The Journal of Systems and Software 114 (2016) 101–124

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

The daily stand-up meeting: A grounded theory study

Viktoria Stray[a,*], Dag I.K. Sjøberg[a,b], Tore Dybå[a,b]

[a] University of Oslo, Norway
[b] SINTEF, Norway

ARTICLE INFO

Article history:
Received 17 July 2014
Revised 17 November 2015
Accepted 4 January 2016
Available online 11 January 2016

Keywords:
Daily meeting
Daily Scrum meeting
Agile software development

ABSTRACT

The daily stand-up meeting is one of the most used agile practices but has rarely been the subject of empirical research. The present study aims to identify how daily stand-up meetings are conducted and what the attitudes towards them are. A grounded theory study with 12 software teams in three companies in Malaysia, Norway, Poland and the United Kingdom was conducted. We interviewed 60 people, observed 79 daily stand-up meetings and collected supplementary data. The factors that contributed the most to a positive attitude towards the daily stand-up meeting were information sharing with the team and the opportunity to discuss and solve problems. The factors that contributed the most to a negative attitude were status reporting to the manager and that the frequency of the meeting was perceived to be too high and the duration too long. Based on our results, we developed a grounded theory of daily stand-up meetings and proposed empirically based recommendations and guidelines on how to organize them. Organizations should be aware of the factors that may affect the attitude towards daily stand-up meetings and should consider our recommendations and guidelines to make this agile practice as valuable as possible.

1. Introduction

Common to all agile methods is an emphasis on communication and the human side of software development (Merisalo-Rantanen et al., 2005). Conducting a *daily stand-up meeting* (DSM) is an important practice in the agile methods Scrum and Extreme Programming (XP) to improve communication in software projects. The DSM is often conducted as a 15-min morning meeting to share information that is supposed to be relevant to the teams' progress. The term DSM used in this article originates from XP. Other names of the practice are *frequent, short meetings* (Rising, 2002), *morning roll call* (Anderson, 2003), *daily huddle meeting* (Paez et al., 2005), *daily meeting* (Pikkarainen et al., 2008), and *daily Scrum meeting* (Sutherland and Schwaber, 2013a).

The software development industry has extensively adopted agile practices, many of which have been thoroughly investigated (Dingsøyr et al., 2012). However, little research has been conducted on the DSM, which may be surprising given that the DSM is the most used agile practice according to a 2014 survey (VersionOne, 2015). In that survey, the DSM was used by 85% of the organizations that employed agile development. The global cost of conducting the DSM is immense if one supposes that the majority of the software development teams in the world daily interrupt their development tasks to spend 15 min on the DSM.

In this article, we report a study on how DSMs are conducted and what affects the attitude towards them. We propose a theory of DSMs that includes propositions among DSM constructs, with explanations grounded in data. The data was generated from 79 observations of DSMs of eight software teams in three companies and 60 interviews with team members, Scrum Masters and product owners that worked in these teams and an additional set of four teams.

A few studies have investigated the DSM as one of several agile practices. Pikkarainen et al. (2008) studied the impact of agile practices on communication and found that DSMs kept developers, project leaders and customers aware of the project status and helped the developers resolve design issues faster. Paasivaara et al. (2008) examined agile practices in global software development and found that DSMs helped reveal problems early and improved transparency between sites. Moe et al. (2010) studied the nature of self-managing agile teams and found that DSMs were mostly used by a Scrum Master to obtain an overview of the progress and ongoing project activities. McHugh et al. (2012) examined how agile practices impact trust and found that DSMs helped the teams function more cohesively. Dorairaj et al. (2012) studied dynamics in distributed teams and found that the practice promoted team interaction and the building of a "one team" mindset. Yu and Petter (2014) argue that the DSM may contribute to build

* Corresponding author. Tel.: +47 22840107.
E-mail addresses: stray@ifi.uio.no (V. Stray), dagsj@ifi.uio.no (D.I.K. Sjøberg), tore.dyba@sintef.no (T. Dybå).

http://dx.doi.org/10.1016/j.jss.2016.01.004
0164-1212/

shared mental models within a team. The results of an experiment conducted by Hasnain et al. (2013) suggest that introducing DSMs may be a powerful way to improve trust in agile software teams.

The DSM was the primary study topic in some of our earlier research. In a longitudinal study, DSMs led to a greater commitment to a failing course of action (Stray et al., 2012b). In another study, we investigated the proportion of time spent on different topics. The largest topic category was discussing problems and possible solutions (Stray et al., 2012a). In yet another study, we identified thirteen obstacles to efficient DSMs and suggested ways to overcome them (Stray et al., 2013).

