12-6-2010

On the Developer Adoption of Scrum: A New Acceptance Model for Agile Methodologies

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

Sebastian Schlauderer
University of Augsburg, sebastian.schlauderer@uni-bamberg.de

Dominik Birkmeier
University of Augsburg, dominik.birkmeier@wiwi.uni-augsburg.de

Jonas Miller
University of Augsburg, jonas.miller@gmx.de

芽|Sprouts Working Papers on Information Systems ISSN 1535-6078

On the Developer Adoption of Scrum: A New Acceptance Model for Agile Methodologies

Sven Overhage
University of Augsburg, Germany

Sebastian Schlauderer
University of Augsburg, Germany

Dominik Birkmeier
University of Augsburg, Germany

Jonas Miller
University of Augsburg, Germany

Abstract
In recent years, the agile Scrum methodology has become a popular software development approach. It significantly differs from traditional approaches as it promotes communication, self-organization, flexibility, and innovation instead of extensive planning and codified processes. While such a paradigm shift promises to better support the timely delivery of high-quality software in turbulent business environments, its success considerably depends on the willingness of developers to adopt the agile methodology. In this paper, we present a framework with drivers and inhibitors to the developer acceptance of Scrum. It combines analytical with empirical findings and can be used as a theoretical basis to empirically evaluate the actual support of Scrum in concrete scenarios. The introduced framework is based on the extended Technology Acceptance Model (TAM), which has been proven to be also applicable to describe the intention of developers to use a methodology. Building upon results from qualitative in-depth interviews with six experienced Scrum experts of a German DAX-30 company, we refine the general determinants of adoption contained in the TAM with several observed factors that influence the willingness of developers to use Scrum in practice.

Keywords: Agile methodologies, Scrum, developer acceptance, Technology Acceptance Model

Permanent URL: http://sprouts.aisnet.org/10-76

INTRODUCTION

As the persisting software crisis demonstrates, the question of how software development should be organized to support the production of high-quality results in a cost-efficient, flexible, and fast way continues to pose research challenges. Over the last years, agile methodologies have been proposed, which promote communication, self-organization, flexibility, and innovation instead of extensive planning and codified processes. As a reaction to heavyweight plan-based approaches, agile software development since then attracted a rapidly increasing attention in practice. A recent industry survey with 2252 participants in North America and Europe revealed that 26 percent of the companies were already using agile methodologies and an additional 42 percent were aware of them (Forrester Research, 2007). Among the various agile approaches, Scrum plays a particularly important role as it has become the most widespread agile methodology to date. A global industry survey on agile development in 88 countries e.g. documented that „Scrum or a [customized] variant were by far the most common Agile Methodologies employed" (VersionOne Inc., 2009).

Agile methodologies like Extreme Programming (XP) and Scrum introduce a whole new approach to manage development processes in a flexible fashion. With the fundamental changes compared to traditional development, the need to create theories on how agile methodologies impact IT personnel, development practice, and the resulting IT artifacts becomes obvious. Nevertheless, a survey of scientific evaluations published in 2008 revealed „a clear need for more empirical studies of agile development" (Dyba and Dingsoyr, 2008). Not only found the authors the number of scientific evaluations of agile methodologies to be very limited. They furthermore criticized that the majority of studies concentrated on less widespread approaches like XP. Scrum as the most widespread approach instead was hardly in the focus. Accordingly, they considered Scrum (together with its customizations) to be „clearly the most under-researched compared to their popularity in practice" (Dyba and Dingsoyr, 2008). The few published examinations of Scrum mostly evaluate its effect in terms of increased productivity, product quality, or customer satisfaction (Dyba and Dingsoyr, 2008, Ilieva et al., 2004, Layman et al., 2004, Macias et al., 2003, Mann and Maurer, Rising and Janoff, 2000). Such studies document the potential gains resulting from a successful introduction of Scrum. However, they do not immediately contribute to the building of the above-mentioned theories.

As Scrum postulates self-organizing team structures and flexible collaborations in a flat hierarchy, its successful introduction especially depends on the willingness of the developers to adopt the agile methodology. Only consequently were the ability to adjust organizational culture, resistance to change, and lacking of the necessary agile experience in teams identified as major barriers to the successful introduction of agile methodologies (VersionOne Inc., 2008). It is hence important to focus on the building of theories „on human and social factors in order to succeed" (Dyba and Dingsoyr, 2008). However, the current state of research regarding the influence of such factors on the success of agile methodologies is still nascent. We therefore contribute to the building of such theories by introducing a framework with drivers and inhibitors to the developer acceptance of Scrum. In particular, we examine the following research questions: *Which observable human and social factors positively or negatively impact the developer acceptance of Scrum? How can these factors be combined with existing theories on* *the developer acceptance of methodologies to provide an explanative model for the adoption of Scrum?*

We use the constructs of the extended Technology Acceptance Model (Davis, 1989) as a theoretical basis to explain the developer acceptance of Scrum. This model has been proven to also describe the developer acceptance of methodologies in general (Riemenschneider et al., 2002). To specifically explain the developer acceptance of Scrum, we refine the general determinants contained in the Technology Acceptance Model (TAM) with antecedents that were observed as influencing factors in practical projects. Aiming at identifying such factors, we firstly compared Scrum with traditional, plan-based methodologies and analyzed its particular strengths and weaknesses. Based on these results, we conducted an exploratory study in which we performed semi-structured, qualitative in-depth interviews with six Scrum experts from a German DAX-30 company. The goal of these interviews was to gain insights into the social factors that influenced the adoption of Scrum in positive or negative ways during their projects. We therefore selected participants that (i) had different roles in the Scrum development process and (ii) already had mature experiences with the introduction and usage of Scrum. The findings were analyzed, aligned with the results gained during the analytical comparison, and related to the constructs of the TAM.

