12-13-2018

Toward a Model of Managing Interruptions in Agile IT Projects

Manuel Wiesche
Technical University of Munich

Agile IT Projects: How Teams Manage Interruptions

Manuel Wiesche
Technical University of Munich
wiesche@in.tum.de

ABSTRACT

Working in uncertain environments fundamentally changes how we organize work. Using agile methodologies for IT projects helps teams to better meet user needs and ensure flexibility in uncertain environments. But using agile methods increases interactions with fellow team members and external stakeholders such as customers. These interactions are either embedded in agile practices or occur unplanned in the work context, which both cause interruptions in the workplace. While those can be helpful in terms of task completion, meeting user needs, and increased process flexibility, interruptions hinder employees in being efficient and productive. We thus conducted a Grounded Theory study analyzing four cases to understand the nature and consequences of interruptions in agile ISD teams and how the team manages these interruptions. We find that IT project teams formalize interruptions to reduce negative consequence, channel interruptions during their daily routines based on expertise and workload, and use digital tools both to reduce the number of interruptions and also to prioritize incoming interruptions. Our analysis suggests that IT project teams use practices embedded in the agile method to exploit the positive aspects of interruptions and find ways to reduce the negative.

Keywords

Agile information systems development, interruptions, teams, project management.

INTRODUCTION

Working in uncertain environments fundamentally changes how we organize work (Rigby et al. 2016). Adapting to uncertain requires continuous adaption and coping with change. Therefore, organizations need to balance flexibility and stability (Bazigos et al. 2015). In Information Technology (IT) projects, agile project management approaches are used to continuous re-correct by enforcing continuous interaction with extern and intern stakeholders (Rigby et al. 2016; Slaughter et al. 2006). Pre-planned tasks are revised in an iterative manner, agile practices such as daily standups, planning sessions, and burndown charts foster continuous feedback and refinement, informal knowledge exchange and problem solving is encouraged, and collaborative workplaces are created as co-located or digitally connected work environments (Dery et al. 2017; Lee and Xia 2010; Maruping et al. 2009).

Agile IT projects increase collaboration with different stakeholders in IT project team (Pflügler et al. 2018). This collaboration has many benefits for the agile team, as it ensures, that customer needs are met (Recker et al. 2017; Rigby et al. 2016; Vidgen and Wang 2009), fosters knowledge sharing (Ghobadi and Mathiassen 2017; Kudaravalli et al. 2017), and increases employee motivation (Tripp et al. 2016). However, collaboration also increases the number of interruptions within the IT team (Tregubov et al. 2017). We understand interruptions as "external, unpredictable events that create attentional conflict between the competing demands of the interruption and primary task" (Addas and Pinsonneault 2015). Interruptions can be human- or IT-induced and examples include colleagues asking for help in solving a programming task, customers requesting to change backlog items, incoming emails, social media or phone calls. Such interruptions are considered process impediments (Wiklund et al. 2013) and recovering from interruptions is a central problem amongst software development teams (LaToza et al. 2006).

Given the ubiquity of interruptions in work environments (Murphy 2016; The Economist 2017), our understanding of interruptions has been around negative consequences such as time pressure for important tasks, procrastination, and mediocre performance (Addas and Pinsonneault 2015; Jett and George 2003). This negative understanding has converged to a more balanced perspective on interruptions, which are now also considered helpful for informal feedback and information sharing, individualizing work pace, enhancing performance and mindful information processing (Chua et al. 2012; Jett and George 2003; Pendem et al. 2016; Zellmer-Bruhn 2003).

While existing research has developed an understanding of interruptions in the work place, only little work has been conducted on interruptions for IT project teams (Tregubov et al. 2017). More specifically, literature remains silent on how interruptions are handled in agile Information Systems Development (ISD) contexts and how the agile teams

handle these. Therefore, the following research question guides or study: *"What are nature and consequences of interruptions in agile IT teams and how do teams respond to these?"* We conduct an exploratory study of four agile software development teams and use Grounded Theory Methodology to understand interruptions in our context (Urquhart 2012). In addition to identifying interruptions in agile work environments, we develop a model of gatekeeping, building on (1) the usage of the agile method to invite, yet buffer interruptions, (2) channel interruptions within the IT team, and (3) tool support to reduce unwanted interruptions.

BACKGROUND

Types and consequences of interruptions in organizations

Interruptions are understood as events that impede or delay organizational members during work tasks (Addas and Pinsonneault 2015; Perlow 1999). Organizational theorists conceptualize interruptions as intrusions, when an individual's work is interrupted by another person, breaks, when a self-initiated halt is posed on the task, distractions, when external stimuli interrupt concentration, and discrepancies, when an individual perceives inconsistencies between their knowledge and the external environment (Jett and George 2003). For understanding interruptions, the interrupted task, the interruption content, the timing and quantity, as well as the consequences are important (Galluch et al. 2015; Jett and George 2003).

Interruptions have negative consequences such as increased time pressure for the task that was interrupted, procrastination, and mediocre performance (Jett and George 2003). However, interruptions are considered to have positive consequences as well, including being helpful for informal feedback and information sharing, individualizing work pace, enhancing performance and mindful information processing (Bechky and Okhuysen 2011; Chua et al. 2012; Jett and George 2003).

