Investigating the Long-Term Acceptance of Agile Methodologies: An Empirical Study of Developer Perceptions in Scrum Projects

Sven Overhage  
University of Augsburg  
sven.overhage@wiwi.uni-augsburg.de

Sebastian Schlauderer  
University of Augsburg  
sebastian.schlauderer@wiwi.uni-augsburg.de

Abstract

Agile development methodologies have gained great interest in research and practice. As their introduction considerably changes traditional working habits of developers, the long-term acceptance of agile methodologies becomes a critical success factor. Yet, current studies primarily examine the early adoption stage of agile methodologies. To investigate the long-term acceptance, we conducted a study at a leading insurance company that introduced Scrum in 2007. Using a qualitative research design and the Diffusion of Innovations Theory as a lens for analysis, we gained in-depth insights into factors influencing the acceptance of Scrum. Particularly, developers felt Scrum to be more compatible to their actual working practices. Moreover, they perceived the use of Scrum to deliver numerous relative advantages. However, we also identified possible barriers to acceptance since developers felt both the complexity of Scrum and the required discipline to be higher in comparison with traditional development methodologies.

1. Introduction

Volatile enterprise environments and ever more rapidly changing requirements increasingly complicate the work of information systems development (ISD) teams today. As a result, a large number of IT development projects is either unsuccessful or at least seriously challenged [12]. In response to this development, agile methodologies such as Scrum, Feature Driven Development, or Extreme Programming (XP) have been proposed. They allow developers to more flexibly adapt to changing requirements by promoting close collaboration with customers, self-organizing teams, communication, and responsiveness to change over extensive planning and codified processes [27]. Agile methodologies have gained a significant interest over the last years [38], especially since both anecdotal evidence and the results of more rigorously conducted studies indicate that they can indeed enhance the flexibility of development teams in practice [13, 21].

However, the introduction of agile methodologies considerably changes the working habits of developers as such methodologies intentionally set a polar opposite to traditional development approaches. For example, Scrum introduces a whole set of team meetings in which developers discuss the current project state, flexibly take over new tasks, and continuously act upon the feedback of customers. Since agile methodologies moreover postulate flat hierarchies with self-organizing teams, the developers’ commitment to agile principles and their willingness to actively participate in agile projects become critical success factors. Therefore, the agile manifesto advises to build “projects around motivated individuals” [5].

To ensure the success of agile approaches, it is especially important that developers do not only adopt agile methodologies in the first place, but that they assimilate agile principles and stay committed in the long-term. So far, little research has evaluated how developers perceive agile methodologies in the long-term, however. In current literature, the success of agile methodologies is mostly measured in terms of cost reduction or product quality [18, 22, 29]. Moreover, there exist studies investigating social acceptance factors of agile development methodologies. However, these studies mainly focus on early stages of the assimilation process and evaluate if developers support the initial transitioning from conventional to agile methodologies [20, 36, 37]. Consequently, there remains a “need for a better understanding of agile methods beyond the adoption stage” [1].

To help closing this literature gap, we present the results of an empirical study, in which we examined the continued use of an agile methodology in a later assimilation stage. Particularly, we investigated factors that foster or impede the acceptance of agile methodologies in the long-term. Using the Diffusion of Innovations Theory [31] as a lens for analysis, we address the following research questions: which factors of agile methodologies are perceived as benefits or drawbacks by developers? How does this influence the commitment of developers to agile development?
We address the questions using qualitative research methods from social science. Doing so enables us to get in-depth insights by examining rich data that was gathered during a field study. The field study was conducted at a leading German insurance company, which started introducing Scrum back in 2007. We collected the data in late 2009 when the company was employing Scrum already for a considerable portion of its in-house development projects. All in all, about 200 developers were working in 23 agile projects. In order to study their perception of agile methodologies, we conducted semi-structured interviews with five experts who had privileged insights into the projects and developer opinions.