The remaining presentation is organized as follows: after discussing related work to confirm the research gap (section 2), we summarize differences between Scrum and traditional development approaches in section 3. In section 4, we introduce the TAM and describe adoptions necessary to explain the acceptance of methodologies. Section 5 presents details about the conducted interviews and the influencing factors which were identified out of the gathered information. The framework of drivers and inhibitors is presented in section 6. We conclude by discussing implications of our work and by highlighting future research directions.

RELATED WORK

Generally, empirical studies of the social factors affected by the introduction of agile methodologies are still rare (Dyba and Dingsoyr, 2008). From the few available studies, most are inappropriate to explain the developer acceptance of Scrum for several reasons: first of all, these studies usually examine in what respect the introduction of agile methodologies changes the way of development (Chong, 2005, Ilieva et al., 2004, Mann and Maurer, 2005, Mannaro et al., 2004, Robinson and Sharp, 2005). While such research helps clarifying how the implementation of an agile methodology affects development practices, it does not immediately give information about the factors that lead to an acceptance or resistance. Secondly, nearly all studies focus on examining the less widespread XP methodology and do not take Scrum into account at all (Dyba and Dingsoyr, 2008, Hossain et al., 2009). Even worse, the majority of studies has serious limitations as they were not following a sound research strategy and/or did not focus on examining IT personnel with mature experiences in agile development (Dyba and Dingsoyr, 2008). In 2008, a survey on empirical studies of agile development e.g. revealed that only one research group had examined teams with mature agile experience (Dyba and Dingsoyr, 2008).

Examinations of the human and social factors that determine the acceptance of Scrum by developers – to the best of our knowledge – do not exist yet. Among the very few closely related studies, Bahli et al. show that technology acceptance models can basically be used to explain the developer acceptance of agile methodologies (Bahli and Zeid, 2005): they built upon the TAM to explain the impact of knowledge creation on the acceptance of XP. However, technology acceptance models cannot arbitrarily be reused to explain the acceptance of development methodologies. Riemenschneider et al. (2002) have examined under which conditions such models are applicable as an appropriate kernel theory. In our approach, we therefore build upon their findings to choose a suitable acceptance model as kernel theory. Our strategy of using qualitative in-depth interviews to get insights into the factors that influenced the acceptance of Scrum in practice furthermore follows recommendations of Dyba et al. (2008). They identified a need for exploratory studies as the state of theories on agile software development methodologies „is clearly nascent" (Dyba and Dingsoyr, 2008). To better understand the human and social factors that particularly influence developer acceptance, we accordingly start by identifying key differences between Scrum and traditional development methodologies.

SCRUM AND TRADITIONAL APPROACHES

Compared to traditional plan-based approaches, agile methodologies such as Scrum introduce fundamental changes to the software development process. To show the resulting implications for the IT personnel, we briefly discuss the major differences between traditional development approaches and the Scrum methodology (Table 1 summarizes the results). In contrast to traditional approaches, Scrum is defined as „a management and control process that cuts through complexity to focus on building software that meets business needs" (Schwaber and Beedle, 2002). It emphasizes practical applicability, incremental development, and flexibility. Traditional development methodologies instead focus on rigorous process management. As a consequence, they are said to be predictable, repeatable, and to allow processes to be optimized (Boehm, 2002).

Aspect	Scrum	Traditional methodologies
Planning	Development process managed from iteration to iteration. Planning is done on different strategy levels. Assignment of tasks is discussed in daily meetings.	Development process is managed in advance. Planning is done using a work breakdown structure and milestones. Project manager assigns tasks in advance.
Collaboration	Self-organizing and self-dependent team structure. Tight collaboration with the customer throughout the whole project.	Project manager leads the team. Collaboration with the customer usually during the definition of requirements only.
Project controlling	Burndown Charts show a daily updated summary of remaining tasks. Working piece of software delivered to the Product Owner at the end of each Sprint.	Team members often return a percentage of completion (e.g. by using tools) for milestone or status reports.
Requirements	Continuous discussion of requirements with the customer.	Requirements are fixed in a contract-like document.
Documen-tation	No instruction to explicitly write down knowledge in a documentation. Instead tacit knowledge is built through various meetings.	Documentation is considered to be an integral part of the development process.
Retro-spectives	Retrospective Meetings after each Sprint.	Lessons learned usually only once at the end of a project.

Table 1. Identified differences between Scrum and traditional methodologies

The different ways of approaching software development first of all lead to different strategies for the planning and controlling of development processes. Scrum uses a so-called empirical process control which is based on the assumption that the analysis, design, and implementation of software is generally unpredictable and hence difficult to be planned ahead (Schwaber, 1995). As a consequence, Scrum manages the development process from iteration to iteration and uses different levels of planning: Release Planning, Sprint Planning, and Daily

Scrum. During the Release Planning, basic strategic aspects to govern the whole development process are determined. Such aspects only describe the overall functionality or maximum costs of a release, however. Operational details are to be determined separately for each iteration of the development process. In Scrum, such iterations are called Sprints. The according Sprint Planning provides the relevant details: it defines the requirements for the iteration as well as the resulting tasks in the Sprint Backlog. The time for a meeting to plan a one month Sprint is suggested to take eight hours and has to be adjusted proportionally to the Sprint's length (Schwaber and Sutherland, 2010). The third and most detailed level of planning is the Daily Scrum meeting: during daily 15-minute meetings, each team member explains what (s)he accomplished since the last meeting and what (s)he aspires to achieve before the next meeting.

Traditional development methodologies instead rely on predefined process models with phases to capture the whole progress of a project in advance. The planning typically is based on a work breakdown structure with work packages and milestones. On that basis, the project manager assigns the activities to specific team members. Scrum instead postulates a self-organizing team where the assignment of tasks is not discussed before the Daily Scrum meeting. Such an organization requires a higher commitment and sense of responsibility from each team member, however. While it makes Scrum more flexible, transparent, and adoptable to changing requirements, such an organization furthermore makes detailed cost planning and controlling difficult. Last but not least, it burdens the IT personnel with additional effort to be spent for the various kinds of meetings.