Interruptions are associated with increased creativity, as they prompt attention shifts toward different perspectives, increase knowledge transfer, increase team learning through interactions, and invoke a "wakeup" call on routine work to conscious information processing (Watson-Manheim et al. 2012; Zellmer-Bruhn 2003). Further, unexpected breaks are associated with increased performance if they allow employees to uphold attention the primary task (Pendem et al. 2016).

Interruptions in agile IT projects

Agile IT teams are groups of software developers that jointly work on the development of new or modifications of existing software system (Tripp et al. 2016). Agile IT teams use agile methods such as SCRUM, eXtreme Programming (XP), or KANBAN, while SCRUM is the most common agile method applied in practice (Bazigos et al. 2015; Rigby et al. 2016). In SCRUM, there are three dedicated roles: the development team consisting of developers that implement product functionalities, the product owner, who represents the customer and is responsible that the team delivers business value, and the SCRUM master, who is accountable for removing work impediments to the IT team (Lee and Xia 2010; Maruping et al. 2009; Rigby et al. 2016).

Agile IT teams are characterized by high levels of collaboration including internal coordination and knowledge sharing, as well as extern integration of core stakeholders (Ghobadi and Mathiassen 2017; Kudaravalli et al. 2017; Przybilla et al. 2018; Recker et al. 2017; Rigby et al. 2016; Vidgen and Wang 2009). Further, agile IT teams use a continuous high number of meetings for agile practices including daily standups, planning sessions, and burndown charts (Przybilla et al. 2018; Tripp et al. 2016). Agile IT teams rely on informal control mechanisms such as clan control that establish group norms and shared beliefs (Chua et al. 2012; Wiedemann and Wiesche 2018).

These characteristics have consequences on the occurrence and management of interruptions in agile IT teams. While agile IT teams benefit from knowledge sharing, close collaboration increases the amount of direct communication, which ultimately results in intrusions and distractions (Jenkins et al. 2016; Melnik and Maurer 2004). Similarly, close collaboration with the customer causes intrusions and discrepancies (Hoda et al. 2011). The high number of planned meetings provide breaks for IT team members and the distributed decision making causes discrepancies and additional work (Hoda et al. 2011). These interruptions are in line with non-agile IT literature, which highlights the high number of interruptions during development work and the high costs of switching between tasks (Abad et al. 2017; Meyer et al. 2017; Perlow 1999; Tregubov et al. 2017).

In addition, agile practices such as pair programming create additional interruptions. Here, developers are interrupted by peers during joint activities, instantly correcting code written by the developer (Balijepally et al. 2009). In addition to these very low level interruptions, todays work practices in IT teams involve developers, simultaneously working in multiple teams (Cameron and Webster 2013). This brings interruptions to the developer, who's tasks fundamentally mix even throughout a work day, but also interrupts the IT team, as a missing team member can cause delay in decision making, quality assurance, or other path dependencies.

Contrasting the co-location in agile IT teams, the topic of communication-caused interruptions has been intensively discussed in the literature on virtual, i. e., distributed teams (Maznevski and Chudoba 2000). In virtual teams, direct communication is challenged though missing verbal cues, delayed feedback, and communication pauses (McGrath 1991). The temporal and physical dispersion creates distractions and involuntary breaks due to misunderstandings and delay (Colazo and Fang 2010).

Based on observations of the dynamics nature of boundaries such as time, distance, and culture in virtual IT work, Watson-Manheim et al introduce organizational discontinuity theory (ODT) (Watson-Manheim et al. 2012). ODT suggests that boundaries, which are often present in IT work, may not always be problematic for virtual work. Only when organizational members perceive discontinuity at boundaries, this creates difficulties for virtual teams. ODT has been introduced as a fruitful lens to study coordination conflicts in large-scale agile IT teams (Crowston et al. 2016), but the theory does not cover how the use of the agile method helps agile IT teams manage interruptions in their daily work (Tregubov et al. 2017).

While these promising perspectives suggest that interruptions affect agile IT teams, it remains unclear how interruptions affect agile IT teams and how the team responds to them.

METHODS

We applied an inductive exploratory approach to answer our research question, given the complexity of the phenomenon. We find Grounded Theory Methodology particular useful to understand the phenomenon of interruptions in its natural context (Urquhart 2012). We interviewed 19 agile team members and asked about interruptions they experience in their daily work. We explored these interruptions and particularly focused on understanding how the agile method helped team members deal with interruptions. In table 1, we describe the four teams we studied and report on analysis procedures we applied in this study.

Data analysis

Four our analysis, we used techniques from Grounded Theory Methodology (Glaser 1978; Urquhart 2012). We applied theoretical sampling as we selected interviewees and cases based on the analysis of the previous collected data (Wiesche et al. 2017). For example, we sampled team MONITOR to gain a better understanding of unique interruptions to SCRUM in the other cases and to understand if our mechanisms hold for other agile methods such as Kanban. We constantly comparing our data with extant literature, as well as other data points collected before. We followed Glaser's guidance in applying open coding and selective coding (Glaser 1978). During this analysis, we gave specific attention to the causes and consequences of interruptions in the agile ISD process.

