THE PERCEIVED IMPACT OF THE AGILE DEVELOPMENT AND PROJECT MANAGEMENT METHOD SCRUM ON INFORMATION SYSTEMS AND SOFTWARE DEVELOPMENT PRODUCTIVITY

Karlheinz Kautz
Faculty of Business, University of Wollongong,
Wollongong NSW 2522, Australia
Kautz@uow.edu.au

Thomas Heide Johansen
Progressive AS,
DK-2730 Herlev, Denmark
thj@progressive.dk

Andreas Uldahl
Ernst & Young Denmark,
DK-2860 Søborg, Denmark
Andreas.Uldahl@dk.ey.com

ABSTRACT

This research contributes to the body of knowledge in information systems development (ISD) with an empirical investigation in form of a case study that demonstrates the positive impact of the agile development and project management method Scrum on information systems and software development productivity and it provides a useful operationalization of the concept through seven identified indicators for productivity. Despite the fact that the case unit had challenges with the use of Scrum, the indicators identified the areas where the company had managed to exploit the potential of Scrum and its practices with regard to increasing productivity. The research results are discussed both with regard to the existing Scrum literature as well as to complex adaptive systems (CAS) as a foundation for ISD and agile development.

INTRODUCTION

Over the last decade agile information systems and software development has received much attention from researchers and practitioners as an approach for dealing with change and the unpredictable and hardly controllable elements of ISD in a dynamic environment. While numerous publications claim a positive impact of agile development and in particular Scrum on information systems and software development, only little empirical work exists to verify these claims. The literature review, which was part of the study reported here uncovered some notable exceptions. To further contribute to this body of knowledge we set out to answer the following two research questions: What impact has the introduction of the agile development and project management method Scrum on information systems and software development? What is the effect of any deviations from the guidelines for Scrum? The results we present in the following are part of a larger project where we developed a framework for investigating the impact of Scrum (see Johansen & Uldahl 2012). In this paper however we concentrate on one of these concepts, namely Scrum's impact on productivity in information systems and software development. In the remainder of the paper we first briefly introduce Scrum, and then we describe our theoretical background and the research setting and method. Subsequently we present and discuss our findings against the existing literature on Scrum and relate them to complex adaptive systems (CAS) theory, a theory which is considered to provide a theoretical foundation for ISD (Kautz 2012) and in

particular agile development (Highsmith 2002). We finish with some conclusions and an outlook to future research.

SCRUM – AN AGILE DEVELOPMENT AND PROJECT MANAGEMENT METHOD

Scrum is an agile information systems and software development method with a strong focus on project management, which was formalized and tested by Schwaber and Sutherland in the mid 1990ties (Schwaber & Beedle 2002, Schwaber 2004). Scrum focuses on an iterative and nimble development process, on transparency, visibility and on cooperation in and between the development team and the customers. In Scrum the development team is called the Scrum team. Unlike traditional development projects where analysts, developers and testers are typically separated, Scrum teams are built on an interdisciplinary basis and comprise all these roles in one team preferably in one physical location. This structure, as well as Scrum's focus on self-organization aims at creating team dynamics and a better understanding of the tasks to be performed jointly. In this context the role of the Product owner has the responsibility to represent the project and product externally to other stakeholders and customers and to handle and manage the tasks that appear in the product and release backlogs (see below) (Schwaber & Beedle 2002). Internally, the role of the Scrum master will provide leadership, motivate and facilitate the team in line with the Scrum values, practices and development process.

A Scrum development process is structured through a product backlog, which is a prioritized list of required business and technical functions of the envisioned product. It might change in line with the customer's new needs. A release backlog is a prioritized subset of the total product backlog and defines the functions to be included in a release. A Scrum, performed in so-called sprints, is a set of development tasks and processes which a Scrum team carries out to achieve a given sprint goal. The length of a sprint is predefined. It typically lasts between 5 and 30 calendar days (Schwaber & Beedle 2002). What needs to be done during a sprint is determined by a prioritized sprint backlog, which is determined together with a sprint goal before the start of each sprint by the team and Scrum master and others, if necessary, at a planning meeting. Throughout a project a burn-down chart shows the amount of work left to do versus time over a given period (Schwaber 2004). In short daily Scrum meetings project members briefly present what they have done during the preceding day, which tasks they take on that day, as well as any challenges and obstacles that might have prevented them from carrying out their work without any solution being discussed. Scrums of scrums are additional short meetings by the Scrum masters of projects, which consist of several Scrum teams. At the end of a sprint a sprint review meeting takes place where the Scrum team, the Product owner, other management, and one or more representatives from the customer (Schwaber & Beedle 2002) assess the team's development process and progress in relation to the predefined sprint goal. Finally the Scrum team, the Scrum master and possibly the Product owner hold a meeting, called a retrospective, to secure learning and further improvement in the team where both the process and the product are assessed and discussed by each individual team member.

