ADAPTATION PATTERNS IN AGILE INFORMATION SYSTEMS DEVELOPMENT TEAMS

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

Saskia Bick
University of Mannheim, Mannheim, Germany, bick@uni-mannheim.de

Christoph Schmidt
University of Mannheim, Mannheim, Germany, christoph.schmidt@uni-mannheim.de

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

Thomas Kude, Saskia Bick, Christoph Schmidt, and Armin Heinzl, 2014, "ADAPTATION PATTERNS IN AGILE INFORMATION SYSTEMS DEVELOPMENT TEAMS", Proceedings of the European Conference on Information Systems (ECIS) 2014, Tel Aviv, Israel, June 9-11, 2014, ISBN 978-0-9915567-0-0
http://aisel.aisnet.org/ecis2014/proceedings/track13/11

ADAPTATION PATTERNS IN AGILE INFORMATION SYSTEMS DEVELOPMENT TEAMS

Complete Research

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

Bick, Saskia, University of Mannheim, Germany, bick@uni-mannheim.de

Schmidt, Christoph, University of Mannheim, Germany, christoph.schmidt@uni-mannheim.de

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

Abstract

This research draws on team adaptation theory to study how agile information systems development (ISD) teams respond to non-routine events in their work environment. Based on our findings from a qualitative case study of three ISD teams, we identified non-routine events that could be distinguished according to the three categories task volatility, technological disruption, and team instability. In addition, we found three patterns of reacting to these events that differed regarding complexity and team learning. Our results show that the theoretical link between different types of events and adaption patterns depends on the type of event and the reach of the events' impact as well as on the extent to which the teams followed an iterative development approach. While previous literature either examined ISD team agility as the extent to which agile techniques and methods are applied, or as a capability to adapt to changes, this research is the first to study how more or less agile teams react to non-routine events. By taking a process view and examining the influence of iterativeness on the link between events and adaptation patterns, this study helps reconcile the behavioral and capability perspectives on agility that have so far been disconnected.

Keywords: Information Systems Development; Software Development Teams; Agile Software Development; Team Adaptation; Iterativeness

1 Introduction

Information systems development (ISD) has increasingly shifted from plan-driven to agile approaches in recent years (Dybå & Dingsøyr 2008). Traditional software development relies on extensive up-front requirements analysis, documentation, and sequential execution of predefined plans to cope with challenges in the development process (Sommerville 2007). Agile ISD, by contrast, suggests a more iterative approach with short development cycles and collaborative work within development teams, combined with frequent customer interaction (Cockburn 2006). In order to implement agile ISD, several methods and techniques have become popular, such as eXtreme programming or Scrum (VersionOne 2012). Existing research and anecdotal evidence suggest that by applying these methods and techniques, agile ISD teams may be able to develop better software and to better meet customer requirements (Lee & Xia 2010; Balijepally et al. 2009; Sfetsos et al. 2006; Cao & Ramesh 2008).

Previous work mainly ascribed these improved outcomes to an enhanced adaptability of ISD teams, i.e. to an improved ability to effectively and efficiently respond to changes (Lee & Xia 2010). In fact, "welcoming change" has been one of the foundational pillars of agile software development approaches (Agile Manifesto, 2001; Conboy, 2009). However, it is so far not clear how applying agile methods and techniques (i.e., agile behavior) is theoretically linked to an ISD team's ability to adapt to changes (i.e., agile capabilities). In particular, a theoretical understanding of the conditions under which ISD teams with more or less agile behavior are indeed better able to react to changes is still missing (Dingsøyr & Dybå 2012; Dingsøyr et al. 2012). Such an improved understanding would allow for more specific theorizing on ISD team agility and for deriving actionable insights regarding the situations in which agile methods and techniques can be effectively applied. In fact, team adaptation processes that are triggered by non-routine events have been intensively studied in social psychology literature (Burke et al. 2006; Rosen et al. 2011; Salas et al. 2005). Yet, in order to transfer these insights to agile ISD, the idiosyncrasies of ISD teams have to be taken into account. As such, the specific events that call for adaptation within ISD teams are so far not clear. While some authors focus on changes in software requirements (Lee & Xia, 2010), others refer to environmental or various other changes in a more abstract way (Henderson-Seller 2005, Erickson et al. 2005). Moreover, while there is general agreement about the steps involved in team adaption (Burke et al. 2006), the question of how more or less agile ISD teams vary in performing these steps is yet to be answered.

Hence, the goal of this study is to theoretically link different types of ISD-specific non-routine events, varying degrees of ISD team agility, and consequent team adaptation patterns. In order to achieve this goal, we conducted an in-depth, exploratory study of three ISD teams within a large software company. Our multiple-case study draws on interviews with software developers and managers as well as observational data from team meetings and informal interactions within the teams. While the three selected teams had all implemented agile methods and techniques to a certain extent, they varied in terms of the cycle times with which software was delivered to the customers. Thus, we focus on the impact of iterativeness as one key aspect of agile behavior on the link between non-routine events and adaptation processes. The next section provides a literature review on agile ISD and presents team adaptation as discussed in social psychology. We then introduce our research design and report our empirical findings. Finally, we discuss our results in light of existing literature.