Team acronym	CAR	FLEET	MONITOR	REAL ESTATE
Team members	4	8	4	9
Agile method	SCRUM	SCRUM	Kanban	SCRUM
Industry	Automotive	Automotive	Insurance	Banking
Software product	Customizable online store for car configuration	Website to support company fleet management	Automation, administration, and monitoring of insurance software	Information system to support real estate financing consulting
Characteristics	Co-located, 2 week iteration length	co-located, high complexity through number of users and HR system connection	co-located, focus on operating landscape of different insurance software systems	co-located, 3 week iterations, 18 release planning
Agile practices	automated unit testing, backlog, sprint planning, definition of done, daily stand-ups, coding standards, burndown charts, retrospective	continuous integration, automated unit testing, sprint planning, daily customer involvement, burndown charts, retrospective	ticket system that is linked to their e-mail accounts, daily stand-ups, weekly planning meetings, Kanban chart, waiting line, backlog, retrospective meetings	daily stand-up meeting, iterative planning, retrospectives, burndown charts, automated testing, coding standards
Team roles	SCRUM master, a product owner, and two developers	SCRUM master, product owner, frontend developers, backend developers, UX expert, developer in charge for deployment	three developers and project lead	SCRUM master, two product owner, four developers, and two testers
Interviewees	3	6	4	6

Table 1. case overview

RESULTS

Our open coding identified three categories of interruptions across all four teams. Interruptions were related to the task of developing software (development-related), the use of the agile method (method-imposed), and to context variables (context related). In the following, we document each category by describing the interruptions, it consequences, and how the IT project team managed the interruptions.

Development-related interruptions

Requirement-based interruptions occurred when management re-prioritized requirements, customers needed to fully specify the requirement, or when the customer changed a requirement. In project FLEET, the customer changed a requirement in the course of an iteration. The team was working on a user story and the product owner identified changed text in a user story in their issue tracking software JIRA. The change was so fundamental, that certain features of the software product had to be changed. The developer described that "*the customer restated the user story. This slacked our pace. Totally different goal, resulting in different task for us … For the birds … We had to do another sprint planning and needed new code*." (Project FLEET, developer 2)

Interruptions around the existing code occurred frequently, when the developed software was not developed as a standalone solution, but integrated in an existing application environment. When describing their regular work during each iteration, developers often noted that there were additional tasks that occurring during the sprint. A developer described that *"in the hot phase, the [backend] was down for several days. And we had to do other stuff. The [backend] cannot be simulated. We needed to stop developing. Jump to other tasks like preparing unit tests or developing a shot in the dark."* (Project CAR, developer 1)

Across all teams, interviewees highlighted the importance of the interruptions in solving problems and guiding the project. Especially the customer perspective was highlighted as fundamental to understand in which direction the project should go. *"The customer is very important to give feedback on the progress and direction of the project […]*

Of course, it is interrupting if that come like this, but we try to prepare for this. There are regular project review meeting where we gather feedback from our management and customers. […] This helps us in identifying dead ends, problems, and new developments which we were not aware of." (Project REAL ESTATE, developer 3)

Agile IT project team members responded by channeling interruptions to the relevant colleague. All teams reported that the customer often did not follow agile practices exactly and interrupt different team members at different points in time, rather than postponing issues until the next formal team meeting. The teams developed mechanisms to channel these interruptions to the responsible colleague and ensure that they would take care of this issue later during the project phase. Usually, every developer knows what others are working on and had a brief understanding of the feature that was commented on by the customer. So they used digital tools like slack or JIRA to document the issue and assigned the correct team member. Thereby, the new detail was neither lost, nor did it interrupt the colleague directly. Team REAL ESTATE's tester 1 reported that he *"tr[ies] to capsulate [these interruptions]. I ask the customer for details, provide a first evaluation in terms of feasibility and time horizon and then put it in the system. If I consider it urgent, I will raise the issue in the next daily meeting, otherwise, it will sit in the system, waiting for [developer 1] to resolve."*

The teams FLEET, MONITOR, and REAL ESTATE reported that they scheduled "quiet time", where developers could concentrate on their work without being interrupted. The SCRUM master in team FLEET explained that in his team, developers showed up as early as 7:00 am to get work done until 9 am. He estimated that during this "quiet time", developers got 80% of their work done. After 9 am, meetings started and work was interrupted by meetings such as stand ups, personal breaks, and socializing, e.g. during smoking breaks.

Interruptions imposed by the agile method

The second category of interruptions were imposed by the agile method. These method-imposed interruptions include the encapsulating interruptions in agile practices, the usage of tools to track tasks, and the interaction with the customer.

Across all four cases, we observed that the team formalized many unplanned interruptions in agile procedures. Especially the SCRUM master helped the team encapsulate problems, issues, and discussions in continuously occurring SCRUM meetings. Team members collected issues that needed discussion with fellow team members that were not urgent to be discussed within the next regular SCRUM meeting, most of the time the next daily stand-up meeting in the morning of the following day. Developers learned that if the consequences would not cause long delay, s/he would decide to wait for the next team meeting rather than interrupting fellow team members. The SCRUM master explained that he decided for this way in managing the interruption as he thought that it was more important to have a certain result ready that could be changed rather than a sudden stop in the coding.

