**Communications of the Association for Information Systems**

Volume 36 Article 15

2-2015

# The Role of Social Agile Practices for Direct and Indirect Communication in Information Systems Development Teams

Markus Hummel
*Faculty of Economics and Business Administration, Goethe University Frankfurt*, hummel@wiwi.uni-frankfurt.de

Christoph Rosenkranz
*Faculty of Management, Economics and Social Sciences, University of Cologne*

Roland Holten
*Faculty of Economics and Business Administration, Goethe University Frankfurt*

Communications of the Association for Information Systems

CAIS

# The Role of Social Agile Practices for Direct and Indirect Communication in Information Systems Development Teams

Markus Hummel

*Faculty of Economics and Business Administration, Goethe University Frankfurt*

*hummel@wiwi.uni-frankfurt.de*

Christoph Rosenkranz

*Faculty of Management, Economics and Social Sciences, University of Cologne*

Roland Holten

*Faculty of Economics and Business Administration, Goethe University Frankfurt*

Abstract:

Methods for Agile information systems development (ISD) are widely accepted in industry. One key difference in comparison to traditional, plan-driven ISD approaches is that Agile ISD teams rely heavily on direct, informal face-to-face communication instead of indirect and formal documents, models, and plans. While the importance of communication in Agile ISD is generally acknowledged, empirical studies investigating this phenomenon are scarce. We empirically open up the "black box" of the Agile ISD process to enhance the knowledge about the communication mechanisms of Agile ISD teams. We conducted a case study at two medium-sized ISD companies. As our primary data collection technique, we carried out semi-structured interviews, which we complemented with observations and, in one case, a survey. Our study's main contribution is a set of so-called social Agile practices that positively impact the direct communication of team members. Our data suggests including the Agile practices co-located office space, daily stand-up meeting, iteration planning meeting, pair programming, sprint retrospective, and sprint review in this set. Furthermore, we investigate the role of more formal, indirect communication in Agile ISD projects. We highlight areas in which formal documents remain important so that a trade-off between indirect and direct communication is necessary.

**Keywords:** Agile Information Systems Development, Agile Practices, Communication, Case Study.

Volume 36, Article 15, pp. 273-300, February 2015

The manuscript was received 13/06/2014 and was with the authors 4 months for 2 revisions.



Communications of the Association for Information Systems

## I. INTRODUCTION

Software engineering offers a large body of knowledge on how to develop software and how to support and manage the development process (Abran, Moore, Bourque, & Dupuis, 2004; Iivari, Hirschheim, & Klein, 2004). Information systems development (ISD) methods range from being disciplined, sequential, and iterative (Black, Boca, Bowen, Gorman, & Hinchey, 2009; Boehm, 1988) to emergent or even amethodical approaches (Truex, Baskerville, & Travis, 2000). During the last few decades, Agile ISD methods such as Scrum (Schwaber & Beedle, 2002), Extreme Programming (XP) (Beck, 1999), and most recently Lean Software Development (LSD) (Poppendieck & Poppendieck, 2003) have been proposed in order to address the issue of exceedingly dynamic and complex environments by explicitly embracing and incorporating change (Beck et al., 2001). Following these methods, the adaption to changing requirements is achieved by trading strict, plan-driven control for more flexibility and self-organized team autonomy; the overall ISD process is not planned and scheduled upfront, and progress is made in small iterative phases, which encourages constant change, frequent delivery of features and working software, and many customer contacts and team interactions (Cockburn & Highsmith, 2001; Highsmith & Cockburn, 2001). The latter is one of the most striking aspects of Agile ISD because business customers and developers are encouraged to work together daily by coordinating and continuously sharing project information mostly through direct communication such as face-to-face conversations rather than through plans, specifications, documentation, or models (Melnik & Maurer, 2004; Pikkarainen, Haikara, Salo, Abrahamsson, & Still, 2008; Ramesh, Cao, & Baskerville, 2010).

Despite the broad acceptance and adoption of Agile ISD in industry (Dingsøyr, Nerur, Balijepally, & Moe, 2012), little research has conceptually or empirically examined its underlying theoretical relationships and working mechanisms (Ågerfalk, Fitzgerald, & Slaughter, 2009; Iivari & Iivari, 2011; Lee & Xia, 2010). Research efforts have contributed to the theoretical foundations of Agile ISD by pinpointing the facets of agility (Sarker, Munson, Sarker, & Chakraborty, 2009; Sarker & Sarker, 2009) or by suggesting a conceptual basis for Agile ISD methods (Batra, VanderMeer, & Dutta, 2011; Nerur & Balijepally, 2007), but further research is needed that contributes to the missing "theoretical glue" (Conboy, 2009, p. 330) of Agile ISD. In this paper, we argue that communication, as one of the allegedly most fundamental aspects of Agile ISD, is a key factor that needs to be understood to contribute towards this missing theoretical glue. The importance of communication for ISD in general (Corvera Charaf, Rosenkranz, & Holten, 2013; Gallivan & Keil, 2003; Guinan & Bostrom, 1986; Rosenkranz, Corvera Charaf, & Holten, 2013) and Agile ISD in particular (Hummel, Rosenkranz, & Holten, 2013; Pikkarainen et al., 2008; Sarker & Sarker, 2009) has been highlighted by several previous studies. In this paper, we further enhance the knowledge on the theoretical underpinnings of Agile ISD by investigating which Agile practices have a strong impact on the communication mechanisms of project teams. Furthermore, we do not only focus on direct communication, but we also examine the role of indirect communication, which is supposedly neglected in the Agile ISD process (Beck et al., 2001). Consequently, we investigate the following two research questions: (1) Which Agile practices are essential for enabling direct communication of team members?, and (2) In which areas of an Agile ISD process is indirect communication still important?

In this paper, we shed light on this nascent research area and provide an empirical description of team communication behavior on the basis of Agile ISD practices. Our findings are based on a qualitative case study with two ISD companies that succeeded in using Agile practices by satisfying customer needs and delivering software of high quality in an efficient and effective way.

The remainder of the paper is structured as follows. In Section 2, we discuss the theoretical background in the field of Agile ISD and communication. In Section 3, we describe our research design, including the case study setting, data collection, and data analysis techniques. In Section 4, we present our findings and, in Section 5, discuss our results and their limitations. Lastly, in Section 6, we conclude the paper.

## II. THEORETICAL BACKGROUND & RELATED WORK

### Information Systems Development and Agile Methods

Even though ISD has been investigated in various fields such as software engineering, information systems research, and project management for several decades (Iivari et al., 2004; Kitchenham et al., 2009; Sambamurthy & Kirsch, 2000), various studies point to recurring problems that still arise in ISD (e.g., ISD projects failing to meet cost

estimates or time constraints, ISD projects delivering software with fewer functionalities than planned, or ISD projects that are total failures) (e.g., Agrawal & Chari, 2007; Kautz, Madsen, & Nørbjerg, 2007; Nelson, 2007; Xia & Lee, 2005). The majority of traditional ISD methods are structured and plan-driven, and rely on mechanisms of indirect communication such as project plans, specification documents, or models to support and control coordination, communication, and knowledge sharing among team members (Black et al., 2009; Boehm, 1988; Kraut & Streeter, 1995; Royce, 1970). In dynamic and rapidly changing environments, however, these formal mechanisms impede quick reactions to changes (Kraut & Streeter, 1995); that is, "Rather than being bastions of order in an uncertain world, traditional teams may indeed become chaotic should their plan-driven organization be overwhelmed by events" (Vidgen & Wang, 2009, p. 374).

During the last decades, the traditional structured approaches have been complemented with iterative and Agile principles and practices for management and development. The resulting Agile ISD methods such as Scrum (Schwaber & Beedle, 2002), Extreme Programming (XP) (Beck, 1999), and most recently lean software development (LSD) (Poppendieck & Poppendieck, 2003) are increasingly adopted in industry (Dingsøyr et al., 2012). In the following discussion, we will focus on Agile practices of Scrum and XP, which are the most widely used Agile ISD methods in industry (Dingsøyr et al., 2012; VersionOne, 2012).

Although there is no widely accepted definition of Agile ISD, research efforts define the different facets of agility more clearly (Conboy, 2009; Sarker et al., 2009). Following these attempts, we understand "agility" as:

> *the continual readiness of an ISD method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment.* (Conboy, 2009, p. 340)

In industry, an employed Agile ISD approach consists of a set or combination of specific Agile practices, which can originate from different methods (Abrahamsson, Salo, Ronkainen, & Warsta, 2002). Practices describe specific actions taken by specific people at specific times when they are engaged in a higher-level organizational routine (Feldman & Pentland, 2003), which is the routine of ISD in our context. For example, such practices include executing prescriptive, formalized processes such as continuous integration (Beck, 1999), and adopting emergent, unformalized routines such as reflective and retrospective discussions during regular meetings (Derby & Larsen, 2007; Schwaber & Beedle, 2002).

Over many years, the body of knowledge on Agile ISD has constantly grown (Dingsøyr et al., 2012) to overcome the lack of empirical and rigorous studies (Dybå & Dingsøyr, 2008; Erickson, Lyytinen, & Keng, 2005); however, there is still a "backlog of research problems" (Rajlich, 2006, p. 70). For example, there is a clearly identified need for more and better research to determine in which situations Agile ISD should be applied (Ågerfalk et al., 2009). Most importantly, researchers are calling for stronger theoretical foundations of Agile ISD and conceptual frameworks to understand Agile ISD (Ågerfalk et al., 2009; Dingsøyr et al., 2012). Indeed, "despite the growing popularity and importance of Agile approaches, little research has empirically examined their key concepts and underlying theoretical relationships" (Lee & Xia, 2010, p. 82).

## Agile Information Systems Development and Communication

Investigating communication offers a promising theoretical lens for examining Agile practices as a key concept of Agile ISD, and for empirically observing the effects of different practices on communication (Hummel et al., 2013). The importance of communication in Agile ISD is generally recognized (Hummel et al., 2013; Melnik & Maurer, 2004; Pikkarainen et al., 2008). For example, Agile ISD practices strongly emphasize constant communication among team members and customers, particularly through face-to-face interaction (Cockburn & Highsmith, 2001; Highsmith & Cockburn, 2001). Some Agile approaches are based almost entirely on spoken communication with the customer (Ramesh et al., 2010, p. 451), and Agile ISD is vividly described as a "cooperative game of invention and communication" (Cockburn, 2002, p. 28). However, the nature and effects of communication mechanisms in Agile ISD project teams are not well understood, and the literature refers to a very broad understanding of communication (Hummel et al., 2013).

An important way to categorize different communication mechanisms is to use direct versus indirect communication channels. Communication channels are the media chosen to convey the message from sender to receiver. The literature distinguishes two main categories: direct and indirect channels of communication (Dennis, Fuller, & Valacich, 2008; Kock, 2004; Te'eni, 2001). Direct communication involves using those channels that involve synchronous, spoken discourse (e.g., oral conversations involving face-to-face, telephone, or other media) and non-verbal aspects such as gestures, facial expressions, scent, and so forth. Indirect communication involves using those channels that use asynchronous, written discourse (e.g., documents, emails, books, or other media).