2 Literature Review and Theoretical Foundations

2.1 Agile Information Systems Development

Two fundamentally different perspectives on ISD agility have been assumed in prior literature. On the one hand, studies have conceptualized agility as a behavior. In most cases, these studies referred to agile ISD teams as those that apply a certain set of agile ISD techniques or methods (Conboy 2009). As such, more and more ISD teams in professional software development companies rely on Scrum (Schwaber

& Sutherland 2011; VersionOne 2012). In its essence, Scrum is a project management method that prescribes certain roles (software developers, Scrum master, product owner), a specific cycle time called Sprint (usually two or four weeks), well-defined meetings (such as daily stand-up meetings, Sprint planning and review meetings, as well as team retrospectives), and certain processes that specify how work items are designed and distributed among team members (Schwaber & Sutherland 2011). In addition, various agile development techniques, for instance pair programming, test-driven development, or refactoring, have been proposed under the umbrella term eXtreme programming (Beck & Andres 1999). Given the prevalent trend among many software development companies to apply these practices, previous literature often assessed an ISD team's agility in terms of its intensity of adopting such techniques and methods that are considered to be "agile" (c.f. Maruping, Venkatesh, & Agarwal, 2009; Maruping, Zhang, & Venkatesh, 2009). While this approach yields value when studying the impact of individual agile techniques and methods, such a view does not allow for studying the agility of ISD teams independently of currently applied approaches. To our best knowledge, there is no unambiguous indicator reflecting the agility of an ISD team based on the usage intensity of different agile techniques. Recent conceptual attempts to overcome these challenges suggested to abstract from specific techniques and instead measure agility in terms of the degree to which an ISD team performs standard software development activities in an iterative and collaborative way (Schmidt et al. 2013). In this study, we build on this conceptualization and focus on a team's iterativeness.

On the other hand, instead of an actual behavior, existing studies conceptualized ISD agility as a capability of an ISD team. Most often, this capability was interpreted with the notion of being able to adapt to changes. As such, agility has been characterized as promoting the ability to adapt to various changes (Henderson-Sellers & Serour 2005), as an organizational learning capability (Lyytinen & Rose 2006), as "the continual readiness of an ISD method to rapidly or inherently create change" (Conboy 2009, p.338), or as "a software team's ability to efficiently and effectively respond to user requirement changes" (Lee & Xia 2010, p.88). While there is general agreement about the key role of responding to changes when studying agility, such a capability perspective only allows indirect observations and consequently remains rather vague regarding the specific triggers of adaptation and distinct ways of responding to changes. In this study, we aim at providing a more integrated perspective on behavioral and capability aspects of agile ISD teams. In fact, our key assertion is that adaptability is an important capability of ISD teams and may be an *outcome* of agile behavior, but that team adaptation is also a process that may take different shapes. To take such a behavioral perspective on team adaptation, we borrow from social psychology literature.

2.2 Team Adaptation

For teams working in highly dynamic environments or facing innovation tasks, a team's adaptability was found to be a key determinant of team effectiveness (Burke et al. 2006). Previous literature defined such team adaptation as "a change in team performance, in response to a salient cue or cue stream, that leads to a functional outcome for the entire team" (Burke et al. 2006, p.1190) . Essentially, 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 of its performance. Rosen et al. (2011, p.108) further specify team adaptation in stating that it is "a complex phenomenon, one comprising multiple inputs, interaction processes, and emergent states that results in event-driven changes in team properties and processes, enabling higher levels of effectiveness in complex environments".

During the first phase of team adaptation – *situation assessment* – the environment is scanned by team members for potential cues. A *cue* is any kind of non-routine event, whether previously known or unknown, that has the potential to disturb or affect the current process (Louis & Sutton 1991). First, the cue has to be recognized and identified as being a source of possible disruption for the ongoing process. Then, the individuals who recognize the cue ascribe a certain meaning to the raw data based on previous experience. After interpreting the situation, the information is communicated to the rest of the team

(Burke et al. 2006; Rosen et al. 2011). During the *plan formulation* phase, the team works on a plan how to address the cue in order to still meet the originally desired goal. This includes a mission analysis, creation of a ‘Plan A’ and a contingency ‘Plan B’ (Marks et al. 2001), differentiation of team member roles (Kozlowski et al. 1999), and conflict management. The third phase of the adaptive cycle, called *plan execution*, is the actual performing phase. During this phase, several processes on the individual and the team level happen dynamically, recursively, and simultaneously. The central process during the plan execution phase is coordination. To successfully coordinate, mutual monitoring, back-up behavior, and systems monitoring are key processes (Marks et al. 2001; Salas et al. 2005). The last phase of Burke et al.’s adaptive cycle describes *team learning*. During this retrospective phase, the team recaps prior actions to build a common understanding of what has happened. Finally, the team formulates lessons learned with the aim to benefit from them in similar situations in the future. Figure 1 illustrates a simplified adaptive cycle.

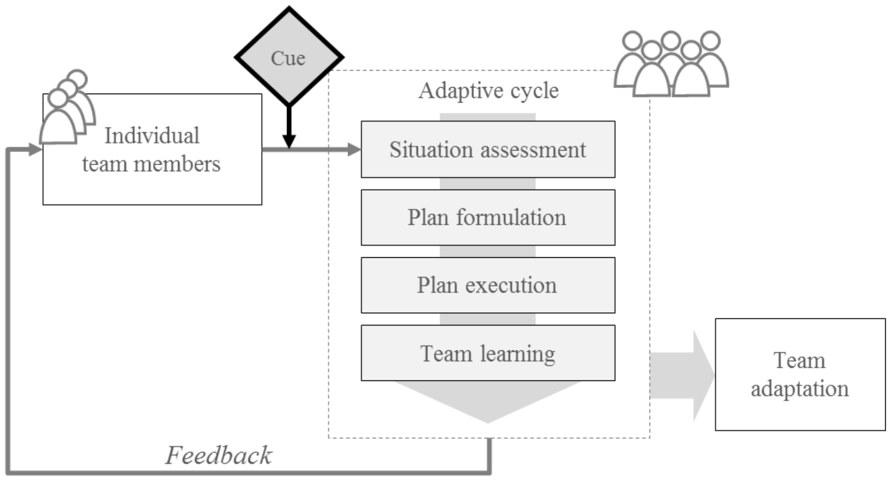

Figure 1: Simplified Team Adaptation Model (based on Burke et al., 2006; Rosen et al., 2011)

Despite the growing importance of team work in ISD and the dynamic nature of software development, there is a substantial gap in the literature regarding the topic of team adaptability in IS (Dingsøyr et al. 2012). Recently, Schmidt et al. (2013) and Tripp (2012) applied the model of team adaptability to study the relationship between agility and performance. However, these studies feature a strong variance character. Instead, our goal is to shed more light on the processes of team adaptation in the context of ISD. In particular, the non-routine events that trigger the adaptive cycle have not yet been examined in detail. Furthermore, no attempt has been made so far to identify team adaptation patterns of more or less agile ISD teams and to link them to different cues.