One strength of the agile method is to put the customer at the core of the process. Many practices target at updating, simulating, or integrating the customer and his/her needs. On the one hand, this is helpful in calibrating the solution, prioritizing, and planning the next steps, but it also interrupts the development process. Asked about how customer interrupt daily activities, one developer described: "*They usually send emails. And they are the customer, so you have to drop all other work and respond […] Stop your current task, understand the problem, solve the problem, and update customer about the solution.* […] *You want to give a good impression, provide a great service and high quality code. He is the one who pays the bill that is why we do not bother about customer interruptions. […] they increase at the end of releases, I guess because then, the customer 'wakes up', but also, it is easier to criticize a working solution rather than imagining potential functionalities."* (Project REAL ESTATE, developer 2)

In team MONITOR, developers reported that customers usually used an e-mail based ticketing system to report and track incidents. Only on particularly urgent issues, customers were asked to approach the team via phone. However, developer 1 reported that there were some customers, who interpreted every issue as urgent and called immediately. If the call was not answered, they would come to the teams office, which was located in the basement of the headquarter building and interrupt the team in whatever tasks they were currently facing. He explained: "*It really is annoying. And takes you out of whatever you are doing. But we have a great work environment here, so you do not send them to h***, but agree to help and ask for the matter. If it is something that takes less than 10 minutes, like rebooting a system, you just fix it immediately, just to get rid of the guy. If not, you just open another ticket and signal that you understood the urgency and importance*." (Project MONITOR, developer 1)

Agile practices helped IT project teams buffer interruptions in formal meetings. These formal meetings help reduce the number of unplanned interruptions within the daily work. Practices such as daily stand-ups, sprint planning and review, pair programming, time boxing, and retrospective capsulate interruptions in meetings. The sprint and the daily stand-ups are the most important agile practices to buffer interruptions related to development work within the team. Every morning, developers have the chance to raise issues where they struggle or need feedback with the whole team. Thus, the time to the next meeting is short and many non-urgent requests, which would interrupt peers, can be discussed in the next stand-up meeting. The sprint is an important vehicle to capsulate the team from external interruptions to concentrate on development tasks. Across all cases, the SCRUM master tried to protect the team from unnecessary interruptions. One SCRUM master described that he *"tr[ies] to keep all further interruptions from the team if possible. […] I consider a sprint successful, when the developers can spend more than 80% of their time on tasks, that we agreed on during sprint planning. So I will do everything to protect my team from interruptions."* (Team REAL ESTATE, SCRUM master). Similarly, the customer is involved in sprint planning, where he can prioritize tasks to develop a focus and sprint review meetings to correct directions and demand changes to the current increment.

In teams FLEET and CAR, the project team scheduled additional meetings, which they referred to as refinement meetings. These meetings were scheduled in the middle of each sprint to formalize the continuous refinement process. One developer explained that he sees these meetings *"as artificial interruptions in an ongoing process. But the formalizing [in an official meeting] helps in updating the backlog and tracking progress. This opens up the view on the next iteration that starts a week from now. You can do slight adaptions with an eye what will come next week."* (Project FLEET, developer 2)

Interruptions imposed by the project environment

The third category of interruptions were grouped around the work environment. Here, the office set-up, the daily schedule including breaks, and the way the teams worked together imposed positive and negative interruptions. All teams balanced interruptions of daily work practices with the advantages of co-location. Several developers described a work culture of helping and the willingness to let teammates interrupt one's work to ensure the overall project goal.

One developer explained that he did not make use of the company's home office policy very often. He described that this limited his ability to solve problems by reflection: "*If I sit a home and work on a topic for four hours, so I need to set an alarm to get lunch, […] and continue afterwards. This is when you write a heck lot of code. But it cannot help you on thinking problems. When you need to reflect. Chances are high that you get on the wrong track with your solution. And then you are stuck. Here in the office, I can join the guys for a cigarette and I either share my current ideas or just by getting back to the screen after five minutes helps me think 'shoot, that can't possibly work this way'. I would call these breaks organic breaks."* (Project CAR, developer 1)

Across all teams, team members report interruptions related to non-work-related incidents as well. These interruptions occurred on different channels, including telephone, e-mail, and private surfing. The most dominant interruption was the smart phone. *"It takes time to get back to what you are doing. This is not helpful. And […] you catch yourself once in a while doing stuff that is not work related. That is interrupting. Checking your phone, your news-feed, social media. And there is a video that is more interesting or a link to something…"* (Project FLEET, UX designer)

In project MONITOR, the developers used a chat tool to pose questions as soon as they occur. Several team members described that they used this chat tool permanently for small questions amongst each other. So there were ongoing interruptions that even popped up on the developers screen as soon as someone asked something. However, these were perceived as positive: *"[spontaneous things], I find these positive. It might not be sorted or queued, but there are these things of informal, loyal forms of collaboration that involves asking and helping, that I think has more advantages than disadvantages."* (Project MONITOR, developer 1)

In addition, interruptions also helped team members take breaks and get their heads free. Several team members regularly went outside to smoke and used the break to think about the problem. Smoking team members described that the smoking break helped them to develop a different perspective or just to wait for another idea. One team member even described that she took smoking breaks, although she did not smoke: "*Usually, you get a coffee and discuss. But sometimes, I really need a break… And I do not smoke. And I love joining the smokers outside. This was either planned or unplanned."* **(**Project FLEET, UX designer)

IT project teams used digital tools to reduce external interruptions related to information requirements. In every team, the developers used automation to reduce the number of interruptions related to software development tasks. Automatic deployment solutions reduced additional efforts from fellow team members during deployment and automated testing such as unit tests, reduced interruptions due to error detection and functional testing.