Previous research on Agile ISD and communication has focused mainly on XP practices. One of XP's most beneficial aspects is the increased direct communication among team members because of less documentation (Barlow et al., 2011; Cao, Mohan, Ramesh, & Sarkar, 2013; Fruhling & de Vreede, 2006; Kollmann, Sharp, & Blandford, 2009; Layman, Williams, & Cunningham, 2004; Ramesh et al., 2010). As a result, most studies claim that improved communication is one of the main benefits of employing XP practices because it facilitates formal and informal communication (e.g., Pikkarainen et al., 2008). Surprisingly, however, some research has shown that XP does not necessarily affect the communication of teams with their surrounding organization or with customer representatives (Svensson & Höst, 2005). Table 1 summarizes previous findings on the impact of the twelve XP practices on communication[1].

**Table 1: Previous Findings on XP Practices and Communication (based on Hummel et al., 2013)**

| XP practice | Impact on the communication of ISD teams |
|---|---|
| Coding standards | By giving all developers the same rules for developing code, communication among team members is facilitated and simplified (Hazzan & Dubinsky, 2003; Maruping, Zhang, & Venkatesh, 2009b). |
| Collective code ownership | When a whole team is responsible for all of the code, communication is benefited indirectly by relying on timely communication between developers when sharing code (Xiaohu, Bin, Zhijun, & Maddineni, 2004). |
| Continuous integration | When continuously implementing new code, regular feedback on the quality of the code in the form of communication is enabled (Xiaohu et al., 2004). |
| On-Site customer / whole team | When locating the team and the customer nearby, a positive impact on communication quality was observed (Wojciechowski, Wesolowski, & Complak, 2010). The co-located office space serves as a communication channel (e.g., Dorairaj, Noble, & Malik, 2012; Svensson & Höst, 2005). |
| Pair programming | When programming in pairs, communication skills are fostered and interactions between the programmers are improved (e.g., Vidgen & Wang, 2009). |
| Planning game | Regular planning game meetings provide a frame for regular communication among the team and with the customer (e.g., Cao, Mohan, Peng, & Ramesh, 2009; Dorairaj et al., 2012). |
| Refactoring | Influences communication indirectly by enabling simplicity, which is needed to communicate efficiently (Conboy, Pikkarainen, & Wang, 2007). |
| Simple design | |
| Small releases | Enables more-efficient communication in meetings (Pikkarainen et al., 2008) and provides rapid feedback mechanisms between the customer and developers (Xiaohu et al., 2004). |
| Sustainable pace | We could not identify any study that deals with the "sustainable pace" practice and its implications on communication. |
| System metaphor | Serves as a communication platform with a common vocabulary that is stated to improve mutual understanding and communication quality (Beck, 1999; Dall'Agnol, Janes, Succi, & Zaninotto, 2003; Sharp & Robinson, 2004). |
| Unit testing (incl. automated acceptance tests) | Automated acceptance tests are mentioned to be an effective communication medium for customers and developers to exchange business requirements, domain knowledge, and the implementation status (Melnik, Maurer, & Chiasson, 2006; Park & Maurer, 2009). This view is questioned by Haugset and Hanssen (2008), who find that specifying tests is hard for customers because they lack understanding of the system, which may hinder communication. |

As for Scrum and direct communication, researchers have observed that Scrum enables a higher communication frequency and communication quality because it provides a frame for communication that reminds team members to interact closely and regularly (Paasivaara, Durasiewicz, & Lassenius, 2008). Table 2 summarizes previous findings of individual Scrum practices on communication[2].

**Table 2: Summary of Previous Findings on Scrum Practices and Communication (based on Hummel et al., 2013)**

| Scrum practice | Impact on the communication of ISD teams |
|---|---|
| Daily stand-up Meetings | The daily meetings are judged to be one of the most important mechanism for communication because it ensures that team members talk regularly with each other (Pikkarainen et al., 2008). It is reported to have positive effects on the communication frequency among the whole team and one-on-one communication after the meetings is also encouraged (Paasivaara, Durasiewicz, & Lassenius, 2009). |

[1] For a detailed description of each XP practice, please consult the textbook on XP (Beck, 1999).
[2] For a detailed description of each Scrum practice, please consult the Scrum textbook (Schwaber & Beedle, 2002).

| | |
|---|---|
| Iteration planning meetings | These meetings, which are scheduled once during an iteration, are used for communication about features, requirements, and lessons learnt (Pikkarainen et al., 2008). Both the development team and the customer representative are engaged in this regular communication (Pikkarainen et al., 2008; Schwaber & Beedle, 2002). |
| Sprint review & retrospective meetings | |
| Scrum-of-Scrums | Regular meetings of representatives of different teams enable communication and coordination between several teams (Paasivaara & Lassenius, 2010). Negative effects of those meetings include the possibility of miscommunication and misunderstanding because of the reliance on a single person that represents the team (Hossain, Babar, Hye-young, & Verner, 2009). |

In total, Scrum and XP offer 17 different practices that are to a large extent supposed to be related to improved communication. In industry, an often limited combination of practices from different Agile ISD methods are frequently employed (Abrahamsson et al., 2002; Fitzgerald, Hartnett, & Conboy, 2006), and our knowledge on which practices should be employed and in which combination in order to improve communication among team members is limited. This also applies to the use of more indirect communication in Agile ISD such as formal documentation because few studies or practitioner reports provide guidance for using documents in Agile ISD projects. The following list includes the main results of previous studies on the role of indirect communication in the Agile ISD process:

- Architectural and requirements specifications should be documented at a high level of abstraction (Selic, 2009; Stettina, Heijstek, & Fægri, 2012)
- Change decisions and knowledge that is relevant for all project members should be documented in a wiki (Hoda, Noble, & Marshall, 2010; Stettina & Heijstek, 2011)
- Customer feedback should be documented (Hoda et al., 2010)
- Documentation of user requirements and major decisions in large projects is needed (Batra, Weidong, VanderMeer, & Dutta, 2010)
- Documentation at the code level is judged to be not useful (Selic, 2009)
- Project dictionary for documenting business terms, their meanings, and contexts of use is reasonable (Hoda et al., 2010), and
- User stories should be documented (Stettina & Heijstek, 2011).

In sum, first insights on the role of indirect communication in Agile ISD projects exist, but research studies remain scarce, and our knowledge on the areas of Agile ISD projects in which indirect communication is limited.

In this study, we open up the "black box" of the ISD process (Siau, Long, & Ling, 2010, p. 92) and uncover the concept of communication in Agile ISD by empirically studying the direct and indirect communication mechanisms of project teams. While affirming the relationship between Agile practices and communication mechanisms in general and on a broad level, the literature is surprisingly silent about conceptualizing, evaluating, or refuting this claim (Hummel et al., 2013). Few studies (e.g., Maruping, Venkatesh, & Agarwal, 2009a; Pikkarainen et al., 2008; Rosenkranz, Vranesic, & Holten, 2014; Sawyer, Guinan, & Cooprider, 2010) have actually investigated and compared communication and knowledge sharing in teams using different ISD methods in any detail. The lack of studies has been recognized for a long period, as Bostrom noted in 1989: "While many authors stress the importance of effective communication in the (C)IS development process, few provide empirical evidence or concrete suggestions" (Bostrom, 1989, p. 280). We conclude that this statement still seems to hold today and we find it astounding when contrasted with the rich statements highlighting the alleged pivotal role of communication for ISD in general and Agile ISD in particular. We contribute toward filling this gap by fleshing out the necessary conceptualizations of these phenomena, which we achieve by conducting a case study that focuses on the communication mechanisms, patterns, and behaviors of Agile ISD teams.

## III. RESEARCH DESIGN

### Method

Because of the still-limited knowledge on the precise role of communication mechanisms in Agile ISD, we studied two cases in depth. Qualitative research in general (Sarker, Xiao, & Beaulieu, 2013), and more specifically case studies, are well accepted in the IS field (Dubé & Paré, 2003; Lee, 1989). They allow researchers to develop a deep understanding of the phenomena in their socially embedded and organizational contexts (Myers, 2009; Orlikowski & Iacono, 2001).



Communications of the Association for Information Systems

In essence, we conducted a positivist, exploratory case study following established guidelines (Dubé & Paré, 2003; Eisenhardt, 1989; Lee, 1989; Yin, 2003). We were profoundly informed by prior literature (see Section 2); however, we do not test an existing model or theory in our case studies because the communication mechanisms in Agile ISD are still unclear. Instead, we elaborate and flesh out relevant Agile practices and the communication behavior of project teams using the collected empirical data (Eisenhardt, 1989). Despite the problems inherent in case studies' subjectivity, this approach is suitable for gaining new insights in under-researched areas by investigating ISD in a real-life context (Yin, 2003). While conducting the case study, we took special care in following widely accepted principles of qualitative research in information systems research (Sarker et al., 2013). Specifically, comparisons among multiple sites help demonstrate the influence of variability in context and, therefore, yield more general research results than single cases (Dubé & Paré, 2003). On the basis of a literal replication strategy (Yin, 2003) in an exploratory analysis of two firms developing software for two different industries, we expect that the conditions of the cases will lead to similar results, which would also demonstrate that the impact of Agile practices on team communication and behavior is not industry specific.

## Case Settings

As an appropriate case site, we sought two organizations that had undergone an allegedly successful transformation to a more-Agile ISD approach for developing software. As such, we chose two medium-sized companies that develop software and provide IT consulting services. For reasons of confidentiality, we name the case companies as Alpha and Omega. Table 3 overviews the two case sites.

**Table 3: Overview of Case Sites**

| | Alpha | Omega |
|---|---|---|
| Domain | IT support and consultancy for financial services; ISD for own software products | IT support and consultancy in general; ISD for external customers |
| Employees | 60 | 80 |
| Country | Germany | Germany |
| Team Size | 3-15 | 3-9 |
| Team Educational Level | High | High |
| Domain Expertise | High | High |
| Culture | Customer-centric consulting and development culture | Customer-centric coaching, consulting, and development culture |
| Agile Practices Used | All Scrum meeting practices and selected XP practices (continuous integration, coding standards, collective ownership, co-located office space, refactoring, unit testing) | All Scrum and XP practices |

At the time of the study, Alpha employed about 60 employees in its software-development department, of which 40 employees were reported to regularly use a mix of Scrum and XP practices in their work. Consulting in the financial sector generated the main income of the company. To take advantage of the knowledge that is acquired in the consulting activities, the company started to develop their own specialized software products for customers. Three years prior to our investigation, the managers decided to start developing in an Agile way to increase customer satisfaction and project performance. The software developers did not employ a specific Agile ISD method such as Scrum or XP in a strict way. Instead, the organization adopted several practices of different Agile ISD methods that fitted their needs.

Omega had carried out all development activities using Agile practices since its founding. It had its first experience with XP concepts as early as 1998, whereas it adopted Scrum in 2002. A total of 80 employees were employed at this company. The company offered three equally important business areas that all built on the use of Agile practices: first, the company developed in-house solutions for individual customers based on their respective needs. Second, the company offered consulting at the customer's site in order to coach ISD teams in how to successfully use Agile practices. Third, the company held Agile ISD training seminars. In contrast to Alpha, Omega followed Scrum "by the book". Since Scrum's focus is mainly on project management, the company additionally followed XP practices for implementing code.

We considered the selected case sites to be good samples on at least three counts: first, the case organizations were past the adoption stage and were actively using Agile ISD approaches and practices in their ISD projects. Second, both companies worked in different domains, which ensures that our findings are applicable in two different contexts. Also, differences in the applicability of Agile ISD in varying contexts need to be considered (Ågerfalk et al., 2009). Alpha's domain was predominantly developing software for financial industries, whereas Omega developed software for clients across multiple industries and also engaged in consulting activities on the use of Agile practices

independent of industry. Third, the case organizations differed in that Alpha employed only selected Agile practices and Omega employed "pure" Scrum and XP, which allowed us to examine specifically how Agile practices can contribute to team communication and behavior, independent of a specific ISD method.