3 Research Design

Investigating team adaptation patterns of ISD teams requires an in-depth understanding of the team and its working context. Therefore, we chose a qualitative research approach (Yin 2009) and conducted a multiple-case study that included three ISD teams (Eisenhardt 1989). Our study was exploratory in nature and guided by the team adaptation model introduced above. The setting of our exploratory case studies was a large enterprise software vendor that has been using Scrum and agile techniques for several years. Two of the authors were embedded as researchers in the studied organization for over two years and six months respectively. Thanks to this research setup, we were able to obtain a fair understanding about the company’s development organization that helped us examine the phenomenon of team adaptation from multiple perspectives.

All three teams worked within the same development unit having a similar setup of team members’ skills and experience. The three case teams were purposefully selected to vary in their frequency with which their software product is developed and delivered to the customers (release-to-customer, RTC).

With short development cycles as a key aspect of agile software development, the RTC represents one central facet of ISD team agility. Besides the similar organizational context, additional criteria such as Scrum experience, cultural background, and Sprint length were sought to be comparable across the teams to control for other effects that may influence team adaptation behavior. Table 1 presents more detailed information on the three studied ISD teams.

	PEACH	CHERRY	MANGO
Team Members	9	12	10
Team's Scrum Experience	> 2 years	> 2 years	> 3 years
Sprint Length *[in weeks]*	2	2	2
RTC Length *[in weeks]*	2	26	4
Software Product	Cloud-based platform-as-as-service software component	Business process monitoring software application	Cloud-based platform-as-as-service software component

Table 1. Team Context Details

We conducted our data collection over a period of three months during summer 2013 by means of on-site face-to-face interviews and observation of several team meetings. To triangulate our findings and avoid methodological learning effects between the three case studies, we pursued an iterative data collection and analysis approach (Yin 2009). More specifically, we initiated our study with a series of eight open interviews including experts on agile software engineering, senior software developers, and leaders of agile teams. Based on these insights, we developed an initial categorization of non-routine events for ISD teams. Afterwards, we conducted semi-structured interviews with the product owner and Scrum master of each of the three studied teams (PEACH, CHERRY, and MANGO). These interviews helped us better understand each team's particular working style and challenges, and also served as another round of refining our categorization frame. We kept records of the provided answers via extensive note-taking. Third, based on the gathered insights, structured interviews with five developers from each team were conducted to learn about non-routine events and how the teams reacted to these events. In this phase, 15 interviews of 30 to 40 minutes were recorded and transcribed. The critical incident technique (Langley 1999) helped us to find details about specific events in the past. More-over, it mitigated the drawback of not having the possibility for a longitudinal data collection. Finally, informal follow-up interviews with one key developer of each team as well as two software architecture experts served to clarify inconsistencies among the collected information. In addition to these various interviews, we also directly observed the teams in their daily business. This included the attendance of daily Scrum meetings, sprint planning meetings, and retrospective meetings during which we intensively took notes on a pre-defined template for later analysis.

We analyzed the transcribed data with the software tool NVivo 10. In line with our research question, we looked for three different aspects when making sense of our data. First, we coded for interview statements related to non-routine events (i.e., cues) that affected the teams. This coding resulted in a total number of 57 different cues that were found to trigger adaptation processes. By comparing the various cues, we were able to consolidate 57 cues into a total number of 39 types that represent cues which are comparable across different circumstances and teams (see Table 2). Moreover, a structure emerged that categorizes all identified cues into three broad categories and three sub-classes for each category. In addition, the reach of the cue's impact emerged as a property that helped us distinguish cues that stem from a lower/higher level of the organizational hierarchy or from less/more fundamental technologies (i.e., higher/lower levels of the technology stack). After several adjustments to this structure during the process of data analysis, we were able to assign all encountered non-routine events within the three teams in an exhaustive and unambiguous way. Second, we identified episodes in the data that represent sequences of reactions to non-routine events. We coded these episodes according to the four steps of team adaptation introduced above. This process resulted in a total number of 39 episodes, one for each identified cue type. By abstracting from context-specific peculiarities and studying systematic similarities and differences between the episodes regarding the four steps of team adaptation, we were able to identify three typical reaction patterns (Langley 1999). Third, the findings gathered from the interviews, observations, and secondary data were, in a first step, analyzed in a within-case analysis to gain familiarity with the data (Yin 2009). Then, a cross-case analysis served to compare and theoretically link types of non-routine events with applied adaptation patterns and unearth contingency factors of this link (Sabherwal & Robey 1995). The results of our exploratory multiple-case study are presented next.

4 Findings

4.1 Cue Categorization

The types of non-routine events that emerged from our empirical analysis are shown in Table 2, categorized into the identified classes and sub-classes. The table further distinguishes types of cues regarding the reach of the cue's impact. The three categories of cues that emerged from our empirical analysis are *task volatility*, *technological disruption*, and *team instability* (Ferreira et al. 2009; Hollingshead et al. 2010; Carbonell & Rodríguez-Escudero 2009; Song & Montoya-Weiss 2001; Slotegraaf & Atuahene-Gima 2011). Many cues referred to *task volatility* which comprises new requirements, requirement reprioritizations, and ex-post requests. For instance, one developer within CHERRY commented on a change request, an instantiation of a requirements reprioritization: "All of a sudden, there are things that become important. These things could have been planned for a long time, because they were not new. But then, for whatever reason, the priority has changed and it became extremely important and urgent. For instance, right before the release, certain features became important for one customer, and then we had to turn around by 180 degrees." Other examples mentioned by the interviewees include bug reports or additional information about existing requirements.

