TEAM ADAPTABILITY IN AGILE INFORMATION SYSTEMS DEVELOPMENT

Research-in-Progress

Christoph T. Schmidt
SAP AG &
University of Mannheim, Germany
christoph.schmidt@uni-mannheim.de

Thomas Kude
University of Mannheim
Mannheim, Germany
kude@uni-mannheim.de

John Tripp
Baylor University
Waco, Texas, USA
John_Tripp@baylor.edu

Armin Heinzl
University of Mannheim
Mannheim, Germany
heinzl@uni-mannheim.de

Kai Spohrer
University of Mannheim
Mannheim, Germany
spohrer@uni-mannheim.de

Abstract

While agile development methods have enjoyed widespread adoption, literature has lagged behind in the development of theory to explain its impact on team performance. In this paper, we develop a theoretical model to explain the impact of agile software development on the performance of information systems development (ISD) teams. We first propose a new construct “ISD team agility” which we conceptualize along the iterativeness and collaborativeness of conducting fundamental activities of software development. Based on team adaptability theory as well as on empirical exploration, we then develop a model that explains the impact of ISD team agility on team performance. Central team aspects of adaptive teams are proposed to be directly and indirectly influenced by team agility and, therefore, help us explain the effects of agile ISD on team performance. This research project intends to quantitatively test the model using data from professional software developers.

Keywords: Agile Software Development, Team Adaptability, Team Performance, Shared Mental Models
Introduction

Agile information software development (ISD) has fundamentally changed the way that many companies address challenges in the software development process (Dybå et al. 2008; VersionOne 2012). Traditionally, software development companies tried to cope with uncertainty through intensive upfront planning, detailed specifications of customer requirements, extensive project documentations, and rigid execution of predefined plans. Agile ISD, by contrast, “welcomes change” (Fowler et al. 2001) during the development project and tries to cope with unforeseeable situations by using an iterative approach, combined with close developer and customer collaboration (Cockburn 2001).

Agile ISD “evolved from the personal experiences and collective wisdom of consultants” (Dingsøyr et al. 2012a). There are various ISD practices and methods which have collectively been labeled as agile (Conboy 2009). Some are based on accepted management principles, but in general, there is still a lack of, both, theoretical and empirical support. During the last ten years, numerous studies have been conducted to assess and understand the new approach. However, while some empirical studies have been performed, (e.g. Dybå et al. 2008; Erickson et al. 2005; Maruping et al. 2009), many are anecdotal, or based on single case studies or student experiment data. Thus, a theoretical understanding of the impacts of the agile development paradigm is still lacking (Dybå et al. 2008). This is reflected in the repeated call for theory-based investigations and for more rigorous studies on agile software development (Abrahamsson et al. 2009).

The agile ISD approach follows the general trend in product development organizations towards team-based structures (Salas et al. 2005). Collaborative work in teams promises the potential of greater adaptability, productivity, and creativeness as compared to individuals (Gladstein 1984) and to provide better solutions to organizational problems (Sundstrom et al. 1990). Indeed, teamwork was found to be a key driver for successful software development (Hoegl et al. 2001). Against this backdrop, it is surprising that there are only a few studies which draw on theories from team effectiveness research (Moe et al. 2010). Only recently, Dingsøyr et al. (2012a) called for “better theories and models of software team effectiveness” in agile ISD teams. With this study, we seek to contribute by beginning to addressing this gap.

There is still no general understanding among researchers when to consider ISD teams as ‘agile’. There are two fundamental conceptualizations (Sarker et al. 2009). On one hand, software development teams are described as being agile when they adopt a set of ISD methods that are considered to be ‘agile’, (Conboy 2009). On the other hand, teams are described as agile when they have the “ability to efficiently and effectively respond to user requirement changes” (Lee et al. 2010). Thus, previous research has related team agility either to a certain team behavior (adoption of agile ISD methods) or to an ability of an ISD team to react to changes.