We studied multiple organizations to increase generalizability; that is, with two organizations, we reassured ourselves that the events and processes in one well-described setting are not wholly idiosyncratic (Miles & Huberman, 1994). Therefore, the result of the explanation-building process was also the creation of a cross-case analysis, not simply an analysis of each individual case.

## Data Collection

We investigated both case sites from different angles by employing a multi-method design (Gable, 1994), which allows for within-case triangulation concerning the data and includes both quantitative (e.g., questionnaires) and qualitative data-collection techniques (e.g., field observations and interviews).

At Alpha, we conducted eight in-depth semistructured interviews with employees on the manager and team level. This helped us to gain initial insight from different angles into how communication and Agile ISD were implemented in practice at Alpha. We employed a pre-designed interview guideline as a checklist, but allowed room for deviations and open questions. The guideline comprised questions derived from prior literature on Agile ISD (e.g., Beck et al., 2001; Maruping et al., 2009a; Pikkarainen et al., 2008) and on communication (e.g., Te'eni, 2001) (see Appendix A for the interview guideline). The interviews lasted 30-90 minutes (50 minutes on average), and we audio-recorded and later transcribed them. After eight interviews, we reached theoretical saturation because few new categories or themes emerged. Secondly, we conducted in-depth field observations at the company's location. Field observations are suitable for a detailed analysis of social interactions in ISD contexts (Cavaye, 1996; Silverman, 1998). During this data collection, the first author was involved as an observer. He mostly collected the data by taking handwritten notes, which are "naturally occurring data" (Silverman, 1998, p. 9), including inter-subjective descriptions of the physical setting, people, and activities, but also reflective notes in terms of subjective ideas, speculations, and thoughts. The goal of this part of the case study was to learn how Agile ISD was actually performed at Alpha in day-to-day work. We observed which practices were being used for a typical ISD project at Alpha to identify and explore what was occurring with respect to Agile practices and communication. After one week of observations, the emerging insights ceased to be different, so that we concluded that our results adequately represented the particular setting of our study (Maxwell & Chmiel, 2013). For the most part of the week, the researcher observed one specific project that included about 15 persons in five rooms on the same level of the office building. Besides the observations in the office rooms, the researcher also participated in five daily-standup meetings and one sprint planning meeting. Throughout the observation phase, the first author conducted nine interviews with developers, product owners, and project leaders that lasted 20 minutes on average. The first author asked selected questions of the interview guideline during those interviews, depending on the specific development situation. Those interviews were captured by taking handwritten notes. The researcher also collected structured data which was collected in two sheets that he filled out during the observation phase: it contained notes on the use and frequency of Agile practices and on the employed communication media. The structured sheets helped us to identify Agile practices and the role of different communication media that are important for communication, collaboration, and interaction among team members. Third, we distributed a questionnaire online in the development department. We used the survey tool LimeSurvey to create the online survey based on measurement scales of existing surveys (e.g., Lee & Xia, 2010; Maruping et al., 2009a; Misra, Kumar, & Kumar, 2009) (see Appendix B for the survey questions). We distributed the questionnaire by e-mail among all 60 employees of the development department. The responses were anonymized and the employees were encouraged to participate on a voluntary basis by their supervisors. In total, we were able to collect 43 completely answered and returned questionnaires. Due to the small sample size and the exploratory nature of our study, we used the questionnaire to reflect on and corroborate the findings of the previous two data-collection stages, and did not analyze models or conduct multivariate analyses.

At Omega, we conducted semistructured interviews with six developers. Due to access restrictions and since the interviews validated and hardly extended the findings from company Alpha, we focused solely on interviews at Omega and did not engage in field observations and surveys (theoretical saturation). To enable triangulation, we intentionally focused on the developer level at Omega, including lesser-experienced persons that developed individual software for specific customers.

Appendix C provides information about the interviewees and their background information, including selected project characteristics. We conducted all interviews in German; to present the results, the first author translated selected quotes into English.

## Data Analysis

For data analysis, we transcribed and coded all qualitative data. We used the software MaxQDA to carry out open coding for identifying relevant practices and the communication patterns in an inductive-deductive manner (Miles & Huberman, 1994). For example, we deduced codes for the Agile practices based on existing literature (see Section 2), while we coded their effect on social interaction inductively based on the case data, where we also coded the relationships between the practices and communication. This open procedure is also necessary to capture essential practices that impact on the communication patterns of Agile ISD teams. Appendix D provides selected examples on how the codes and conclusions emerged.

Due to the limited sample size and the supporting nature of the survey, which we never intended to be a theory-testing device, we used descriptive statistics to analyze the questionnaire data to support our qualitative analysis. All conclusions and percentages presented in the analysis section are based on the combined percentages of persons who rated the respective questions as “6” (agree) or “7” (strongly agree) on a 7-point Likert scale. We applied basic statistical analyses (e.g., measuring central tendencies in frequency distribution and spreads) and summarized the data in a meaningful way such that we could match this to our data from the previous data collection stages.

# IV. ANALYSIS AND RESULTS

## Identified Categories

Table 4 presents and defines the final list of practices that we included in the set of so-called social Agile practices after our data analysis. Recognizing the importance of social interactions, social behavior, and communication in the ISD process, we chose the term “social” to acknowledge the fact that the practices relate to the interaction of team members (e.g., Corvera Charaf et al., 2013; Gallivan & Keil, 2003; Guinan & Bostrom, 1986; Orlikowski & Gash, 1994; Rosenkranz et al., 2013).

Previous research has highlighted that major problems of ISD are “not so much technological as sociological in nature” (DeMarco & Lister, 1987, p. 4). Consequently, we define social Agile practices as a subset of ISD practices from several methods that directly relate to a considerable degree to the social interaction, collaboration, and direct communication of ISD team members, and that we found to have a direct impact on communication mechanisms in the ISD process (Maruping et al., 2009a; So & Scholl, 2009). Direct communication in ISD refers to synchronous communication between team members, mainly characterized by informal face-to-face conversations (Dennis et al., 2008; Kock, 2004; Te'eni, 2001), whereas indirect communication is characterized by asynchronous communication between team members, mainly characterized by formal documentation, models, or plans (Dennis et al., 2008; Kock, 2004; Te'eni, 2001) (see Section 2).

**Table 4: Overview and Definition of Identified Categories (Social Agile Practices)**

| Categories | Description and definition |
|---|---|
| Co-located office space | The “co-located office space” / whole team practice refers to a high degree of physical proximity of the team members, preferably in the same room in a co-located office space (So & Scholl, 2009). This also includes the customer who is supposed to use the system (Beck, 1999). |
| Daily stand-up meetings | Daily stand-up meetings or stand-up meetings are regular daily meetings in which project members gather around a table and discuss what they are doing at the moment and what they have done before (Schwaber & Beedle, 2002). |
| Iteration planning meetings | Iteration planning meetings are regular meetings at the beginning of each iteration that are used to decide on the user stories or features that will be implemented next (Schwaber & Beedle, 2002). |
| Pair programming | Pair programming means that developers form teams of two and implement code together on one machine (Beck, 1999). |
| Sprint retrospective meetings | Sprint retrospectives are regular meetings at the end of each iteration that are used to reflect on past efforts (Schwaber & Beedle, 2002). |
| Sprint review meetings | Sprint reviews are regular meetings at the end of each iteration that are used to present the results of the last iteration (Schwaber & Beedle, 2002). |

In the following sections, we first overview how these concepts influence direct and indirect communication at both of our case sites. We then provide a cross-case analysis.

## Case Alpha

### Direct Communication

To avoid disturbances from outside the project team, our interviewees at Alpha used and considered a shared working environment as promoted by the co-located office space practice to be useful because distractions from people outside of the project were less likely due to the physical separation. The Alpha team members had no fixed workplace and used laptops. Each office was equipped with displays and the employees could plug in their laptops to have a fully equipped workplace anywhere in the building. Some developers preferred a constant workplace, but most interviewees considered this flexibility to be positive: “Most people like it. A few people rather do not like it because they are not that flexible, they want to sit at their own desk with their colleagues.” (Project leader A3).

Co-located office spaces facilitated direct face-to-face communication. If all the team members were working in the same office or on the same level of the building, the interviewees perceived face-to-face as the most efficient form of communication because of the reduced distance between people who intend to communicate: “I definitely think that face-to-face is the most efficient method.” (Project leader A3).

At Alpha, the co-located office space practice also entailed that customer representatives, embodied by product owners, were actively included throughout the entire ISD process. The product owners were employed with Alpha and not the actual end-customer. The reason for this is that Alpha developed standard software, which was supposed to be developed for many different end customers. In addition, a customer in the typical sense did not exist when financial regulatory requirements (e.g., for ensuring compliance with reporting standards) had to be implemented. The product owners were responsible for discovering and specifying the requirements, so they bridged the gap between consultants and analysts from Alpha’s consulting activities and business departments and the developers. They had the last word on all product-related decisions such as necessary functions and features. Their focus was more on the business side because they often lacked technical knowledge and expertise. According to 18 percent of the survey respondents, the daily collaboration of product owners and the development team was perceived to be very close and very positive. In addition, 75 percent of the managers stated that there was a lot of face-to-face conversation with the product owner due to their more frequent involvement in the development process: “Those persons who adopt those roles have definitely a very strong communication task.” (Senior manager A1).

The survey results also show that daily stand-up meetings were well established at Alpha, which 65 percent of respondents reported. During the observation phase, the researcher even witnessed daily stand-up meetings that were carried out in maintenance projects that were not considered to be run in an Agile way. Daily stand-up meetings were used for identifying problems or hindrances, and to simply talk about the current status of the project, future procedures, and the tasks’ personal status:.

> *We are coming together with the project leader who starts and says: ok, what have you done and (it) goes around and everybody has to say what he is currently working on and if he has any blocker.* (Product owner A7)

The other team members could help to solve the problems that an individual developer was facing. Our interviewees noted that the content of the daily stand-up meetings became more and more detailed during the course of a project:

> *So if you’re coming more to an end (of the project) you have much more details to discuss (than) at the beginning of the iteration or beginning of the project. Daily stand-ups at the beginning of a project are on a very high level.* (Product owner A7)

During the daily stand-up meetings, respondents mentioned that they preferred face-to-face communication as their preferred medium for task description and distribution, and for answering questions. Our structured data on the employed communication media, which indicates that those meetings lead to more frequent and higher amounts of communication compared to when the case company was not carrying out those meetings, also supported this finding. Specifically, the daily gatherings triggered subsequent communication events after the meeting that may not have been relevant to all participants (e.g., two developers later discuss a problem in more detail, which has been brought up in the daily meeting): “The communication increased not only for the time of the daily meetings, but also for the rest of the day.” (Project leader A2).

Moreover, the observer witnessed several occasions at Alpha when problems in daily work were solved in joint face-to-face discussions. During the observation phase, however, the researcher also noticed that there were individual,

introverted team members who seemed to have problems with the Agile mindset of frequent face-to-face communication during the daily stand-up meetings.

The iteration planning meeting involved the product owner, project leader, stakeholders, and developers that updated the project plan and decided which requirements or features would be implemented in the iteration: “Mostly at the beginning of a sprint, we plan together with the developers the activities for the next iteration.” (Project leader A2).