LITERATURE REVIEW AND THEORETICAL BACKGROUND

In our study we were interested in the impact of a specific method, namely Scrum on information systems and software development. Our literature review was therefore focused on that particular approach and not in general on project management methods' or agile methods' impact on information systems and software development. This limited our sources to writings which take their starting point in agile software development. We combined a concept-centric with an author-based approach (Webster & Watson 2002) and applied backward referencing of sources. Our original search with keywords such as 'impact of Scrum', 'effect of Scrum', 'impact of Scrum implementation', and 'effect of Scrum implementation' primarily in Google, Google Scholar and IEEE sources lead to about 90 publications of which 8 dealt more precisely with our research problem. An additional 8 articles were

identified through the other mechanisms. From that literature we derived a number of concepts and for these concepts indicators for the impact of Scrum on information systems and software development processes and projects. The resulting framework consisted of the identified, interrelated concepts productivity, quality, role of leadership, transparency, customer satisfaction, as well as employee satisfaction and a total of 38 indicators, which defined the concepts on a more detailed level. Here - due to page limitations - we are focusing on Scrum's impact on the first one.

Productivity in the agile literature is an expression of the development team productivity (Moe & Dingsøyr 2008). There are a number of interrelated indicators that are linked to different areas that may impact on productivity. Dybå & Dingsøyr (2008) describe the results of a comparative case study where productivity was measured in projects driven by traditional and agile development methods based on the number of lines of code (LOC) per hour, month or employee. Guang-Yong (2011) describes the measurement of productivity in the number of lines of code, and demonstrates how productivity increases gradually as a team becomes more self-organized and manages to review its development processes to avoid the repetition of mistakes. Appelo (2010) has a different view how productivity can be measured. He highlights the increased functionality to the final product as a direct indicator of improved productivity. The way he measures the functionality is the number of story points that have been completed within a given period. A story point is a number that reflects the severity of a given task. Mahnic and Vrana (2007) and Mahnic and Zabkar (2008) define the assessment of productivity as a ratio of the added value versus the associated financial costs as well as costs associated with bug fixes. We use these sources to investigate the indicators employee performance, the time associated with fixing bugs, and repetition of the same mistakes. Sutherland and Altman (2010) use the term "perfect hours" as a label for a project participant's undisturbed and uninterrupted work. They emphasize that a project's progress and productivity should be measured by taking perfect hours combined with other indicators into account. The number of interruptions and the number of uninterrupted development hours were the two indicators we descended from these authors. Moore et al. (2007) argue that increased productivity through the use of Scrum is grounded on its focus on delivering functional software in short time intervals with fixed deadlines where developers do not end up in endless development cycles in an attempt to provide perfect solutions with a product that can handle everything at one time. The avoidance of continuous development cycles and compliance with deadlines are the last two indicators we derived from these authors.

RESEARCH SETTING AND METHOD

We chose a case study approach to research the impact of Scrum on information systems and software development processes and projects. The chosen case organization has approximately 40 years of experience in solving complex IT tasks. Some years ago it changed from being publically owned to private company. It has about 3,000 employees, who are involved in the development of administrative and statutory software solutions. The investigated case department falls into the latter category and has 45 employees. Its sole product is a case management system for municipal job centers, which gives administrators the opportunity to work across different platforms. For the development of the case management system, the department previously followed the traditional waterfall model. In 2011 it launched the implementation of Scrum as the preferred development model. At the time of our investigation, the department had completed three full releases with the use of Scrum. As such the department had the profile of the unit of analysis we were looking for, an organization that had recently, within the past year, chosen to implement Scrum, and that had previously used the traditional waterfall model. With the former model still in their minds we expected the employees to make candid assessments of the impact of Scrum as compared to the past.

As we were not able to make direct measurements, such as number of interruptions, errors, uninterrupted development hours, etc., we chose to directly ask respondents about their perceptions of

the given concepts. The indicators, which we had derived from the literature review, were therefore transformed into direct questions for our interviews, which we validated with 2 employees in a small pilot study before putting them to the 11 interview partners, who were available for the study. We developed 3 largely overlapping interview guides for the three stakeholder groups, with 6 developers as respondents, 4 respondents in leadership roles such as Scrum master, Product owner or unit managers and one representative from the service department, which is responsible for customer liaisons. All interviews were recorded, transcribed and handed over to the respondents for approval. The results of our analysis were also presented to the participants of this study and the case organization at large.