In this study, we take the first perspective and consider team agility as a behavior of an ISD team. Therefore, we do not define team agility by the adoption intensity of various agile ISD methods. This is because the practices proposed by so called “agile” methods (e.g., pair programming, refactoring) are not novel, and can be used in non-agile team environments. Instead, we regard agile ISD methods and techniques as instantiations of a general behavioral team agility concept. Hence, we consider pertinent ISD activities on a team level. Second, we consider team agility as a team-level construct (Klein et al. 2000). We conceptualize ISD team agility as a behavior of the entire team that is reflected in the degree to which standard software development activities are performed in an iterative and collaborative way by the entire team. We further propose how ISD team agility directly and indirectly influences team performance through central team work mechanisms known from team adaptability theory. More specifically, this study aims at answering the following research question:

How does ISD team agility influence the performance of ISD teams?

This study contributes to research on agile ISD by first developing then testing a new concept of ISD team agility. Second, we intend to test our research model that explains effects of ISD team agility on team performance. We intend to do this using data using multiple teams from a very large software development organization. For our model, we draw on team effectiveness research and team adaptability theory. Thus, we respond to frequent calls (Dingsøyr et al. 2012a; Dybå et al. 2008) for theory-based
studies in industrial study contexts. Practitioners may learn from our findings how agile ISD influences team adaptability and team performance in ISD teams.

The paper is structured as follows. The first section presents our new concept of ISD team agility and lays the theoretical foundations of this study. Then, we discuss our proposed research model that explains how agile software development influences teams’ shared mental models as well as teams’ backup and monitoring capabilities. These aspects are hypothesized to directly and indirectly influence ISD team performance. Subsequently, we outline the research design of the study. Finally, the expected contributions of the study are discussed.

Foundations

ISD team agility

Theory-based empirical research on agile ISD requires a precise conceptualization of the studied subject. Our study conceptualizes agility at the team level. Conboy (2009) provides a definition when particular development methods can be considered as agile. Even though this definition is very helpful when studying the effect of particular ISD methods or to categorize them as agile, it is not directly applicable for team-level studies due to multi-level aggregation issues (Klein et al. 2000). How many team members would have to apply which agile ISD methods and to which extent in an ISD team to be considered an agile team? Against this backdrop, we see agile techniques and methods as instantiations of agile ISD team behavior, which is fundamentally different from other development paradigms. This particular agile behavior of a team might be motivated by various project context factors or influence various team emergent states (Mathieu et al. 2008). For instance, agile team behavior might affect the “ability to make the necessary modifications in order to meet new challenges” (Klein et al. 2001), i.e. the adaptability of an ISD team (Burke et al. 2006). Our perspective on ISD team agility is different from the existing team-level conceptualization by Lee et al. (2010). In their paper, they conceptualize agility as an ability of a team to react effectively and efficiently to requirement changes. By contrast, we have a behavioral perspective on ISD team agility as we want to study effects of that team behavior on central drivers of team adaptability.

In software engineering, there is a generally accepted understanding of necessary activities to be done in each software development project - regardless of the development methodology or paradigm that is used. These are (1) software specification & planning, (2) software design/implementation, (3) software validation, and (4) software release & evolution (Sommerville 2007). Arguably, all these activities are essential elements of the traditional plan-driven and the agile development paradigms. As discussed in the introductory section of this paper, this study defines ISD team agility as a behavior of an ISD team. In particular, ISD team agility is defined as the frequency of iterating the aforementioned ISD activities (iterativeness) and the number of people involved in conducting these activities (collaborativeness).