Our observations indicate that the degree of participation of the development team depended on the project leader who was carrying out the project. The requirements were defined to a large extend at the beginning of a project, so the meeting was used to agree on which of the upfront defined requirements were going to be implemented in the current iteration and how the implementation is carried out in detail: “Nearly everything was fixed before the project. But not in detail.” (Project leader A3).

Our interviewees considered sprint reviews to be a preparation for the next iteration planning meeting. Our observational data suggests that it depended on the project leader whether the meeting regularly took place or not. During the review meetings, the Scrum master, the product owner, stakeholders, and the development team spoke about the results of the last sprint: “In the review meetings, the employees of the projects are also taking part, and you talk about the most recent information about the project.” (Senior manager A1).

Our interviewees perceived the review meetings to be an extended daily stand-up meeting because relevant problems that hindered the successful progression of the project were also discussed and reflected on: “So you are saying (in those meetings): ok we have finished these tasks and we have these blockers. It is more like daily stand-up meetings so to say.” (Product owner A7).

There was always the danger that those meetings lead to a questioning of the whole course of action of the project: “I think those meetings can be carried out in a better way than we do right now because those meetings often escalate in a debate of principles.” (Senior manager A1).

Similarly, all team members employed sprint retrospectives as direct face-to-face meetings. The meetings were used to reflect on past efforts to identify areas that needed to be improved. However, those meetings were often carried out at the end of the project and not at the end of each iteration, which was mostly due to time constraints:

> *We mostly do a retrospective at the end of an iteration, but we did not do it the last two times because of time constraints.* (Project leader A2)

> *What we always do is a lessons learned meeting at the end of the project, about the whole project.* (Project leader A3)

### Indirect Communication

Only about 5 percent of the survey respondents felt that documentation was important for communication, but 50 percent of the survey respondents stated that indirect communication such as documentation still played an important or very important role in projects. Our observations and interviews confirm this: although Alpha was supposedly developing in an Agile way, the amount of documentation was still high: “Personally, I feel that the ratio of documentation is still quite high. This means that many things that were discussed face-to-face have to be written down.” (Project leader A3).

One interviewee indicated that the quantity of documents had not been reduced since using Agile ISD, but the way in which documents are created had changed: “We still create certain formal documents. Whether there is less documentation now…hm…maybe it is different, I would say.” (Project leader A2).

Our observational data confirms that there were less formal, textual specifications and more freely designed models, diagrams, and graphics. All in all, the interviewees agreed that it was not desirable to abandon all documentation because it imposes obligations:

> *Personally, I think it is relatively important that the written word exists in certain areas, so I do not think much of the extreme that nothing has to be written down because this puts employees in a position in which nothing is obligatory.* (Senior manager A1)

There were few obligations to create certain documents, so it was up to the specific project leader to decide on what should be documented: “We really have very very few formal must’s , and do’s, and don’t’s.” (Senior architect A8).

Documentation was still stated to be needed in several areas of an Agile project. An important document was the product backlog in which the user stories were stored. The product backlog was administered by the product owners, who collected and distributed the input for the backlog document:

> *So everybody in our company is able to create a feature or a ticket (as input for the product backlog) and my job is to assign these to a version and discuss it with the project leader on how it fits in the iteration.* (Product owner A7)

User stories are an indirect communication medium which team members used to structure a problem. They were not to exceed one paragraph of text:

> *Yeah, for the user stories, the most important point was that the user stories structure the problem and they have a hierarchy…. So the user stories had to be written down in Microsoft Word document with every user story being one paragraph.* (Senior manager A6)

Our interviewees considered documents in form of Excel lists to be necessary to trace and validate tests: “In my opinion, test documentation is absolutely necessary.” (Team leader A5).

Furthermore, they considered documentation of architecture and decisions to be important: “There are certain things that you have to document in any case, such as architecture things and so forth, and decisions.” (Project leader A3).

Our informal interviews and the structured data on the employed communication media show that documentation was also necessary for generating mutual understanding in employees’ training phase because it helped to define important terms. It was also needed for controlling and supporting the administration of open tasks and for time- and budget-calculation. Customers needed documentation not only as a manual, but also as a description of the source code so that they could implement their own modifications: “There is more documentation of the software itself. We deliver the software with the source code to the customer, so there has to be an appropriate (documentation) quality.” (Project leader A2).

The team members used documents to specify data tables, primary keys, and other information, which helped to create mutual understanding. Employees of quality management checked the documents for undefined terms and called for more information if terms were unclear. In larger teams, our interviewees stated that documentation was highly important for creating mutual understanding:

> *At a certain size of the team, I mean we are working in a team of about 15 persons or so in the development of a certain project, so you do not have any other choice than to build up common understanding by certain documents.* (Senior manager A1)

Furthermore, documentation was needed for team members or stakeholders to agree on a certain meeting’s outcome. Writing down the basic facts in a document avoided misunderstandings and generated accountability: “Documents are needed that retain how the facts are understood which helps to make sure that both sides share a common understanding.” (Senior manager A1).

## Case Omega

### Direct Communication

At Omega, if possible, all persons involved in the development process sat co-located, as advised by the co-located office space practice. Our interviewees felt that the best development outcome is achieved when all team members are located close-by: “I think Agile works definitely the best when all persons sit in the same room”. (Developer O4).

Especially in larger projects, interviewees reported that it was helpful if several offices on the same level of the building were reserved for one project team. Our interviewees indicated that benefits of a co-located office space include timely responses to requests and a high quality of communication: “In general, I think that that you should prefer to work together at one site, at best in one room, so there are simply quick interactions, and a good communication can take place.” (Developer O1).

As for on-site customers at Omega, in contrast to Alpha, a customer representative of the actual client was situated several days a week with the development team because the company was developing individual software. The development team preferred if the customer was co-located for the whole week, but this was not feasible because the product owner usually had other responsibilities at the customer's company as well: "The product owner has this double role; he has to do other things as well. For this reason, I think it is difficult to say 'hey move your office to our firm now'." (Developer O4).

If the product owner was not available, short response times were possible by using cell phones. Either this or the physical proximity of the customer representative allowed parties to clarify questions quickly: "If I want to know something from the customer, I have to be able to contact him and ask." (Developer O3).

Especially for complex tasks, our interviewees emphasized that a close cooperation with the customer was essential: "The cooperation with the product owner had to be very tight because the business side was complicated." (Developer O5).

We also found that the daily stand-up meetings at Omega relied heavily on direct communication: "I think the daily meetings support the communication strongly." (Developer O1).

The daily meetings were used to exchange information that had not been noticed or exchanged during the rest of the day. They were not supposed to be used as a replacement for the regular communication. Our interviewees considered it as a platform that offered the possibility for good communication among the team members, but it highly depended on the communication skills and personalities of the participating individuals. Since the expertise was supposed to be evenly distributed among the team members, the problems of other team members were relevant to all team members during the daily stand-up meetings.

Our interviewees stated that pair programming was one of the crucial Agile practices that fostered communication at Omega: "Pair Programming is essential because, well, you can't get away from one another, you have to talk." (Developer O5).

Pair programming was partly done at Omega depending on the task. Our interviewees considered pair programming useful for developing complex algorithms because they entail many questions. The developers thought that simple tasks could be done more efficiently without a partner and they thought it was boring to implement straight-forward tasks in pairs. In order to program in pairs successfully, we found that the developers needed good communication skills so that they were able to express what they wanted to do and also understand what other people meant. Pair programming was also important for training new employees:

> *For training purposes, we simply did pair programming for at least two weeks with those persons, because we looked at different areas of the system within those two weeks so that they have seen everything and they know where to find something. (Developer O1)*

The iteration planning meetings were important for communicating with the customer representative because the development team sat together and talked with the customer representative about the user stories that should be implemented next: "The iteration planning meetings are crucial for communication with the product owner." (Developer O5).

Before each meeting, the team sat together for a "grooming" session in order to talk about the user stories for the next planning meeting. This helped to prevent wrong decisions and problematic estimations during the actual planning meeting. In planning meetings, it was also decided which tasks should be implemented in pairs, depending on the complexity of the tasks. One of the implications of those meetings was more frequent communication with the customer in contrast to traditional, plan-based methods. The developers preferred the direct communication during the iteration planning meetings in contrast to formal documents:

> *There is a real communication because otherwise, you have the tendency to communicate by documents in terms of sending out requirements specifications. Then the customer is saying 'I don't like this' and adds a few comments, then I work those changes in, send it out again, and things like that. That is not communication, that is something that smells a little bit like communication but actually it is error-prone, tedious and for me personally, not a way to spend my everyday life.* (Developer O4)

At Omega, sprint reviews were carried out regularly at the end of a sprint. During the review meetings, the Scrum master, product owner, and development team communicated in a direct way about the results of the last sprint: "At

the end, there is the review which all parties use for communicating. What is the current status, what has been completed, what has not, what are new problems. All this contributes to the communication." (Developer O6).

After trying out the latest version of the product, the customer said whether the results met their expectations or not: "Hopefully, the product owner sees what he has already seen before (while actively participating in the iterations). You talk about things that you have heard before, but in a more detailed way." (Developer O5).

Sprint retrospectives took place regularly at the end of an iteration at Omega. The product owner, Scrum master, and development team reflected on the past sprint, so it was also important for the communication with the customer, and among the team members. The reflection was based on questions such as: "What have we done? Did it work? What should we do in a different way next time?" (Developer O6).

Our interviewees considered it important for communication because those meetings enabled them to talk about how to improve communication among the team: "Retrospectives are meaningful because you can talk about communication problems, and you can try to solve them in a detailed way." (Developer O5).

Our interviewees emphasized that those meetings were only a success if the involved persons are willing to learn: "You have to be able to reflect; without reflection, there is no retrospective, at least no one that is useful, and you have to be interested in improving yourself." (Developer O4).

### Indirect Communication

At Omega, the most important form of indirect communication was the product backlog in which the user stories were stored. The aggregated user stories in the product backlog summarized the functionality of the system. The product backlog was also the basis for the customer's manual, but, generally, there was no explicit documentation about the software functionality. Only if the customer explicitly wanted technical documentation was it created and priced in:

> *If the customer is saying that he needs a technical documentation, then I will ask him why and what for because I will probably not need it. If he still wants to have it, he will get it, but the effort for creating the documentation will be estimated and has to be paid for.* (Developer O6)

Documentation was mainly necessary for communicating with people from outside of the development team, whereas documentation about the code was not necessary when following clean code principles: "And then there is code documentation which is omitted in our case because we follow the clean code principles which means that you should always produce readable code." (Developer O1).

The developers stated that code documentation was not necessary if one had basic knowledge about how the different components of a web application worked. Even if there was documentation, the developers preferred to look directly in the code because reading documentation was tedious to read and it took a lot of time to become acquainted with it: "even if you have a big book as a documentation, until you worked through that in order to find some explanation, it is quicker to look into the code." (Developer O1).

A way to get a feeling for the code without reading through documentation was executing test cases because the developer could quickly see how the specific code fragment is working. Comments in the code were judged to be hardly helpful. Documents were considered more important in larger projects that included several teams: "There was a phase in which I was sitting several weeks with the other team in order to handover our work. I can imagine that this handover would have been a few days shorter if there was documentation." (Developer O4).

As for the requirements-engineering process, the requirements were not gathered in detail at the beginning of the project at Omega. Instead, only the broad picture of the project was captured in a short document and details were discussed directly: "Since we did not specify all the requirements concerning the design and notifications in detail, we did do that in dialogue." (Developer O5).

At Omega, documenting the retrospective results is also judged to be useful: "Yes, we document the retrospective results, simply because we can see at the end of the iteration whether the issue improved or not." (Developer O5).