The data collection with standardized interviews allowed both collections of qualitative and quantitative data. We first asked the respondents to numerically assess, on a scale from -5 to + 5, for each indicator its individual change, improvement or decline, as compared to the situation before the implementation of Scrum and then to evaluate its impact on the concept in question, here productivity. After that quantitative judgment we asked into the reasons for these assessments, which provided rich qualitative data.This combination of data allowed for data and method triangulation to improve the validity of our findings (Andersen 2006). The subsequent analysis was based on mean values for the quantitative data within each indicator; these were interpreted on the basis of the qualitative opinions. The results were then compared and discussed with regard to published Scrum guidelines, findings from the literature, and CAS theory. It is worth pointing out that the numerical element of the collected data should be considered secondary. The interviews were intended as the primary source to collect qualitative data with a statistical element - and not vice versa. The quantitative data was exclusively used to create an indication and an overview over any specific area.

RESULTS – SCRUM'S IMPACT ON PRODUCTIVITY

Table1 summarizes the respondents' assessment of Scrum's impact on productivity. Despite some individual variations the respondents' mostly positive scores indicate their favourable assessment and an improvement in productivity after the implementation of Scrum.

	Improve-ment	Impact on productivity	Range of score in both dimensions
No of interruptions	1.4	2.0	0 - 4
Endless development cycles	2.8	2.8	1 - 5
Repetition of mistakes	1.1	1.4	-1 - 4
Compliance with deadlines	2.9	2.1	0 - 5
Bug fixing time	0.5	1.7	-2 - 3
No of uninterrupted development hours	0.8	1.3	0 - 3
Employee performance	3.5	4	3 - 4

Table 1: Mean Values of Scores for Scrum's impact on productivity

Number of Interruptions

The respondents largely agreed that the number of interruptions had been reduced, and that this had affected productivity in a positive direction. At the high end of the scale one developer argued as follows: "Well, mainly because I refer to my Scrum Master now, go through her, I say. I'll do it, but it must come from the right path. If it comes from the right direction, then it is not so much an interruption, it's just a part of Scrum."

This respondent also scored the impact on productivity high. He underlined the way Scrum worked and emphasized that the number of interruptions had not necessarily decreased, but as the process had changed, he perceived the interruptions no longer as disturbances.

All other respondents signalled a moderate improvement. There was broad consensus that they still were upset in their everyday routines when interruptions occurred, however, they felt that the number of interruptions was less than before. The following quote supported this; respondent A said: "Yes I [a developer] would say +2, and that's based on that there are disruptions; although we try to avoid them, they are there nevertheless. But when I think back to the previous projects I have been involved in I think there have been even more disturbances (...)."

In slight disagreement another developer answered: "So, I would say 0, I think that I have neither got more nor less interruptions (...)."

Both respondents still decided to give the impact on productivity the same score they had given to the first dimension. The general higher score for impact was due to the individual respondents' opinion that the benefits of fewer disturbances had lifted productivity significantly.

Endless Development Cycles

The respondents generally agreed that the introduction of Scrum had resulted in that they were not to the same degree than before in endless development cycles. One respondent had an interesting position on this: "Yes, because I think what, among others, happened in the our tradition approach was that we developed for two months and then we found out that it was too large and complex; we did not recognize that we had bitten off more than we could chew. So there was too much work in it, we had to squash it and that happened many times, the consequence being, it had to be unpacked six months later again, or when we had the necessary resources, and then we had to start again from the very beginning with much of what had been done earlier."

This statement provided evidence that the case unit had previously experienced projects that had been developed for a number of months, only to be shelved and then started all over again. This was further supported by another developer: "(...) Before we got Scrum, it happened many times that an entire delivery was actually thrown away (...) now we think much about that it [a sprint or sprint goal] should be clear in itself, and that the next one should also be clear in itself (...)."

This respondent rated the improvement and impact as significant. Both accounts point to that Scrum has minimized the risk that the case unit did unnecessary development work. This had happened because, according to the respondents, the focus on delivery of functional software had increased and because the development process now was iterative and took place in short cycles.

Another developer was however slightly less positive, because, while he thought that there had been an improvement, he argued that there had not been any designated endless development cycles before the implementation of Scrum. He said: "No, not really, but that has something to do with the fact that we have always been under a bit of time pressure. So, this endlessness where you sit and fiddle with things, it has been non-existent. We have a little history that we may have focused on providing 80% solutions. Because it was more important that we came out with new functionality than that it was 100% correct. So in that way we did not produce these endless tracks."

This perception, however, was unique among the respondents, but pointed to another issue, namely the delivery of less complete solutions.

Repetition of Mistakes

With regard to this indicator there was a noticeable difference in how respondents with greater responsibilities replied compared to the other respondents. We also asked respondents directly to decide

whether the indictor's influence was due to the Scrum processes' focus on self-organization and retrospectives. One manager, who had rated both dimensions rather high, explained why he felt that the tendency to repeat the same mistake had been reduced after the introduction of Scrum: "Yes. I think so. Well, I think due to the fact that each individual has more responsibility one just does not make the same mistakes. That's what I think. It has nothing to do with retrospectives, the assessments one makes, but more with the fact that the responsibility lies with the individual. I think more that's what makes that a person will remember next time, here, we have to maybe do something else."

