5-9-2012

HOW SUSTAINABLE ARE AGILE METHODOLOGIES? ACCEPTANCE FACTORS AND DEVELOPER PERCEPTIONS IN SCRUM PROJECTS

Sven Overhage
University of Augsburg

Sebastian Schlauderer
University of Augsburg

HOW SUSTAINABLE ARE AGILE METHODOLOGIES? ACCEPTANCE FACTORS AND DEVELOPER PERCEPTIONS IN SCRUM PROJECTS

Overhage, Sven, University of Augsburg, Universitätsstraße 16, 86159 Augsburg, Germany, sven.overhage@wiwi.uni-augsburg.de

Schlauderer, Sebastian, University of Augsburg, Universitätsstraße 16, 86159 Augsburg, Germany, sebastian.schlauderer@wiwi.uni-augsburg.de

Abstract

The introduction of agile methodologies such as Scrum considerably changes the working habits of developers. To ensure their successful dissemination, it is therefore particularly important that developers assimilate and remain committed to agile principles. In this paper, we examine the long-term acceptance of Scrum and present the results of a study conducted at a world-wide leading insurance company that began transitioning to Scrum in 2007. Taking the Diffusion of Innovations theory as a lens for analysis, we identify several acceptance factors of Scrum and hypothesize how they are perceived in comparison to traditional methodologies. We evaluate our hypotheses using a multi-method research approach that combines analyses of quantitative and qualitative field data. The results suggest that several factors of Scrum are perceived as relative advantages or as more compatible to the way developers prefer to work. Factors that characterize the complexity of Scrum are identified as potential barriers to acceptance, however.

Keywords: Agile methodologies, Scrum, Acceptance Factors, Diffusion of Innovations Theory.

1 Introduction

Volatile business environments and ever more rapidly changing requirements pose significant challenges for information systems development (ISD) teams. To better cope with these challenges, agile methodologies such as Scrum, Feature Driven Development, or Extreme Programming (XP) have been proposed. They oppose the established practice of traditional methodologies as they promote close collaboration with customers, self-organization, communication, and responsiveness to change over codified processes and extensive planning (Nerur and Balijepally, 2007). Agile methodologies are rapidly becoming popular in practice and both anecdotal evidence as well as initial studies indicate that they can indeed enhance the flexibility of development teams (Dyba and Dingsoyr, 2008).

As *software process innovations* (SPI; Conboy et al., 2007), agile methodologies considerably change the working habits of developers, however. Since they rely on flat hierarchies with self-organizing teams, the developers' commitment becomes a critical success factor. The agile manifesto therefore advises to build "projects around motivated individuals" (Beck et al., 2001). To ensure a successful dissemination of agile methodologies, it is particularly important that developers do not only adopt agile methodologies initially, but that they assimilate agile principles and stay committed in the long term. The diffusion of an innovation is a multi-stage process comprising its initiation, adoption, adaptation, acceptance, routinization, and infusion (Gallivan, 2001). Most studies on agile methodologies concentrate on the adoption and adaptation stages, however. As later assimilation stages have not been sufficiently studied yet, there remains a "need for a better understanding of agile methods beyond the adoption stage" (Abrahamsson et al., 2009). Especially "an understanding of the factors that facilitate or hinder their acceptance and use in organizations would be invaluable" (Mangalaraj et al., 2009).

With the study presented in this paper, we contribute to closing this literature gap. Building on the body of knowledge about SPI and taking the *Diffusion of Innovations Theory* (Rogers, 1995) as a lens for analysis, we identify specific acceptance factors for agile methodologies. Particularly, we address the following research questions: *Which acceptance factors are characteristic for agile methodologies? Compared to traditional methodologies, are they perceived as advantages or drawbacks by developers in the long term?* To address the second question, we use a multi-method research approach that combines quantitative with qualitative methods. Doing so allows us to deliver both statistical evidence and rich explanations to cross-verify the results (Kaplan and Duchon, 1988). The empirical data was collected at a world-wide leading German insurance company that began introducing Scrum in 2007. When we conducted our study in 2010, the company was employing Scrum already in 23 in-house development projects. The setting provided access to a large number of developers in order to study the perception of agile methodologies at the individual level. Thereby, we have chosen Scrum as a specific study subject for several reasons. Most importantly, Scrum is by far the most widespread agile methodology in industry today (West and Grant, 2010). Nevertheless, there exist comparatively few studies of Scrum as most instead evaluate XP (Dyba and Dingsoyr, 2008). While XP focuses on supporting technical aspects of development, Scrum concentrates on the planning and tracking of agile projects as a whole, however. Management-oriented approaches such as Scrum are hence "clearly the most under-researched compared to their popularity in industry" (Dyba and Dingsoyr, 2008).

Next, we discuss related work to highlight the literature gap. In section 3, we discuss the theoretical background. Against this background, we identify acceptance factors and hypothesize about their perception in section 4. Section 5 contains the empirical research methodology. In sections 6 and 7, we discuss the quantitative and qualitative results. Finally, we describe limitations and implications.

2 Related Work

In order to evaluate the current body of knowledge about the perception of agile methodologies, we conducted a literature review based on the recommendations of Webster and Watson (2002). In a first

step, we analyzed leading journals and conferences for relevant literature. Afterwards, we reviewed the references of identified literature to identify prior approaches. Especially the literature studies of Petersen and Wohlin (2009) and Dyba and Dingsoyr (2008) provided a rich basis of citations.