The teams regularly updated and shared their status documents, such as product backlog, sprint backlog, and burndown charts with external stakeholders to increase transparency in the development process. The team thereby reduced interruptions related to status reports, as external stakeholder including management and customer could access these systems to get a status overview of the project.

Within the team, developers developed an informal hierarchy of tools for communicating an understanding of the urgency of the interruption. For example in team FLEET, the developers used a chat tool for internal communication, a ticket system to structure their work, and e-mail to communicate with external stakeholders. Developer 2 explained that: *"I immediately respond to requests via [chat tool]. Because we all know that the person asking benefits from a quick feedback. If not, he would have asked via [ticketing system] or sent an e-mail. For these topics [ticketing system and e-mail] I block some time in the afternoon and analyze and prioritize. So this turns into a planned interruption."* Similarly, in team CAR, fellow team members understood the sense of urgency by the medium of communication. One developer explained that a person rushing in the office is looking for urgent help. These colleagues are willing to pull someone of his/her tasks to solve their problem. So such problems are considered important enough to interrupt colleagues, as otherwise, these developer would be on halt with their work.

Type	Source	Observed action
Development-related	Missing information	In team CAR, developers spend a lot of time clarifying requirements. While they strive to get as much details during planning sessions, during the course of the sprint, things pop up or are changed by the customer, causing delay or double work. The product owner in team REAL ESTATE explained that either missing customer input caused delay for specific development tasks, or external partners such as UI marking experts caused delay.
	Malfunctioning code	Team FLEET's SCRUM master described an incident, when the team produced a potentially shippable product increment, but fully deploying this version of the software in the customer's system required several days and interrupted the team's development activities. In team CAR, developer coined the term "waiting for" time to describe backend issues causing interruptions. The developers frequently had to integrate their solution in the existing architecture – the backend. Thereby, not only ill specified APIs, but also malfunctioning backend code caused delays, where team CAR had to stop working and could not test their solution.
Requirement-reconfiguration	Customer changes backlog	All teams experienced situations, when the customer changed backlog items during the course of the project. The less interrupting point in time for interruptions was, when the customer changed the backlog during sprint planning or sprint review meetings. However, all teams reported, that customers regularly changed items during the sprint. In some cases, the new backlog items could be postponed for the next iteration. In others, the sprint was altered or even cancelled. Team CAR's SCRUM master explained the need to moderate these situations with the team.
	Ill-specified requirements	One developer in team FLEET reported that developers often struggle with requirements that we not clearly specified in the backlog. For example, planning errors occurred when the front-end was integrated with a back-end system using an ill-specified API.
	Re-prioritization	In team REAL ESTATE, senior management approached a new developer with an additional task. This task was scheduled in the backlog for a later iteration, but management pushed the importance and singled out the developer to work on this task. This interrupted the new developer, but also the team, which was dependent on his deliverable to achieve the intended sprint result.

Information impediment	Waiting for customer input	In team CAR, developers struggled with getting access to client systems. They reported weeklong delays in accessing the customer's Jenkins system [tool for continuous integration] and documentation system. They developed workarounds by using customers' access data to prevent project delay. Similarly, team REAL ESTATE described challenges in accessing customer systems. For example, the SAP testing system was not available for a couple of days, which required to re-plan work and simulate test situations.
Method imposed	Scheduled meetings	All team highlighted the importance of meetings to collect and discuss issues relevant for all team member. In team MONITOR, developers explained that often, incidents had path dependencies – for example hardware reboots that affected other pieces of software as well – that were discussed in the group and identified as relevant by other team members. In team REAL ESTATE, the SCRUM master explained that team members shared their status in daily standup meetings. In these meetings, developers could seek or give feedback instead of interrupting colleagues on other tasks. Thus, they postponed problems that were not that urgent until the next day's standup meeting. In addition, developers grasped an overview of issues, problems, and solutions, other developers experienced, which might become relevant for their work in the future.
	Training others	In team FLEET, one developer learned a new development framework, which was a planned task during their iteration. Afterwards, she spent a couple days training other team members using pair programming rather than working on new tasks by herself more efficiently. This way, the team gained expertise and was more efficient on future tasks. As a senior developer, she reflected that her own contribution was in distributing knowledge within the team rather than solving backlog items.
Project management	Re-staffing team members	A former team member in team FLEET had been assigned to a different project, which was struggling and required his expertise. However, he had to help immediately and thus, team FLEET was interrupted in their sprint, which failed to deliver results as planned.
	Onboarding new team members	Developer 1 from team MONITOR explained the importance of training new team members not only in the basic understanding of the agile method, but even more important in the team's developed routines and ceremonies when using agile methods in their particular work environment.
	Status reports	Both teams, FLEET and CAR implemented internal and external status report reports. Internal meetings document development status on a regular basis and identify impediments. External meetings give customers the chance to provide feedback on increments. Several teams used JIRA or Trello for providing transparency to internal and external stakeholders.
	Multi-team membership	Two teams reported that team members were working on more than one project in parallel. This can result in sudden shift of resources due to bottlenecks in other projects. This makes sprint planning more difficult and requires spontaneous re-allocation.
Daily work	Inspiration from information exchange	Developer 2 from team FLEET described their open office space, which allowed them to easily collaborate on joint tasks and interact with others. However, he also described that sometimes – often in the afternoon – team members interacted with nearshore colleagues via skype, which created a continuously high level of noise in the office, which interrupted all co-workers. In team REAL ESTATE, the product owner was co-located with the team, sitting in the same room. Developers valued instant feedback and ongoing discussions, which avoided slack, delay, and double work.
	Helping behavior	Across all teams, team members described an open culture, where they were happy to help other team members on their problem. Taking ownership for the project goal, developers were willing to help peers to ensure that the overall goals are met.
	Telephone calls, People running by	Team FLEET's SCRUM master described the positive "open door" atmosphere within the team, where all team members implicitly agreed on establishing a culture of reciprocal helping behavior within the team. Team members actively asked for help or advice and also let other team members interrupt their work.