A negative aspect of documentation was that it was outdated most of the time. If the documentation was updated at a certain point in time and work on the system continued, the documentation was outdated quickly. Instead of updating code documentation, direct communication was preferred (e.g., via phone): "I rather invest this effort in a quick call in which I explain to the persons what has just happened." (Developer O4).

At Omega, we found that documentation did not always affect the building of mutual understanding in a positive way; for example, in terms of requirements documents. In some cases, it encouraged confusion:

> *If I read a document containing requirements, then I do have my understanding of it, but this is not necessarily what the customer wants.* (Developer O3)

> *I can write 10 pages, and if 10 people read it, then I do have 10 opinions at the end, 10 interpretations. Direct communication helps to clarify ambiguity.* (Developer O3)

## Cross-Case Analysis

The results of our cross-case analysis confirm the findings in prior literature that communication plays a fundamental role in the Agile ISD process. Moreover, our data suggests that a specific subset of Agile practices, what we call social Agile practices (co-located office space, daily stand-up meetings, iteration planning meetings, pair programming, sprint review and sprint retrospective meetings), are the main contributors to direct communication, collaboration, and interaction among team members. As for the use of social Agile practices across the two cases, Table 5 aggregates and summarizes our findings at both case sites regarding the impact of social Agile practices on direct communication in a cross-case display. Other often-enacted Agile practices (such as unit testing or coding standards) did not have a strong direct impact on the communication mechanisms of the ISD teams at our case sites and are, therefore, not included in this set of social Agile practices.

**Table 5: Summary of Social Agile Practices and the Impact on Direct Communication**

| Social Agile practice | Impact on direct communication |
|---|---|
| Co-located office space | Enables quick and efficient direct communication due to close proximity.<br>The customer representative is able to provide immediate, direct feedback and answers to questions.<br>Enables to clarify questions face-to-face, which was considered most efficient by our interviewees. |
| Daily stand-up meetings | Enables high frequency and high amount of direct communication in and after the meetings.<br>Project members are kept up-to-date by direct communication.<br>Problems are identified and solved with direct communication. |
| Iteration planning meeting | Enables direct, frequent communication in the team and with the customer representative while discussing the features that will be implemented in the current iteration.<br>A grooming session of the development team before the actual planning meeting helps to avoid misunderstandings by direct communication. |
| Pair programming | Enables frequent, direct communication because developers have to talk to each other when implementing code.<br>Necessary know-how for the training of new employees is conveyed in a direct manner. |
| Sprint retrospective | Enables direct, frequent communication in the team while identifying and solving problems, and while discussing lessons learnt.<br>Enables the improvement of communication because it offers a platform to talk about the communication behavior of the team. |
| Sprint review | Enables direct, frequent communication in the team and with the customer representative while discussing the results of the last iteration.<br>Was considered as an extended daily stand-up meeting by our interviewees because it allows direct communication on major issues that hinder project progression. |

We found that developers at both case sites strongly preferred direct over indirect communication. This emphasis on direct, oral communication in turn led to a reduced amount of documentation. Information concerning the daily development process was exchanged face-to-face rather than through formal documents. In the end, however, a compromise between formal documentation and informal communication was necessary because both extremes were neither feasible nor desirable in real-life projects. Combining the insights of both our cases, Table 6 presents specific areas in which documentation was still considered to be important at Alpha and Omega, especially for enabling efficient communication and mutual understanding.

**Table 6: Useful Documents in an Agile ISD Context (Indirect Communication)**

| Document | Content |
|---|---|
| Documentation of important decisions | Includes, for example, agreements with stakeholders or decisions concerning the architecture of the software that cannot be reversed easily. |
| Product backlog | Includes the requirements in the form of high-level user stories. |

| | |
|---|---|
| Retrospective agreements | Includes lessons learnt of previous iterations. Those agreements should be documented in order to see whether the lessons learnt were subsequently implemented. |
| Source code documentation | Is necessary when the customer wants to create own modifications of the software. In that case, the source code documentation is an essential part of the product for which the customer has to pay for. |
| Test documentation | Includes information about the traceability and validation of tests. |
| Time, budget and task coordination | Supports the administration of the available time and budget, and the coordination of open tasks. |
| Training material | Includes the definition of important terms and procedures in order to generate mutual understanding with new employees early on. |

Other, more-complex documentation such as detailed formal requirements specifications, either up-front or in-between, impedes efficient communication because there are different perceptions and interpretations of the same document, which is caused by differing views of business and IT employees. In addition, formal requirements documents may be outdated quickly. Source code documentation is also outdated quickly and should be downplayed because it is easier and quicker to look directly into the code instead of reading documents about it. The source code should only be documented if it is supposed to be delivered to the customer (see Table 6).

## V. DISCUSSION

Our case study findings contribute to the body of knowledge on Agile ISD by gathering knowledge about the relation between Agile practices and communication on the team and project level. In short, we define a set of social Agile ISD practices and provide empirical evidence that this set of practices is a facilitator of direct communication. Our main contribution is an empirically corroborated conceptualization (the set of social Agile practices) for explaining and predicting social interactions and communication mechanisms in Agile ISD. We include the practices co-located office space, daily stand-up meetings, iteration planning meetings, pair programming, sprint retrospective, and sprint review meetings in this set. Additionally, we highlight areas in which more indirect communication mechanisms such as documents are still important for Agile ISD projects. Documentation is required in terms of training material, test documentation, and documenting decisions, whereas it seems to be counterproductive in other areas such as detailed requirements specifications and code documentation.

Our findings have important implications for research and practice. For theory, our results contribute to establishing the theoretical underpinnings of Agile ISD by shedding further light on the role of communication in the Agile ISD process. We extend previous findings that emphasize the social side of Agile practices (Maruping et al., 2009a; Robinson & Sharp, 2005; So & Scholl, 2009) by offering a concrete set of six Agile practices that have a strong influence on direct communication, collaboration, and interaction among ISD teams. This serves as a blueprint for developing hypotheses in terms of using Agile practices and their effect on the communication behavior of Agile ISD teams. We also find support for the media naturalness proposition (Kock, 2004) in our empirical data because direct, natural communication seems to have a positive impact on communication efficiency, whereas more indirect, unnatural communication may also inhibit communication efficiency and the building of mutual understanding. Similarly, one of the propositions of media synchronicity theory (Dennis et al., 2008); namely, the need for synchronous, direct communication for agreeing on the meaning of an information (convergence process) is reflected in our data. The reason for this is that direct communication, as emphasized by social Agile practices, seems to enable a high communication performance in terms of mutual understanding, which may or may not come at a cost in terms of retaining and storing knowledge for later. We cannot comment on the latter because our study was not longitudinal in nature. Lastly, our findings complement existing literature on indirect communication in Agile ISD projects (e.g., Batra et al., 2010; Selic, 2009) by specifying clear boundary conditions for distinguishing areas in which indirect communication is needed or not. While direct communication is most efficient during daily development work, documentation is needed for documenting important decisions, test cases and training materials.

For practice, our study highlights the importance of choosing the right communication media in the right situations, as already indicated by media fit studies (Daft & Lengel, 1986; Daft, Lengel, & Trevino, 1987; Kock, 2004). As we describe in Section 4, we provide concrete reference points for media selection in similar development situations. We also reveal the essential set of Agile practices that should be implemented in order to improve direct communication among team members and with customers. Our investigation on the role of direct and indirect communication gives a rationale for why especially large ISD projects or distributed ISD endeavors may have problems when adopting Agile ISD methods “by the book”. Additional complexity may be induced with rising numbers of stakeholders in large projects, which inhibits communication and coordination. Furthermore, purely informal communication may be difficult or even not doable in distributed projects (Batra, 2009; Batra et al., 2011). Direct communication may, therefore, not always be the best choice and may also become an inhibitor (Adolph,

Kruchten, & Hall, 2012; Vidgen and Wang, 2009). In consequence, indirect communication is still of high importance in the Agile ISD process in certain areas. Our findings show that it is not recommendable to abolish all kinds of documentation because situations remain where it contributes to establishing mutual understanding of team members and customers.

We encourage future research to develop and test hypothesis to validate our exploratory findings (e.g., using laboratory experiments or surveys, and with longitudinal, cross-sectional, or panel studies). Research is also needed for detailing our conceptualizations of the theoretical concepts "direct" and "indirect" communication in order to transfer our findings to the empirical measurement level. We can also expect to learn valuable insights if we investigate how more technical Agile practices affect communication. A first crucial step would be to define this set of technical Agile practices.

Our study is not without limitations. Since we only investigated two cases, our findings cannot be generalized to a high level. Therefore, we cannot yet extrapolate that teams using social Agile ISD practices are more likely to have better communication and positive development outcomes compared to teams using traditional, plan-driven ISD approaches in each and every situation. Our qualitative data does not allow us to present clear-cut causal relationships. Moreover, our data collection was limited in the sense that the observation phase of our research at Alpha lasted for only one week, and we did not distribute a questionnaire nor did we engage into field observations at Omega. Nevertheless, we are confident that we captured the essential insights of our case organizations due to our multiple data sources and since we reached theoretical saturation.

## VI. CONCLUSION

We conducted a case study to extend our knowledge on the impact of Agile practices on the communication mechanisms of ISD teams. We investigated two case sites in detail by using several data collection techniques. In total, our data basis included 14 semistructured interviews, one week of field observations (including nine informal interviews and two structured sheets), and 43 completely returned questionnaires. Our data analysis was based on an inductive-deductive coding procedure, which helped us to flesh out the underlying conceptualizations of direct and indirect communication in an Agile ISD setting, which, in turn, led to a set of what we call social Agile practices.

An improved communication behavior is one of the main reasons why Agile practices are adopted in practice, but research lags behind in corroborating this claim. We take important steps towards narrowing the large number of Agile practices down to those that improve the communication behavior of project teams. The main contribution of this paper is the set of social Agile practices, which includes the practices co-located office space, daily stand-up meetings, iteration planning meetings, pair programming, sprint retrospective, and sprint review meetings. Our conceptualizations of social Agile practices and the role of direct and indirect communication serve as blueprint for the development of hypotheses. Based on our empirical results, we generally propose that those practices have a strong impact on the communication mechanisms, interactions, and collaboration behavior of Agile ISD project teams. Testing specific hypotheses based on this general proposition will increase the generalizability of our findings.

In addition, we provide an extensive discussion on the role of indirect communication in the ISD process by providing concrete reference points when to rely on more indirect communication media (see Table 6). We found that a compromise between direct and indirect communication is necessary. The empirical results show that complex, extensive documentation such as detailed requirements specifications and code documentation should be downplayed, whereas documents such as the product backlog and documentation of tests, decisions, and retrospective agreements are important. In a practical context, it is not feasible nor desirable to abolish all documentation.

In sum, we help to resolve the contradictory and inconclusive findings concerning Agile ISD and communication, which may contribute towards the missing "theoretical glue" of Agile ISD. Our findings support the conclusion that specific Agile practices indeed influence the manner of and the frequency with which team members communicate. Further confirmatory research is needed, perhaps using surveys or experiments, in order to test this claim.

## ACKNOWLEDGMENTS

We are indebted to the editorial review for the constructive feedback and suggestions, which helped to considerably improve the paper. In addition, we thank Marc Räkers, Kurt Jäger, and Ferdinand Davertzhofen for their support of the study. The German Research Foundation (DFG) funded part of this study under record no. HO 2196/4-1 and record no. RO 3650/8-1.