Much can be learned from case studies by doing a secondary grounded theory analysis (Glaser, 2001, p. 97). This study builds on our previous research. Among the 60 interviews of this study, 7 were reused from the study reported in (Stray et al., 2011), 17 were reused from the study reported in (Stray et al., 2012b) and 9 were reused from the study reported in (Stray et al., 2013). The remaining 27 were new interviews for this study. We reanalyzed the case study material and iteratively compared it with newly collected material. This study also contributes to increasing the understanding of the costs and benefits of DSMs, which is important for improving agile software development. Finally, our work answers a call for more empirically based theories in software engineering (Herbsleb and Mockus, 2003; Hannay et al., 2007; Sjøberg et al., 2007).

The remainder of this paper is organized as follows. Section 2 outlines relevant background literature. Section 3 describes the research methods used. Section 4 reports our results. Section 5 discusses the results, limitations of the study and future work. Section 6 concludes.

2. Background

This section gives a brief introduction to the field of meetings in general, the DSM in agile development and daily meetings in other disciplines.

2.1. Meetings

Meetings are necessary for teamwork to be successful (Kauffeld and Lehmann-Willenbrock, 2012). They provide a venue for information exchange, decision making, coordination, planning and monitoring progress, each of which is an essential component of the team processes associated with team performance (O'Neill and Allen, 2012). According to Boden (1994, p. 84), a meeting is "a planned gathering, whether internal or external to an organization, in which the participants have some perceived (if not guaranteed) role, have some forewarning (either longstanding or quite improvisatorial) of the event, which has itself some purpose or 'reason,' a time, [a] place, and, in some general sense, an organizational function."

Employees spend a lot of time in meetings, and the amount seems to increase (Rogelberg et al., 2006). A great portion of meeting time is perceived as ineffective, and over one third of the time is wasted, with annual losses up to USD 37 billion in the United States alone (Elsayed-Elkhouly et al., 1997). Furthermore, meeting demands also affect employee productivity beyond the meeting setting (Allen et al., 2012). For example, a meeting is a particular kind of interruption (Rogelberg et al., 2006), which may affect employees' subsequent readiness to perform by influencing their psychological state (Zijlstra et al., 1999). After an interruption, people have to scan and evaluate all new information that they have encountered; several short interruptions have a greater effect than one long interruption (Zijlstra et al., 1999). Parnin and Rugaber (2011) analyzed 10,000 programming sessions and found that in 57% of the sessions, the developers needed 15 min or more to collect their thoughts and make the first edit after an interruption, such as a meeting.

Very few empirical research studies have specifically focused on team meetings; most studies use meetings only as a context for studying other variables of interest (Scott et al., 2012), although there are exceptions: Anderson et al. (2007) explored the nature of communication in virtual team meetings. They found that the communication was influenced by the way in which the technologies were used. For example, the person controlling the keyboard dominated cross-site communication even though the audio facility made contributions from any team member perfectly audible at either site. Sonnentag and Volmer (2009) studied how individuals in software design teams contributed to teamwork processes during team meetings. They found expertise to be a strong predictor of individuals' contributions. Team members with a high level of expertise were more involved in problem analysis and goal specification than those with less expertise. Kauffeld and Lehmann-Willenbrock (2012) analyzed videotaped team meetings and linked their observations with objective data on team productivity and organizational success. Their findings show that team meeting interaction processes affect meeting satisfaction, team productivity and organizational outcomes.

2.2. DSM in agile software development

The DSM in agile software development is supposed to be a brief gathering of team members that is planned and has a prearranged time and place, and a purpose and thus satisfies the definition of a meeting given in the previous section. Based on Boden's (1994, p. 84) division of formal and informal meetings, DSMs may be characterized as informal because they are task and decision oriented, have casual conversation styles and are generally unrecorded, and members are gathered for a narrow organizational goal. Often, the purpose of the DSM is that every team members should share their response to a set of questions. A survey (VersionOne, 2009) reported that 69% of agile practitioners adhered to the three questions:

1. What have you done since we last met?
2. What are you planning to do until we meet again?
3. What, if any, impediments are you encountering that are preventing you from making forward progress?

In software engineering, conducting DSMs in development teams became popular with the introduction of agile methods, in particular Scrum, where it is a mandatory practice. Scrum describes the DSM as a 15-min time-boxed event for the team to synchronize activities and create a plan for the next 24 h (Sutherland, 2013a). Schwaber and Beedle (2002, p. 40) claim that "the Daily Scrum meeting gets people used to team-based, rapid, intense, co-operative, courteous development. Daily Scrums improve communication, eliminate other meetings, identify and remove impediments to development, highlight and promote quick decision-making, and improve everyone's level of project knowledge."