As a specific methodology [34], we have chosen to examine the use and acceptance of Scrum for several reasons. Firstly, Scrum is by far the most widespread agile methodology in practice to date [38]. Despite that, there exist comparatively few empirical studies of Scrum as most focus on XP [11]. Dyba and Dingsøyr accordingly found management-oriented approaches such as Scrum to be “clearly the most under-researched compared to their popularity in industry” [10].

The remainder of the paper is organized as follows: in the next section, we relate our work to that of others and highlight the existing literature gap. Section 3 discusses the theoretical background before we set up our research model in section 4. Thereafter, the research design is described in section 5. In section 6, the results of our study are presented and discussed before we conclude by emphasizing limitations and implications of our study in section 7.

2. Related Work

During the introduction of agile methodologies such as Scrum [34] or XP [4], the promotion and description of these approaches was driven by consultants or practitioners. With their growing popularity in practice, agile methodologies were subsequently addressed by an increasing number of scientific articles. However, literature analyses still attest that the current body of knowledge about agile methodologies has several shortcomings [1]. First of all, the lack of rigorously conducted and described studies is criticized [1, 10]. In a literature review of empirical studies, Dyba and Dingsøyr consequently determined the evidence about strengths and weaknesses of agile approaches as “very low” and “very uncertain” [10]. Following calls to enhance the evidence about strengths and weaknesses of agile methodologies, more rigorous studies were published [7, 17, 18, 22]. Yet, these studies mainly focus on measuring the success of agile methodologies in terms of project costs, code quality, or completion time. Since they do not aim at examining social acceptance factors of agile methodologies, they are not suited to reason about the developers’ commitment and motivation.

Besides the lack of rigorously conducted studies, it was moreover criticized that studies which examine social acceptance factors of agile methodologies mainly concentrate on the early adoption stage [1, 10]. As the subsequent use of agile methodologies is not evaluated, no conclusions about the long-term acceptance by developers and hence about the sustainability of agile methodologies can be drawn from these studies. Although this criticism was back in 2009, so far only very few approaches examined the use of agile methodologies beyond the adoption stage. Amongst the very few approaches is an article by Mangalaraj et al. [20]. In their study, they examined XP practices and identified factors that affect the acceptance of these practices. However, the study only concentrates on practices that are used when XP is employed. While Scrum and XP arguably share similarities, Scrum targets the planning and management of development projects whereas XP concentrates on supporting technical implementation steps [13]. Compared to Scrum, XP describes a number of best practices which shall be utilized during the implementation of software (e.g. pair programming). Due to these differences between the methodologies, it is necessary to additionally assess the perceptions of developers during projects in which Scrum is used.

Some recently published articles investigate the long-term use of Scrum [15, 17]. Yet, they do not explicitly focus on the commitment of developers. Since evaluating social acceptance factors of Scrum was not the primary aim of these papers, findings about developer perceptions and the resulting implications for their commitment were only mentioned as an aside. So far, rigorously conducted studies which examine how developers perceive Scrum as a specific agile methodology beyond the adoption stage – to our best knowledge – do not exist.

3. Theoretical Background

To understand which factors might be perceived as advantages or drawbacks by developers, we discuss Scrum and its differences to traditional development approaches. Thereafter, we describe innovation adoption theories and particularly the Diffusion of Innovations Theory, which we use as a basis to systematically identify acceptance factors later on.

3.1 Scrum and Traditional Approaches

Scrum and other agile methodologies set a polar opposite to traditional development methodologies
such as the waterfall model. Traditional approaches typically follow a rigorous process management, in which the planning is based on a work breakdown structure with milestones and work packages. Enforcing a predefined process model allows to plan the phases of the development process ahead. Therefore, traditional methodologies are said to be predictable, repeatable, and to enable the optimization of processes [6]. Instead of focusing on rigorous process management, Scrum emphasizes practical applicability and flexibility. It was envisioned as a “management and control process that cuts through complexity to focus on building software that meets business needs” [34].