The second category of non-routine events, *technological disruption*, is caused by technological novelty or technological turbulence. In the context of our study, technological disruptions can be subdivided into platform issues, program-related issues, and external issues. The cue types within these groups found in the three ISD teams comprised, for instance, non-routine events such as database breakdowns, incompatible new platform versions, changes in the shared code line or infrastructure, or problems with the test framework. For example, a member of PEACH reported about problems with the so-called main build: "Our platform, the main build, cost us several days recently. There were certain things that didn't work well, and we had to come up with a workaround". Another member of PEACH explained that his/her team was actively involved in addressing the problem, even though it was not part of PEACH's original responsibilities. Similarly, a member of MANGO commented on changes in open source software the team draws on: "If there are updates in the open source software, we cannot wait too long to implement them. Otherwise it's highly likely that our own code won't work anymore. Sometimes we are doing workarounds and contributions ourselves."

The third group of cues reported by the teams comprises issues related to *team instability*. A stable team consists of members who have known each other for quite some time. Team instability may imply severe impacts on cognitive structures of ISD teams and, hence, be harmful for team effectiveness (Davern et al. 2012). Reported non-routine events were, for instance, changing product owners, new or leaving team members, cross-functional teams, or the addition of another sub-team. A member of PEACH gave examples for such a case of team instability: "Because [the new platform] is important for the organization, there are new project groups being set up. Right now, one of my colleagues has been working for such a project group and not for our team for more than a year. I did the same for several months. These are the more serious things, other than that: Of course, people from the team call in sick from time to time."

		Organizational/technological impact	
		Local	Broad
Task Volatility (via product management)	New Requirements	− New feature for existing product − New module	− New user interface (UI) − Feature for new product − Event-related feature request
	Reprioritizations	− Additional information about a particular user story	− Additional information about a particular requirement − New 'hype' topic − Change in an existing requirement
	Ex-post Request	− Internal consultancy request − Internal bug report − Internal maintenance request	− External/customer bug report
Technological Disruption (via platform, program, ext.source)	Platform Issues	− Change of proxy server − Regression bug	− Database breakdown − Platform problems − Incompatible, new platform version − Infrastructure changes − Network issues − Server update
	Program-Related Issues	− Change of runtime environment − New test framework	− Incompatibilities in the shared code line − Main build problems − New programming language
	External Issues	− External browser update − Open source software update	n.a.
Team Instability (via line management)	Internal Changes	− Team member leaving team − Parental leave − New team member joining	− Team reorganization
	Leadership Changes	− Scrum master fill-in	− Product owner change
	External Requests	− Team member to work in task force − Team member to work in other team − Team member illness / vacation	− Team member to work in other cross functional team − Team member to work in other project

Table 2. Categorization of Cues

In addition to the three categories, the cues can also be characterized in terms of the reach of the cue's organizational or technological impact (local versus broad). For the categories task volatility and team instability, local or broad impact refers to the level in the organizational hierarchy the cue originates from. For instance, the source of task-related cues could be the solution or product manager (e.g., feature for a new product), thus having a broader impact within the organization, or a peer team (e.g., internal bug report). Similarly, team instability can have its origin within the team (e.g., team member calling in sick) or outside the team (e.g., product owner change). Technological cues do not always originate from organizational hierarchies. Rather, the impact of technological cues is reflected by its location in the architectural stack.

In order to clarify the impact of the identified cues, one program manager and one software architect of the case organization were consulted as additional experts. In two separate sessions, the experts were asked to cluster and sort the cues, which were written on cards prior to the meetings. Both agreed on the viewpoint that depending on where in the architectural stack the cue comes from, its reach and thus its potential impact differs. For example, a change of the runtime environment might potentially have a more narrow impact than a database breakdown, which likely influences wider parts of the organization.

4.2 Adaptation Patterns

In order to better understand how agile ISD teams adapt to non-routine events, we analyzed the empirical episodes that were triggered by different types of cues. This analysis was guided by the four general steps identified by Burke et al. (2006), i.e., situation assessment, plan formulation, plan execution, and team learning. Interestingly, for each of the four steps, we were able to group the observed adaptation episodes according to rather clear dimensions. As such, in the situation assessment phase, we found teams to either respond to a cue with *one or few* assessors or with the *entire team involved* in the assessment. During the plan formulation phase, the team behaviors could be distinguished depending on whether a *routine is available* or if the team needs to actively go into a *team discussion* to prioritize and formulate a plan. The plan execution phase differed as to whether *little* or *intensive coordination* was needed. Team learning, the last phase of the team adaptation cycle, could be characterized by either *few* or *many learnings* for the team. Thus, considering all possible permutations, a total of 4^2=16 patterns exist on the theoretical plane. However, only three adaptation patterns were empirically observed that reflect systematic similarities and differences between the identified adaptation episodes. Figure 2 illustrates these patterns. Table 3 provides more details on the three patterns, together with rich data from our empirical findings. In particular, two typical episodes for each pattern are presented.

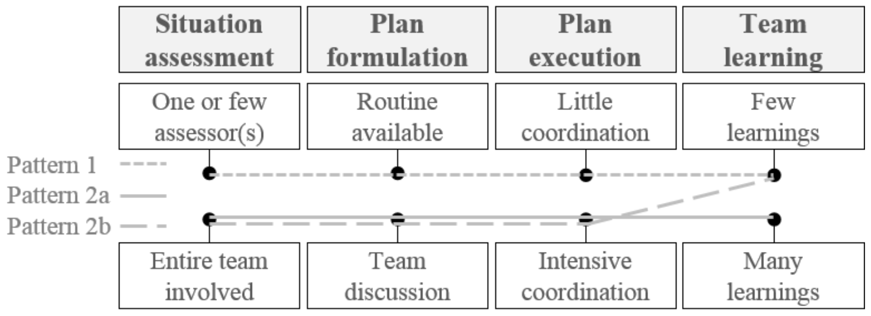

Figure 2. Overview of Adaptation Patterns

Pattern 1 describes a routinized reaction process, which teams have established over time. Typically, one or few team members recognize the cue and assess its nature. Then, the assessor(s) evaluate the problem by estimating urgency, potential severity, and effort to solve the issue. Established routines exist for these types of cues, so that little active and collaborative planning needs to be done. As these cues can usually be handled by one or very few developers who rely on routines for the plan execution, little coordination effort is needed. Given that established routines are applied that have already existed before, there are only few learnings for the team.