Researchers attest that the current body of literature about agile methodologies has several shortcomings. Especially, the lack of rigorously conducted empirical studies is criticized (Abrahamsson et al., 2009). Dyba and Dingsoyr (2008) even state that the strength of empirical evidence regarding the benefits and limitations of agile methodologies is "very low". Addressing this call for more rigorously conducted studies, several newer studies evaluate the advantages of agile methodologies in terms of costs or product quality (Conboy, 2009, Lee and Xia, 2010). In contrast, we focus on evaluating social factors and in particular how Scrum is perceived by developers. As other agile methodologies, Scrum considerably changes the working habits of developers. Its success hence depends on the continuous commitment and motivation of the developers to support agile principles (Beck et al., 2001).

However, only very few empirical studies examine how developers perceive agile methodologies. Most studies concerning social acceptance factors furthermore focus on XP (Dyba and Dingsoyr, 2008). Bahli and Zeid evaluate the adoption of XP using the Technology Acceptance Model (2005). Other studies examine the satisfaction of employees in XP projects (Mannaro et al., 2004) or the acceptance of specific XP practices (Mangalaraj et al., 2009, Ilieva et al., 2004). While Scrum and XP arguably share similarities, they have a different focus. Scrum addresses management aspects whereas XP focuses on technical implementation steps (Fitzgerald et al., 2006). As opposed to Scrum, XP for instance describes best practices for the implementation of software (e.g. pair programming). Consequently, the results of studies concerning XP cannot easily be transferred to Scrum. Yet, the perception of specific characteristics of Scrum seldom was in the focus of research. Dyba and Dingsoyr (2008) hence conclude that Scrum is most under-researched compared to its popularity. Among the few studies available, Mann and Maurer (2005) report developer perceptions that were observed during the introduction of Scrum at a software company. Yet, it is not clear in which assimilation stage the company exactly was, how the data was collected, and how it was analyzed.

Study	Method	Subjects	Assimilation stage	# Subjects
Bahli and Zeid (2005)	XP	Developers, Managers, Customers	Adoption	14
Mannaro et al. (2004)	XP	Developers, Managers, Researchers	Not defined	122
Mangalaraj et al. (2009)	XP	Developers, Tester, Architects	Acceptance	13
Ilieva et al. (2004)	Customized (XP/PSP)	Developers, Customers	Not defined	4
Mann and Maurer	Scrum	Developers, Customers	Not defined	8
Friis et al. (2011)	Scrum	Developer, Managers	Not defined	55
Bonner er al. (2010)	Agile principles	Analysts, Developers, Managers	Not defined	479

Table 1. Empirical studies that include developer perceptions on agile development projects

In addition, only few studies examine the perception of agile methodologies beyond the adoption stage (Abrahamsson et al., 2009). Specifically regarding Scrum, we only found one such study. Friis et al. (2011) analyze the role of managers in Scrum projects three years after the methodology was introduced. However, they mainly highlight challenges from a manager perspective and mention developer perceptions only as an aside. Bonner et al. (2010) conducted an online survey to determine how development process agility as a general concept affects the participant's perception of development complexity, compatibility, and benefits. Yet, the percentage of participants who worked with agile methodologies like Scrum (2.1%) or XP (4.2%) was relatively small compared to that working with traditional methodologies (e.g. waterfall model 40.3%). Moreover, the authors only use generic measures of general acceptance theories. The results hence remain abstract. In particular, they provide no insights into specific factors that caused participants to perceive agile methodologies as a benefit or drawback. It is thus not clearly apprehended yet what exactly fosters or impedes the acceptance of agile methodologies. To provide more concrete insights and to complement the above-mentioned approaches (see Table 1), the study presented in this paper aims at identifying specific acceptance factors for Scrum. Thereby, we follow the advice to evaluate the use of agile methodologies beyond the adoption stage (Abrahamsson et al., 2009) by studying a company in which Scrum was already in use as a normal activity.

3 Theoretical Background

Agile software development methodologies in general and Scrum in particular set a polar opposite to traditional methodologies such as the waterfall model. While Scrum emphasizes flexibility, traditional approaches typically follow a rigorous process management. The latter enforce a predefined process model in which planning is based on a work breakdown structure with milestones and work packages. Since this enables organizations to plan the phases of a development project ahead, traditional methodologies are sometimes said to be repeatable, predictable, and to enable the optimization of processes (Boehm, 2002). Scrum instead was envisioned as a "management and control process that cuts through complexity to focus on building software that meets business needs" (Schwaber and Beedle, 2002). Assuming that requirements are likely to change during a development project, Scrum builds upon an empirical process control that manages projects from iteration to iteration. Instead of planning the complete project ahead, Scrum has three levels of planning: the Release Planning, the Sprint Planning, and the Daily Scrum. The Release Planning is thought to discuss only basic strategic aspects like the overall costs or functionality of a development project. Operational details are only planned for the next Sprint, i.e. a time box of one month or less (Schwaber and Sutherland, 2011). The Daily Scrum is a daily 15-minute meeting, in which team members discuss the current project state and new tasks.