## REFERENCES

*Editor's Note*: The following reference list contains hyperlinks to World Wide Web pages. Readers who have the ability to access the Web directly from their word processor or are reading the paper on the Web, can gain direct access to these linked references. Readers are warned, however, that:

1. These links existed as of the date of publication but are not guaranteed to be working thereafter.
2. The contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.
3. The author(s) of the Web pages, not AIS, is (are) responsible for the accuracy of their content.
4. The author(s) of this article, not AIS, is (are) responsible for the accuracy of the URL and version information.


Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: Review and analysis. In *Espoo 2002* (VTT Technical Report 478). VTT Technical Research Centre of Finland.

Abran, A., Moore, J. W., Bourque, P., & Dupuis, R. (Eds.). (2004). *Guide to the software engineering body of knowledge* (SWEBOK®). Los Alamitos, CA: IEEE.

Adolph, S., Kruchten, P., & Hall, W. (2012). Reconciling perspectives: A grounded theory of how people manage the process of software development. *Journal of Systems and Software*, *85*(6), 1269-1286.

Ågerfalk, P. J., Fitzgerald, B., & Slaughter, S. A. (2009). Flexible and distributed information systems development: State of the art and research challenges. *Information Systems Research*, *20*(3), 317-328.

Agrawal, M., & Chari, K. (2007). Software effort, quality, and cycle time: A study of CMM level 5 projects. *IEEE Transactions on Software Engineering*, *33*(3), 145-156.

Barlow, J. B., Giboney, J. S., Keith, M. J., Wilson, D. W., Schuetzler, R. M., Lowry, P. B., & Vance, A. (2011). Overview and guidance on Agile development in large organizations. *Communications of the Association for Information Systems*, *29*(1), 25-44.

Batra, D. (2009). Modified Agile practices for outsourced software projects. *Communications of the ACM*, *52*(9), 143-148.

Batra, D., VanderMeer, D. E., & Dutta, K. (2011). Extending Agile principles to larger, dynamic software projects: A theoretical assessment. *Journal of Database Management*, *22*(4), 73-92.

Batra, D., Weidong, X., VanderMeer, D., & Dutta, K. (2010). Balancing Agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry. *Communications of the Association for Information Systems*, *27*(21), 379-394.

Beck, K. (1999). *Extreme Programming explained: Embrace change*. Boston, MA: Addison-Wesley.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Agile manifesto*. Retrieved from http://www.Agilemanifesto.org/

Biocca, F., Harms, C., & Gregg, J. (2001). The networked minds measure of social presence: Pilot test of the factor structure and concurrent validity. Retrieved from http://www.soc.napier.ac.uk/~cs181/Modules/CM/Biocca.pdf

Black, S. E., Boca, P. P., Bowen, J. P., Gorman, J., & Hinchey, M. G. (2009). Formal versus Agile: Survival of the fittest. *IEEE Computer*, *49*(9), 39-45.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, *21*(4), 61-72.

Bostrom, R. P. (1989). Successful application of communication techniques to improve the systems development process. *Information & Management*, *16*(5), 279-295.

Cao, L., Mohan, K., Peng, X., & Ramesh, B. (2009). A framework for adapting Agile development methodologies. *European Journal of Information Systems*, *18*(4), 332-343.

Cao, L., Mohan, K., Ramesh, B., & Sarkar, S. (2013). Adapting funding processes for Agile IT projects: An empirical investigation. *European Journal of Information Systems*, *22*(2), 191-205.

Cavaye, A. L. M. (1996). Case study research: A multi-facetted research approach for IS. *Information Systems Journal*, *6*(3), 227-242.

Cockburn, A. (2002). *Agile software development*. Boston, MA: Addison-Wesley.

Cockburn, A., & Highsmith, J. (2001). Agile software development: The people factor. *IEEE Computer*, *34*(11), 131-133.

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, *20*(3), 329-354.

Conboy, K., Pikkarainen, M., & Wang, X. (2007). *Agile practices in use from an innovation assimilation perspective: A multiple case study*. Paper presented at the International Conference on Information Systems, Atlanta, GA.

Corvera Charaf, M., Rosenkranz, C., & Holten, R. (2013). The emergence of shared understanding: Applying functional pragmatics to study the requirements development process. *Information Systems Journal*, *23*(2), 115-135.

Daft, R. L., & Lengel, R. H. (1986). Organizational information requirements, media richness and structural design. *Management Science*, *32*(5), 554-571.

Daft, R. L., Lengel, R. H., & Trevino, L. K. (1987). Message equivocality, media selection, and manager performance: Implications for information systems. *MIS Quarterly*, *11*(3), 355-366.

Dall'Agnol, M., Janes, A., Succi, G., & Zaninotto, E. (2003). Lean management—a metaphor for Extreme Programming? In M. Marchesi & G. Succi (Eds.), *Extreme Programming and Agile processes in software engineering, XP 2003* (pp. 26-32). Heidelberg, Germany: Springer.

DeMarco, T., & Lister, T. (1987). *Peopleware: Productive projects and teams*. New York, NY: Dorset House Publishing.

Dennis, A. R., Fuller, R. M., & Valacich, J. S. (2008). Media, tasks, and communication processes: A theory of media synchronicity. *MIS Quarterly*, *32*(3), 575-600.

Derby, E., & Larsen, D. (2007). *Agile retrospectives: Making good teams great!* Washington, DC: Pragmatic Bookshelf.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of Agile methodologies: Towards explaining Agile software development. *Journal of Systems and Software*, *85*(6), 1213-1221.

Dorairaj, S., Noble, J., & Malik, P. (2012). Knowledge management in distributed Agile software development. In *AGILE 2012* (pp. 64-73). Los Alamitos, CA: IEEE.

Dubé, L., & Paré, G. (2003). Rigor in information systems positivist case research: Current practices, trends, and recommendations. *MIS Quarterly*, *27*(4), 597-635.

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

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of Management Review*, *14*(4), 532-550.

Erickson, J., Lyytinen, K., & Keng, S. (2005). Agile modeling, Agile software development, and Extreme Programming: The state of research. *Journal of Database Management*, *16*(4), 88-100.

Feldman, M. S., & Pentland, B. T. (2003). Reconceptualizing organizational routines as a source of flexibility and change. *Administrative Science Quarterly*, *48*(1), 94-118.

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising Agile methods to software practices at Intel Shannon. *European Journal of Information Systems*, *15*(2), 200-213.

Fruhling, A., & de Vreede, G.-J. (2006). Field experiences with Extreme Programming: Developing an emergency response system. *Journal of Management Information Systems*, *22*(4), 39-68.

Gable, G. G. (1994). Integrating case study and survey research methods: An example in information systems. *European Journal of Information Systems*, *3*(2), 112-126.

Gallivan, M. J., & Keil, M. (2003). The user-developer communication process: A critical case study. *Information Systems Journal*, *13*(1), 37-68.

Guinan, P., & Bostrom, R. P. (1986). Development of computer-based information systems: A communication framework. *SIGMIS Database*, *17*(3), 3-16.

Haugset, B., & Hanssen, G. K. (2008). Automated acceptance testing: A literature review and an industrial case study. In G. Melnik, P. Kruchten, & M. Poppendieck (Eds.), *AGILE 2008* (pp. 27-38). Los Alamitos, CA: IEEE.

Hazzan, O., & Dubinsky, Y. (2003). Bridging cognitive and social chasms in software development using Extreme Programming. In M. Marchesi & G. Succi (Eds.), *Extreme Programming and Agile processes in software engineering, XP 2003* (pp. 47-53). Heidelberg, Germany: Springer.

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *IEEE Computer*, *34*(9), 120-122.

Hinds, P. J., & Mortensen, M. (2005). Understanding conflict in geographically distributed teams: The moderating effects of shared identity, shared context, and spontaneous communication. *Organization Science*, *16*(3), 290-307.

Hoda, R., Noble, J., & Marshall, S. (2010). How much is just enough? Some documentation patterns on Agile projects. In *European Conference on Pattern Languages of Programs 2010.* New York, NY: ACM.

Hossain, E., Babar, M. A., Hye-young, P., & Verner, J. (2009). Risk identification and mitigation processes for using Scrum in global software development: A conceptual framework. In S. Sulaiman & N. M. M. Noor (Eds.), *Asia-Pacific Software Engineering Conference 2009* (pp. 457-464). Los Alamitos, CA: IEEE.

Hummel, M., Rosenkranz, C., & Holten, R. (2013). The role of communication in Agile systems development: An analysis of the state-of-the-art. *Business & Information Systems Engineering*, *5*(5), 343-355.

Iivari, J., Hirschheim, R., & Klein, H. K. (2004). Towards a distinctive body of knowledge for information systems experts: Coding ISD process knowledge in two IS journals. *Information Systems Journal*, *14*(4), 313-342.

Iivari, J., & Iivari, N. (2011). The relationship between organizational culture and the deployment of Agile methods. *Information & Software Technology*, *53*(5), 509-520.

Kautz, K., Madsen, S., & Nørbjerg, J. (2007). Persistent problems and practices in information systems development. *Information Systems Journal*, *17*(3), 217–239.

Keskin, H. (2009). Antecedents and consequences of team memory in software development projects. *Information & Management*, *46*(7), 388-396.

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering—a systematic literature review. *Information and Software Technology*, *51*(1), 7-15.

Ko, D.-G., Kirsch, L. J., & King, W. R. (2005). Antecedents of knowledge transfer from consultants to clients in enterprise system implementations. *MIS Quarterly*, *29*(1), 59-85.

Kock, N. (2004). The psychobiological model: Towards a new theory of computer-mediated communication based on Darwinian evolution. *Organization Science*, *15*(3), 327-348.

Kollmann, J., Sharp, H., & Blandford, A. (2009). The importance of identity and vision to user experience designers on Agile projects. In Y. Dubinsky, T. Dybå, S. Adolph, & A. Sidky (Eds.), *AGILE 2009* (pp. 11-18). Los Alamitos, CA: IEEE.

Kraut, R. E., & Streeter, L. A. (1995). Coordination in software development. *Communications of the ACM*, *38*(3), 69-81.

Layman, L., Williams, L., & Cunningham, L. (2004). Exploring Extreme Programming in context: An industrial case study. In *AGILE 2004* (pp. 32-41). Los Alamitos, CA: IEEE.

Lee, A. S. (1989). A scientific methodology for MIS case studies. *MIS Quarterly*, *13*(1), 32-50.

Lee, G., & Xia, W. (2010). Toward Agile: An integrated analysis of quantitative and qualitative field data. *MIS Quarterly*, *34*(1), 87-114.

Majchrzak, A., Beath, C. M., Lim, R., & Chin, W. W. (2005). Managing client dialogues during information systems design to faciliate client learning. *MIS Quarterly*, *29*(4), 653-2672.

Maruping, L. M., Venkatesh, V., & Agarwal, R. (2009a). A control theory perspective on Agile methodology use and changing user requirements. *Information Systems Research*, *20*(3), 377-399.

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

Maxwell, J., & Chmiel, M. (2013). Generalization in and from qualitative analysis. In U. Flick (Ed.), *Handbook of qualitative data analysis* (pp. 540-553). London, England: Sage.

Melnik, G., & Maurer, F. (2004). Direct verbal communication as a catalyst of Agile knowledge sharing. In *AGILE 2004* (pp. 21-31). Los Alamitos, CA: IEEE.