To achieve this goal, Scrum builds upon an empirical process control. In contrast to traditional methodologies, Scrum emphasizes that the development process cannot be completely planned ahead since requirements are likely to evolve throughout the project. Scrum hence has three different levels of planning: the Release Planning, the Sprint Planning, and the Daily Scrum. During the Release Planning, the basic strategic aspects like the overall costs or functionality of a development project are discussed. Operational details are instead planned from iteration to iteration during the development project. Scrum defines these iterations as Sprints, which are supposed to take one month or less. In the according Sprint Planning meeting, which should take eight hours for a one month Sprint [35], the requirements and tasks for the next Sprint are fixed. The most detailed level of planning takes place in the Daily Scrum meetings. During these 15-minute meetings, the team members report their current development progress and new tasks are appointed.

Compared to traditional methodologies, in which the project manager assigns tasks to team members, Scrum builds upon a self-organizing team structure where the assignment of tasks is discussed in the team. This not only makes Scrum projects more transparent and flexible, but it also presumes that team members have a high commitment and sense of responsibility. Due to their involvement into the project planning, the members in Scrum teams have to spent additional effort for the various meetings.

Next to the planning of development projects, Scrum also affects the collaboration within the team and with the customer. Scrum development teams usually consist of four to nine members and are coached by a Scrum Master. In contrast to a project manager in traditional methodologies, the Scrum Master only guides the team during the development project and does not prescribe how requirements should be implemented. Scrum therefore promotes self-organizing teams with a flat hierarchy. Besides the collaboration within the team, the collaboration with the customer also plays an important role in Scrum projects. As Product Owners, customers have their own role in Scrum projects and are integrated into several project stages. They should for instance participate in Sprint and Daily Scrum meetings so that they are always aware of the current development state.

The integration of customers into the development project also affects the management of requirements. In Scrum projects, customers evaluate working pieces of software at the end of each Sprint and continuously discuss the current project state with the development team. Even during later stages of the project, they may add or change requirements if necessary. In traditional development projects, the requirements are fixed at the beginning of a development project in a contract-like document. Unlike in Scrum projects, working pieces of software can only be inspected by customers in late stages of the development process. Since most of the implementation is already finished then, the integration of new requirements becomes a difficult task, however.

The close collaboration with customers and the evaluation of potentially releasable software are two important controlling techniques in Scrum projects, since both make it more likely that the actual needs of customers are met. In addition, Scrum proposes so-called Burndown Charts which show a daily updated summary of the remaining tasks of an iteration. In traditional projects, team members typically return a percentage of completion for milestone or status reports. However, while the close collaboration with customers seems to result in many advantages, it also burdens developers with more effort as they continuously have to interact with customers.

Another difference between traditional and agile methodologies is the documentation of the resulting software. Traditional methodologies emphasize the general importance of documentation and regard it as an integral part of the development process. Scrum instead values “working software over comprehensive documentation” [5]. The knowledge gained during a development project with traditional methodologies is hence explicitly written down, whereas Scrum relies on the knowledge transfer between team members which results from the various meetings. Due to these meetings and the increased communication, Scrum not only facilitates the transfer of knowledge within the team but also reduces the effort needed for writing an explicit documentation. On the downside, knowledge can easily get lost if team members leave the team or meetings are not taken seriously.

Finally, the feedback mechanisms are handled differently. With traditional methodologies, team members typically only discuss lessons learned at the end of a development project. In contrast, Scrum
enhances the transfer of knowledge right after the first iteration. During Sprint Review and Sprint Retrospective meetings, team members not only discuss the current project state, but moreover identify measures that turned out to be successful during the last Sprint. While Scrum enhances the learning processes for team members, their effort is again increased as retrospective meetings are supposed to take approximately seven hours for a one month Sprint [35]. Table 1 summarizes the major differences.