Aspect / Method	Planning	Requirements	Documentation	Project controlling	Collaboration	Retrospectives
Scrum	Process managed from iteration to iteration.	Customers continuously discuss requirements with the team.	Explicit documentation not prescribed.	Various meetings to discuss progress. Burndown Charts show remaining tasks.	Teams with a flat hierarchy and a "servant-leader".	Meetings at the end of each Sprint to foster communication.
Traditional	Process planed in advance using a work breakdown structure.	Requirements are fixed at the beginning of the project.	Documentation is integral part of the development process.	Team members return a percentage of completion for milestones.	Project manager leads the team and assigns tasks to developers.	Lessons learned are usually discussed at the end of a project.

Table 2. Differences between Scrum and traditional methodologies

Besides the different planning principles, Scrum also establishes a different form of collaboration within the team as well as with customers. While traditional methodologies typically have a project manager who leads the team and assigns tasks to team members, Scrum development teams should organize their work self-responsibly. Scrum promotes a flat hierarchy by allowing team members to discuss the assignment of tasks and by providing them with a so called "servant-leader", i.e. the Scrum Master. Scrum Masters should only coach the team and ensure that Scrum is done right (Schwaber and Sutherland, 2011). Scrum furthermore emphasizes the importance of collaborating with customers and integrates them into the development process by giving them their own role, namely that of the Product Owner. In Scrum projects, Product Owners ought to evaluate working pieces of software at the end of each Sprint. This gives them the opportunity to continuously evaluate the development progress and to add or change requirements during the project. Compared to that, traditional methodologies handle the management of requirements much more rigidly. Traditionally, requirements are fixed at the beginning of a development project in a contract-like document and working pieces of software can only be inspected in late development stages. However, changing requirements or integrating new ones is a difficult task then, since most of the implementation is already finished. Actively involving customers into the development project also is an important aspect when it comes to controlling the project state, since this makes development projects more transparent. To further increase transparency, Scrum uses so-called Burndown Charts which show a daily updated summary of remaining tasks. In traditional methodologies, project controlling is done by inquiring a percentage of completion for milestone or status reports. However, the integration of customers into the development project also burdens both groups with additional effort as they have to actively drive the collaboration.

Another difference between Scrum and traditional methodologies regards the relevance of documentation. While Scrum values "working software over comprehensive documentation" (Beck et al., 2001), traditional methodologies emphasize the general importance of documentation and consider it as an

integral part of a project. In traditional projects, knowledge is hence explicitly written down whereas Scrum relies on the transfer of knowledge due to various meetings and the increased communication. On the one side, this reduces the effort developers in Scrum projects need to spend for writing an explicit documentation. On the other side, knowledge can easily get lost if team members leave or if meetings are not taken seriously. Finally, feedback mechanisms are handled differently. In traditional development projects, team members usually only discuss lessons learned at the end of a project. In contrast, Scrum gives developers the opportunity to obtain feedback right after the first Sprint. During Sprint Review and Sprint Retrospective meetings, team members discuss measures that turned out to be successful. While this establishes a continuous learning process for team members, their effort is again increased due to the additional meetings. Table 2 summarizes the major differences.

The discussion illustrates that Scrum differs from traditional development methodologies in several aspects. From a theoretical perspective, its innovative way to manage the development of information systems hence qualifies as a *software process innovation* (SPI, Conboy et al., 2007). An innovation thereby is broadly defined as an idea or practice that is perceived as new by adopters (Rogers, 1995). The assimilation of innovations is explained by innovation diffusion theories. To explain the *process* in which an innovation is assimilated by an individual, a group, or an organization, Gallivan (2001) introduces a model consisting of six stages. During the initiation stage, the overall suitability of the innovation for the adopting unit is determined. In the adoption stage, the decision to introduce the innovation is made. During the adaptation stage, the innovation is adjusted, installed, and members of the adopting unit are trained to use it. In the acceptance stage, members of the adopting unit are committing to use the innovation. Its usage is then encouraged as a normal activity during the routinization stage. In the infusion stage, the innovation is used comprehensively and in a sophisticated manner.

Theories such as the Diffusion of Innovations (DOI) Theory, the Technology Acceptance Model, or the Theory of Planned Behavior explain the *factors* that determine the assimilation of innovations. They have many commonalities as to the considered factors and have repeatedly been used to explain the assimilation of information systems, tools, and SPI (Mustonen-Ollila and Lyytinen, 2003, Riemenschneider et al., 2002). They can hence provide a theoretical fundament and starting point for the identification of specific acceptance factors for agile methodologies. In this paper, we build upon the DOI theory. It assumes that individuals have different degrees of willingness to assimilate an innovation (Rogers, 1995). Consequently, it is generally observed that the portion of individuals that assimilate an innovation is distributed over time. The DOI theory defines five perceived attributes of innovations as generic factors that affect the willingness to assimilate innovations (Rogers, 1995):

Relative advantage: the degree to which an innovation is perceived as better than the superseded idea.
Compatibility: the degree to which an innovation is perceived as consistent with actual needs.
Complexity: the degree to which an innovation is perceived as difficult to understand or use.
Trialability: the degree to which an innovation is perceived as easy to experiment with.
Observability: the degree to which an innovation is visible to others.

Except complexity, all factors are positively correlated to the rate of acceptance. Complexity is negatively correlated. Regarding the individual developers' acceptance and willingness to use a SPI within organizations, research has found trialability and observability to be non-significant (Mustonen-Ollila and Lyytinen, 2003). In this context, only relative advantage, compatibility, and complexity turned out to be significant determinants (Mustonen-Ollila and Lyytinen, 2003, Riemenschneider et al., 2002). We therefore consider the latter as conceptual basis to derive specific acceptance factors of Scrum.