Melnik, G., Maurer, F., & Chiasson, M. (2006). Executable acceptance tests for communicating business requirements: Customer perspective. In J. Chao, M. Cohn, F. Maurer, H. Sharp, & J. Shore (Eds.), *AGILE 2006* (pp. 35-46). Los Alamitos, CA: IEEE.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: A sourcebook of new methods*. Beverly Hills, CA: Sage.

Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting Agile software development practices. *Journal of Systems and Software*, *82*(11), 1869-1890.

Myers, M. D. (2009). *Qualitative research in business & management.* London, England: Sage.

Nelson, R. R. (2007). IT project management: Infamous failures, classic mistakes, and best practices. *MIS Quarterly Executive*, *6*(2), 67-78.

Nerur, S., & V. Balijepally (2007). Theoretical reflections on Agile development methodologies. *Communications of the ACM*, *50*(3), 79-83.

Orlikowski, W. J., & Gash, D. (1994). Technological frames: Making sense of information technology in organisations. *ACM Transactions on Information Systems*, *12*(2), 174-207.

Orlikowski, W. J., & Iacono, C. S. (2001). Research commentary: Desperately seeking the "IT" in IT research: A Call to theorizing the IT artifact. *Information Systems Research*, *12*(2), 121-134.

Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2008). Distributed Agile development: Using Scrum in a large project. In *International Conference on Global Software Engineering 2008* (pp. 87-95). Los Alamitos, CA: IEEE.

Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2009). Using Scrum in distributed Agile development: A multiple case study. In *International Conference on Global Software Engineering 2009* (pp. 195-204). Los Alamitos, CA: IEEE.

Paasivaara, M., & Lassenius, C. (2010). Using Scrum practices in GSD projects. In *Agility Across Time and Space (*pp. 259-278). Berlin Heidelberg, Germany: Springer.

Park, S., & Maurer, F. (2009). Communicating domain knowledge in executable acceptance test driven development. In P. Abrahamsson, M. Marchesi, & F. Maurer (Eds.), *Agile processes in software engineering and Extreme Programming, XP 2009* (pp. 23-32). Berlin Heidelberg, Germany: Springer.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of Agile practices on communication in software development. *Empirical Software Engineering*, *13*(3), 303-337.

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An Agile toolkit* (1st ed.). Amsterdam: Addison-Wesley Longman.

Rajlich, V. (2006). Changing the paradigm of software engineering. *Communications of the ACM*, *49*(8), 67-70.

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: An empirical study. *Information Systems Journal*, *20*(5), 449-480.

Robinson, H., & Sharp, H. (2005). The social side of technical practices. In H. Baumeister, M. Marchesi, & M. Holcombe (Eds.), *Extreme Programming and Agile processes in software engineering, XP 2005* (pp. 1342-1344). Heidelberg, Germany: Springer.

Rosenkranz, C., Corvera Charaf, M., & Holten, R. (2013). Language quality in requirements development: Tracing communication in the process of information systems development. *Journal of Information Technology*, *28*(3), 198-223.

Rosenkranz, C., Vranesic, H., & Holten, R. (2014). Boundary interactions and motors of change in requirements elicitation: A dynamic perspective on knowledge sharing. *Journal of the Association for Information Systems*, *15*(6), 306-345.

Royce, W. W. (1970). Managing the development of large software systems: Concepts and techniques. In *IEEE WESCON 1970 (*pp. 1-9). TRW.

Sambamurthy, V., & Kirsch, L. J. (2000). An integrative framework of the information systems development process. *Decision Sciences*, *31*(2), 391-411.

Sarker, S., Munson, C. L., Sarker, S., & Chakraborty, S. (2009). Assessing the relative contribution of the facets of Agility to distributed systems development success: An analytic hierarchy process approach. *European Journal of Information Systems*, *18*(4), 285-299.

Sarker, S., & Sarker, S. (2009). Exploring Agility in distributed information systems development teams: An interpretive study in an offshoring context. *Information Systems Research*, *20*(3), 440-461.

Sarker, S., Xiao, X., & Beaulieu, T. (2013). Qualitative studies in information systems: A critical review and some guiding principles. *MIS Quarterly*, *37*(4), iii-xviii.

Sawyer, S., Guinan, P. J., & Cooprider, J. (2010). Social interactions of information systems development teams: A performance perspective. *Information Systems Journal*, *20*(1), 81-107.

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum*. Upper Saddle River, NJ: Prentice Hall.

Selic, B. (2009). Agile documentation, anyone? *IEEE Software*, 26(6), 11-12.

Sharp, H., & Robinson, H. (2004). An ethnographic study of XP practice. *Empirical Software Engineering*, *9*(4), 353-375.

Siau, K., Long, Y., & Ling, M. (2010). Toward a unified model of information systems development success. *Journal of Database Management*, *21*(1), 80-101.

Silverman, D. (1998). Qualitative research: Meanings or practices? *Information Systems Journal*, *8*(1), 3-20.

Smith, K. G., Smith, K. A., Olian, J. D., Sims, H. P., Jr., O'Bannon, D. P., & Scully, J. A. (1994). Top management team demography and process: The role of social integration and communication. *Administrative Science Quarterly*, *39*(3), 412-438.

So, C., & Scholl, W. (2009). Perceptive Agile measurement: New instruments for quantitative studies in the pursuit of the social-psychological effect of Agile practices. In P. Abrahamsson, M. Marchesi, & F. Maurer (Eds.), *Agile processes in software engineering and Extreme Programming, XP 2009* (pp. 83-93). Heidelberg, Germany: Springer.

Stettina, C. J., & Heijstek, W. (2011). Necessary and neglected? An empirical study of internal documentation in Agile software development teams. In *International Conference on Design of Communication 2011* (pp. 159-166). New York, NY: ACM.

Stettina, C. J., Heijstek, W., & Fægri, T. E. (2012). Documentation work in Agile teams: The role of documentation formalism in achieving a sustainable practice. In *AGILE 2012* (pp. 31-40). Los Alamitos, CA: IEEE.

Svensson, H., & Höst, M. (2005). Views from an organization on how Agile development affects its collaboration with a software development team. In F. Bomarius & S. Komi-Sirviö (Eds.), *Product focused software process improvement* (pp. 487-501). Heidelberg, Germany: Springer.

Te'eni, D. (2001). Review: A cognitive-affective model of organizational communication for designing IT. *MIS Quarterly*, *25*(2), 251-312.

Truex, D., Baskerville, R., & Travis, J. (2000). A methodical systems development: The deferred meaning of systems development methods. *Accounting, Management and Information Technology*, *10*, 53-79.

VersionOne. (2012). 7th annual state of Agile development survey. Retrieved from http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf

Vidgen, R., & Wang, X. (2009). Coevolving systems and the organization of Agile software development. *Information Systems Research*, *20*(3), 355-376.

Wallace, L., Keil, M., & Rai, A. (2004). Understanding software project risk: A cluster analysis. *Information & Management*, *42*(1), 115-125.

Wojciechowski, A., Wesolowski, M., & Complak, W. (2010). Experimental evaluation of "on-site customer" XP practice on quality of software and team effectiveness. In *On the Move to Meaningful Internet Systems: OTM 2010 Workshops* (pp. 269-278). Berlin Heidelberg, Germany: Springer.

Xia, W., & Lee, G. (2005). Complexity of information systems development projects: Conceptualization and measurement development. *Journal of Management Information Systems*, *22*(1), 45-83.

Xiaohu, Y., Bin, X., Zhijun, H., & Maddineni, S. (2004). Extreme Programming in global software development. In *Canadian Conference on Electrical and Computer Engineering 2004* (pp. 1845-1848). Los Alamitos, CA: IEEE.

Yin, R. K. (2003). *Case study research: Design and methods* (3rd ed.). Thousand Oaks, CA: Sage.

## APPENDIX A: INTERVEW GUIDELINE

The interview guideline was composed of the following questions:

- What is your current role in projects?

- How many years of experience do you have in software development?
- How many years of experience do you have in Agile software development?
- Please describe the current development process at Alpha.
- Why did Alpha decide to use Agile software development methods?
- In which case do you consider software development to be “Agile”, and in which case is it not?
- Do you consider the development process at Alpha to be Agile?
- Which Agile practices do you know?
- Which Agile practices are employed at Alpha?
- Who are the different stakeholders in an Agile project?
- What are the success factors of Agile software development? Is communication one of them?
- Which Agile practices have implications on communication within the teams?
- What is the role of communication with the internal customer / stakeholder when developing Agile?
- What is the role of communication within the team when developing Agile?
- Which factors affect the communication process in general when developing Agile?
- Which communication media do you use at Alpha when employing Agile practices?
- Do you employ Agile practices at Alpha that help to develop mutual understanding within the team?
- What is the role of relationships among the team members and how are relationships supported by Agile practices?
- Is there more communication in Agile projects in comparison to traditional development projects?
- How important is formal communication such as documentation for an Agile project?
- Is there more informal than formal communication?
- Are there any aspects of the subject that we did not talk about so far?

## APPENDIX B: SURVEY ITEMS

The scale labels range on a 7-point Likert scale from “strongly disagree” to “strongly agree” (unless noted otherwise). We newly developed the items or based them on existing literature (Biocca, Harms, & Gregg, 2001; Hinds & Mortensen, 2005; Keskin, 2009; Ko, Kirsch, & King, 2005; Lee & Xia, 2010; Majchrzak, Beath, Lim, & Chin, 2005; Maruping et al., 2009; Misra et al., 2009; Smith et al., 1994; Wallace, Keil, & Rai, 2004).

**Collective code ownership**

- Anyone on this team can change existing code at any time.
- If anyone wants to change a piece of code, they need the permission of the individual(s) that coded it.
- Members of this team feel comfortable changing any part of the existing code at any time.

**Continuous integration**

- Members of this team integrate newly coded units of software with existing code.
- We combine new code with existing code on a continual basis.
- Our team does not take time to combine various units of code as they are developed.

**Daily stand-up meetings**

- All team members come together daily for a meeting.
- The whole team meets only when it is necessary.
- The daily meetings do not take longer than 15 minutes.

**Pair programming**

- The tasks of the developers at Alpha are divided in a way that participation of other employees is not necessary.
- On this team, we do our software development using pairs of developers.
- At Alpha, problems during development are solved by two persons in front of a PC.

**Refactoring**

- Where necessary, members of this team try to simplify existing code without changing its functionality.
- We periodically identify and eliminate redundancies in the software code.
- We periodically simplify existing code.

**Sprint planning meetings**

- We plan the entire project before the start of the project.
- The planning is conducted and updated in a regular fashion.

- The features are cut to an iteration so that the maximum length of an iteration (four weeks) is not exceeded.

**Sprint retrospective meetings**
- After each iteration, the team and management reflect jointly on the effectivity of the development work.
- The team and management reflect jointly at the end of the project on the effectivity of the development work.
- The team is included during the reflection of the effectivity of the development work.
- At regular intervals, our team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

**Unit testing**
- We run unit tests on newly coded modules until they run flawlessly.
- Test cases are created before the actual implementation of the component.
- The team tests continuously in order to find errors early.

**Simplicity**
- Development at Alpha is complex.
- We practice simple designs, processes, and approaches in our software development.
- We implement features that are required by the customers - nothing more.
- Development at Alpha is designed so that the code is used for the current project and not for further projects.
- The high complexity of the requirements impedes the development work.

**Communication informality**
- Development team meetings tend to be very formal in nature (reverse-coded).
- Meetings between members of the development team are very informal.
- Communication between members of the TMG is always in writing (reverse coded).
- The development team employs informal rather than formal communication channels.