		Team MONITOR reported on an implemented phone ring, where the team's phone calls were automatically handed over to fellow team members if a call was not answered. This ongoing ringing was perceived as interrupting and they approached management to deactivate the phone ring.
	Breaks	Several developers described the work environment as helpful and inspiring as there were interruptions once in a while. Rather than solely focusing on a task, they found that discussing ideas – which they refer to as "thinking through a task" - is helpful for problem solving. There were many situations, where developers would struggle with finding a solution on their own and benefit from explicating their problem to colleagues and getting feedback and ideas.

Table 2. Sources of interruptions and derived types of interruption

DISCUSSION

We identify interruptions related to the task of developing software, the use of the agile method, and to context variables. Teams use the agile method to invite, yet buffer interruptions by transferring unplanned into planned interruptions in formal meetings and processes. We highlight the importance of the SCRUM master in channeling interruptions within the team. Finally, we identify tool support as third mechanism to manage interruptions and reveal the importance of immediate feedback in agile IT teams.

Toward a model of handling interruptions in agile IT project teams

In the following, we develop a model handling interruptions in agile IT project teams. This is based on the underlying mechanism that agile IT teams filter interruptions based on the context and the current work situations. Following the agile method, IT teams invite feedback in processes, work setups, and decision-making. However, the team uses and develops mechanisms to filter and channel useful interruptions and cope with hindering interruptions. The model is separated into three means if handling interruptions.

The first mean relates to the team's usage of the agile method to invite, yet buffer interruptions. We refer to this as formalizing interruptions. Our results suggest that using agile project management methods invites interruptions, which are addressed in formal agile processes. External stakeholders like the customer are invited early and continuously to project meetings, giving them the opportunity to provide feedback and correct the direction of the project. Similarly, internal team members are invited to interrupt fellow team members in daily stand-ups, pair programming, and continuous delivery, but also by the co-located work set up. Given the breadth of expertise and missing hierarchical structure of agile work increases the number of interruptions related to decision making. However, our results suggest that the formalization of interruptions in agile procedures reduces the negative perceptions of interruptions within the project team.

The second mean relates to practices the team uses to channel interruptions during their daily routines. This helps the development teams to focus on critical tasks, where interruptions would reduce the amount of time available and ultimately increase stress and anxiety. Our results suggest that agile IT project teams channel interruptions within the team to first, identify the right expert without additional overhead and, second, identify "bottleneck" situations with the team and suggest alternative solutions to cope with the interruption. In addition, frequent interruptions prevent developers from concentrating on solving one particular problem and reach a state of high involvement on that particular topic. The agile teams developed a set of practices (e.g., protecting sprints against external interruptions, strategies for "escaping" from daily work, physical cues to signal openness for interruptions) to channel interruptions within their daily work.

The third mean relates to using digital tools to prioritize interruptions. IT project teams used digital tools to reduce external feedback and information request. Similarly, during development, developers used automation to reduce the number of interruptions related to software development tasks. Within the team, developers developed an informal hierarchy of tools for communicating an understanding of the urgency of the interruption. Overall, these means helped the IT project team to prioritize potential interruptions. Team members can distinguish interruptions that require instantly pausing their current work and others that can be postponed until the work is finished or even rescheduled for the next iteration.

Contributions

Our analysis revealed several positive consequences of interruptions in agile IT teams, that go beyond individual level search (Jett and George 2003). The dominance of real-time tools in collaboration within the team highlights the importance of fast feedback during ISD work. We found that this personal help reduces slack and double work. Similarly, all cases highlight the importance of feedback from IT project stakeholders. While such feedback might interrupt a team's daily routines, it also reduces double work by ensuring the early recognition of need for change and action (Bechky and Okhuysen 2011). We further highlight that formalized interruptions can solve problems by increasing communication or just explicating the problem. Finally, using the example of training fellow developers via pair programming, we show that short-term consequences of interruptions might be negative, but in the end will increase overall project performance.