By contrast, *Pattern 2a* is a much more complex pattern. In the first phase, the whole team is involved in assessing and evaluating the situation, either because the trigger reached the team via an unknown channel, or because it puts the whole team's activities on hold. Then, during the planning phase, the team engages in a discussion on what to do and how to proceed. Oftentimes, there are no routinized actions to fall back on. In some cases, emergency measures such as escalating the problem to the management are discussed. After agreeing on a more or less structured plan, intensive coordination is needed to execute it. In case a team decides to escalate the issue, most development activities are set on hold. Otherwise, the majority of the team is needed to solve the issue and if only one or few team members can actually be active in finding a solution, still a high level of coordination and communication effort with the rest of the team is needed. Whether a suitable solution was found or not, there are usually many learnings for a team after having gone through Pattern 2a. *Pattern 2b* is similar to Pattern 2a in the first three phases. However, as opposed to Pattern 2a, there are hardly any learnings. Instead, the observed level of confusion, discussion, and disruption was so high that teams ended up in a rather chaotic situation which prevented them from drawing lessons learned from the non-routine event.

4.3 Linking Non-Routine Events to Adaptation Patterns

Our next goal was to examine how cues and patterns are linked to each other across teams as well as across different cue types. Our comparative analysis suggests that the link between cues and patterns is influenced by the reach of the cue's impact, the RTC (i.e., a team's iterativeness), and the type of cue. These theoretical relationships that emerged from our data analysis are also reflected in Table 4, which shows the adaptation patterns of the two teams with high iterativeness (PEACH and MANGO) and the one with low iterativeness (CHERRY) for the three cue categories and for cues with local or broad impact. Table 5 summarizes our results by proposing three theoretical relationships.

First, when comparing the reach of the cue's impact and the consequent adaptation patterns of teams, our empirical results suggest that cues with a rather local impact, such as within the team or from peers, tend to result in routinized responses, whereas cues with a broader impact lead to more complex adaptation processes. For instance, one team member of PEACH explained that if "[the company's CTO] wants to show something new at a customer event, then [he/she] gets it right away of course." Indeed, the disruptive nature of the cue that resulted from the direct interest of the board member was also highly noticeable at the team's retrospective meeting. As a result of this urgency and importance of the request, the team's adaptation process was characterized by intensive planning, high coordination effort, and strong learning effects (Pattern 2a). On the other hand, the team can decide rather independently when to fix bugs that are reported internally. "If we are supposed to develop something that we don't want to develop, there needs to be a lot of pressure [from somebody like the CTO] that we do it right away" (team member PEACH). This substantially reduces urgency in these cases and therefore eases the adaptation process. Therefore, local cues seem to result in adaptation processes with less planning and coordination effort (Pattern 1). The link between reach of a cue's impact and entailing adaptation patterns is also observable in Table 4. In particular, Pattern 1 was only followed as a response to local impact cues, whereas none of the cues with a broad impact resulted in Pattern 1.

Second, the comparison of the three analyzed teams suggests that the teams' iterativeness has a substantial impact on adaptation processes. Our data suggests that this impact is in fact twofold. First, our teams with short RTC cycle time tended to respond to a larger number of cues with a more routinized adaptation process (Pattern 1) than the team with a long RTC cycle. The shorter cycle time enabled PEACH and MANGO to limit the complexity and urgency of many cues to a minimum. This was different for CHERRY, where problems often seemed to pile up over time before the team becomes aware of them in the form of a non-routine event. This is illustrated by the situation described by one of CHERRY's team members. "We had two long test phases at the very end that came too late from my point of view. Many complex, program-wide bugs and issues were detected. These issues would have been much easier to handle if we had tested more often and in a more systematic way." This clearly points to the role of iterativeness as one important aspect of agile ISD. Interestingly, the RTC seemed to have an additional impact on the adaptation processes. Those teams that delivered their software more frequently seemed to be more often able to draw learnings from complex adaptation processes (Pattern 2a as opposed to 2b). In fact, due to their shorter development cycles, PEACH and MANGO were able to move some issues to the next release and therefore address problems in a more systematic and calmer way. "Even though discussions can happen at the middle of the Sprint, we often move problems to the next Sprint. We don't let these problems spoil our Sprint" (developer MANGO). As a result, teams with a high iterativeness had the chance to reflect on non-routine events more regularly and thereby learn for future occurrences of comparable incidents. This was different at CHERRY, where particularly towards the end of a release cycle, the pressure seemed to overwhelm the team.

Thus, systematically learning from non-routine events was not feasible. "I don't think about the RTC too much, unless it is pending [laughs]" (developer CHERRY). The effect of RTC can also be inferred from Table 4, as teams with a high iterativeness tended to follow Pattern 1 in more cases than the low-iterativeness team. If more complex adaptations processes were followed, then teams with a high iterativeness tended to benefit more by learning (Pattern 2a) than the team with a long RTC cycle, which more often was unable to learn from complex adaptation processes (Pattern 2b).