With its agile philosophy, Scrum also impacts the collaboration within the team and with the customer. To stay focused on direct and frequent collaboration, Scrum teams should only consist of five to nine developers (Schwaber and Sutherland, 2010). The customer, who is represented by the role of the Product Owner in Scrum teams, furthermore plays a key role and is to be integrated into several process stages. The Product Owner permanently has to be aware of the current development state and, consequently, is participating in all Sprint Planning and review meetings. When following traditional methodologies, developers usually collaborate with the customer during the requirements definition and not again until the product is handed over. On the one hand, Scrum thus demands considerably more time and involvement from the customer. On the other hand, Scrum teams are able to evaluate development steps with the customer early. They can hence better ensure that actual needs are met. Furthermore, Scrum promotes a self-organizing team structure, in which the so-called Scrum Master only coaches and guides the team. In contrast to traditional methodologies, the team itself decides how to implement requirements (Schwaber and Sutherland, 2010).

With Scrum, project controlling is done using so-called Burndown Charts, while traditional methodologies use milestones and status reports to control the process status. To report the progress of a Sprint, a Burndown Chart shows a daily updated summary of all remaining tasks during the iteration. Additionally, the Product Owner is able to attend the Daily Scrum meetings to get a direct feedback on individual tasks. Scrum furthermore aims at delivering a working piece of software to the Product Owner at the end of each Sprint. For the customer, the delivered software gives insights into the implementation and allows for an immediate reaction to possible misunderstandings. In traditional approaches, it is common that team members only return a percentage of completion to the project manager at the end of each phase. Working software usually is delivered late in the process.

Another major difference between Scrum and traditional methodologies is the identification, analysis, and management of requirements. Scrum requires the IT personnel to work together with the customer to discuss, identify, and add all requirements to a priority list (the so-called Product Backlog). During the project, the Product Owner can add new requirements into the Product Backlog and change priorities, whereas in traditional methodologies the initial requirements are usually fixed in a contract-like document. If new requirements occur during later phases of the development, they have to be included using a change-management process. With Scrum, potentially releasable software is presented to the Product Owner at the end of each iteration. During the discussion of these releases, new requirements are likely to be identified in a timely manner and can be implemented during the next iteration(s). While Scrum therefore is more flexible with respect to new requirements, its agile methodology requires more communication within the Scrum team and with the Product Owner.

Traditional methodologies emphasize the general importance of documentation as they consider it to be an integral part of the development process. Unlike these, agile methods do not instruct explicit documentation since the agile manifesto values „working software over comprehensive documentation“ (Beck et al., 2001). As a consequence, Scrum preferably relies on tacit team knowledge that originates from the various team meetings, whereas traditional methodologies build upon explicit knowledge elicited from the development teams in the form of documentation (Boehm, 2002). While Scrum facilitates the transfer of knowledge due to the intensified collaboration, it also bears a risk of losing knowledge if, for instance, meetings are not taken seriously or if members leave the team. Yet compared to traditional methodologies, time and effort for documentation are clearly reduced.

Finally, the handling of retrospectives is different between both methodologies. Scrum not only foresees so-called Sprint Reviews in which the results of a Sprint are discussed. It additionally prescribes retrospective meetings after each Sprint. In these meetings, the team should discuss improvements learned during the last Sprint. Compared to the so-called Lessons Learned in traditional methodologies, which are usually only discussed once at the end of a project, Scrum retrospectives facilitate the learning process as well as the transfer of knowledge right after the first Sprint. As the Scrum Guide recommends a four hour meeting for the Sprint Review and another three hours for the retrospective meeting of a one month Sprint, the effort required from the IT personnel is further increased, however.

METHODOLOGY ACCEPTANCE MODEL

As can be seen from the comparative discussion of Scrum and traditional development approaches, a successful implementation of agile methodologies largely depends on the acceptance of their development philosophy by the IT personnel. To explain why individuals accept or resist information technologies, IS researchers make use of so-called *acceptance models* that were derived from general theories of motivated human behavior. Among these models, specifically the (extended) Technology Acceptance Model, the Theory of Planned

Behavior, the Perceived Characteristics of Innovating, and the Model of Personal Computer Utilization have become accepted and confirmed theories to explain why individual users accept or resist technologies. Building upon the fact that these models stem from general theories of social science, Riemenschneider et al. (2002) have examined if they can also be applied to explain the adoption of development methodologies such as Scrum.

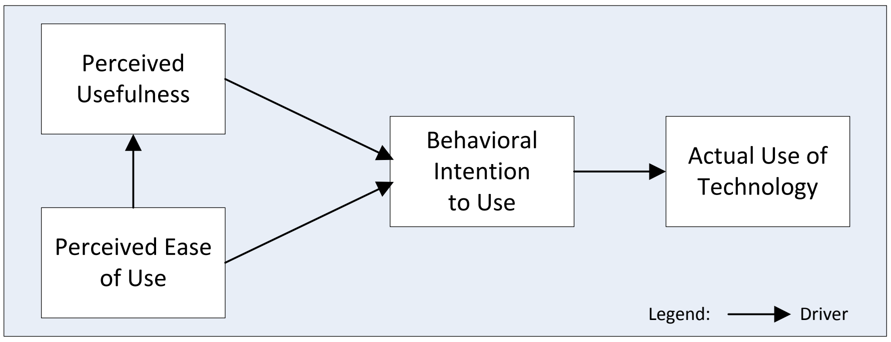

Figure 1. Technology Acceptance Model

Among the models they examined, especially the extended TAM, also called TAM2 (Venkatesh and Davis, 2000), turned out to contain nearly all of the factors that were found to determine the acceptance of development methodologies (Riemenschneider et al., 2002). For that reason, we decided to use the extended TAM and its constructs as the basis to explain the developer acceptance of Scrum. Furthermore, we implemented minor adoptions to the extended TAM that were recommended by Riemenschneider et al. (2002) to better cover the acceptance of methodologies.

