, enables the use of theoretical lenses in the future. This is necessary to better understand the dynamics of the adoption process of Scrum in a wider context.

7. References

[1] O. Salo and P. Abrahamsson, "Agile Methods in European Embedded Development Organizations: a survey study of Extreme Programming and Scrum." *IET Software,* 2008, vol 2, pp. 58-64.

[2] T.Dybå and T.Dingsøyr, "Empirical Studies of Agile Software Development: A Systematic Review", *Information and Software Technology,* 2008, doi: 10.1016/j.infsof.2008.01.006
.

[3] G. Cloke, "Get Your Agile Freak On! Agile Adoption at Yahoo! Music", in *AGILE 2007*, 2007, pp. 240-248.

[4] A. Begel and N. Nagappan, "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study", in *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, 2007, pp. 255-264.

[5] B. Greene, "Agile methods applied to embedded firmware development", in *Agile Development Conference, 2004*, 2004, pp. 71-77.

[6] ITEA-AGILE, "Agile software development of embedded systems", *Newsletter, no.2,* 2006, available online at http://www.agile-itea.org/public/deliverables/D.6.4.4_ITEA-AGILE-Newsletter2-2006.pdf

[7] L. Rising and N. S. Janoff, "The Scrum software development process for small teams", *Software, IEEE,* vol. 17, pp. 26-32, 2000.

[8] M. Lindvall, D. Muthig, A. Dagnino, C. Wallin, M. Stupperich, D. Kiefer, J. May, and T. Kahkonen, "Agile Software Development in Large Organizations", *Computer,* vol. 37, pp. 26-34, 2004.

[9] J. Sutherland, C. R. Jakobsen, and K. Johnson, "Scrum and CMMI Level 5: The Magic Potion for Code Warriors", in *AGILE 2007*, 2007, pp. 272-278.

[10] B. Boehm and R. Turner, "Management challenges to implementing agile processes in traditional development organizations", *Software, IEEE,* vol. 22, pp. 30-39, 2005.

[11] K. Schwaber and M. Beedle, "Agile software development with Scrum", in *Series in agile software development* Upper Saddle River, NJ: Prentice Hall, 2002, p. 21.

[12] K. Schwaber and M. Beedle, "Agile software development with Scrum", in *Series in agile software development* Upper Saddle River, NJ: Prentice Hall, 2002, pp. 31-56.

[13] K. Schwaber, *Agile project management with Scrum*. Redmond, Wash.: Microsoft Press, 2004.

[14] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology", *MIS Quarterly,* vol. 13, pp. 319-340, 1989.

[15] A. Langley, "Strategies for Theorizing from Process Data", *The Academy of Management Review,* vol. 24, pp. 691-710, 1999.

[16] K. Silva and C. Doss, "The Growth of an Agile Coach Community at a Fortune 200 Company", in *AGILE 2007*, 2007, pp. 225-228.

[17] R. Moore, K. Reff, J. Graham, and B. Hackerson, "Scrum at a Fortune 500 Manufacturing Company", in *AGILE 2007*, 2007, pp. 175-180.

[18] N. Sridhar, M. RadhaKanta, and M. George, "Challenges of migrating to agile methodologies", *Commun. ACM,* vol. 48, pp. 72-78, 2005.

[19] B. Schatz and I. Abdelshafi, "The agile marathon", in *Agile Conference, 2006*, 2006, p. 8 pp.

[20] M. Striebeck, "Ssh! We are adding a process... [agile practices]", in *Agile Conference, 2006*, 2006, p. 9 pp.

[21] A. Mahanti, "Challenges in Enterprise Adoption of Agile Methods - A Survey", *Journal of Computing and Information Technology,* vol. 14, p. 10, 2006.

[22] M. Cohn and D. Ford, "Introducing an agile process to an organization [software development]", *Computer,* vol. 36, pp. 74-78, 2003.

[23] J. Packlick, "The Agile Maturity Map A Goal Oriented Approach to Agile Improvement", in *AGILE 2007*, 2007, pp. 266-271.

[24] R. Hugh, S. Judith, and S. Helen, "Ethnographically-informed empirical studies of software practice", *Inf. Softw. Technol.,* vol. 49, pp. 540-551, 2007.

[25] D. A. Scho□n, *The reflective practitioner : how professionals think in action*. Aldershot: Arena, 1995.

[26] B. G. Glaser and A. L. Strauss, *The discovery of grounded theory : strategies for qualitative research*. Hawthorne, N.Y.: Aldine de Gruyter, 1967.

[27] A. L. Strauss and J. M. Corbin, *Grounded theory in practice*. Thousand Oaks: Sage Publications, 1997.