We find that IT project teams filter interruptions based on the context and the current work situations. While the agile method provides practices that invite interruptions in processes, work setups, and decision-making, teams filter and channel useful interruptions and cope with hindering interruptions. These mechanisms confirm the importance of implementing separate roles in agile IT projects and underline the importance of the SCRUM master in protecting the IT team and removing impediments (Lee and Xia 2010; Maruping et al. 2009; Rigby et al. 2016). While extant research discusses the importance of timely and constant customer feedback (Recker et al. 2017; Vidgen and Wang 2009), we extend this view in suggesting a timing perspective. Customers can help SCRUM masters protect the team, by using pre-specified agile practices like sprint review meetings or daily stand-ups to provide feedback and course corrections.

Finally, our results highlight how different implementations of agile, namely SCRUM and KANBAN affect agile IT projects. We found that team MONITOR, which used KANBAN experienced more interruptions than the other teams that followed scrum. However, tasks in team MONTOR were smaller, reducing the impact of interruptions for the team. To cope with lengthy interruptions, teams CAR, FLEET and REAL ESTATE, all following a SCRUM approach, used sprints as "safety zones" in iterations to protect the team from additional interruptions. Finally, our results highlight the importance of the role of SCRUM master in agile ISD. When following KANBAN, team MONITOR had a strong manager, who planned the process and imposed the method on the team, as well as external stakeholders.

Our study has practical implications as well. Practitioners benefit from the three mechanisms to manage interruptions for the agile IT team. Further, the list of interruptions (Table 2) can be used to examine agile IT projects to identify and manage potential sources of interruptions. For designing work environments, it is important to co-locate or establish other mechanisms to exchange informal communication, e.g. through digital collaboration tools (Dery et al. 2017), however, managers are advised to leave space for quiet time and meetings.

Our study is subject to limitations. First, we considered teams that actively used the agile method and thus, we might have missed mechanisms that completely reduce interruptions within the process. However, we examined four different cases and asked interviewees about how the agile procedures were applied and what the consequences were. Second, our analysis was mainly from an internal perspective, reducing external views on how gatekeeping was perceived from the outside. We interviewed one customer and asked respondents about how their actions were perceived by customers. Third, it is inherent to exploratory qualitative work that generalizing the results is challenging. For example, we derived our results from agile teams in large and professional organizational contexts, thus they cannot be taken for granted for smaller, informal organizations like startups or open source projects. Further research should extend our theoretical sampling to other agile methods such as eXtreme Programming as this might provide additional evidence of mechanisms to handle interruptions, as well as distributed teams as their way of collaboration differs compared to co-located teams.

References

Abad, Z. S. H., Ruhe, G., and Bauer, M. 2017. "Understanding Task Interruptions in Service Oriented Software Development Projects: An Exploratory Study," *Software Engineering Research and Industrial Practice (SER&IP), 2017 IEEE/ACM 4th International Workshop on*: IEEE, pp. 34-40.

Addas, S., and Pinsonneault, A. 2015. "The Many Faces of Information Technology Interruptions: A Taxonomy and Preliminary Investigation of Their Performance Effects," *Information Systems Journal* (25:3), pp. 231-273.

Balijepally, V., Mahapatra, R., Nerur, S., and Price, K. H. 2009. "Are Two Heads Better Than One for Software Development? The Productivity Paradox of Pair Programming," *MIS Quarterly* (33:1), pp. 91-118.

Bazigos, M., Smet, A. D., and Gagnon, C. 2015. "Why Agility Pays," *McKinsey Quarterly*:12).

Bechky, B. A., and Okhuysen, G. A. 2011. "Expecting the Unexpected? How Swat Officers and Film Crews Handle Surprises," *Academy of Management Journal* (54:2), pp. 239-261.

Cameron, A.-F., and Webster, J. 2013. "Multicommunicating: Juggling Multiple Conversations in the Workplace," *Information Systems Research* (24:2), pp. 352-371.

Chua, C. E. H., Lim, W.-K., Soh, C., and Sia, S. K. 2012. "Enacting Clan Control in Complex It Projects: A Social Capital Perspective," *MIS Quarterly* (36:2), pp. 577-600.

Colazo, J. A., and Fang, Y. 2010. "Following the Sun: Temporal Dispersion and Performance in Open Source Software Project Teams," *Journal of the Association for Information Systems* (11:11), p. 684.

Crowston, K., Chudoba, K., Watson-Manheim, M. B., and Rahmati, P. 2016. "Inter-Team Coordination in Large-Scale Agile Development: A Test of Organizational Discontinuity Theory," *Proceedings of the Scientific Workshop Proceedings of XP2016*: ACM, p. 2.

Dery, K., Sebastian, I. M., and van der Meulen, N. 2017. "The Digital Workplace Is Key to Digital Innovation," *MIS Quarterly Executive* (16:2).

Galluch, P. S., Grover, V., and Thatcher, J. B. 2015. "Interrupting the Workplace: Examining Stressors in an Information Technology Context," *Journal of the Association for Information Systems* (16:1), p. 1.

Ghobadi, S., and Mathiassen, L. 2017. "Risks to Effective Knowledge Sharing in Agile Software Teams: A Model for Assessing and Mitigating Risks," *Information Systems Journal* (27:6), pp. 699-731.

Glaser, B. G. 1978. *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory*. Mill Valley, CA: Sociology Press.