In its original fashion, the TAM postulates that *perceived usefulness* and *perceived ease of use* determine an individual's behavioral intention to use a technology (see Figure 1). Intention to use a technology thereby serves as a mediator of actual use (Davis, 1989). Perceived usefulness describes the extent to which an individual thinks that using a technology will enhance his/her performance. Perceived ease of use means the degree to which the use of a technology is thought of as being free of effort. It not only influences the behavioral intention to use a technology, but often also impacts the perceived usefulness (Davis, 1989).

To encompass voluntary and mandatory usage situations, the TAM has been extended with two „correction factors" (Venkatesh and Davis, 2000). In the extended TAM, *subjective norm* is defined as the extent to which individuals believe that important other individuals want them to use a technology. This construct has been proven to be a significant determinant of behavioral intention to use in mandatory usage scenarios (Venkatesh and Davis, 2000). In contrast, *perceived voluntariness* describes the degree to which individuals perceive a decision to use a technology as non-mandatory. This variable significantly moderates the effect of subjective norm on the behavioral intention to use (Venkatesh and Davis, 2000).

Regarding the behavioral intention of developers to use a software development methodology, *perceived usefulness* was found to be a strong and highly significant determinant (Riemenschneider et al., 2002). Accordingly, we included it into our adopted acceptance model to explain the developer acceptance of Scrum (see Figure 2). Perceived ease of use was found to be nonsignificant, though (Riemenschneider et al., 2002). Instead, *compatibility* - a more specific determinant than ease of use - was found to have a significant influence on developer acceptance (Riemenschneider et al., 2002). This construct describes the extent to which the introduced methodology is perceived as being consistent with the actual needs and past experiences of the developers (Moore and Benbasat, 1991). As Scrum postulates its empirical process control to better represent the reality in software development, we included this determinant instead of the more general ease of use construct.

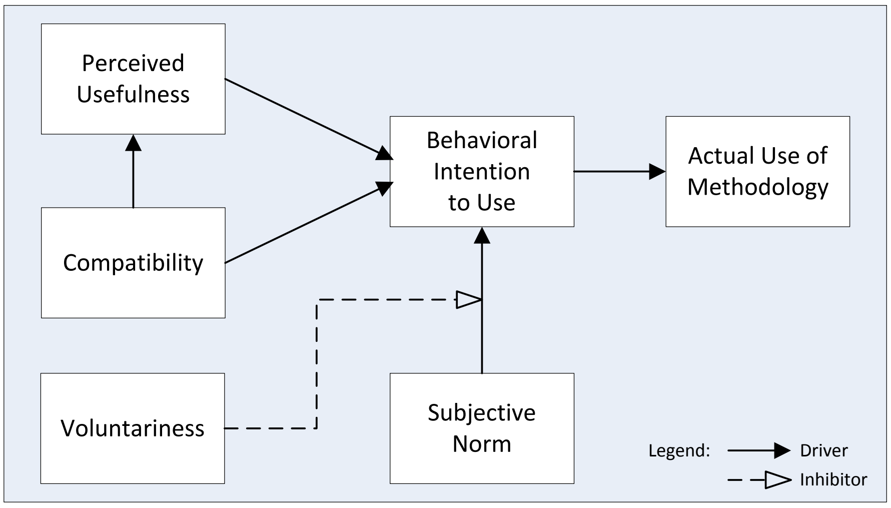

Figure 2. Adopted acceptance model

We also included *subjective norm* and *perceived voluntariness* as relevant factors to describe the acceptance of Scrum, because these factors were proven to have a significant influence on the usage of development methodologies in general (Riemenschneider et al., 2002). In this respect, findings regarding the acceptance of methodologies differ from those for the acceptance of technologies, where subjective norm and perceived voluntariness were oftentimes found to be nonsignificant. Riemenschneider et al. (2002) accordingly concluded that it seems to be less important for the acceptance of development methodologies how easy the behavior prescribed by the methodology is to perform. Instead, the perceived normative pressure and the compatibility of the required behavior with the current way of performing working steps gain more importance.

Figure 2 shows the adopted acceptance model resulting from this discussion along with its constructs. They are used as the theoretical basis to explain the adoption of Scrum by developers and will be refined with concrete influencing factors that have been observed in actual projects.

EXPERTS EVALUATION

To determine such influencing factors, we decided to follow an exploratory qualitative research strategy. In such a setting, „the qualitative interview is the most common and one of the most important [...] tools" (Myers and Newman, 2007) to gather information on a topic. In the following, we describe the design of our interviews and then analyze the obtained results.

Design of interviews

In order to be able to identify relevant factors that influence the constructs of the model introduced in section 4, in-depth information on the topic had to be gathered. In social sciences, so-called *expert interviews* are the most common approach for studies on such a problem (Bogner et al., 2009, Denzin and Lincoln, 2000). Thereby, the number of interviewed experts can be rather low, as long as they were selected carefully. An expert, in this setting, is characterized as someone who bears responsibility for the design, the implementation, or the control of a problem solution or has privileged knowledge on teams and processes. Usually, such an expert is not found at the top of an organization, but on slightly lower hierarchy levels, where the most knowledge on inner structures and events is present (Meuser and Nagel, 2009). Furthermore, as the comparison of traditional development approaches and Scrum is in the focus, experts with profound knowledge in both fields are required.

Additionally, we had to decide on the specific type of qualitative interview. In this case, to achieve the described goals, unstructured interviews are the most commonly used, as they „can provide a greater breadth of data than the other types“ (Fontana and Frey, 2000). Unstructured interviews can either be completely free, or roughly follow a previously defined interview guideline. The latter ones, which were utilized in our approach, are also known as semi-structured interviews. Their concrete form corresponds to the conceptual interview as described by Kvale and Brinkmann (2009).

The interview guideline, which was established to ensure a basically equal structure and content of all interviews in the study, covers a general information part as well as a specific section on development methodologies. In each interview, both parts were completed one after the other. However, there was no strict order of all questions within the sections. In the general part, the interviewee was asked about his experiences and the roles he has taken during Scrum-based and traditional software development projects.