Traditional plan-driven teams split these four ISD activities into separate project phases that are sequentially processed. By contrast, agile ISD teams iterate these activities in small cycles. Hence, we consider the frequency of iterating these activities (iterativeness) as a central distinction factor between the two development paradigms. Furthermore, agile ISD teams embrace a collaborative approach. While the traditional approach focuses on individuals - mostly project managers - who are in charge of planning the project, assigning development tasks to individual developers, and monitoring progress, agile software development emphasizes collaboration between developers and collaborative customer involvement. Traditional and agile approaches differ in terms of the iterativeness and collaborativeness of all fundamental ISD activities (see table 1). We propose a conceptualization of ISD team agility around the aforementioned four core activities of software development and their distinctive iterativeness and collaborativeness. We only consider teams as agile teams that pursue, both, iteratively and collaboratively these tasks. Consider, for example, a team that perceives quality assurance as a team effort, but strictly follows a detailed project development plan. Further, imagine a team that iteratively implements, tests, and releases its software, but all task are managed, assigned, and monitored by a single project manager. Both teams would not fit to our conceptualization of ISD team agility. Hence, we frame team agility as a team-based, multi-dimensional, latent, second-order construct (MacKenzie et al. 2011) that is reflected by a team’s approach to collaboratively and iteratively conduct the previously discussed software development activities.
In order to understand determining factors of ISD team performance, this study draws on team effectiveness research (Cohen et al. 1997; Gladstein 1984; Kozlowski et al. 2006; Mathieu et al. 2008). A long-lasting research history on team effectiveness resulted in a vast amount of models and theories that intend to explain the effectiveness of teams. Salas et al. (2005) consolidated prior team literature streams by reviewing 138 work team models. For teams working in highly dynamic environments as well as for teams facing innovation tasks, team adaptability was found to be an essential antecedent for high performance. Team adaptability was defined as a team’s “ability to change team performance processes in response to cues from the environment in a manner that results in functional team outcomes” (Burke et al. 2006). Team adaptation theory holds that adaptive teams successfully manage to (1) assess situations appropriately and build a coherent understanding of a new situation, (2) adjust their plans accordingly, (3) coordinate their work to fit the new situation, and (4) learn by evaluating their effectiveness. Salas et al. (2005) further propose that high adaptability is positively related with a high ability of the team to provide backup (Marks et al. 2001), a high capability of team monitoring (McIntyre et al. 1995), and high congruence of mental models.

ISD teams constantly face new problems in mostly dynamic work environments (Schmidt et al. 2001). Hence, team adaptation is a key for performing ISD team work (Salas et al. 2005), as adaptive teams are -by definition - able to do the necessary modifications in order to meet new challenges (Klein et al. 2001). Adaptation can be external, i.e. modification of what the team plans to do, or internal, i.e. modification of how something is done. Burke et al. (2006) propose a process model that explains how teams go through adaptive cycles that lead to team adaptation. In accordance with Salas et al. (2005), shared mental models, team backup behavior, and mutual monitoring are central elements of these cycles. Before proposing our research model which builds on these “behavioral markers for team adaptability” (Rosen et al. 2011), we describe our research design in the next section.

### Table 1: Iterativeness and Collaborativeness of Software Development Activities in Agile ISD Teams

<table>
<thead>
<tr>
<th>Iterativeness</th>
<th>Collaborativeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Incrementally specify and prioritize requirements</td>
<td>- Intensive customer involvement</td>
</tr>
<tr>
<td>- Incremental planning (limited upfront planning)</td>
<td>- Team-based planning/discussion sessions</td>
</tr>
<tr>
<td>2. Incremental design</td>
<td>- Team goals (tasks not assigned to individuals)</td>
</tr>
<tr>
<td>- Time-boxed, incremental development</td>
<td>- Collective code ownership</td>
</tr>
<tr>
<td>- Continuous integration of software increments</td>
<td>- Technical decisions are taken by the team</td>
</tr>
<tr>
<td>3. Continuous quality assurance and focus on work excellence</td>
<td>- Culture of collective quality responsibility</td>
</tr>
<tr>
<td>- Continuous refactoring of code</td>
<td>- Collective quality assurance as team effort,</td>
</tr>
<tr>
<td>- Continuous delivery of software increments</td>
<td>- rather than individuals in charge</td>
</tr>
<tr>
<td>4. Frequent delivery of software increments</td>
<td>- Team-based customer presentation</td>
</tr>
<tr>
<td>- Regular presentations of valuable deliverables</td>
<td>- Team-based maintenance / error fixing</td>
</tr>
<tr>
<td>- Small releases</td>
<td>- Collaborative stakeholder involvement</td>
</tr>
</tbody>
</table>

### Team adaptability theory

In order to understand determining factors of ISD team performance, this study draws on team effectiveness research (Cohen et al. 1997; Gladstein 1984; Kozlowski et al. 2006; Mathieu et al. 2008). A long-lasting research history on team effectiveness resulted in a vast amount of models and theories that intend to explain the effectiveness of teams. Salas et al. (2005) consolidated prior team literature streams by reviewing 138 work team models. For teams working in highly dynamic environments as well as for teams facing innovation tasks, team adaptability was found to be an essential antecedent for high performance. Team adaptability was defined as a team’s “ability to change team performance processes in response to cues from the environment in a manner that results in functional team outcomes” (Burke et al. 2006). Team adaptation theory holds that adaptive teams successfully manage to (1) assess situations appropriately and build a coherent understanding of a new situation, (2) adjust their plans accordingly, (3) coordinate their work to fit the new situation, and (4) learn by evaluating their effectiveness. Salas et al. (2005) further propose that high adaptability is positively related with a high ability of the team to provide backup (Marks et al. 2001), a high capability of team monitoring (McIntyre et al. 1995), and high congruence of mental models.