4 Research Hypotheses

As shown before, Scrum organizes the development process in a fundamentally different way than traditional methodologies. Below, we analyze the differences to identify seven specific acceptance factors F1 - F7 for Scrum. To illustrate how these factors influence the developer acceptance, we classify them as instances of the perceived attributes of innovations (Figure 1). Moreover, we hypothesize how

Scrum developers perceive the factors in comparison to traditional methodologies: H1 - H3 claim that Scrum brings relative advantages, H4 - H5 postulate that it is more compatible to actual working practices and H6 - H7 posit a tradeoff between lower task complexity and higher demand for discipline.

For the developer, the relative advantage of a SPI is determined by its perceived ability to enhance the job performance (Riemenschneider et al., 2002). We suppose that reorganizing development projects according to the principles of Scrum will impact this criterion in several ways. Firstly, we expect this to affect the perceived *meeting of requirements* (F1). Other than in traditional projects, Scrum developers profit from constant customer feedback on the emerging product. They will hence better be able to evolve their product in coordination with the customers. Moreover, Scrum's empirical process control facilitates the adaptation to changing requirements (Fitzgerald et al., 2006). The developers will so better be able to incorporate any necessary changes during the project. Therefore, we suppose:

H1. Scrum developers perceive the meeting of requirements as better than in traditional projects.

In addition, we expect the introduction of Scrum to affect the perceived *time to market* (F2). As requirements are prioritized constantly in Scrum projects, developers will be able to realize mission-critical functionality first. Also, this functionality can be delivered earlier than in traditional projects because working software is created in every Sprint (Schwaber and Beedle, 2002). We hence posit:

H2. Scrum developers perceive the time to market as better than in traditional projects.

We suppose that the introduction of Scrum also impacts the perceived *learning effects* (F3). Compared to traditional projects, the frequent Sprint reviews and Retrospectives facilitate the advancement of development skills. As development tasks furthermore have to be taken over flexibly in Scrum projects (Schwaber and Beedle, 2002), the expertise of developers will be broadened. We thus conclude:

H3. Scrum developers perceive the learning effects as better than in traditional projects.

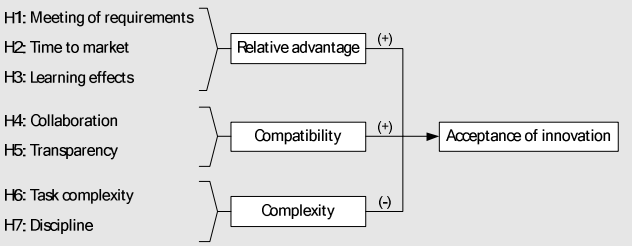

Figure 1. Acceptance factors (left) and DOI theory constructs (right)

The compatibility of a SPI is determined by its perceived conformance to the way developers prefer to perform their work (Riemenschneider et al., 2002). Specifically, we expect the introduction of Scrum to affect the perceived *collaboration* (F4) within the team. Developers prefer to work in close collaboration with their colleagues as this helps them utilizing the different skills in an ideal manner (Sawyer et al., 2008). Compared to traditional projects, in which developers are assigned to separate tasks, Scrum developers discuss project matters in daily meetings. The collaboration moreover profits from the established immediate communication in Scrum projects (Fitzgerald et al., 2006). We hence posit:

H4. Scrum developers perceive the collaboration as better than in traditional projects.

We expect the introduction of Scrum to also impact the perceived *transparency* (F5). Developers need to be aware of the overall status and the remaining tasks to identify themselves with the project and coordinate their work with others (Sawyer et al., 2008). In contrast to traditional projects, Scrum developers can review the project status anytime as it is publicly documented in Burndown Charts. They also receive frequent updates since the status is discussed in the Daily Scrum. Therefore, we suppose:

H5. Scrum developers perceive the transparency as better than in traditional projects.

The complexity of a SPI is determined by its perceived ease of use (Riemenschneider et al., 2002). We suppose that a transitioning to Scrum particularly affects the *task complexity* (F6). In contrast to

traditional projects, which are structured into milestones and complex, interdependent work packages, Scrum projects are organized in small, controllable iterations. As a consequence, both the size and the interdependency of development tasks are reduced (Schwaber and Beedle, 2002). We hence propose:

H6. Scrum developers perceive the task complexity to be lower than in traditional projects.

We expect the introduction of Scrum to also impact the perceived *discipline* (F7). Unlike in traditional projects, Scrum developers are not led by project managers who assign and control tasks. Instead, they have to organize their teams self-responsibly and in flat hierarchies. Consequently, they have to bring in a high degree of commitment and personal responsibility (Beck et al., 2001). To live up to the increased demand, they will need to show a higher amount of discipline. This is why we posit:

H7. Scrum developers perceive the discipline to be higher than in traditional projects.

Note that we did not study any effects on software quality as such effects are rather caused by methodologies that support the implementation process (e.g. XP). Also, we did not study effects of neglecting documentation as the company we examined had to document software to ensure its auditability.