The second part of each interview covered several questions on possible advantages and disadvantages of Scrum in general and in specific project management areas. Furthermore, the interviewees were asked to report their experiences with Scrum in different projects, expected benefits of Scrum in new projects, and possibilities to measure the change in organizational culture associated with a swap to Scrum. All interviews were audio recorded in the first place to avoid distraction of the interviewer and the subject. Immediately afterwards, a transcript of the interview was created by the interviewer. Overall, our interviews closely followed the suggested guidelines for interviewers in IS research as proposed by Myers and Newman (2007).

Analysis of interviews

The study was performed at a large German DAX-30 company, which recently has adopted the Scrum methodology for a considerable portion of their in-house software development projects. The company belongs to the world-wide leading insurance firms and, until a few years ago, has developed all of its projects with traditional methodologies only. Since then, the proportion of Scrum projects is constantly increasing and large parts of the IT personnel have gained mature knowledge in agile development.

All interviews were held within one week in late 2009 and have taken place in the subjects' own offices. Due to their open form, interviews varied between 40 and 60 minutes. Overall, six experts have been included in the study. Three of them performed the role of a Scrum Coach and two took on the role of a Scrum Master or Product Owner. The last expert was an executive from the higher management who was strongly involved in the company's turn to Scrum. All of them had profound experience in Scrum and traditional project management. On average, they had been working with Scrum for 4.1 years and stated their experience in traditional development with 20.3 years. On a scale from one (marginal knowledge) to four (profound expert knowledge) they rated their own knowledge in Scrum with 3.7 and in traditional methodologies with 3.2.

For the identification of possible factors influencing the constructs of the adopted TAM (cf. Figure 2), the recorded interviews were examined for consistencies and distinctive features. Furthermore, correlations to the differences between Scrum and traditional development approaches as deduced in section 3 were determined.

One of the major differences between Scrum and traditional methodologies are the frequent meetings and the opportunity to promptly react to changing requirements. During the interviews, the experts described several advantages for the development teams due to these characteristics. They emphasized better information about the development progress and the ongoing planning for everyone involved. In all previous projects, the „development teams and customers reported the increased *transparency* as a major advantage“, as „unpleasant topics and problems are discussed” in a timely manner. In general, the *planning and scheduling* in Scrum projects was judged to form another advantage over conventional methodologies. Especially the fact that it takes place on different levels has shown to be profitable in practice since it presumably resembles the reality in projects more closely.

In the experts' previous projects the *self-organization* of teams, which enables an „improvement of team performance through more communication”, was also identified as an advantage of Scrum. Furthermore, „teams are better organized with Scrum” as the team members themselves are responsible for the detailed planning. Together with the short Sprints and the reduction of communication bottlenecks, the self-organization of teams furthermore leads to a higher *flexibility* in Scrum projects compared to traditional methodologies. Additionally, the experts reported the simple inclusion of customer requests through prioritization into the Product

Backlog as well as fast reactions of customers during the short cycles, to increase flexibility. As revealed during the interviews, the higher flexibility of Scrum not only benefits the management and customers, however, but also „allows the developers more freedom in their activities” than conventional methodologies.

The *empirical process* control of Scrum is another characteristic that was presumed to benefit all involved parties. According to the experts, such a process control is „more suitable [than] defining almost everything in the beginning” and moreover satisfies the desired flexibility of work as well. The regular short meetings and the close collaboration between all developers were judged to improve the *knowledge transfer* and to enable learning effects between team members. Furthermore, through the permanent involvement of the customer, the interviewees reported that Scrum „increases business knowledge on the IT side.”

The experts, however, also described several possible disadvantages and problems inherent with a swap to Scrum. Among these is the *neglecting of documentation* as they experienced it in Scrum projects. Overall, they judged that the Scrum-specific and comparably abstract „User Stories do not suffice documentation requirements” in general and, further, that the documentation handling in Scrum especially hindered the fulfillment of legal documentation requirements. The fulfillment of documentation requirements and the increased responsibilities of all team members in planning and self-organization, therefore, “requires a particularly *mature team discipline* in Scrum projects“.

Factor	Expert Statements
Transparency	"Unpleasant topics and problems are discussed" "Current development status is well-known" "Velocity enables insights on productivity changes" "Scrum controls overall progress more closely than traditional methods" "User stories increase communication among team members and with product owner"
Planning and Scheduling	"Long- and short-term planning represent reality" "Duration of each development step can more quickly be estimated, which enables better planning" "Long-term planning requires velocity" "Beginning of project remains unconsidered"
Self-organization	"Team decides on course of action, which requires according skills" "Improvement of team performance through more communication" "Team is better organized with Scrum… detailed planning is done by team"
Flexibility	"Higher flexibility through reduction of communication bottlenecks and self-organization of teams" "Requests can easily be included through prioritization" "Transparency and short cycles enable fast reactions of customers"
Empirical Process	"Defining almost everything in the beginning is wrong" "Empirical processes are more suitable"
Knowledge Transfer	"Knowledge transfer between developers through XP practices and close collaboration" "Learning effects among team members" "Increase business knowledge on IT side"
Neglecting of Documen-tation	"User stories do not suffice documentation requisites" "Documentation handling hindered" "Revision requirements need to be fulfilled"
Mature Team Discipline	"Scrum requires a high team discipline"
Resistance to Change	"Large change process for personnel, which needs to consider their growing responsibilities" "Consequences of changes on psychological aspects are often underestimated" "Difficulties of Scrum adoption often underestimated"
Teamwork	"Overall team performance instead of single valuation increases collaboration and enhances teamwork" "Stronger teamwork reduces pressure on single developers"
Team Morale	"Satisfaction increases for most involved parties" "Team satisfaction is measurable" "Increased satisfaction through more fun in development" "Increased team morale if Scrum is done right"

Table 2. Identified factors and corresponding expert statements

All interviewees described the inherent change process as a further obstacle for the adoption of Scrum. In their experience, the „difficulties of Scrum adoption" and the

„consequences of changes on psychological aspects are often underestimated". „The large change process for personnel, which needs to consider their [constantly] growing experiences", oftentimes results in an initial *resistance to change* of many developers. The experts even recognized this as the major factor that hampered the adoption of Scrum in many projects. However, this is not necessarily a Scrum-specific observation, as resistance to change is a well-known general behavioral phenomenon (Watson, 1971).