ISD teams constantly face new problems in mostly dynamic work environments (Schmidt et al. 2001). Hence, team adaptation is a key for performing ISD team work (Salas et al. 2005), as adaptive teams are -by definition - able to do the necessary modifications in order to meet new challenges (Klein et al. 2001). Adaptation can be external, i.e. modification of what the team plans to do, or internal, i.e. modification of how something is done. Burke et al. (2006) propose a process model that explains how teams go through adaptive cycles that lead to team adaptation. In accordance with Salas et al. (2005), shared mental models, team backup behavior, and mutual monitoring are central elements of these cycles. Before proposing our research model which builds on these “behavioral markers for team adaptability” (Rosen et al. 2011), we describe our research design in the next section.

### Research design

Research on agile software development has been described to be at an intermediate state (Dybå et al. 2008) for which Edmondson and McManus (2007) propose a mixed-method approach of qualitative studies in combination with quantitative research methods. The authors share this view and pursue a two-stage study approach for developmental research purposes and reasons of completeness (Venkatesh et al. 2013). The first phase of our study was exploratory in nature. Over a period of two years, one of the authors was embedded in one of the largest software companies in the world with several hundred ISD teams. These teams are very heterogeneous concerning their programming languages (Java, C, ABAP) and types of developed software (database software, on-premises and cloud-based business applications, development tools, mobile applications, etc.). Over the past two years, this company has introduced thousands of developers to agile ISD. Each development team participated in a one-week course to get acquainted with the new approach followed by a three-week coaching phase. Afterwards, the teams voluntarily decided whether to apply the learned methods.
For this study, we focus on teams that had voluntarily adopted agile methods. For the exploratory part of the study, the authors conducted more than 30 interviews with developers from various teams, did interviews with trainers and coaches who had, in total, insights into more than 50 teams, conducted interviews with several team managers, and carried out four team observations of at least three working days. Moreover, one researcher participated in the training program. Even though the study was conducted at one single company, the size of the company and the heterogeneity of the teams allowed for a broad diversity of investigated teams. We consider our first research phase as deductive in nature, as we derive a research model to explain effects of agile software development on team performance from team effectiveness research (Cohen et al. 1997; Mathieu et al. 2008; Salas et al. 2005). Even though deductive in nature, the developed research model is empirically informed as we enriched it with our insights into the actual usage patterns of agile software development in a professional ISD context.

In the second research phase, we plan to test the proposed research model quantitatively. We are in the process of developing a questionnaire-based measurement instrument based on insights from the first research phase as well as a review of extant literature (see Table 2 in the Appendix). For those constructs that have not been properly operationalized in existing literature, scales are developed following MacKenzie et al. (2011). The measurement instrument is currently under development; we plan to have first pre-test results ready for presentation at ICIS in order to obtain feedback. The final dataset is intended to include responses from about 70 teams. The performance of the studied teams will be measured with items covering the reliability of a team to deliver software quality and quantity (Henderson et al. 1992; Hoegl et al. 2001). In order to reduce a common method bias and particularly common rater effects, we will not solely collect ratings from team members, but also from customers and managers if possible (Podsakoff et al. 2003).

**Research model**

During the first phase of our study, we intensively observed and interviewed several teams that developed software components at various levels. One team developed database components to store data from various software applications. Another team delivered a software platform which is directly sold to customers, but also used by other teams within the company. Members of both teams described frequent requirements changes in their working context. In yet another team, one senior developer, who drew from more than 15 years of experience, described a remarkable situation of his latest project. The team had been developing software on top of other teams’ software components. Most surprisingly, these software components had not changed for about three months due to various reasons. He emphasized that he had not experienced a similar situation since the beginning of his career as a professional software developer. In general, many developers described a constant change of priorities in their tasks. These were either caused by technology, requirement or task priority changes that were often triggered by new customer expectations or occurring customer bugs which needed to be instantly fixed.
explain performance of ISD teams by drawing from team adaptability theory (Burke et al. 2006) and discuss its main elements. The proposed research model (Figure 1) explains how ISD team agility influences team performance through central behavioral markers of team adaptability (Rosen et al. 2011). Following Salas and colleagues’ (2005) consolidated team work model, a team’s shared mental models, its mutual performance monitoring capability and team backup capability are the central drivers. We propose that these mechanisms are directly and indirectly affected by ISD team agility.