5 Research Methodology

The hypotheses were analyzed using a multi-method approach that includes analyses of quantitative and qualitative field data. Such an approach has the advantage that it mitigates the weaknesses of each individual approach and supports a triangulation of the results. Especially, it can be used to gain statistical objectivity by analyzing quantitative data and to provide in-depth insights by examining qualitative data (Kaplan and Duchon, 1988, Lee and Xia, 2010). We proceeded in four steps: (i) we conducted a survey to collect quantitative data; (ii) we collected qualitative data by interviewing experts; (iii) we used the quantitative data to analyze our hypotheses; (iv) we used the qualitative data to cross-verify the results of the quantitative analysis and provide deeper insights (Kaplan and Duchon, 1988).

Both quantitative and qualitative data was collected at a world-wide leading insurance company which began gradually turning its in-house development to Scrum in 2007. Before Scrum was introduced, the developers worked with the V-model, a traditional methodology that builds upon the waterfall model. When we collected the data in 2010, the company already employed Scrum for a considerable amount of its ISD projects: 200 team members were using Scrum in 23 different projects. With respect to Gallivan's innovation assimilation model, the company was in the routinization stage, in which the usage of the innovation becomes a normal activity. Scrum was used regardless of the application domain or the project complexity. Scrum projects hence ranged from developing standard business software to mission-critical database platforms. Overall, the implementation of Scrum followed the textbook and encompassed all meetings and principles. Only the Sprint length varied between two and four weeks.

Quantitative data was collected by conducting an online survey using the company's intranet since the participants were assigned to different headquarters. All of the study participants worked as developers in Scrum projects and had worked with traditional methodologies before. As they had expertise in both methodologies, they were best suited to assess differences between them. We hence decided to ask comparative questions. For each factor F1 - F7 we wanted to know if it has changed positively or negatively after the transitioning to Scrum. For instance, we asked "Compared to traditional projects, how do you perceive the meeting of requirements in Scrum projects?". Responses were measured on a five-point Likert scale mostly with the following items: (1) much worse, (2) somewhat worse, (3) about the same, (4) somewhat better, (5) much better (Vagias, 2006). To measure the complexity, we used items ranging from much lower to much higher. Participants moreover were allowed to leave out answers if they felt uncertain. To analyze the data for statistical significance, we performed one-sample student's t-tests (Walczuch et al., 2000). T-tests were found to deliver valid results for five-point Likert scales regardless of the discreteness and the possibly non-normal distribution of their values (de Winter and Dodou, 2010). If the mean value of a question significantly deviated from the mid-

dle scale value, we considered the perception of a factor to have changed. Although our hypotheses suggest a one-tailed testing, we decided to conduct two-tailed tests since this provides stricter results.

Qualitative data was collected by interviewing five experts from the company. Generally, an expert is characterized as someone with privileged knowledge about the developer perceptions across the various projects. To enhance the reliability of the interview results, we decided to interview experts with different roles. All in all, we interviewed three Scrum Coaches, one Scrum Master, and one executive from the higher management. Both Scrum Coaches and Scrum Masters had frequent contact to the development teams since they either counseled or guided them during their development projects. The higher management executive helped strategically managing the company's transitioning to Scrum and hence maintained a close contact to development teams. We decided to conduct semi-structured face-to-face interviews since they provide a great breath of results and nevertheless follow an elementary guideline (Given, 2008). Our interview guideline consisted of two parts: in the first part, we gathered general information about the participants such as the experience they had with traditional and agile methodologies. In the second part, we asked for general advantages and disadvantages and then specifically inquired how participants perceived the factors considered in this study. As this was only an elementary guideline, further questions were added if necessary. To analyze the obtained data, the recorded interviews were transcribed to English language and independently coded by two researchers in order to identify consistencies. The factors depicted in Figure 1 served as a classification schema to thematically group the statements into "intellectual bins" (Miles and Huberman, 1999). Afterwards, coding results were compared and discrepancies were eliminated based on a discussion.

6 Quantitative Results

Overall, we received responses from 50 developers. As the participants were allowed not to give an answer to individual questions, the number of actual responses varied between 43 and 50 however. At first, participants were asked how long they had worked with traditional methodologies and with Scrum to accomplish their everyday activities. Moreover, they were required to rank their experience with both methodologies on a Likert-scale from (1) very little experience to (8) very much experience. Most of the participants had worked with traditional methodologies for more than five years and accordingly ranked their experience as 6.5 on average. With Scrum about one third had respectively worked for less than six months, six to twelve months, and more than one year. As Scrum was new to the participants when they started working with it in the company, they found their experience with it to be lower. On average, they ranked it as 4.4. The second part of the survey asked for their perceptions of factors F1 - F7. Table 3 depicts the results of the conducted statistical analysis.

Variable	N	Test			Summary statistics			
		t-value	p-value	Mean≠3?	Min.	Max.	Mean	Std. dev.
H1: Meeting of requirements	44	5,006	0,000***	✓	2	4	3,41	0,54
H2: Time to market	45	4,119	0,000***	✓	1	5	3,53	0,87
H3: Learning effects	48	6,615	0,000***	✓	2	5	3,75	0,79
H4: Collaboration	50	4,876	0,000***	✓	2	5	3,56	0,81
H5: Transparency	50	12,322	0,000***	✓	2	5	4,26	0,72
H6: Task complexity	43	1,360	0,181	✓	1	5	3,16	0,78
H7: Discipline	49	5,039	0,000***	✓	2	5	3,55	0,77

Legend: ***: 0,1% significance, **: 1% significance, *: 5% significance

Table 3. Test and summary statistics for the conducted survey

With respect to the *meeting of requirements*, the results support H1. 43 of 44 respondents found the meeting of requirements to be about the same or better. Only one developer perceived the meeting of requirements to be somewhat worse in Scrum projects. Regarding the *time to market*, the results support H2. 20 of 45 respondents perceived the time to market either to be somewhat or even much better. Only two judged the time to market to be somewhat or much lower. As for H3, the results are even more significant. 25 of 48 respondents stated that the *learning effects* in Scrum projects were somewhat better. Seven additional even found them to be much better. Only three respondents said that their learning effects were somewhat worse in Scrum projects. Consequently, the results support H3.