Due to its concepts, Scrum „increases collaboration and enhances *teamwork*" between the developers more than traditional methodologies. Consequently, a „reduced pressure on single developers" was observed in many projects utilizing Scrum. Finally, the experts judged Scrum to „increase the *team morale* if [it] is done right". They furthermore observed an „increased satisfaction for most involved parties" in general and for developers in particular. As one possible reason for the last observation, they claimed that Scrum projects cause „more fun in development".

Overall, several factors influencing the acceptance of Scrum can be derived from the qualitative interviews. With the derived factors, we were able to confirm indications gathered during the theoretical comparison between Scrum and traditional methodologies. Table 2 summarizes all factors together with the major supporting expert statements. The identified factors, however, are not generally disjoint to each other, but might be partly correlated.

FRAMEWORK OF DRIVERS AND INHIBITORS

The empirical findings gathered during the expert interviews can be used to refine the theoretically motivated acceptance model with concrete influencing factors that were observed in practical projects. To that end, we classified influencing factors as antecedents to those constructs of the acceptance model which they affect. Thereby, we preferably considered theoretical constructs to be dependent variables. Except for the correction factors, relationships between identified influencing factors and constructs are hence unidirectional. A construct was determined as being affected, if its characteristic measures were impacted by corresponding statements of the interviewed experts. For each construct, we therefore examined the characteristic measures (see Table 3) as proposed in the according publications

Construct	Characteristic Measures
Perceived Usefulness	Using Scrum enhances my effectiveness on the job. Using Scrum improves my job performance. Using Scrum allows me to finish tasks more quickly. Using Scrum increases my productivity. Using Scrum makes my job easier.
Compatibility	Using Scrum complies to the way I work. Using Scrum better represents the reality of my job. Using Scrum better satisfies my needs to perform my job.
Subjective Norm	People influencing my behavior think I should use Scrum. People who are important to me think I should use Scrum.
Voluntariness	My use of Scrum is voluntary. The use of Scrum is generally helpful. My supervisor does not need to require me to use Scrum.

Table 3. Characteristic measures of model constructs (Adams et al., 1992, Davis, 1989, Moore and Benbasat, 1991, Venkatesh and Davis, 2000).

For developers, the expectation of increasing *team morale* positively influences the perceived usefulness of the Scrum methodology. As a higher team morale makes the job of developers easier and it is likely to increase the job performance, the perceived usefulness construct is directly impacted (Adams et al., 1992). Higher team morale is also considered as being generally helpful (Venkatesh and Davis, 2000) so that it increases the voluntariness of the decision to accept Scrum in mandatory usage scenarios.

An intensive *knowledge transfer* between team members has a positive impact on the perceived usefulness of the Scrum methodology as the productivity (Adams et al., 1992) is directly increased. The perceived usefulness of Scrum is furthermore positively impacted by the expected increase in *flexibility*, which better allows developers to organize their work according to situational requirements, as well as by the principle of *self-organization*, which helps teams to autonomously decide on the most promising course of action. Both factors contribute to increasing the productivity and making the job of developers easier (Davis, 1989).

The perceived usefulness of Scrum is moderated by expected problems stemming from the *neglecting of documentation* throughout the development process, however. As requirements and architectural specifications are only recorded in a comparatively imprecise manner, it is oftentimes difficult to verify and report if existing requirements and constraints have been satisfied during development. Accordingly, productivity and effectiveness are likely to be compromised (Davis, 1989).

Combining long-term with short-term *planning and scheduling* in an agile way positively impacts the compatibility of Scrum with developer needs. As the development of software is a significantly creative process, the possibility to adjust short-term planning from Sprint to Sprint allows developers to flexibly react to unforeseen complexities as they arise. It hence better represents the reality (Moore and Benbasat, 1991). As a flexible planning of development iterations simultaneously makes the job easier for developers, it also increases the perceived usefulness of Scrum (Davis, 1989).

The compatibility of Scrum is furthermore increased by the expected improvement of *transparency* and *teamwork*. As a consequence of Daily Scrum meetings, developers can expect their information about the current project status to increase. While this does not directly affect perceived usability, it fulfills the developer's need (Moore and Benbasat, 1991) for information and increases the communication with other developers working on the same project. Reported improvements of teamwork, furthermore, seem to not directly affect the perceived usefulness as well, but rather reduce the pressure on individual team members so that creative tasks can be carried out more thoroughly. Similar to the improvement of transparency, this enhances the compatibility of Scrum to the developer's needs (Moore and Benbasat, 1991).

With its *empirical process* control, Scrum introduces a principle to handle development activities as a workflow that needs not to be well understood. Accordingly, the actual development activity is treated as a black box and periodically evaluated in checkpoints, the so-called Daily Scrum meetings. While such a loose process management was not reported to directly increase the productivity or job performance, it was judged to be better suited to represent reality. In fact, software development is often performed as a loosely structured, creative activity that differs from developer to developer. The empirical process control is hence better compatible with actual practice (Moore and Benbasat, 1991).

For loosely managed and self-organized processes, a *mature team discipline* was reported to be a critical success factor by the experts, however. As such a team discipline will have to be actively built first, we classified it to be a moderator for the compatibility of Scrum with actual practice (Moore and Benbasat, 1991). To some part, it can be raised by executing normative pressure, though, as the experts attested that developers are willing to accept higher requirements regarding the discipline in mandatory settings (Venkatesh and Davis, 2000).

The reported *resistance to change* represents a general behavioral phenomenon during the introduction of a new methodology. Any subjective resistance to change decreases the voluntariness of a developer's decision to use Scrum (Venkatesh and Davis, 2000). In part, however, it can be controlled by increasing the normative pressure so that developers feel they should adopt the new methodology.