**Shared mental models**

Mental models are organized knowledge structures consisting of the content as well as the structure of the concepts in the mind of individuals (Cannon-Bowers et al. 1993). They can represent knowledge, structures, or relationships. Mental models help people describe, explain, and predict a system or people they interact with. Shared mental models are team members’ shared, organized understandings and mental representation of knowledge about key elements of the team’s relevant environment (Klimoski et al. 1994). They allow team members to interpret information in a similar way which leads to better communication and coordination within a team (Burke et al. 2006; Cannon-Bowers et al. 1993; Klimoski et al. 1994; Mohammed et al. 2001; Salas et al. 2005). Cannon-Bowers et al. (1993) have suggested that task-related, i.e. shared knowledge about the task, and team-related mental models, i.e. shared knowledge about roles and responsibilities of team members, exist in parallel in teams. Both have been found to have a positive impact on the performance of teams.

According to Marks et al. (2001), teams alternate between transition and action periods. During transition periods, the mission of the team is analyzed, goals are specified, and plans are made before actions are taken to accomplish the goals of the team in subsequent action periods. Effective planning before action periods increases the congruence of mental models among team members (Stout et al. 1999). In their study, effective planning included the definition of common team goals, exchange of expectations, role clarification, and sequential timing of tasks. Long-term plans face the risk to be either inaccurate or to require a lot of planning effort. Iterative development helps teams to overcome this tradeoff as teams can make detailed plans for the next increment while limiting the risk of excessive and unnecessary planning efforts. Incremental planning and short, frequent meetings allow team members to align their mental models without spending too much time for worthless planning activities. Many agile teams meet daily for about 15 minutes. During these daily (iterative) Scrum meetings, all team members (collaboratively) make their progress transparent and explicate challenges they face. Agile teams iteratively develop small software increments which are planned accordingly. Consequently, all team members frequently share their understanding of the task their teammates work on (team mental model) and how the tasks are solve (task mental models).

Agile teams regularly present software increments to their customers. A study by Xinwen et al. (2006) indicates that team feedback after task performance episodes increased the congruence of mental models compared to no-feedback situations. Feedback is a central aspect of team agility on different levels. First, developers working with a programming partner continuously offer and receive intra-team feedback on their work. Furthermore, agile teams frequently present software increments to their customers to receive team-external feedback on work results. Both types of feedback provide the entire team with the ability to monitor and reflect on the work. This leads to a high congruence in shared mental models (Gurtner et al. 2007). Team member interaction is one of the prime factors through which shared mental models develop over time (Levesque et al. 2001). When team members communicate with each other, they are more likely to develop a shared understanding of each other and their tasks (Klimoski et al. 1994). Team interaction and collaboration are at the core of agile software development according to the agile manifesto (Fowler et al. 2001).

In summary, iterative and collaborative software development is proposed to positively impact the similarity of task and team mental models in software development teams. Hence, we put forth the following proposition.

**P1:** The higher the ISD team agility, the higher the congruence of the mental models.
Mutual performance monitoring capability

Salas et al. (2005) state in their refined team work model that members of effective teams are aware of team functioning and of changes in the team environment. While carrying out their own work, team members make sure that everyone in the team is following his or her task correctly and in a timely manner. A team’s mutual performance monitoring capability is thus defined as a team’s “ability to develop a common understanding of its environment and to apply appropriate task strategies to accurately monitor teammate performance” (McIntyre et al. 1995). Software development teams mostly face highly abstract, cognitively challenging, non-repetitive tasks. Even the best programmers are not immune to errors. However, software bugs are mostly difficult to detect and, therefore, harmful for a team’s performance. Hence, mutual performance monitoring to find bugs early on is considered a central necessity to ensure high performance in software development teams.