Table 1. Scrum and traditional methodologies

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Scrum</th>
<th>Traditional methodologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Development process managed from iteration to iteration. Planning is done on different strategy levels. Assignment of tasks is discussed in daily meetings.</td>
<td>Development process is managed in advance. Planning is done using a work breakdown structure and milestones. Project manager assigns tasks in advance.</td>
</tr>
<tr>
<td>Collaboration</td>
<td>Self-organizing teams with a flat hierarchy. Close collaboration within the development team and with the customer throughout the project.</td>
<td>Project manager leads the team. Collaboration with the customer usually during the definition of requirements only.</td>
</tr>
<tr>
<td>Requirements</td>
<td>Continuous discussion of requirements with the customer.</td>
<td>Requirements are fixed in a contract-like document.</td>
</tr>
<tr>
<td>Project controlling</td>
<td>Customers can evaluate a working piece of software at the end of each Sprint. Burndown Charts show daily updated summary of remaining tasks.</td>
<td>Team members typically return a percentage of completion (e.g. by using tools) for milestone or status reports.</td>
</tr>
<tr>
<td>Documentation</td>
<td>No instruction to write down knowledge in a documentation. Instead the transfer of knowledge is fostered through various meetings.</td>
<td>Documentation is considered to be an integral part of the development process.</td>
</tr>
<tr>
<td>Retrospectives</td>
<td>Retrospective Meetings after each Sprint. Lessons learned usually only once at the end of a project.</td>
<td></td>
</tr>
</tbody>
</table>

3.2 The Diffusion of Innovations Theory

As the discussion illustrates, Scrum introduces a novel methodology to develop information systems. From a theoretical viewpoint, it can be classified as innovation, which are broadly defined as ideas or practices that are perceived to be new by adopters [31]. The adoption and diffusion of such innovations is explained by innovation adoption theories [3, 8, 16, 31, 32].

To describe the process in which an innovation is assimilated by an organization, a group, or an individual, Gallivan introduces a model consisting of six stages [16]. Each stage describes a certain level to which an innovation penetrates an adopting unit (see Figure 1). In the initiation stage, a match between an innovation and its application by the adopting unit is defined. During the adoption stage, a decision to adopt the innovation is made. As part of 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 using the innovation. The innovation is employed in the unit. During the routinization stage, the usage of the innovation is encouraged as a normal activity. In the infusion stage, it is used in a comprehensive and sophisticated manner so that the effectiveness of the adopting unit increases. With respect to Gallivan’s model, the study presented in this paper focuses on the later stages of assimilation where initiation, adoption, and adaptation already have occurred.

Figure 1. Innovation assimilation stages [16].

Complementary to descriptions of the assimilation process, theories such as the Technology Acceptance Model [8], the Theory of Planned Behavior [3], or the Diffusion of Innovations Theory [31] explain why and at what rate innovations spread within a social system over time. These theories have many commonalities and have repeatedly been used to explain both the diffusion of innovative information systems as well as that of ISD methodologies [25, 30]. However, they are based on different levels of analysis, such as the individual, group, and organization level.

The Diffusion of Innovations (DOI) Theory was defined by Rogers to explain the assimilation of innovations [31]. In this theory, individuals are seen as having different degrees of willingness to assimilate innovations. Consequently, it is generally observed that the portion of the population assimilating an innovation is distributed over time. The DOI theory defines five perceived attributes of innovations that affect the rate of assimilation [31]: relative advantage, compatibility, complexity, trialability, and observability. Except for the complexity, the perception of these attributes is positively correlated to the rate of assimilation. The perception of complexity is negatively correlated [31].
With respect to innovative information systems and ISD methodologies, empirical research has consistently found three perceived attributes of innovations to be significant determinants for the rate of assimilation [2, 25]. We therefore consider these well-established attributes as facets to identify factors that influence the developer acceptance of Scrum:

- **Relative advantage**: the level to which an innovation is perceived to be better than the idea it supersedes.
- **Compatibility**: the level to which an innovation is perceived to be consistent to actual needs and past experiences.
- **Complexity**: the level to which an innovation is perceived to be difficult to use.

4. **Research Model**

As highlighted by the discussion in section 3, several factors of Scrum differ from traditional, plan-based approaches. In the following, we classify these factors as instances of the perceived attributes of innovations to explain their effect on the developer acceptance of Scrum (see Figure 2). Furthermore, we hypothesize how these factors are perceived by the developers. Hypotheses H1 to H4 claim that Scrum introduces relative advantages for developers. H5 and H6 postulate that Scrum is more compatible to the actual working practices of developers than traditional methodologies. With respect to the perceived complexity of Scrum, H7 and H8 posit a tradeoff between a lower process complexity on the one hand and a higher demand for discipline on the other hand.