		Situation Assessment	**Plan Formulation**	**Plan Execution**	**Team Learning**
		Who assesses the cue?	*How is a plan formulated?*	*Level of coordination?*	*Level of lessons learned?*
Pattern 1		One or few assessor(s)	Routine available	Little coordination	Few learnings
Team	**PEACH** (2-week RTC)	At any point during the Sprint, an internal bug report reaches the developer in charge of bug fixing via the established bug report system. He performs an initial problem and effort assessment.	Together with the Product Owner, the developer in charge of the bug estimates priority and urgency of fixing the bug. He then takes care of the progress, collects necessary information, and informs the team during regular synchronization meeting about current bug status.	The developer in charge solves smaller problems himself, in case the workload is too high, he consults the team for help. The responsibility for bug assessment and solving rotates every Sprint. If the bug is neither crucial nor urgent, it may stay in the pipeline for several Sprints, if it is urgent a hotfix might be necessary.	The developer in charge passes on relevant information concerning the problem solution or potential workarounds. No learnings about the process itself are drawn from this incident.
Class	Task volatility				
Subclass	Current system request				
Instance	Internal bug report				
Impact	Local				
Team	**MANGO** (4-week RTC)	A new team member joins the team. Several days before, the team is notified in the daily synchronization meeting about the change by its line management.	Standard procedures for the on-boarding of the new colleague are planned and one or two colleagues reserve some time for initial knowledge transfer sessions.	The new colleague is trained on the job by various team members. Knowledge transfer mostly happens hands on in pair programming development sessions.	No lessons learned.
Class	Team instability				
Subclass	Structural capacity change				
Instance	New team member				
Impact	Local				
Pattern 2a		Entire team involved	Team discussion	Intensive coordination	Many learnings
Team	**MANGO** (4-week RTC)	A request for contribution to a new, very high prioritized product reaches the team via its product owner in the daily Scrum meeting. The request comes from the solution management level. The Product Owner passes on all known details to the team.	Together with their Product Owner, the team discusses internally the request and how to proceed - whether blocking the cue is possible, pushing it into the next Sprint, or whether it has to be taken care of right away. As much information as possible is gathered from various sources outside the team.	After deciding that blocking the cue is not possible, two developers are assigned to gather more information and regularly attend the mobile platform meetings. The overall team capacity is marginally adjusted and few additional tasks are included in the current Sprint backlog.	Key contact persons are now known to the team. Some actions proved to work well for the team: wait and see what really happens, no commitment if not absolutely required, deliver the necessary in good quality, define clear expectations in terms of necessary information.
Class	Task volatility				
Subclass	New requirement				
Instance	Mobile platform				
Impact	Broad				

Table 3. Adaptation Patterns

		Situation Assessment	**Plan Formulation**	**Plan Execution**	**Team Learning**
		Who assesses the cue?	*How is a plan formulated?*	*Level of coordination?*	*Level of lessons learned?*
Pattern 2a (continued)		Entire team involved	Team discussion	Intensive coordination	Many learnings
Team	**CHERRY** (26-week RTC)	Via the line management, the team is informed about three new colleagues joining their team after another team in the program was split up. The new developers bring along old duties and legacy system responsibilities.	The team discusses the extensive on-boarding procedure within and outside of regular team meetings. Knowledge transfer sessions and Pair Programming are planned.	For more than one Sprint, the overall team capacity has to be adjusted, as the many team members are involved in on-boarding activities. Regular tasks are handed over in pair programming and the new colleagues take care of the tasks stemming from their old team.	Knowledge transfer sessions and Pair Programming are perceived as valuable means to train and integrate new team members. Yet, getting back up to speed took longer than expected (from their management). New colleagues brought with them valuable knowledge.
Class	Team Instability				
Subclass	Structural capacity change				
Instance	New colleagues after team split				
Impact	Broad				
Pattern 2b		Entire team involved	Team discussion	Intensive coordination	Few learnings
Team	**CHERRY** (26-week RTC)	Half way through a Sprint, the Scrum Master realizes that their feature has been moved onto a new platform version overnight. Incompatibilities with the new version manifest directly in the system by preventing the code from working. Automatic tests fail, features do not work any longer. The Scrum Master communicates it to the rest of the team immediately to jointly assess the situation.	This is an emergency situation which is discussed within the team right away. A block of the impact is not possible, responsibles are chosen who take care of the problem. Emergency or escalation procedures are discussed.	The team is partially or fully blocked and current and planned tasks are set on hold. Some tasks can be completed manually, yet all automatic tests are broken. The problem is escalated to the program management level and other teams are informed. The team atmosphere is tense as all members are, if not involved directly, at least affected in their work.	The team massively loses time, is not able to deliver a working increment at the end of the Sprint and no structured approach to solve the problem can be identified. Few lessons learned, besides being more alert for future platform issues.
Class	Technological disruption				
Subclass	Platform issues				
Instance	Incompatible new platform version				
Impact	Broad				
Team	**CHERRY** (26-week RTC)	A change of the overall product user interface is detected by various team members in the central code line. The team is immediately alert and tries to obtain information via its product owner to assess the situation. Features do not work any longer.	The situation is rated as an emergency and the immediate reactions are rather chaotic. Tests fail and the team discusses how to escalate the problem. Workaround, offline, and maintenance tasks are assigned.	The team is partially or fully blocked and current tasks are set on hold. No new items are added to the backlog. The problem is escalated to the program management. A high level of discussion, noise, and coordination impacts the team after the event.	The team massively loses time, blocks any additional requests from outside and besides escalating to the management level, no structured approach is followed. Few constructive lessons learned, and moreover resignation towards the product management level.
Class	Task volatility				
Subclass	Program-related issues				
Instance	New UI				
Impact	Broad				

Table 3. Adaptation Patterns (continued)

Third, as can be inferred from Table 4, our empirical findings suggest that the influence of the team's RTC cycle on the adaptation pattern may also depend on the type of cue that is affecting the team. In our sample of three ISD teams within one organization, the pattern of reaction to cues related to task volatility and technological disruption substantially depended on the length of the RTC cycle, both for cues with local impact (Pattern 1 vs. Pattern 2a) and with broad impact (Pattern 2a vs. Pattern 2b). For non-routine events related to team instability, however, the RTC time had no impact within our sample of three teams. Even though this is a rather exploratory finding, our empirical insights may help shed some first light on this interaction effect between type of cue and iterativeness. For instance, the processes to follow when a team member calls in sick were routinized within all three teams (Pattern 1). Similarly, all three teams were able to learn from more substantial team instability cues (Pattern 2a). The reason for this may be that for more severe team instability cues, such as integrating new sub-teams or adapting to a new product owner, learning may be rather implicit and less related to concrete retrospective meeting sessions that happen more frequently in teams with an iterative work approach.