Monitoring each other’s performance requires team members to have an extensive team orientation, i.e. team members believe in the importance of team goals rather than their individual goals exclusively. Moreover, team members must trust each other to not perceive their monitoring teammates as “big brother watching” them, but to see performance monitoring as an accepted norm and opportunity to achieve team goals (McIntyre et al. 1995). A high level of collaboration in the team is expected to contribute to both, team members’ team orientation and mutual trust, because team members learn to know each other better. Several experienced interviewees appreciated a stronger orientation towards team goals in an agile team context. While project managers assigned tasks to individual programmers before agile ISD, they now assigned team tasks. Hence, developers naturally work together to solve them. The high level of collaboration in agile teams makes developers orientate towards the team and build up mutual trust. Agile, iterative development includes various work mechanisms which allow teams to monitor their performance. Iterative planning and status meetings offer team members the opportunity to be familiar with the tasks their teammates are working on. During frequent planning meetings, new requirements are communicated to the team.

Most of today’s programming languages build on the principle of encapsulation. Consequently, it is often not necessary to understand how a module, for instance a class, a function, etc., works, but it is important to know how to use it. On the one hand, it is efficient to use a colleague’s encapsulated functionality without knowing all its details; on the other hand, it is difficult to judge the quality of a colleague’s code quality. Hence, peer monitoring is hampered. Various agile practices, however, induce developers to make errors transparent. For instance, pair programmers constantly provide and receive feedback on the quality of the code they write. Moreover, agile teams often work with automated tests. These tests help developers to continuously check their teammates’ code. One team that we observed had developed an extensive test suite. The team was able to run different test modules to monitor different aspects of their code base. For instance, when making changes in the persistence layer, they hypothesized which modules might be affected and were able to falsify these hypotheses within minutes. This setup allowed the team to continuously monitor their own as well as their teammates’ performance. When errors were detected, everyone in the team was informed via a centrally exposed traffic light in the room that turned red. One team described that due to their fine-meshed test suite, the team was able to integrate a new teammate within three months into the team to productively contribute to the team’s performance. Without pair programming and the test suite, this would have taken at least six months according to their estimation. The new developer was able to quickly write productive code as the test suite was a “security net” for him. In another team, developers explained that the test suite also helps experienced developers “to sleep well at night” as they know that they are not the only one to check the modified code for sever errors during the day, but that their peers keep their eyes open, too. Finally, yet another team explained to have invested four weeks with the entire team to find a simple bug in their legacy code that had not been properly covered with tests. The team stated that if they had a single unit test at the right spot, they could have avoided all these efforts.

P2: The higher the ISD team agility, the better the team’s mutual performance monitoring capability.

P3: The higher the congruence of mental models in the team, the better the team’s mutual performance monitoring capability.
**Team backup capability**

Team backup capability is defined as the ability of a team to provide resources or feedback to a team member to help obtain his or her task goals when potential task failure is apparent (Porter et al. 2003). Marks et al. (2001) identify three means of providing backup, i.e. (a) providing feedback and coaching to improve performance, (b) to assist teammates in performing a task, and (c) to complete a task when overload is detected. Backup behavior is particularly important for software development team performance as research has shown that individuals may not be aware of their own performance deficiencies (Salas et al. 2005). To effectively engage in backing up behavior, team members must be able to not only detect needed help (monitoring capability), but to effectively judge the trade-off between providing help or feedback and accomplishing their own task. Consequently, a common understanding of tasks and team member responsibilities (shared mental models) is essential to effectively provide assistance or performance feedback with regard to team objectives. Only if team members understand their teammates’ individual task, they can provide feedback or assist the performance of other team members (Dickinson et al. 1997). Through mutual performance monitoring, developers realize when teammates have problems and through customer monitoring developers can better judge whether helping others or finishing their own task is advantageous for the entire team’s effectiveness. Hence, both team adaptability behaviors will increase a team’s capability to provide team-internal backup.

P4: The higher the congruence of mental models in the team, the better is the team’s ability to provide backup behavior.

P5: The better a team’s mutual performance monitoring capability, the better is the team’s ability to provide backup behavior.

**Team performance**

Following Salas et al. (2005), we propose that team backup behavior boosts the performance of a group of individual developers to be more than the sum of the individuals’ performance. Team performance is specified as the reliability of the team to deliver the promised scope and quality and an assessment of the software quality of the team. Backup behavior, in turn, is only made possible through the information gathered through mutual performance monitoring and provided through feedback and assistance, if required. In software development, small bugs can have significant effects on the entire system. Often, semantic software errors are not immediately detected. This may lead to fatal consequences, for instance, unpredictable system behavior or unpredictable efforts to fix problems. Hence, the ability of a team to detect such errors early on, e.g. by calling their peers’ attention to flawed software code, is at least as important as a teams’ ability to proactively assist team members to accomplish critical tasks when facing problems.