Scrum has been designed to support the developing of software that meets (volatile) business requirements [34]. The empirical management of development projects from Sprint to Sprint allows customers to gradually evolve their requirements, flexibly add new ones to the Backlog, and continuously give feedback on the results [13]. We hence propose that

**H2. Compared to traditional projects, developers perceive the customer requirements to be better met.**

Because developers flexibly take over development tasks in close collaboration when they have to be fulfilled [34], each team member gets an in-depth knowledge of the whole project. Scrum furthermore defines Sprint Review and Retrospective meetings, which are thought to enhance the learning processes right after the first Sprint. Accordingly, we posit

**H3. Compared to traditional projects, developers perceive the learning effects to be better.**

Although agile methodologies have primarily been designed to better satisfy customers [5], collaborative development and the experiences shared with each other also lead to teamwork effects among developers and promote the emergence of trust into their mutual abilities [19]. We thus suggest

**H4. Compared to traditional projects, developers are more satisfied with the outcomes.**

To be able to achieve their development task efficiently, developers need to be aware of the overall development state and the tasks that remain to be done. In contrast to traditional development methodologies, Scrum explicitly supports this requirement, e.g. by providing so-called Burndown Charts which constantly document the progress. For this reason, we assume

**H5. Compared to traditional projects, developers perceive the transparency of the development status to be better.**

As software development is a highly creative process, a close collaboration within the development team is crucial to utilize the different skills and expertise in an ideal manner [33]. In traditional development projects, developers are assigned to tasks in an inflexible manner that is determined by the project leader. By contrast, Scrum builds upon the principle of self-organizing teams and promotes an immediate, informal communication [34]. We therefore suppose

**H6. Compared to traditional projects, developers perceive the collaboration to be better.**

Scrum furthermore aims at reducing the complexity, e.g. by minimizing extensive planning steps and managing development only from Sprint to Sprint. Consequently, the tasks to be completed are smaller and easier to accomplish [28] so that we expect

**H7. Compared to traditional projects, developers perceive the process complexity to be lower.**
Developers will have to closely work together and continuously collaborate with the customers to ensure the success of the project, though [5]. To live up to this demand, developers will have to show a considerable amount of discipline. Accordingly, we posit

H8. Compared to traditional projects, developers perceive the required discipline to be higher.

Note that we did not hypothesize about software characteristics such as the density of code defects. While improvements of such characteristics have been reported, they have to be attributed to methodologies which concretely support the implementation process (e.g. XP). Also, we did not hypothesize about the effects of neglecting documentation as the company examined had to document ready-made software in detail to ensure auditing certainty.

5. Research Method

In order to examine the research hypotheses developed above, we conducted so-called semi-structured expert interviews [23]. Thereby, the number of interviewees can be rather low as long as they are carefully selected. An expert is defined as someone with responsibility for the design, implementation, and/or the control of a study object and/or privileged knowledge of studied teams and processes. Typically, such experts are neither found on the topmost nor on the lowest hierarchy levels of an organization, but on somewhat higher hierarchy levels where most of the knowledge about inner structures and events is present [23].

Conducting semi-structured interviews has several advantages. On the one side, they allow for an in-depth understanding of the context and provide deeper insights than other data collection approaches [9]. On the other side, they follow an elementary guideline, which ensures a common structure and makes results comparable [14]. Therefore, semi-structured interviews are considered to be the superior data collection technique in interpretive investigations [39].

The data was collected at a world-wide leading German insurance company, which began gradually shifting its in-house software development to Scrum in 2007. Before that, the company mainly employed the V-Model – an extension of the waterfall model – in their development projects. The implementation of Scrum in the company followed the textbook and encompassed all of the foreseen Scrum roles, meetings, and principles. However, the length of the Sprints varied between the projects. For the majority of them, it was between two and four weeks.