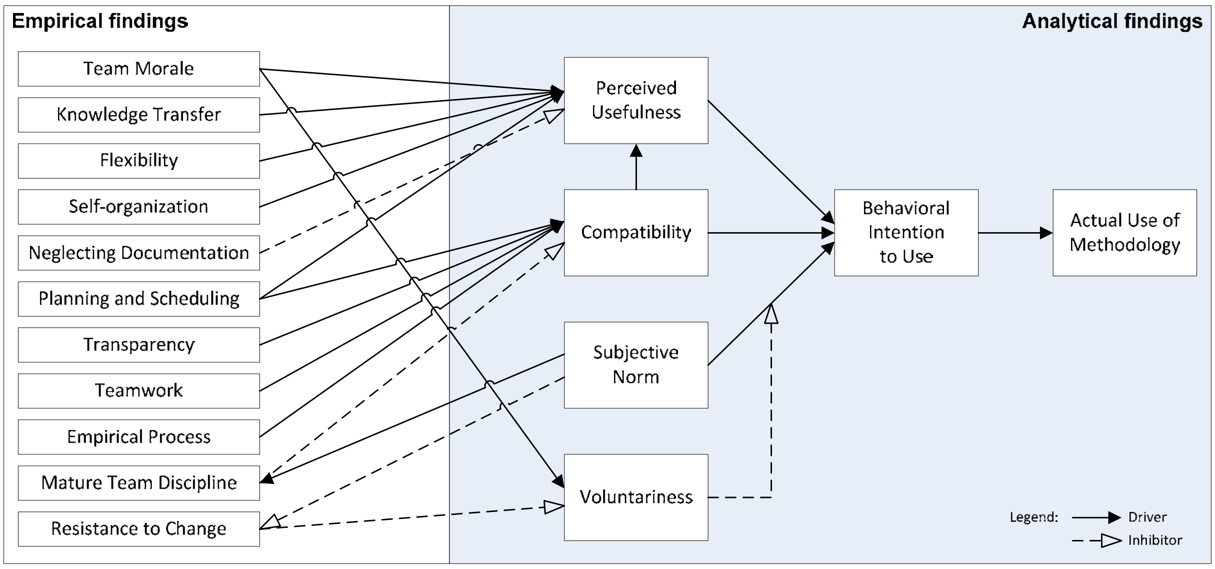

Figure 3. Framework with drivers and inhibitors to the acceptance of Scrum

Figure 3 depicts the resulting framework which combines theoretical and analytical findings in order to show drivers and inhibitors to the acceptance of Scrum. It presents a systematic structuring of human and social factors which influence the acceptance of Scrum from the developer's point of view. As such, it contributes to the building of a theory of relevant acceptance factors, which can e.g. be used to evaluate the acceptance of Scrum in concrete settings. To get to the presented structuring, we followed an exploratory research strategy and used the results of qualitative in-depth interviews. Relationships between influencing factors and theoretical constructs accordingly were included into the framework if they were backed with expert statements. As we did not employ a quantitative approach, the numeric correlations between influence factors and higher-order constructs were not examined yet. Consequently, we are not able to supply information about the question which influencing factors are more dominant than others. Such questions are left as future research directions, instead.

CONCLUSION

In this paper, we have presented a framework of drivers and inhibitors to describe the developer's acceptance of Scrum. As a theoretical foundation, the presented framework makes use of the extended TAM and its concepts, which we adopted to explain the acceptance of methodologies. To refine the theoretical model with influencing factors that can be observed in practice, we have used results of an exploratory study in which we conducted qualitative in-depth interviews with Scrum experts from a German DAX-30 company. The interviewed experts had mature experiences with the introduction and application of the Scrum methodology in industry projects. They were able to provide profound insights into the social and human factors which determine the acceptance of agile methods by the IT personnel. With our research, we hence contribute to satisfy the „need to direct more resources towards investigating the practices of mature teams” in order to examine the true potential of agile methods (Dyba and Dingsoyr, 2008).

The results of our research have implications both for practice and academia. For practice, the presented framework provides a set of influencing factors that can be evaluated to determine the developer acceptance of Scrum in concrete settings. Thereby, the framework can be used to predict developer acceptance before introducing Scrum as well as to validate the level of acceptance afterwards. The conducted expert interviews suggest that it is possible to achieve improvements regarding the team morale, the transfer of knowledge, the flexibility, and the transparency of the development progress when introducing Scrum. However, more demanding requirements with respect to the team discipline and possible resistance to the changes introduced with the agile philosophy might represent major barriers to a successful integration of Scrum. Especially, it seems that a high level of individual autonomy needs to be balanced with a high level of responsibility and interpersonal skills among the team members. As Scrum neglects documentation compared to traditional development methodologies, our findings furthermore suggest that its introduction might become difficult in large projects where multiple interdependencies and higher requirements for the reporting exist. Consistent with the recommendations by others, we hence recommend that practitioners study the project's characteristics and examine them for compatibility with the agile management philosophy of Scrum carefully (Boehm, 2002, Dyba and Dingsoyr, 2008).

As regards academia, our results contribute to the building of theories about the acceptance of agile development methodologies. In recent time, agile development has had a deep impact on the software industry. Researchers therefore need to investigate more closely what has driven the trend and which effects arise from the changes implemented with the turn to agile development methodologies. Apart from economic effects, the agile organization of software development particularly impacts the way in which developers organize their work and collaborate. Social and human factors that lead to the support or resistance of agile methods therefore need to be examined in more detail.