With respect to the *collaboration*, the results support H4. 22 of 50 respondents found the collaboration in their teams to be somewhat or much better after the introduction of Scrum. Only two stated that the collaboration is somewhat worse in their projects. Almost all respondents moreover found the *transparency* to have increased after the transitioning to Scrum. 44 of 50 respondents stated that the transparency is somewhat or even much better. Consequently, the results support H5.

However, the respondents viewed the overall complexity in Scrum projects rather negatively. Regarding the *task complexity*, developers reported ambivalent perceptions so that no judgment about H6 can be made. Seven respondents found the task complexity to be higher or even much higher. 14 developers instead found it to be lower or much lower, while seven felt unable to answer the question. Although Scrum was designed to cut through complexity (Schwaber and Beedle, 2002), developers in our study did not perceive the task complexity to have significantly lowered. As to the required *discipline*, the results support H7. 25 respondents felt the discipline to be higher or even much higher.

All in all, the developers perceive the introduction of Scrum to bring them relative advantages. Especially the meeting of requirements, the time to market and the learning effects are perceived to be better in Scrum projects. Compared with traditional methodologies, developers also perceive Scrum projects to be more compatible to the way developers prefer to work. In particular, they perceive the collaboration within the team and the transparency of project status to have changed for the better. However, developers seem to perceive the complexity of Scrum projects as higher than that of traditional projects. While they have mixed feelings regarding the task complexity, they perceive the required discipline to be higher in Scrum projects. To further identify if the perceptions of developers vary depending on how long they have been working with Scrum, we conducted an analysis of covariance (ANCOVA). Only the perceived time to market, the learning effects, the task complexity, and the collaboration were significantly correlated to the time Scrum was already used. The longer developers used Scrum, the more they felt that the collaboration within the team, the learning effects, and the time to market has changed for the better. However, the more experienced developers were, the more they also perceived that the task complexity has increased since Scrum was introduced.

7 Qualitative Results

The results from the first interview part show that the interviewed experts had profound experience with Scrum and traditional methodologies. They had worked with Scrum for 4.5 years and with traditional methodologies for 23.4 years on average. On a scale from (1) marginal to (4) profound knowledge, they ranked their expertise in Scrum with 3.6 and their knowledge of traditional methodologies with 3.4. The coded results of the second interview part provide details about the hypotheses H1 - H7.

The results confirm that the developers perceive the meeting of requirements as better in Scrum projects (H1): *"The developers appreciate that they do not work for nothing anymore as they often did with the waterfall model"* (Scrum Master). *"An important improvement for the developers is that the customer demands can be satisfied better"* (Scrum Coach 1). They appreciate both the continuous adjustment to changing requirements as well as the frequent customer feedback: *"The developers like the idea of taking over changes already during development and not having to work on change requests afterwards"* (Scrum Coach 1). *"Developers appreciate that they do not any longer have to work for a year just to realize that the results are not useful for the customer"* (Scrum Master).

The interview results also confirm that the developers perceive the time to market as better in Scrum projects (H2): *"The developers are happy that they achieve a good requirements covering and that it is already achieved at an early point during the project"* (Scrum Coach 3). *"The developers can provide results much faster now and for them this is a big advantage"* (Scrum Master). Specifically, they appreciate the continuous prioritization of requirements by the customer: *"Developers can now focus on what customers actually want and defer less relevant things. They still deliver 20 pieces of software in let's say a week, but 20 useful ones. For them, that is a major improvement"* (Scrum Coach 3).

The results corroborate that the developers perceive the learning effects as better in Scrum projects (H3). They acquire a deeper development knowledge due to the flexible takeover of tasks and the retrospective meetings: *"A frequently mentioned advantage is that everyone now takes over any task. On the one hand, this strengthens communication as colleagues exchange their know-how actively. On the other hand, everyone gets a broad expertise and can overlook the whole project. Before Scrum, we had a lot of bottleneck resources as many developers had a unique expertise and defended their niche"* (Scrum Master). As a result of the collaboration with the customer, they also acquire a deeper domain knowledge: *"The tight collaboration with the customer facilitates the creation of domain knowledge among the developers. They appreciate this effect very much"* (Management Executive).

The results confirm that the developers perceive the collaboration within the team as better in Scrum projects (H4): *"The communication among the developers now is much more efficient and immediate"* (Management Executive). *"The developers appreciate that they now conceptualize new things together. The team members really act as fellows"* (Scrum Master). The daily meetings are perceived as nuclei of the intensified collaboration: *"Triggered through the Daily Scrum, developers started to help each other. A colleague who has a problem instantaneously gets help from the others"* (Scrum Coach 2). The increased collaboration better conforms to the way developers prefer to work: *"Teamwork is valued much more by the developers than having to work as lone fighters"* (Scrum Coach 3).