Since Scrum was introduced, the proportion of Scrum projects constantly increased and the development teams gained mature knowledge about Scrum. When we began conducting our study in late 2009, a considerable amount of software development projects was managed using Scrum. Scrum was already in use for 23 different in-house development projects with approximately 200 team members.

The projects were not limited to certain domains or applications. Instead, Scrum was used for projects ranging from developing software for database platforms to business software. Regarding the project significance, Scrum was used in areas which belonged to the core competencies of the company as well as in areas that handled the support of standard processes. With respect to Gallivan’s innovation assimilation model (Figure 1), the company was in the midst of the routinization stage, in which the usage of the innovative methodology becomes a normal activity.

The interviews were conducted face-to-face in the interviewees own offices. As the interviews basically had an open form and only followed an elementary guideline, their length varied in between 40 and 60 minutes. Regarding the overall structure of the interviews, we closely followed the recommendations of Myers and Newman [26]. The first part of the interview guideline contained questions about the participants such as the department they worked for, their expertise with Scrum and traditional methodologies, or the role they had in their Scrum projects. In the second part, we collected information about the experiences with Scrum and the developer perceptions. We began by generally asking what factors of Scrum were perceived as advantages or disadvantages. Afterwards, we specifically wanted to know how the factors examined in this study were perceived. We moreover asked why this was the case and how this affected the developers’ commitment to agile development projects. Finally, we asked for which projects the interviewees would recommend Scrum and what benefits they expect for the studied company. However, the guideline only provided an elementary basis and further questions were added during the interviews depending on what the interviewees responded.

In order to enhance the reliability of our results, we collected data from experts with different roles. Since we decided to interview experts, we considered asking Scrum Masters, Scrum Coaches, and higher management executives. All of the interviewed experts were internal resources of the company.

Scrum Masters guide the development teams during the project and are responsible for their teams to adhere to agile values, practices, and rules [35]. Scrum Coaches are not directly involved into development projects, but are responsible for educating team members and making them familiar with agile principles. Additionally, they regularly counsel team members and reflect gathered experiences together

5457
with them. In the examined company, both groups have constant and direct contact to the development teams. Furthermore, higher management executives were strategically managing the company’s turn to Scrum. To ensure a smooth transitioning, they maintained a close contact to the development teams and were hence able to also provide privileged insights into the perceptions of developers in Scrum projects.

In total, we conducted interviews with five experts. Of them, three were Scrum Coaches, one was a Scrum Master, and one was a higher management executive. To analyze the data, we recorded the interviews and transcribed the statements to English language immediately after the interviews were conducted. Afterwards, two researchers independently coded the interview statements in order to identify consistencies and distinctive parts in the interview statements. Thereby, the acceptance factors shown in Figure 2 served as a classification schema to thematically group the statements into “intellectual bins” [24]. We compared the coding results and eliminated any discrepancies based on a discussion.

6. Results and Discussion

The analysis of the data gathered in the first interview part shows that the interviewed experts had profound experience both with Scrum and traditional development methodologies. On average, they had worked with Scrum for 4.5 years and with traditional methodologies for 23.4 years. On a scale from (1) marginal knowledge to (4) profound knowledge, they rated their expertise in Scrum with 3.6 and their knowledge of traditional methodologies with 3.4.

The coded data that was gathered in the second interview part gives indications regarding the research hypotheses that we posited earlier. The interview results confirm that the developers perceive the time to market to be better in Scrum projects (H1). The reduced time to market is viewed as an advantage that fosters the acceptance of Scrum: “In Scrum projects, the developers especially value that they can react faster and flexibly develop what is really required. [...] The developers can provide results much faster now and for them this is a big advantage” (Scrum Master). “The developers can focus on what the customer actually wants and leave out things that are less relevant. So they still deliver 20 pieces of software in let’s say a week, but 20 useful ones. And for them that really is a major improvement” (Scrum Coach 3). “The developers are very pleased that they can present outcomes fast with Scrum” (Scrum Master).