The findings presented in this paper are just first steps into this direction, which need to be complemented with additional qualitative and quantitative research. As regards qualitative research directions, we will need to conduct additional studies both to confirm the identified relationships between influence factors and TAM constructs as well as to identify further correlations which might have remained undiscovered. To provide a more comprehensive view, non-adopters of Scrum will have to be interviewed for inhibitors that caused a return to traditional development. To explain how the inhibitors depicted in the current version of the model can be managed successfully, we will furthermore have to analyze the existing interview data for critical success factors that served as a solution in the examined projects. To make the model more parsimonious, we finally plan to examine to what extent its determinants are suited to also explain the acceptance of other agile methodologies like e.g. Extreme Programming. As the project management strategies of most agile methodologies in fact are similar, analyzing the presented acceptance model for generalizability suggests itself (Abrahamsson et al., 2002).

Quantitative research will on the one hand have to investigate into the numeric effects that the identified influence factors have on the acceptance of Scrum. On the other hand, the model can be used as a theoretical foundation to empirically study the impact of Scrum on social and human aspects of the software development process. For such research activities, structured models with acceptance factors – such as the one proposed in this paper – can e.g. serve as a basis to create questionnaires. The mentioned topics describe future directions that should be addressed in order to get a more complete picture of Scrum and its impact on the developers – a factor the field cannot do without.

REFERENCES

Abrahamsson, P., O. Salo, J. Ronkainen, and J. Warsta. (2002) *Agile Software Development Methods: Review and Analysis*.

Adams, D. A., R. R. Nelson, and P. A. Todd (1992) "Perceived Usefulness, Ease of Use, and Usage of Information Technology: A Replication," *MIS Quarterly* (16) 2, pp. 227-247.

Bahli, B. and E. S. A. Zeid. (2005) The role of knowledge creation in adopting extreme programming model: an empirical study. *ITI 3rd International Conference on Information and Communications Technology: Enabling Technologies for the New Knowledge Society, 2005*.

Beck , K., M. Beedle, A. van Bennekum , A. Cockburn et al. (2001) *Manifesto for Agile Software Development*.

Boehm, B. (2002) "Get Ready for Agile Methods, with Care," *IEEE Computer* (35) 1, pp. 64-69.

Bogner, A., B. Littig, and W. Menz (2009) *Interviewing Experts*: Palgrave Macmillan.

Chong, J. (2005) Social behaviors on XP and non-XP teams: a comparative study. *Proceedings of the Agile Development Conference, 2005*, pp. 39-48.

Davis, F. D. (1989) "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *MIS Quarterly* (13) 3, pp. 319-340.

Denzin, N. K. and Y. S. Lincoln (2000) *Handbook of Qualitative Research*: Sage Publications.

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

Fontana, A. and J. H. Frey (2000) The Interview: From Structured Questions to Negotiated Text, in N. K. Denzin and Y. S. Lincoln (Eds.) *Handbook of Qualitative Research*: Sage Publications, pp. 645-672.

Forrester Research, I. (2007) *Enterprise And SMB Software Survey, North America and Europe*.

Hossain, E., M. A. Babar, and H.-Y. Paik. (2009) Using Scrum in Global Software Development: A Systematic Literature Review. *Proceedings of the International Conference on Global Software Engineering, 2009*, pp. 175-184.

Ilieva, S., P. Ivanov, and E. Stefanova. (2004) Analyses of an agile methodology implementation. *Proceedings of the 30th Euromicro Conference, 2004*, pp. 326-333.

Kvale, S. and S. Brinkmann (2009) *InterViews: Learning the Craft of Qualitative Research Interviewing*: Sage Publications.

Layman, L., L. Williams, and L. Cunningham. (2004) Exploring Extreme Programming in context: an industrial case study. *Proceedings of the Agile Development Conference, 2004*.

Macias, F., M. Holcombe, and M. Gheorghe. (2003) A formal experiment comparing Extreme Programming with traditional software construction. *Proceedings of the Fourth Mexican International Conference on Computer Science, 2003*.

Mann, C. and F. Maurer. A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction. *Proceedings of the Agile Development Conference*, pp. 70-79.

Mann, C. and F. Maurer. (2005) A case study on the impact of Scrum on overtime and customer satisfaction. *Proceedings of the Agile Development Conference, 2005*.

Mannaro, K., M. Melis, and M. Marchesi. (2004) Empirical analysis on the satisfaction of IT employees comparing XP practices with other software development methodologies. *Extreme Programming and Agile Processes in Software Engineering, 2004*, pp. 166-174. Lecture Notes in Computer Science 3092.

Meuser, M. and U. Nagel (2009) The Expert Interview and Changes in Knowledge Production, in A. Bogner, B. Littig, and W. Menz (Eds.) *Interviewing Experts*: Palgrave Macmillan, pp. 17-42.

Moore, G. and I. Benbasat (1991) "Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation," *Information Systems Research* (2) 3, pp. 192-222.

Myers, M. D. and M. Newman (2007) "The Qualitative Interview in IS Research: Examining the Craft," *Information and Organization* (17) 1, pp. 2-26.

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, pp. 1135-1145.

Rising, L. and N. S. Janoff (2000) "The Scrum Software Development Process for Small Teams," *IEEE Software* (17) 4, pp. 26-32.

Robinson, H. and H. Sharp. (2005) The social side of technical practices. *Extreme Programming and Agile Processes in Software Engineering, 2005*, pp. 100-108. Lecture Notes in Computer Science 3556.

Schwaber, K. (1995) SCRUM Development Process. *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications, 1995*.

Schwaber, K. and M. Beedle (2002) *Agile Software Development with SCRUM*: Prentice Hall.

Schwaber, K. and J. Sutherland. (2010) *Scrum Guide*.

Venkatesh, V. and F. D. Davis (2000) "A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies," *Management Science* (46) 2, pp. 186-204.

VersionOne Inc. (2008) *The State of Agile Development, 3rd Annual Survey: 2008*. VersionOne Incorporated.

VersionOne Inc. (2009) *The State of Agile Development, 4th Annual Survey: 2009*. VersionOne Incorporated.

Watson, G. (1971) "Resistance to Change," *American Behavioral Scientist* (14) 5, pp. 745-766.

芽|Sprouts
Working Papers on Information Systems | ISSN 1535-6078