**Communication intensity**
Scale Label: Very Low to Very High
Please estimate the intensity of the usage of the following media during software development at Alpha.
- Face-to-face Communication.
- Face-to-face with Whiteboard.
- Formal Communication.
- E-Mail.
- Telephon.
- Video Conferences.
- Instant Messaging.

**Face-to-face conversation**
- In projects at Alpha, there is a lot of face-to-face communication.
- There is a lot of face-to-face communication with the product owner during a project.
- There is a lot of face-to-face communication with the stakeholders outside of the team during a project.

**Documentation**
- Documentation is not playing an important role in projects at Alpha.
- In projects at Alpha, there is as less documentation as possible.
- In projects at Alpha, there is as much documentation as necessary.

**Tacit knowledge**
- A large part of the knowledge about the project is not written down in documentation, it only exists inside of the heads of the team members.
- A large part of the knowledge about software products at Alpha is not written down in documentation, it only exists inside of the heads of the team members.
- Knowledge about the project is fixed mainly in documents and distributed by those.

**Mutual understanding**
- At the beginning of an iteration, it is assured that team members have a mutual understanding about the next steps.

- At the beginning of a project, it is assured that team members have a mutual understanding about the project.
- The mutual understanding among team members is playing an important role.
- Team members understand each other when talking about the content of the project.
- Team members solve problems similarly.
- It is hard for the product owner to transform business requirements into technical tasks of the team.
- My opinions are clear to the other team members.
- The opinions of the other team members are clear.
- My thoughts are clear to my team members.
- The other team members thoughts are clear to me.
- The other team members understand what I mean.
- I understand what the other team members mean.

**Shared context**

Scale Label: Very Infrequent to Very Frequent.

How frequently do you experience the following issues in attempting to coordinate work on a development team?

- Incompatibility between different team members' tools and/or work processes.
- Team members having different priorities.
- Differences in the information held by team members.
- Incomplete or inaccurate information about what other team members are doing.

**Trust**

- Most people in our development group are basically honest and can be trusted.
- Team members in my development team are always interested only in their own welfare.
- Members in our development group at Alpha are always trustworthy.
- In our development group at Alpha, one has to be alert or someone is likely to take advantage of you.
- Team members in our development group at Alpha are willing to help if you need it.

**Frequent delivery of working software**

- The results of an iteration at projects at Alpha is working, runnable software.
- The to-developed software is divided into single features so that working software can be delivered in up to four weeks.
- The length of an iteration changes in the middle of an iteration.
- We deliver working software early during a project.

**Self-organizing teams**

- Important project decisions are decided by the team in joint discussions.
- The team allocates upcoming tasks itself.
- The project leader allocates the tasks on the team.
- Our development team is self-organizing.

**Embrace changing requirements**

- At Alpha, we consider changing requirements even late during development.
- The requirements are fixed at the beginning of the project and then never changed.
- Changing requirements are considered within an iteration as well.
- We welcome changing requirements, even late during development.
- Requirements fluctuated quite a bit in early phases of this project.
- Requirements fluctuated quite a bit in later phases of this project.
- Requirements identified at the beginning of the project were quite different from those toward the end.

**Customer satisfaction**

- In our projects, we give very high priority to achieving customer satisfaction.
- The users are satisfied with the developed applications at Alpha.

Overall, how do you feel customers judge about the software that you develop?

- Scale Label: Extremely Displeased to Extremely Pleased.
- Scale Label: Extremely Frustrated to Extremely Contented.
- Scale Label: Extremely Terrible...Extremely Delighted.
- Scale Label: Extremely Dissatisfied...Extremely Satisfied.

**On-time completion**

- We usually complete projects late according to the original schedule.
- Our software is often launched within or under the original schedule.

**On-budget completion**

- We usually complete projects over budget according to the original budget.
- Our software is often launched within or under the original budget.

**Software functionality**

- The software delivered by our development teams usually achieves its functional goals.
- The software delivered by our development teams meets end-user requirements.
- The capabilities of our software fit end-user needs.
- The software developed at Alpha/ meets technical requirements.

**Product performance**

- The applications developed at Alpha/ are reliable.
- The applications developed at Alpha/ are easy to maintain.
- The applications developed at Alpha/ meet user expectations with respect to response time.
- The software developed at Alpha/ meets user expectations with respect to ease of use.
- The overall quality of the developed applications at Alpha/ is high.

**Process performance**

- The system was completed within budget.
- The system was completed within schedule.

**Experience**

- Work experience Software Development (in years) Scale Label: < 1 Year; 1-2 Years; 2-5 Years; 5-10 Years; > 10 Years
- Work experience in actual position (in years) Scale Label: < 1 Year; 1-2 Years; 2-5 Years; 5-10 Years; > 10 Years
- I have become knowledgable about Agile software development practices for: Scale Label: I have not experienced; < 1 Year; 1-2 Years; 2-5 Years; > 5 Years
- I have been developing software using Agile software development practices for: Scale Label: < 1 Year; 1-2 Years; 2-5 Years; 5-10 Years; > 10 Years
- How do you rate your skill level concerning Agile software development? Scale Label: Beginner, Moderate, Experienced
- What Agile methods do you know of? Scale Label: Scrum, Exterme Programming (XP), Crystal, Lean, DSDM, Kanban, Feature-driven Development, Agile Unified Process, Other (free text)

**Position**

- What is your current position? Scale Label: Employee; Manager; Assistant

**Project role**

- What is your current role in projects? Scale Label: Developer; Architect; Business Analyst/Product Owner; IT Manager; Operation Support; Project Leader; QA/Test; Other (free text)

## APPENDIX C: OVERVIEW OF INTERVIEWEES

Table C-1 provides information about the interviewees and their background information, including selected project characteristics. The nationality of all interviewees was German.

**Table C-1: Overview of Interviewees**

| # | Length | Role | Gender | Age Group | Project Domain | Experience in ISD | Team Size | Software Type |
|---|---|---|---|---|---|---|---|---|
| A13 | 54 Min. | Senior Manager | M | 30-40 | Finance | 11 years | Medium (~15) | Standard |
| A2 | 46 Min. | Project Leader | W | 40-50 | Finance | >10 years | Medium (~15) | Standard |
| A3 | 45 Min. | Project Leader | M | 30-40 | Finance | 8 years | Medium (~15) | Standard |
| A4 | 45 Min. | Developer | M | 30-40 | Finance | 5 years | Medium (~15) | Standard |
| A5 | 27 Min. | Team Leader | W | 30-40 | Finance | 6 years | Medium (~15) | Standard |
| A64 | 88 Min. | Senior Manager | M | 30-40 | Finance | 11 years | Medium (~15) | Standard |
| A7 | 83 Min. | Product Owner | M | 30-40 | Finance | 6 years | Medium (~15) | Standard |
| A8 | 50 Min. | Senior Architect | M | 30-40 | Finance | >10 years | Small (<10) | Standard |
| O1 | 50 Min. | Developer | M | 20-30 | E-Commerce | 1 year | Small (<10) | Individual |
| O2 | 56 Min. | Developer | M | 30-40 | E-Commerce | 3 years | Small (<10) | Individual |
| O3 | 47 Min. | Developer | M | 30-40 | E-Commerce | 4 years | Small (<10) | Individual |
| O4 | 56 Min. | Developer | M | 20-30 | E-Commerce | 2 years | Small (<10) | Individual |
| O5 | 63 Min. | Developer | M | 20-30 | E-Commerce | 2 years | Small (<10) | Individual |
| O6 | 63 Min. | Developer & Coach & Manager | M | 40-50 | E-Commerce | 6-7 years | Small (<10) | Individual |

[3] All interviewee names are anonymized for reasons of confidentiality.
[4] Interviews A1 and A6 were conducted with the same person at different points in time.

## APPENDIX D: EXAMPLES FOR THE DERIVATION OF CONCLUSIONS

Table D-1 shows how codes and conclusions emerged on the basis of selected quotes. We have to point out that it is not our intention to claim that the derived conclusions are clear-cut causal relationships, but our qualitative data provides first indications of possible effects.

**Table D-1: Comparison of Quotes, Codes, and Consequences**

| Quote | Code | Conclusion |
|---|---|---|
| *"The communication increased not only for the time of the daily meetings, but also for the rest of the day."* | Daily stand-up meetings, Indirect communication, Daily stand-up meetings → Indirect communication | Daily stand-up meetings seem to have a positive effect on indirect communication. |
| *"The responses to questions in those meetings are available to all participants. In consequence, a common picture is created."* | | |
| *"I think the Dailys support the communication strongly."* | | |
| *"In the review meetings, the employees of the projects are also taking part, and you talk about the most recent information about the project."* | Sprint review, Indirect communication, Sprint review → Indirect communication | Sprint reviews seem to have a positive effect on indirect communication. |
| *"At the end, there is the review which all parties use for communicating."* | | |
| *"Retrospectives are meaningful because you can talk about communication problems,…"* *Retrospectives are meaningful because you can talk about communication problems ,…"* | Sprint retrospective, Indirect communication, Sprint retrospective → Indirect communication | Sprint retrospectives seem to have a positive effect on indirect communication. |
| *"Pair Programming is essential, because, well, you can't get away from one another, you have to talk."* | Pair programming, Indirect communication, Pair programming → Indirect communication | Pair Programming seems to have a positive effect on indirect communication. |
| *"In general, I think that that you should prefer to work together at one site, at best in one room, so there are simply quick interactions, and a good communication can take place."* | Co-located office space, Indirect communication, Co-located office space → Indirect communication | The co-located office space practice seems to have a positive effect on indirect communication. |
| *"Another advantage when sitting in the same room is that you know where the other persons are at. This means that if there is a question being asked, you can overhear it. So you can directly respond to that and help each other out."* | | |

## ABOUT THE AUTHORS

**Markus Hummel** is a doctoral candidate at the Department of Information Systems Engineering, Goethe-University Frankfurt, Germany. He received a Master of Science degree from the University of Hohenheim. His research focuses on Agile information systems development, organizational communication, and IT project management.

**Christoph Rosenkranz** is Full Professor of Information Systems at the University of Cologne. Previously, he worked as an assistant professor at Goethe University Frankfurt. He holds Diploma and PhD degrees from the University of Münster and Goethe University, respectively. His research focuses on integrated information systems and business process management, information systems development and IT project management, and online communities. He has published articles in such outlets as *ACM Transactions on Management Information Systems, Business & Information Systems Engineering*, *Information Systems Journal*, *Journal of Database Management*, *Journal of Information Technology*, *Journal of the Association for Information Systems*, and *Supply Chain Management*.

**Roland Holten** is Full Professor of Information Systems at the Faculty of Economics and Business Administration, Goethe University. He habilitated in Information Systems at the Faculty of Economics, University of Muenster, Germany. Roland holds two Master degrees in Business Administration (Dipl.-Kfm., TU Dortmund University, Germany) and Computer Science (Dipl.-Inform., RWTH Aachen University, Germany), and a PhD in Information Systems (Dr. rer. pol., University of Muenster, Germany). His research interests include IT-mediated communication structures of groups, information systems development, and conceptual modelling. His work has been published in peer-reviewed journals such as *Journal of Information Technology*, *Information Systems Journal*, *Information Systems*, *Journal of the Association for Information Systems*, *Information Systems and e-Business Management*



Communications of the Association for Information Systems

(ISeB), and *Business & Information Systems Engineering*. Roland is a member of the editorial board of the journal ISeB.

Communications of the Association for Information Systems

ISSN: 1529-3181





Communications of the Association for Information Systems