The results also confirm that the developers perceive the customers’ requirements to be better met in Scrum projects (H2). The better meeting of requirements is viewed as a benefit that fosters the acceptance of the new methodology: “The developers realize that they do not work for nothing anymore as they oftentimes did with the waterfall model. [...] They don’t work for a year and then it is realized that the results are not useful for the customer” (Scrum Master). “An important improvement for the developers is that the customer gets what he wants and his demands can be satisfied better. [...] The developers like the idea of taking over changes already during development and not having to work on change requests afterwards” (Scrum Coach 1). “The developers are happy that they achieve a good covering of requirements in Scrum projects and that this covering is already reached at an early point in the development process” (Scrum Coach 3).

In comparison to traditional methodologies, there seem to be better learning effects in Scrum projects (H3). The interview results indicate that developers learn both from each other and from the customer. The better learning effects are perceived as an advantage and a motivator for the acceptance of Scrum: “Triggered through the Daily Scrum meetings, developers started to help each other. If a colleague has a problem, he gets help from the others simultaneously” (Scrum Coach 2). “A frequently mentioned advantage is that every developer now flexibly takes over any task. On the one hand, this strengthens the 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 own niche” (Scrum Master). “The tight collaboration with the customer facilitates the creation of domain knowledge among the developers. The developers appreciate this effect very much” (Management Executive).

In addition, the interview results indicate that developers are more satisfied with the outcomes of Scrum projects than they were with those of traditional methodologies (H4). The increased satisfaction is perceived as an advantage and positively influences their willingness to accept Scrum: “Most techies are satisfied when they get a challenging but solvable task and can decide for themselves how to approach them. This is exactly what Scrum allows them to do. [...] Developers typically want to solve a task properly so that they can be proud of their solution” (Scrum Coach 2). “The introduction of Scrum has led to a higher satisfaction among developers as the new process introduces a sustainable pace. In the past, we had extreme fluctuations in the workloads and before the milestones everyone was under immense stress” (Scrum Coach 1). “As a consequence of the close
collaboration, the teams moved much more closely together. The satisfaction of the developers is apparent. They really like working with Scrum” (Scrum Master).

The study results furthermore suggest that the transparency of the development status in Scrum projects is better than in traditional projects (H5). Scrum seems to better fulfill an existing requirement for transparency: “The Daily Scrum and the shared task-board make the current project status and the encountered problems transparent. For the developers, this is a benefit as they can immediately react” (Scrum Master). “The Daily Scrum turned out to be an appreciated opportunity for synchronization as everyone gets an overview of the actual status” (Scrum Coach 1). “The developers perceive the increased transparency as benefit” (Scrum Coach 3).

The results indicate that collaboration has increased likewise (H6). Developers welcome this development as it better conforms to their preferred way of working: “Teamwork is valued much more by the developers than having to work as lone fighters” (Scrum Coach 3). “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).

However, the developers have a mixed attitude regarding the complexity of the development process in Scrum projects. We therefore were not able to identify support for H7. On the one hand, they perceive the individual development tasks to be less complex in Scrum 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).

On the other hand, the complexity seems to be increased by various factors. Especially, the teams had difficulties to get projects started: “Scrum doesn’t say much about the up-front long-term planning. But before a project starts, someone has to think ahead and plan the portfolio of products as well as the top-level development strategy” (Scrum Coach 2). “Actually, we need to do a few Sprints before we get an idea of what we are doing and can make any useful long-term planning” (Scrum Coach 1). In addition, the developers found it increasingly difficult to concentrate on developing new software artifacts as the already delivered software increments have to be improved simultaneously (e.g. due to existing faults in the implementation or changing customer requirements): “We also have to account for the software part that already is in use. [...] This leads to constant interference that we cannot avoid” (Scrum Master). “Supporting software is difficult with Scrum. To do this, you’d rather need a Kanban system with rolling backlogs” (Scrum Coach 3). “We do have projects with complex software that do not have any stability anymore. They ended up with a Kanban system were they only discuss bugs that have to be eliminated on a day to day basis” (Scrum Coach 2).