According to this respondent, the improvement had solely come about through the increased personal responsibility, i.e. through self-organization and not through learning in retrospectives. An explanation how the case unit used retrospectives was given by a developer, who saw no improvement nor any change in impact: "There is no difference because we do not use retrospectives to talk about what has been developed. De facto we use them to discuss the method, what has been good and whether there have been more or less disruptions or, if yes, whether we had a problem with that (...)."

A developer, who saw a slightly negative impact on productivity although he meant that the repetitions of mistakes had been reduced, explained his choice with this argument: "Well, it's because before we sat in professional groups. Now a .net developer, a main framer, a tester and an analyst sit together. And we have a good team, that's not it. But there's another dynamic, there is."

Overall, the result of this indicator shows that the respondents felt an improvement, which had a positive impact. The examination of this indicator also identified that the case unit only discussed the Scrum method in their retrospective meetings, but not the developed product.

Compliance with Deadlines

Our respondents generally agreed that there had been a clear improvement in this particular area. One of the main reasons that respondents were so positive was that Scrum broke the development process down into smaller iterations and deliveries. Several respondents highlighted this. For example, one developer stated: "Yes, it is because it is a shorter and more manageable process, it is like that. Well I would say it has become easier to meet the deadlines for when we should deliver."

Another developer added: "Because you have a well-defined task, you have your notes, you follow up on. So that's completely mastered. You have this 14-day sprint to do it (...)."

Another argument for why the impact of deadlines had increased came from a manager, who underlined that it not just had become easier to meet deadlines due to the decomposition and prioritization of tasks, which had had a great effect on productivity. He attributed the increase also to the fact that it had become easier to handle those elements and components that could be cut off at the end of the development process without much loss of functionality. This became possible because Scrum prioritizes and structures its processes different than the waterfall model did. He said: "Well, if you take the components of a car in the right order. If you wait to mount the wheels last, you fail to prioritize. It's the antenna, which should be mounted at the end, which we can leave out if we do not reach it (...)."

It was, however, not everyone, who shared the positive majority attitude; one developer saw no change, although he felt that the work had become easier to handle; he explained: "(...) Now, it's hard in a different way, because you have a sprint, so you have a deadline of 14 days instead of a deadline of 3 months. Plus, now you make minor things, where before you after all sometimes built these big chunks. Now it's more manageable. So on the whole I think it's the same. It's just so hard to achieve it, as it was before. "

This person stood alone with his opinion, thus it did not substantially affect the overall positive picture.

Bug Fixing Time

The respondents were unanimous in assessing this indicator, with the exception of one developer, who chose to evaluate the first dimension quite negative (-2), but its impact as very positive (3). He argued although more time was spent on bug fixes than in the past - an observation, which he interestingly enough did not refer to himself - it now happened at an earlier time in the process, which had a positive effect: "(...) It may not be me personally, who spends so much more time, but I've seen developers spent more time on it, and it's probably a -2 ... Well, we spend much time on bug fixes, but we do it at a better point in time now than we did, because, I feel, we used to spend much time on bugs that were registered quite late by customers; we now simply find more errors as a consequence of our Scrum processes (...)."

There was a broad agreement that the restructuring of the bug fixing and error correction process with short iterations and testers directly on the team had improved and that this had a positive impact on productivity. One developer gave an interesting explanation for this; he felt that productivity had increased, because the individual developers were now spared of being repeatedly and often with a delay reminded of their own errors. "(...) that way one can say that it [the new process] also helps to protect ourselves more. That we can say. The team is protected from mistakes they might have made themselves earlier in the development process."

Several respondents stated that it had made a big difference that there was now dedicated time specifically for bug fixes, and that it happened earlier in the process. Moreover, it was also mentioned that the interaction between testers and developers within the small teams did that both parties were given an insight into the work of the other - resulting in fewer errors, and ultimately less time spent on bug fixes.

Number of Uninterrupted Development Hours

There was much agreement about this indicator across all respondents. The slight improvement was explained with that there were more meetings now, but that the meetings had become better to support the teams in handling challenges. These led to higher scores for the indicator's impact on productivity. The respondents stated that they were now working more efficient because problems were solved within the teams and because they had better knowledge about the tasks. This was best described by two developers, who both stressed the positive side of performing tasks in a team:

"This is because many of the things we now do in the team, so there are not so many of those big formal review meetings. And earlier it took a long time, both the test case writing by the developers and analysts, but this now is done inside and across the team."

"Simply across the table, a ping pong that helps us to avoid endless, stupid time-wasting analysis and dead ends, so we quickly can resolve any issues (...)."