Non-detection of errors might have different negative effects for the performance of the team. Obviously, the quality of the software might deteriorate over time as unpredictable software behavior will occur. Second, the team will have to spend more time on fixing surfacing problems rather than developing new features. Finally, fixing bugs is mostly a time-consuming and unpredictable challenge. Hence, the team is not able to predict and plan its development velocity anymore. As a consequence, the team appears unreliable to its stakeholders regarding the promised software quality and feature scope. Through feedback provision and assisting of development tasks, developers can shift their priorities and workload in the team to counteract such effects which, in turn, will make the entire team better performing from the perspective of its stakeholders.

P6: The better the ability of a team to provide backup behavior, the better is its performance.

**Expected contributions**

Our study is expected to contribute to research on agile software development in various ways: Dingsøyr et al. (2012a) have provided a list of priorities for future research steps in this field that are expected to be answered by this study: (1) better measurement instruments to team behavior and team performance, (2) more rigorous industrial empirical studies, and (3) emphasis on team cognition. A deep understanding of this highly relevant topic among practitioners is still lacking. This study attempts to fill this gap. First, we propose a new conceptualization of the still vague construct of ISD team agility. Based on the central activities of software engineering, we propose that agile teams iteratively and collaboratively conduct
these very activities. Previous studies have mainly measured the use of selected agile methods and practices. With our approach, we intend to provide a way to abstract from the usage of selected methods to the central aspects of ISD team agility. Furthermore, we have been deducing a research model – with in-depth insights from various observational, exploratory studies – in order to explain effects of the ISD team agility on ISD team performance through team adaptability. The theoretical model will be tested empirically with professional software development teams. Hence, this study replies to frequent calls for theory-based and empirical studies (Dingsøyr et al. 2012a) on agile software development and its performance effects (Abrahamsson et al. 2009; Conboy 2009; Dingsøyr et al. 2012b).

References


## Table 2: Construct Definitions

<table>
<thead>
<tr>
<th>Construct</th>
<th>Definition</th>
<th>Reference</th>
<th>Items</th>
</tr>
</thead>
</table>
| ISD team agility                 | Frequency of iterating the following ISD activities: (1) software specification & planning, (2) software design & implementation, (3) software validation, and (4) software release and the number of people involved in conducting these activities                                                                                                                                                                                                                                                                  | (Self-developed)  
*How often does the team conduct the following activities (1-4)?  
*How many people are involved in conducting these activities (1-4)* | (Cannon-Bowers et al. 1993; Mohammed et al. 2001; Rentich et al. 1994)  
Each team member rates his or her agreement on a Likert scale for three randomly selected team members ($x_1, x_2, x_3$). The answers of all team members are averaged.  
**Task mental model**  
• $x_i$ and me, we have a common understanding of the product vision that we develop.  
• For the last three sprints, I could easily explain what $x_i$ worked on.  
• When co-working with $x_i$ we have a similar model about our software code base in mind.  
• $x_i$ and me, we have a similar understanding of how good code looks like.  
**Team mental model**  
• I know exactly $x_i$’s software development skills and experience.  
• I know the areas of expertise of $x_i$.  
**Assistance**  
• At the end of a typical sprint, I know for all team members how good they finished their personal tasks.  
• At the end of a typical sprint, we recognize when my teammates have quality issues in the software they contributed.  
• As a team, we realize when someone has not finished his tasks correctly.  
• We quickly realize changing customer requirements.  
**Team performance**  
• A team’s reliability to deliver a promised software scope and assessment of the software quality the team delivers  
*These questions are answered by a team’s stakeholder.*  
**Software quality**  
• The team’s deliverables are always of high quality.  
• The customers are/will be satisfied with the software quality of the team.  
• The team’s software advances the company’s image as a high quality company.  
• This team has a reputation as a “high quality team” among other teams.  
**Reliability**  
• The team is reliable with respect to its commitments and deliverables.  
• I am satisfied with the progress of the team.  
• The team reliably delivers at the end of the sprint what it promises before the sprint. | | |