Finally, the interview results show that the developers perceive the required discipline to be higher than in traditional projects (H8). The required discipline is viewed to be a disadvantage that negatively affects the acceptance of Scrum: “It is a totally different life. Developers have to be self-responsible and live up to that responsibility everyday” (Management Executive). “In practice, it is sometimes a problem that Scrum requires so much discipline from every developer” (Scrum Coach 1). “The lightweight methodology appears to be so easily usable that 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). “Scrum does not say much about how to organize the complementary governance process” (Scrum Coach 1).

All in all, we could identify several factors that act as motivators for the developer acceptance of Scrum as they were either perceived as relative advantage or as more compatible with the actual working practices in comparison to traditional projects. Nevertheless, we also encountered potential barriers to acceptance. Factors that characterize the complexity of Scrum were either perceived with a mixed attitude or even judged to have changed for the worse. As the perception of these factors is in conflict with one of the central goals of the Scrum methodology, they might pose a threat to the sustainability of Scrum projects.

7. Conclusions

So far, little research has examined the use of agile methodologies beyond the adoption stage. This is especially true for studies which examine the perceptions of developers in Scrum projects. To close this literature gap, we presented results of a study in which we investigated how developers perceive Scrum in the long term. Taking the Diffusion of Innovations Theory as a lens for analysis, we identified several acceptance factors of Scrum and hypothesized about their perception by the developers. Using qualitative field data, we evaluated our research hypotheses and provided explanations for the perceptions that we observed in practice.

In order to ensure the reliability of the results, we decided to interview experts with different roles in Scrum projects. Doing so gave us the opportunity to gather insights from various perspectives and allowed
us to cross-verify the gathered data. Nevertheless, some limitations should be taken into account when interpreting the results. Since we only presented qualitative data, we could not provide solid statistical evidence for our research hypotheses yet. An analysis of quantitative data will therefore have to be part of future research. Nevertheless, the collected data provided rich explanations and the gained insights already give valuable indications about the support of the formulated hypotheses. We only examined one concrete agile methodology, however. The study results hence cannot be directly transferred to other agile methodologies like XP or Crystal. They can be interpreted in the light of a theoretical and abstract concept of agility, as defined by Conboy [7], though. Finally, we only examined the use of Scrum in the context of one specific company. To enhance the external validity of the results, the use of Scrum will have to be analyzed in additional contexts.

Despite the mentioned limitations, the presented results have implications for academia and practice. For practice, they show which aspects of Scrum should be observed by companies in the long term. In particular, we provide a framework of factors that might positively or negatively affect the commitment of developers and hence influence the sustainability of Scrum projects. In our study, developers perceived Scrum to be more compatible to their working practices and to deliver relative advantages in comparison to traditional development methodologies. In spite of the assumption that Scrum “cuts through complexity” [34], they perceived the complexity of agile methodologies to be considerably higher, though. Especially, they found the required discipline in Scrum projects to be challenging. The perceived complexity might therefore pose a potential threat to the sustainability of agile development projects. Accordingly, companies ought to carefully monitor this aspect in order to ensure that developers remain committed and actively support agile principles.

For academia, our results signal the need to more comprehensively examine the use of agile methodologies beyond the adoption stage. The findings presented in this paper are an initial step that outlines factors which might contribute to or potentially endanger the sustainability of agile methodologies. In order to triangulate the results, future research should also make use of quantitative data collection methods and examine the use of Scrum in other contexts. Furthermore, additional agile methodologies ought to be evaluated in order to detect similarities between different approaches and to contribute to abstracting the discussion to a more general concept of agility. Future research should also examine how severe potential threats might become. Therefore, studies should investigate further into the causes of potential threats and develop strategies to overcome them. While our findings outline a framework of factors, it would be interesting to moreover compare the effect that the different factors have on the acceptance of Scrum. By determining interrelations between the factors, it is possible to elaborate on the acceptance of agile methodologies in more detail. With our results, we hope to provide a starting point for such endeavors.

8. References


[38] D. West and T. Grant, Agile Development: Mainstream Adoption Has Changed Agility, Forrester Research, 2010.