A third developer emphasised the increase of knowledge, which he related to the fact that there now were new tasks beyond coding, which were considered as development. Although this was novel and a challenge for him, as he had a prior focus on programming, he perceived this change not as necessarily negative and an interruption of his development work: "No, I do not think that, sitting down and analysing actually has the positive effect that one gets to know the tasks much better."

Still he was very conservative in his assessment of the achieved improvement and its impact as he scored both dimensions with 0, interestingly indicating that he did not perceive a significant change of this indicator.

Employee Performance

While all other indicators of productivity were primarily assessed from the developers' perspective this one was based on the opinion of the four respondents in leadership roles. They provided a very positive assessment. One of them explained this by referring to their negative experiences and what had been bad before the implementation of Scrum and the prevailing perception about those involved in the development process: "(...) the waterfall model also represents a kind of hierarchy. So, those on the top [at the very beginning of a project] are the best and the coolest, and at the bottom we find the testers, those, who are low skilled, have little education or are not educated at all. So it was not just the waterfall model, which allowed the analyst to possibly spend too much time, and then they delivered too late down to (...) the developers, who passed it on even later to the testers. (...)"

The respondent clearly felt the waterfall model put a disadvantage on those, who worked towards the end of a development process. All time overruns happened at the expense of the testers, who thus could not test sufficiently, which eventually led to rather erroneous software put into production. Given that this no longer happened as Scrum made sure that there was knowledge and understanding across the different professional disciplines, the respondent believed that the overall performance had increased. Another manager fully agreed that it had increased, but had slightly different reasons: "(...) Because now everyone has to make visible what he or she has been doing every day, that makes that one cannot so much hide behind a task (...)."

This manager argued that the daily Scrum meetings raised the individual employees' performance, since there was greater pressure to either deliver results or to demonstrate what challenges one had run into. Such a result-oriented environment could well have had a negative effect on the unit's culture and its ambience, but this was neither stated by any of the leaders nor by the developers; in contrast a recently performed survey in the organization confirmed these results.

SUMMARY OF RESULTS AND DISCUSSION OF FINDINGS

As mentioned earlier the investigation of Scrum's impact on productivity in information systems and software development was part of a larger study, which both developed and applied a comprehensive framework consisting of seven related concepts. Although a presentation of the overall result would give a much more comprehensive portrait of the method's impact we have here focused on one of the keystone concepts mostly due to page limitations. This still provides some valuable insights and where necessary we will relate to the other concepts. As a starting point for our subsequent discussion we summarize the results of our analysis concerning Scrum's impact on productivity in the case unit as follows:

We found that the decrease in the number of interruptions was limited, but it had led to a significant, perceived impact on productivity. In addition, the changed process with interruptions now coming from an authorized person and thus being less perceived as disturbances was appreciated.

Prioritizing new functionality higher than error-free deliveries had been the organization's strategy to avoid endless development cycles. Nevertheless these were experienced by the majority of the respondents. The increased focus on delivering functional software in defined, short iterations has prevented endless development and has resulted in more productivity, however not on the expense of product quality. Scrum's impact on quality and the interrelation of productivity and quality in the case organization are beyond the scope of this paper, but are documented in Johansen & Uldahl (2012).

With regard to the repetition of mistakes there was also positive development. Primarily the respondents' explained this progress with Scrum's focus on self-organization and not that much with the practice of retrospectives, which are the method's explicit mechanism for the identification of weaknesses and subsequent process and product improvements.

The respondents felt that Scrum's decomposition and prioritization of tasks had positively changed the compliance to deadlines and had had a positive impact on productivity in general.

Bug fixing time was the area with the least perceived improvement compared to the other indicators. Despite low average ratings, respondents expressed that although the actual time spent had not decreased, bug fixing now happened at a much better and appropriate time in the process. Thus, its impact on productivity was assessed significantly higher.

The perception of the number of uninterrupted, continuous development hours had also only seen a very modest increase. The respondents reasoned that the use of Scrum had led to more meetings than in the past, which led to interruptions in the continuity of their work. The frequent meetings resulted, however, in a better understanding of the tasks. This was appreciated by the respondents as having a positive impact on their productivity as they thought they now both worked more efficiently and tackled unforeseen challenges much better.

The managers among the respondents assessed that the employees' performance had increased significantly. They provided two different arguments for this. First Scrum's emphasis on visibility and transparency, which we had identified as a separate concept for investigating Scrum's impact on information systems and software development, made it compulsory for developers to publically present their work and take a position with regard to any challenges they had encountered. As a consequence they put more focus on the execution of their tasks. The other reason was related to the avoidance of project overruns. In the past overruns had always been passed through the chain of development tasks, with the results that the developers or even more so the testers became time-pressured and could not do their job properly. The shorter iterations carried out by a multidisciplinary team avoided this effect and resulted in overall better performance.