	Task Volatility	Technological Disruption	Team Instability
Cues with local impact			
2-4 weeks RTC	Pattern 1	Pattern 1	Pattern 1
6 months RTC	Pattern 2a	Pattern 2a	Pattern 1
Cues with broad impact			
2-4 weeks RTC	Pattern 2a	Pattern 2a	Pattern 2a
6 months RTC	Pattern 2b	Pattern 2b	Pattern 2a

Table 4. Cue Categories, Iterativeness, and Adaptation Patterns

Proposition 1	Cues with a local impact trigger routine responses (Pattern 1), whereas cues with broad impact tend to result in more complex adaptation patterns (Pattern 2a and 2b).
Proposition 2	Teams with a high iterativeness are more likely to develop routine responses to cues (Pattern 1) compared to teams with a low iterativeness. If no routine response is available, teams that work in a highly iterative way are more likely to learn from complex adaptation (Pattern 2a) than teams with a low iterativeness.
Proposition 3	The influence of a team's iterativeness on the adaptation process is contingent upon the type of cue such that the effect of a team's iterativeness is more salient for task volatility and technological disruptions than for team instability.

Table 5. Derived Propositions

5 Discussion of Findings

This study was motivated by the observation that while the benefits of applying agile methods and techniques in ISD teams are often ascribed to increased adaptability, the triggers and patterns of team adaptation as well as the theoretical relationship between the two and the role of agile behavior have so far been disregarded. By drawing on social psychology literature and based on the insights from a multiple-case study of three ISD teams, we attempted to shed some light on these open questions. Our study revealed a variety of non-routine events that affect ISD teams above and beyond requirements changes. Moreover, we were able to consolidate the adaptive behavior within the studied teams into three adaptation patterns that represent rather routinized reactions (Pattern 1), or complex adaptation processes that may or may not result in learning effects for the team (Pattern 2a and 2b). Finally, our results suggest that teams with a more iterative approach are able to follow Pattern 1 more often than those with a longer release-to-customer cycle. This seems to be particularly the case for task- and technology-related cues, but less for team instability issues. Generally, our findings suggest that cues with a broader impact lead to more complex adaptation patterns (Pattern 2a, 2b instead of Pattern 1).

Our findings provide several contributions to the existing debate on the agility of ISD teams. First, our study contributes to extant literature in that it integrates the so far disconnected perspectives on agility as a behavior (in terms of applying agile techniques and methods) and as a capability (in terms of being able to react to changes). By studying team adaptation processes, we were able to add more substance to and drill deeper into the notion of adaptability. As such, while teams that effectively and efficiently incorporate changes (reflected by Pattern 1) may indeed show a high adaptability, more cumbersome adaptation processes are not necessarily detrimental, as long as teams are able to learn (Pattern 2a as opposed to Pattern 2b). Our findings suggest that if teams 'show agile behavior' in terms of high iterativeness, then they are more likely to have a routinized procedure at hand (Pattern 1), and to learn from more complex adaptations (Pattern 2a). Thus, teams that work in an iterative way may be enabled to learn faster and develop new routinized responses to other, comparable events. Instead, less iterative teams seem to learn from cumbersome adjustments to non-routine events to a lower extent and may hence face similar issues over and over again. Thus, our findings provide new insights into the non-trivial link between agility as a behavior and agility as the capability to adapt.

Second, our analysis of non-routine events that affect ISD teams shows that a sole focus on requirements changes may be too narrow. In particular, the ISD teams we studied faced numerous non-routines events that related to task volatility (i.e. requirements changes), but also to technological disruption and team instability. This insight adds clarity to what has been referred to as environmental or other changes in previous studies. Thus, based on our empirical findings, the existing definition of agility as a capability provided by Lee & Xia (2010) could be extended to the ability to efficiently and effectively react to task volatility, technological disruption, and team instability. In fact, our results show that this distinction is particularly important when assessing the influence of agile behavior in terms of iterativeness. Third, our study provides further arguments to support previous attempts to abstract from agile techniques and methods and instead study agile behavior in a more generalizable way (e.g., through examining a team's iterativeness). In the context of our study, all three teams were applying Scrum, but still differed regarding their release-to-customer cycle. As a result, the teams varied substantially in terms of their ability to react to task volatility and technological disruption.

In addition to these theoretical implications, our results also provide insights for software developers, project managers, and other decision makers in the ISD context. First, practitioners may learn from our findings that adaptability does not only mean to have established processes at hand to react to every conceivable type of non-routine event. Instead, it seemed equally important to be able to learn from adaptation processes that were more complex. Second, our findings may help ISD teams and project managers to be more aware of the various non-routine events that can affect and disturb development teams in their daily work. Third, the results of this study point to the pivotal role of working in an iterative way that seemed even more important to promote adaptability than specific agile methods and techniques.

Our study has several limitations that need to be considered when interpreting its results and that may also pave the way for future research. First, while our focus on one organization and three teams enabled us to gather focused and rich data as well as control for spurious factors, it may also limit the generalizability of our findings. Future research may therefore replicate our study in other settings. Second, future studies may gain additional insights by following ISD teams and their adaptation patterns over a longer period of time. Such a longitudinal study may be particularly insightful for teams with a low iterativeness, as the team members' perceptions may be affected by the team's current development phase. Moreover, while we focused on iterativeness as one facet of agile behavior, future studies may also select teams that differ in their collaborativeness (e.g., the degree to which these teams apply pair programming). In particular, while we found no impact of iterativeness on the reaction to team instability cues, teams that work in a more collaborative way may be better able to efficiently and effectively react to non-routine events related to team instability.

Acknowledgements

We thank the Associate Editor and the reviewers for their helpful comments. We also thank Kai Spohrer for providing valuable feedback on earlier versions of this paper.