Hoda, R., Noble, J., and Marshall, S. 2011. "The Impact of Inadequate Customer Collaboration on Self-Organizing Agile Teams," *Information and Software Technology* (53:5), pp. 521-534.

Jenkins, J. L., Anderson, B. B., Vance, A., Kirwan, C. B., and Eargle, D. 2016. "More Harm Than Good? How Messages That Interrupt Can Make Us Vulnerable," *Information Systems Research* (27:4), pp. 880-896.

Jett, Q. R., and George, J. M. 2003. "Work Interrupted: A Closer Look at the Role of Interruptions in Organizational Life," *Academy of management Review* (28:3), pp. 494-507.

Kudaravalli, S., Faraj, S., and Johnson, S. L. 2017. "A Configural Approach to Coordinating Expertise in Software Development Teams," *MIS Quarterly* (41:1).

LaToza, T. D., Venolia, G., and DeLine, R. 2006. "Maintaining Mental Models: A Study of Developer Work Habits," *Proceedings of the 28th international conference on Software engineering*: ACM, pp. 492-501.

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

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

Maznevski, M. L., and Chudoba, K. M. 2000. "Bridging Space over Time: Global Virtual Team Dynamics and Effectiveness," *Organization Science* (11:5), pp. 473-492.

McGrath, J. E. 1991. "Time, Interaction, and Performance (Tip) a Theory of Groups," *Small group research* (22:2), pp. 147-174.

Melnik, G., and Maurer, F. 2004. "Direct Verbal Communication as a Catalyst of Agile Knowledge Sharing," *Agile Development Conference, 2004*: IEEE, pp. 21-31.

Meyer, A. N., Barton, L. E., Murphy, G. C., Zimmermann, T., and Fritz, T. 2017. "The Work Life of Developers: Activities, Switches and Perceived Productivity," *IEEE Transactions on Software Engineering* (43:12), pp. 1178-1193.

Murphy, M. 2016. "Interruptions at Work Are Killing Your Productivity," in: *Forbes*.

Pendem, P., Green, P., Staats, B. R., and Gino, F. 2016. "The Microstructure of Work: How Unexpected Breaks Let You Rest, but Not Lose Focus," Harvard Business School.

Perlow, L. A. 1999. "The Time Famine: Toward a Sociology of Work Time," *Administrative science quarterly* (44:1), pp. 57-81.

Pflügler, C., Wiesche, M., and Krcmar, H. 2018. "Subgroups in Agile and Traditional It Project Teams," *Proceedings of the 51st Hawaii International Conference on System Sciences*.

Przybilla, L., Wiesche, M., and Krcmar, H. 2018. "The Influence of Agile Practices on Performance in Software Engineering Teams: A Subgroup Perspective," *Proceedings of the 2018 ACM SIGMIS Conference on Computers and People Research*: ACM, pp. 33-40.

Recker, J., Holten, R., Hummel, M., and Rosenkranz, C. 2017. "How Agile Practices Impact Customer Responsiveness and Development Success: A Field Study," *Project Management Journal* (48:2), pp. 99-121.

Rigby, D. K., Sutherland, J., and Takeuchi, H. 2016. "Embracing Agile," *Harvard Business Review* (94:5), pp. 40-50.

Slaughter, S. A., Levine, L., Ramesh, B., Pries-Heje, J., and Baskerville, R. 2006. "Aligning Software Processes with Strategy," *MIS Quarterly* (30:4), pp. 891-918.

The Economist. 2017. "Are Digital Distractions Harming Labour Productivity?," in: *The Economist*.

Tregubov, A., Boehm, B., Rodchenko, N., and Lane, J. A. 2017. "Impact of Task Switching and Work Interruptions on Software Development Processes," *Proceedings of the 2017 International Conference on Software and System Process*: ACM, pp. 134-138.

Tripp, J. F., Riemenschneider, C., and Thatcher, J. B. 2016. "Job Satisfaction in Agile Development Teams: Agile Development as Work Redesign," *Journal of the Association for Information Systems* (17:4), p. 267.

Urquhart, C. 2012. *Grounded Theory for Qualitative Research: A Practical Guide*. London, UK: Sage.

Vidgen, R., and Wang, X. 2009. "Coevolving Systems and the Organization of Agile Software Development," *Information Systems Research* (20:3), pp. 355-376.

Watson-Manheim, M. B., Chudoba, K. M., and Crowston, K. 2012. "Perceived Discontinuities and Constructed Continuities in Virtual Work," *Information systems journal* (22:1), pp. 29-52.

Wiedemann, A., and Wiesche, M. 2018. "How to Implement Clan Control in Devops Teams," *Twenty-fourth Americas Conference on Information Systems*, New Orleans, LA.

Wiesche, M., Jurisch, M. C., Yetton, P. W., and Krcmar, H. 2017. "Grounded Theory Methodology in Information Systems Research," *MIS Quarterly* (41:3), pp. 685-701.

Wiklund, K., Sundmark, D., Eldh, S., and Lundqvist, K. 2013. "Impediments in Agile Software Development: An Empirical Investigation," *International Conference on Product Focused Software Process Improvement*: Springer, pp. 35-49.

Zellmer-Bruhn, M. E. 2003. "Interruptive Events and Team Knowledge Acquisition," *Management science* (49:4), pp. 514-528.