The interview results confirm that the developers perceive the transparency as better in Scrum projects (H5): *"The developers appreciate that the overall status and problems are more transparent with Scrum"* (Scrum Master). *"The developers perceive the increased transparency as a benefit of Scrum"* (Scrum Coach 3). Especially the daily meetings and the Burndown Chart establish transparency: *"The Daily Scrum and the shared task-board make the current project status and the encountered problems transparent"* (Scrum Master). The increased transparency better conforms to the way developers prefer to work: *"For the developers this is a benefit since they can immediately react"* (Scrum Master). *"The Daily Scrum turned out to be an appreciated opportunity for synchronization"* (Scrum Coach 1).

Analogous to the quantitative data, the interview results indicate that the developers have a mixed attitude regarding the complexity of development tasks (H6). They perceive the smaller development increments of Scrum projects to be less complex than the tasks in traditional projects: *"Developers find the goals to be more realistic in Scrum projects"* (Scrum Coach 1). *"In Scrum projects, the individual tasks are smaller and easier to manage"* (Scrum Master). Various factors seem to increase complexity, however. Teams had difficulties to get Scrum projects started: *"Scrum does not say much about the up-front planning. But before a project starts, someone has to think ahead and plan the strategy"* (Scrum Coach 2). *"A problem is that we need to do a few Sprints before we get an idea of what we are doing"* (Scrum Coach 1). In addition, the developers find it increasingly difficult to concentrate on building new software as the already delivered increments have to be improved simultaneously: *"We also have to account for the software part that already is in use. This leads to a constant interference"* (Scrum Master). *"We have projects with complex software that do have no stability anymore. They ended up with a Kanban system and only discuss bugs that have to be eliminated on a daily basis"* (Scrum Coach 2). Finally, the execution of distributed development projects is perceived as problematic: *"The management of external dependencies is difficult with Scrum. Maybe, Scrum should be combined with traditional methodologies to define milestones for such deliverables"* (Scrum Coach 1). As rather experienced Scrum developers were assigned to larger and distributed projects with such problems, the findings provide an explanation for the ANCOVA results presented earlier on.

The results confirm that the developers perceive the necessary discipline as higher in Scrum projects (H7). Especially the necessary personal responsibility requires discipline: *"It is a totally different life. Developers have to be self-responsible and live up to that responsibility every day"* (Management Executive). *"In practice, it is sometimes viewed as a problem that Scrum requires so much discipline from every developer"* (Scrum Coach 1). It is perceived as difficult to assure the required amount of discipline: *"Scrum does not say much about how to organize the complementary governance process"* (Scrum Coach 1). *"Developers oftentimes underestimate repercussions and psychological aspects. Many rush into agile development saying 'let's just do it' and then it goes wrong"* (Scrum Coach 3).

In summary, the interview results validate our quantitative findings. However, they also provide rich explanations and give detailed insights into the opportunities and threats accompanying Scrum in use.

8 Conclusions

Little research has examined the use of agile methodologies beyond the adoption stage. In this paper, we have presented the results of a study in which we investigated how developers perceive agile methodologies in use by taking Scrum as a specific study subject. Building upon the DOI theory as a lens for analysis, we identified several acceptance factors of Scrum and evaluated if these factors are perceived as benefits or drawbacks by developers. As our research approach combined quantitative with qualitative data analyses, we could provide statistical evidence for our hypotheses and complement the results with explanations obtained during the interviews. This helped validating the quantitative results, concretizing them, and clarifying why some hypotheses remained unsupported.

We have taken several precautions to ensure the validity of our findings. In particular, we used methodological triangulation to validate the results through a cross-verification of data gathered with different (quantitative and qualitative) methods and from different audiences. To ensure the stability of the quantitative data, we examined if the perception of the developers varied depending on the period they used Scrum. An ANCOVA showed that the perceptions only intensified the longer developers worked in Scrum projects. To ensure the stability of the qualitative data, we interviewed owners of different Scrum roles and checked their statements for consistency during the coding phase. The presented findings were each supported by multiple experts. Nevertheless, we will have to increase the external validity of our findings. So far, we have only examined ISD projects of one company. Although the company has implemented Scrum by the book, the results should be generalized with care. Furthermore, we have only examined a specific methodology. The findings hence should not straightforwardly be transferred to other methodologies.

Our findings have implications for academia and practice. For practice, they provide a framework with factors that affect the developer acceptance and hence influence the sustainability of Scrum projects. These factors should be observed to ensure its successful dissemination and use. While our results suggest that Scrum brings relative advantages and is more compatible to the way developers prefer to work, developers perceived the complexity of Scrum to be higher than that of traditional methodologies. Especially, they found the required discipline to be challenging. Despite the promise that Scrum "cuts through complexity" (Schwaber and Beedle, 2002), they furthermore found several aspects to increase the task complexity. The perceived complexity might therefore pose a potential threat to the sustainability of Scrum projects. Companies ought to carefully monitor this aspect to ensure that developers remain committed and actively support agile principles. With respect to academia, the results contribute to understanding the factors that facilitate or hinder the acceptance and use of agile methodologies in organizations (Mangalaraj et al., 2009). While the presented findings outline a framework of concrete factors, they are just an initial step that ought to be re-evaluated in other contexts. Thereby, future research should also examine other agile methodologies in order to abstract the discussion of acceptance factors to a more general concept of agility (Conboy, 2009). Furthermore, strategies should be developed to overcome identified causes of threats to the acceptance of agile methodologies. Finally, it would be interesting to quantify the effect that the different factors have on the acceptance of agile methodologies. With our results, we hope to provide a starting point for such endeavors.