References

Balijepally, V. et al., 2009. Are Two Heads Better than One for Software Development? The Productivity Paradox of Pair Programmming. *MIS Quarterly*, 33(1), pp.91–118.

Beck, K. et al., 2001. Manifesto for Agile Software Development. *Agile Alliance*. Available at: http://www.agilemanifesto.org/ [Accessed April 19, 2013].

Beck, K. & Andres, C., 1999. *Extreme programming explained: embrace change* 1st ed., Addison-Wesley Professional.

Burke, C.S. et al., 2006. Understanding Team Adaptation: A Conceptual Analysis and Model. *Journal of Applied Psychology*, 91(6), pp.1189–1207.

Cao, L. & Ramesh, B., 2008. Agile requirements engineering practices: An empirical study. *IEEE Software*, 25(1), pp.60–67.

Carbonell, P. & Rodríguez-Escudero, A.I., 2009. Relationships among team's organizational context, innovation speed, and technological uncertainty: An empirical analysis. *Journal of Engineering and Technology Management*, 26(1), pp.28–45.

Cockburn, A., 2006. *Agile software development: the cooperative game* 2nd ed. A. Cockburn & J. Highsmith, eds., Pearson Education.

Conboy, K., 2009. Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. *Information Systems Research*, 20(3), pp.329–354.

Davern, M., Shaft, T. & Te'eni, D., 2012. Cognition Matters: Enduring Questions in Cognitive IS Research. *Journal of the Association for Information Systems*, 13(4), pp.273–314.

Dingsøyr, T. et al., 2012. A decade of agile methodologies: Towards explaining agile software development. *The Journal of Systems and Software*, 85(6), pp.1213–1221.

Dingsøyr, T. & Dybå, T., 2012. Team Effectiveness in Software Development: Human and Cooperative Aspects in Team Effectiveness Models and Priorities for Future Studies. In *Cooperative and Human Aspects of Software Engineering (CHASE)*. Zurich, Switzerland, pp. 27–29.

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

Eisenhardt, K.M., 1989. Building Theories from Case Study Research. *The Academy of Management Review*, 14(4), pp.532–550.

Ferreira, S. et al., 2009. Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. *The Journal of Systems and Software*, 82(10), pp.1568–1577.

Henderson-Sellers, B. & Serour, M.K., 2005. Creating a dual-agility method: The value of method engineering. *Journal of Database Management*, 16(4), pp.1–24.

Hollingshead, A.B. et al., 2010. Communication and Knowledge-sharing Errors in Groups: A Transactive Memory Perspective. In H. E. Canary & R. D. McPhee, eds. *Communication and Organizational Knowledge: Contemporary Issues for Theory and Practice*. Routledge, pp. 133–150.

Kozlowski, S.W.J. et al., 1999. Developing adaptive teams: A theory of compilation and performance across levels and time. In *The changing nature of work performance: Implications for staffing, motivation, and development*. ERIC, pp. 240–292.

Langley, A., 1999. Strategies for Theorizing from Process Data. *Academy of Management Review*, 24(4), pp.691–710.

Lee, G. & Xia, W., 2010. Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data on Software Development Agility. *MIS Quarterly*, 34(1), pp.87–114.

Louis, M.R. & Sutton, R.I., 1991. Switching Cognitive Gears: From Habits of Mind to Active Thinking. *Human Relations*, 44(1), pp.55–76.

Lyytinen, K. & Rose, G.M., 2006. Information system development agility as organizational learning. *European Journal of Information Systems*, 15(2), pp.183–199.

Marks, M.A., Mathieu, J.E. & Zaccaro, S.J., 2001. A Temporally Based Framework and Taxonomy of Team Processes. *The Academy of Management Review*, 26(3), pp.356–376.

Maruping, L.M., Venkatesh, V. & Agarwal, R., 2009. A Control Theory Perspective on Agile Methodology Use and Changing User Requirements. *Information Systems Research*, 20(3), pp.377–399.

Maruping, L.M., Zhang, X. & Venkatesh, V., 2009. Role of collective ownership and coding standards in coordinating expertise in software project teams. *European Journal of Information Systems*, 18(4), pp.355–371.

Rosen, M.A. et al., 2011. Managing adaptive performance in teams: Guiding principles and behavioral markers for measurement. *Human Resource Management Review*, 21(2), pp.107–122.

Sabherwal, R. & Robey, D., 1995. Reconciling variance and process strategies for studying information system development. *Information Systems Research*, 6(4), pp.303–327.

Salas, E., Sims, D.E. & Burke, C.S., 2005. Is There a “Big Five” in Teamwork? *Small Group Research*, 36(5), pp.555–599.

Schmidt, C.T. et al., 2013. Team Adaptability in Agile Information Systems Development. In *International Conference on Information Systems*. Milan, Italy, pp. 1–11.

Schwaber, K. & Sutherland, J., 2011. The Scrum Guide. , pp.1–16.

Sfetsos, P., Angelis, L. & Stamelos, I., 2006. Investigating the extreme programming system–An empirical study. *Empirical Software Engineering*, 11(2), pp.269–301.

Slotegraaf, R.J. & Atuahene-Gima, K., 2011. Product Development Team Stability and New Product Advantage: The Role of Decision-Making Processes. *Journal of Marketing*, 75(1), pp.96–108.

Sommerville, I., 2007. *Software Engineering* 8th ed., New York: Pearson Education Limited.

Song, M. & Montoya-Weiss, M.M., 2001. The Effect of Perceived Technological Uncertainty on Japanese New Product Development. *Academy of Management Journal*, 44(1), pp.61–80.

Tripp, J.F., 2012. *The Impacts of Agile Development Methodology Use on Project Success: A Contingency View*. Michigan State University.

VersionOne, 2012. *7th Annual State of Agile Development Survey*,

Yin, R., 2009. *Case study research: Design and methods* 4th ed., SAGE Publications Inc.