These favorable results are in line with the results for the other concepts and their indicators, which with the exception of customer satisfaction were all very positive (Johansen & Uldahl 2012). As with all qualitative studies of this kind we of course have to take the danger of positive bias and a respondents' tendency of reporting future expectations rather than stating actual perceptions into account.

On this background, we now compare our empirical data first with the literature on agile information systems and software development and in particular the identified writings about Scrum. According to these sources, there are a number of areas that impact on productivity, these being: sprints, a focus on functional software, retrospectives, self-organization, the product backlog and the daily scrum meetings.

The introduction of an iterative sprint development process (Schwaber & Beedle 2002) plays a central role in the use of Scrum. In the case unit, sprints had made it easier to comply to deadlines as tasks were now decomposed in smaller manageable items with clear definitions, which allowed for their easier handling and execution. These results are confirmed in empirical work reported by among others Augstine et al. (2005), Vidgen and Wang (2006) as well as Wang and Vidgen (2007).

Scrum's increased focus on iterative delivery of functional software should according to the literature increase productivity while avoiding falling into endless development cycles in an attempt to develop the 'perfect piece' of software (Moore et al. 2007). This was the effect Scrum had on the case unit, which thus was an area where the method lived up to the expectation.

Retrospectives are intended to increase the productivity among others as a result of the project participants' learning from their own and others' mistakes, so that errors and faults are not repeated in the next sprint or iteration. Retrospectives should address both, the overall application of the method, its processes and practices, but also the more specific experience in the daily development work and its relation to the resulting product (Schwaber 2004). The latter turned out to be an area the case unit did

not focus on and thus did not benefit from in their daily work. This prioritization of topics discussed during retrospectives can be explained with the case unit's early stage of utilizing Scrum and their lack of experience with regular retrospectives. In the case organization this area should therefore get further attention with an increased focus on Scrum's practices to support learning.

Self-organization in a Scrum team has among others the objective to protect and relieve individual team members from certain tasks and create an environment where they are not constantly disturbed in their work. In a successfully self-organized team, everyone has insights into the other team members' tasks, while at the same time a Scrum master is clearly identified and appointed (Schwaber & Beedle 2002). This means that when there is a need for input from a specific team member, the other team members are not unnecessarily disturbed, as the tasks have been clearly defined, broken down and distributed. If in doubt, the Scrum master is available to facilitate or solve the problem. At the case unit this had not yet been fully achieved, which meant that employees were still interrupted and disturbed in their work and further efforts will be needed to progress. However, one of the benefits of the Scrum master role had been achieved already, since the respondents expressed that the interruptions now came from the right person.

In the literature the avoidance of repeating errors is ascribed to retrospectives. As discussed above in the case unit retrospectives had not yet been applied to their full potential, yet the perception of the respondents had been that the repetition of errors had drastically decreased. This was attributed to the influence that self-organization had. As a consequence of the increased individual developer's responsibility now, team members had become more mindful not to repeat the same mistakes. Individual and collective mindfulness have been reported as characteristics of agile development independently of a particular method or agile practice (Matook & Kautz 2008). This supports that the lack of exploiting retrospectives in the case organization has been compensated by self-organization and mindfulness to lead to a positive outcome with regard to avoiding the repetition of mistakes.

In the case organization the introduction of a product backlog had primarily an effect on compliance with deadlines. As the work was now broken down to single items, there were ongoing opportunities to check whether the agreed schedule was met. Additionally, there was now the possibility to prioritize and plan the order of executing the items in an appropriate manner, which according to the literature (see e.g. Schwaber & Beedle 2002) further increases the overall productivity. The introduction of a product backlog at the case unit had affected both of these areas positively, and thus the overall productivity. Product backlogs can also be used to plan a specific test, debugging and error correction period in form of a dedicated item for these tasks (Schwaber 2004). In the case unit this did not lead directly to a reduction of the time spent on bug fixing, but it had resulted in bug fixing happening at a more appropriate point in time, which ultimately had had an impact on productivity.

Finally, daily Scrum meetings, have among others the objective to create visibility in a Scrum team. This helps that everyone in a team gains insight into what the others are working on and at the same time it makes it difficult for employees to conceal modest work efforts, since they publically have to communicate and document their results (Schwaber & Beedle 2002). The latter had a substantial impact on employee performance in the case unit - and as such affected productivity positively as openly explaining why as task took longer than expected, had the psychological effect that it deprived the employees of the opportunity to hide behind a task longer than necessary.

In addition, the meetings had both a positive and negative impact on the number of uninterrupted development hours. The increased number of meetings had reduced the amount of uninterrupted development hours. This was outweighed, however, by the fact that the meetings created better visibility, oversight and knowledge. This allowed employees to tackle unforeseen challenges better, which had a positive effect on productivity, since waste time was avoided.