References

Abrahamsson, P., K. Conboy and X. Wang (2009). 'Lots Done, More To Do': the Current State of Agile Systems Development Research. European Journal of Information Systems, 18 (4), 281-284.

Bahli, B. and E.S.A. Zeid (2005). The role of knowledge creation in adopting extreme programming model: an empirical study. In Proc. 3rd Int. Conf. on ICT.

Beck, K., *et al.* (2001). Manifesto for Agile Software Development.

Boehm, B. (2002). Get Ready for Agile Methods, with Care. IEEE Computer, 35 (1), 64-69.
Bonner, N., J. Teng and S. Nerur (2010). The Perceived Advantage of Agile Development Methodologies by Software Professionals: Testing an Innovation-Theoretic Model. In Proc. 16th AMCIS.
Conboy, K. (2009). Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. Information Systems Research, 20 (3), 329-354.
Conboy, K., M. Pikkarainen and X. Wang (2007). Agile Practices in Use from an Innovation Assimilation Perspective: A Multiple Case Study. In Proc. 28th ICIS.
de Winter, J.C.F. and D. Dodou (2010). Five-Point Likert Items: t test versus Mann-Whitney-Wilcoxon. Practical Assessment, Research & Evaluation, 15 (11), 1-12.
Dyba, T. and T. Dingsoyr (2008). Empirical studies of agile software development: A systematic review. Information and Software Technology, 50 (9-10), 833-859.
Fitzgerald, B., G. Hartnett and K. Conboy (2006). Customising agile methods to software practices at Intel Shannon. European Journal of Information Systems, 15 (2), 200-213.
Friis, D., J. Ostergaard and J. Sutherland (2011). Virtual Reality Meets Scrum: How a Senior Team Moved from Management to Leadership. In Proc. 44th HICSS.
Gallivan, M. (2001). Organizational Adoption and Assimilation of Complex Technological Innovations: Development and Application of a New Framework. The DATA BASE, 32 (3), 51-85.
Given, L.M. (2008). The SAGE encyclopedia of qualitative research methods. Sage, Los Angeles.
Ilieva, S., P. Ivanov and E. Stefanova (2004). Analyses of an agile methodology implementation. In Proceedings of the 30th Euromicro Conference, pp. 326-333, IEEE Computer Society Press.
Kaplan, B. and D. Duchon (1988). Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study. MIS Quarterly, 12 (4), 571-586.
Lee, G. and W. Xia (2010). Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data on Software Development Agility. MIS Quarterly, 34 (1), 87-114.
Mangalaraj, G., R. Mahapatra and S. Nerur (2009). Acceptance of software process innovations - the case of extreme programming. European Journal of Information Systems, 18 (4), 344-354.
Mann, C. and F. Maurer (2005). A case study on the impact of Scrum on overtime and customer satisfaction. In Proc. 1st Agile Development Conference.
Mannaro, K., M. Melis and M. Marchesi (2004). Empirical analysis on the satisfaction of IT employees comparing XP practices with other software development methodologies. In Extreme Programming and Agile Processes in Software Engineering, pp. 166-174, LNCS 3092, Springer.
Miles, M.B. and A.M. Huberman (1999). Qualitative Data Analysis. Sage, London.
Mustonen-Ollila, E. and K. Lyytinen (2003). Why Organizations Adopt Information Systems Process Innovations: A Longitudinal Study using Diffusion of Innovation Theory. Information Systems Journal, 13 (3), 275-297.
Nerur, S. and V. Balijepally (2007). Theoretical Reflections on Agile Development Methodologies. Communications of the ACM, 50 (3), 79-83.
Petersen, K. and C. Wohlin (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. Journal of Systems and Software, 82 (9).
Riemenschneider, C.K., B.C. Hardgrave and F.D. Davis (2002). Explaining Software Developer Acceptance of Methodologies: A Comparison of Five Theoretical Models. IEEE Transactions on Software Engineering, 28 (12), 1135-1145.
Rogers, E.M. (1995). Diffusion of Innovations. 4. Ed. Free Press, New York, NY.
Sawyer, S., P. Guinan and J. Cooprider (2008). Social interactions of information systems development teams: A performance perspective. Information Systems Journal, 20 (1), 81-107.
Schwaber, K. and M. Beedle (2002). Agile Software Development with SCRUM. Prentice Hall.
Schwaber, K. and J. Sutherland (2011). The Scrum Guide.
Vagias, W.M. (2006). Likert-type scale response anchors. Clemson International Institute for Tourism & Research Development, Department of Parks, Recreation and Tourism Management.
Walczuch, R., G. Van Braven and H. Lundgren (2000). Internet Adoption Barriers for Small Firms in the Netherlands. European Management Journal, 18 (5), 561-572.
Webster, J. and R.T. Watson (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. MIS Quarterly, 26 (2), xiii-xxiii.

West, D. and T. Grant (2010). Agile Development: Mainstream Adoption Has Changed Agility. Forrester Research.