Our overall positive assessment of Scrum on the productivity of information systems and software development confirms empirically the expectations and claims, which are made in many of the conceptual and non-academic writings we had identified in our literature review. It also fills a gap in the area of empirical studies of agile software development (Dybå & Dingsøyr 2008). In the absence of quantitative data and with no possibility to make direct measurements and collect such data throughout the project it is however built on subjective perceptions.

Nonetheless, on a more theoretical level our study can be related to complex adaptive systems (CAS) theory to find support for the increase of productivity as one outcome of Scrum. CAS theory underpins agile information systems and software development methods (Highsmith 2000) such as Scrum and the case unit appears to be rather successful after its transition to Scrum. On this background the above results can be linked to CAS concepts and principles. If ISD, in our case agile development supported by Scrum, is understood as CAS, certain characteristics of the process are recognized to facilitate good performance and thus productivity, while others inhibit it (Meso & Jain 2006; Kautz 2012).

A number of concepts are frequently used when discussing CAS. All these core concepts are intertwined and mutually reinforcing. Within the area of ISD Vidgen and Wang (2009) as well as Kautz (2012) have summarized and put them forward as follows: Interconnected autonomous agents are able to independently determine what action to take, given their perception of their environment; yet, they collectively or individually are responsive to change around them, but not overwhelmed by the generated information flow. Self-organization is the capacity of these agents to evolve into an optimal organized form, which results from their interaction in a disciplined manner within locally defined and followed rules. Co-evolution relates to the fact that a complex adaptive system and/or its parts alter their structures and behaviours in response to their internal interactions and to the interaction with other CAS where adaptation by one system affects the other systems, which leads to reciprocal change where the systems evolve individually, but concertedly. Time pacing indicates that a complex adaptive system creates an internal rhythm that drives the momentum of change, which is triggered by the passage of time rather than the occurrence of events; this stops them from changing too often or too quickly. Poise at the edge of time conceptualizes a complex adaptive system's attribute of simultaneously being rooted in the present, yet being aware of the future and its balance of engaging exploitation of existing resources and capabilities to ensure current viability with engagement of enough exploration of new opportunities to ensure future viability. Poise at the edge of chaos describes the ability of a complex adaptive system to be at the same time stable and unstable; this is the place not only for experimentation and novelty to appear, but also for sufficient structures to avoid disintegration; CAS that are driven to the edge of chaos out-compete those that are not. The above analysis has provided examples of interacting interconnected autonomous agents, such as the involved developers and testers, their self-organization as individuals and as project teams, their co-evolution through knowledge sharing and learning from each other, as well as for time pacing in the short iterative development cycles, and for poise at the edge of time and chaos, for instance with regard to compliance to deadlines, bug fixing time and uninterrupted development hours, which thus empirically and theoretically lend support to the identified perceived positive impact of Scrum on performance and productivity of information systems and software development in our case setting.

CONCLUSION

While the usual disclaimers for the shortcomings of qualitative research also apply for our study, our work contributes to the body of knowledge in information systems development with an empirical investigation that demonstrates the positive impact of the agile development and project management method Scrum on information systems and software development productivity and it provides a useful operationalization of the concept through seven indicators. Despite the fact that the case unit had challenges with the use of Scrum, the indicators identified the areas where the company had managed

to exploit the potential of Scrum and its practices with regard to increasing productivity. Through the analysis we found an interesting area where the case unit differed from the Scrum literature's recommendations. The case unit's handling of retrospective meetings only reflected the actual process and method, but not the developed product. This put the unit at the risk of missing out on any knowledge, which could contribute positively to the future iterations and development projects. Therefore future research should further investigate the relationship between team learning and interaction of autonomous interconnected team members in retrospectives and how productivity supported through Scrum stems from learning.

Although several authors underline the importance of an open organizational culture for agile development (Cockburn 2001; Highsmith 2002; Nerur et al. 2005; Robinson & Sharp 2005; Kautz et al. 2009) and argue that an innovative and open organizational culture is necessary to develop software according to agile principles we decided to disregard the concept as such as we assumed that the culture, its elements, the basic assumptions held by all members of that culture, their values and beliefs, and their artefacts and creations (Schein 2004) and the cultural changes as a result of an implementation of Scrum would have an impact and become visible through the indicators. In other words, for culture as a broad concept we thought it would make more sense to be implicitly investigated through the productivity indicators. In hindsight the relationship between culture and productivity in the use of agile methods such as Scrum does however also merit a thorough investigation through future research on its own.

REFERENCES

Andersen, I. 2006. *The apparent Reality (in Danish).* "Samfundslitteratur" Publisher. Frederiksberg, Denmark.

Augustine, S., Payne, B., Sencindiver, F., and Woodcock, S. 2005. "Agile project management: steering from the edges," *Communications of the ACM* (48:12), pp. 85-89.

Appelo, J. 2010. *Management 3.0 - Leading agile Developers, Developing agile Leaders*. Addison-Wesley. Crawfordsville, Indiana, USA.

Cockburn, A. 2001. *Agile Software Development*. Addison-Wesley. Boston, MA, USA.

Dybå, T., and Dingsøyr, T. 2008. "Empirical studies of agile software development: A systematic review," *Information and Software Technology* (50:9-10), pp 833-859.

Guang-Yong, H., 2011. "Study and Practice of Import Scrum Agile Software Development," in *Proceedings of the 3rd International IEEE Conference on Communication Software and Networks (ICCSN)*, Nanjing, Kina.

Highsmith, J. 2000. Adaptive *Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House Publishing. New York, USA.

Highsmith, J. 2002. *Agile Software Development Ecosystems*. Addison-Wesley. Boston, MA, USA.

Johansen, T., and Uldahl, A. 2012. *Measuring the Impact of the Implementation of the Project Management Method Scrum (in Danish).* MSc Thesis. Copenhagen Business School, Copenhagen, Denmark.

Kautz, Pedersen, C. F., and Monrad, O. 2009. "Cultures of Agility – Agile Software Development in Practice," in *Proceedings of the 20th Australasian Conference on Information Systems.* Melbourne, Australia.

Kautz, K. 2012. "Beyond Simple Classifications: Contemporary Information Systems Development Projects as Complex Adaptive Systems", in *Proceedings of 33rd International Conference on Information Systems.* Orlando, FL, USA.

Mahnic,V., and Zabkar, N. 2008. "Using COBIT Indicators for Measuring Scrum-based Software Development," WSEAS *Transactions on Computers* (7:10), pp. 1605-1617.

Mahnic, V., and Vrana, I. 2007. "*Using stakeholder-driven process performance measurement for monitoring the performance of a Scrum-based software development process*," *Electrotechnical Review* (74:5), Ljubljana, Slovenia, pp. 241-247.

Matook, S., and Kautz, K. 2008. "Mindfulness and Agile Software Development," in *Proceedings of the 19th Australasian Conference on Information Systems*. Christchurch, NZ, pp. 638-647.

Meso, P. and Jain, R. 2006. "Agile Software Development: Adaptive Systems Principles and Best Practices," *Information Systems Management (23:3)*, pp.19-30.

Moe, N. & Dingsøyr, T., 2008. "Scrum and Team Effectiveness: Theory and Practice," in *Proceeding of the Conference on Agile Processes in Software Engineering and Extreme Programming*. Springer. Berlin, Germany, pp.11-20

Moore, R., Reff, K., Graham, J. and Hackerson, B. 2007. "Scrum at a Fortune 500 Manufacturing Company", in Proceedings *of the Agile Conference (AGILE)*. St. Paul, Minnesota, USA.

Nerur, S., Mahapatra, R., and Mangalaraj, G. 2005. "Challenges of Migrating to Agile Methodologies," *Communications of the ACM* (58:5), pp73-78.

Robinson, H., and Sharp, H. 2005. "Organizational culture and XP: three case studies," *Agile Development Conference*, IEEE Computer Society. Denver, CO, USA. pp 49-58.

Schein, E. 2004. *Organizational Culture and Leadership*. 3rd Edition. Wiley & Sons. San Francisco, CA. USA.

Schwaber, K., 2004. *Agile Project Management with Scrum*. Microsoft Press. Redmond, Washington, USA

Schwaber, K., and Beedle, M., 2002. *Agile Software Development with Scrum*. Prentice Hall. Upper Saddle River, USA.

Sutherland, J., and Altman, I. 2010. "Organizational Transformation with Scrum: How a Venture Capital Group Gets Twice as Much Done with Half the Work", in *Proceedings of the 43rd Hawaii International Conference on System Sciences*. Kauai, Hawaii USA.

Vidgen, R., and Wang, X. 2006. "Organizing for Agility: A Complex Adaptive Systems Perspective on Agile Software Development Process," in *Proceedings of the 14th European Conference on Information Systems*. Göteborg, Sweden. Ljunberg, J., and Andersson, M. (eds.), pp. 1316-1327.

Vidgen, R., and Wang, X. 2009. "Coevolving Systems and the Organization of Agile Software Development," *Information Systems Research (20:3*), pp. 355-376.

Wang, X., and Vidgen, R. 2007. "Order and Chaos in Software Development: A comparison of two software development teams in a major IT company," in *Proceedings of the 16th European Conference on Information Systems*. Winter, R., et al. (eds.). St. Gallen, Switzerland, pp. 807-818.

Webster, J., and Watson, R.T. 2002. "Analyzing the Past to Prepare for the Future: Writing a Literature Review," *MIS Quarterly* (26:2), pp 13-23.

An earlier version of this paper was presented at the Australasian Conference on Information Systems (ACIS) 2013 in Melbourne, Australia.