The DSM is not a mandatory practice in Kanban. Nevertheless, many teams that practice Kanban use DSMs; for example, the Kanban teams reported in Sjøberg et al. (2012) and the Kanban teams in our study. According to Kniberg and Skarin (2010), Kanban teams tend to use a more board-oriented format in which they focus on bottlenecks on the Kanban board instead of a format in which every person reports one by one.

VersionOne (2015) conducts an annual survey where agile practitioners state which agile practices they employ. Fig. 1 shows that there is an increase in the use of the DSM and a decrease in the use of test-driven development and pair programming from 2007 to 2014. Still, the two most thoroughly investigated agile

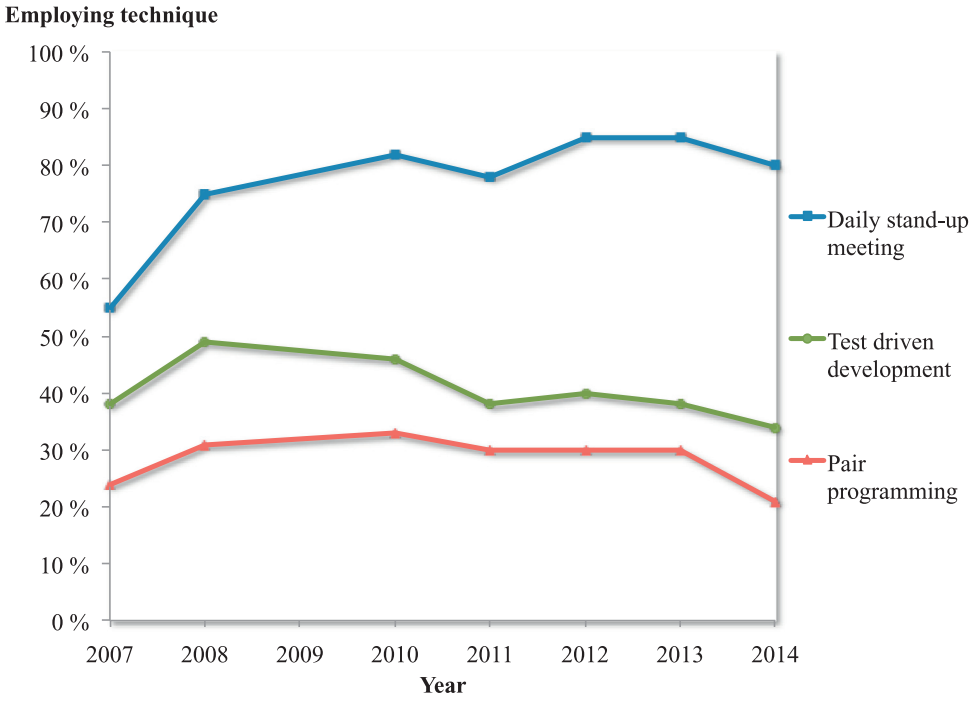


Fig. 1. The percentage of agile practitioners employing the techniques DSM, test driven development and pair programming (VersionOne, 2007, 2008, 2009, 2010, 2011, 2013, 2014, 2015).

practices are test-driven development and pair programming (Dingsøyr et al., 2012), while the DSM has been largely overlooked.

2.3. Daily meetings in other disciplines

The majority of teaching hospitals conduct morning meetings, often called morning reports. Despite widespread use, the meetings have rarely been examined in research, and studies are lacking to document their effectiveness (Amin et al., 2000). The morning meetings usually last for an hour, and the primary purpose is education (Amin et al., 2000). Thus, these meetings are very different from the DSMs discussed in this article. However, some hospitals have introduced so-called surgical morning meetings, which are more relevant because they aim to improve the communication among the staff, and each meeting involves a discussion of the patients (Aston et al., 2005).

Some types of organizations have meetings at the beginning of every shift; for example, hospitals, police stations and restaurants. In these meetings, there is an explicit hand off of work from one shift to the next, which is quite different from the DSM. These meetings would correspond to meetings in software projects in which teams in one location continue work from another location, also called follow-the-sun development (Carmel and Agarwal, 2001).

3. Research method

The motivation for our research was to increase the understanding of which factors contribute to effective teamwork because teams are the fundamental organizational unit through which agile software projects are executed. We chose grounded theory as our research method because it is considered suitable for pursuing a general understanding of a phenomenon; that is, when a researcher asks "What is going on here?" (Glaser, 1978).

3.1. Research sites

Understanding the context is important for the interpretation of the results of any empirical study (Dybå et al., 2012). Here we describe the three companies that participated in this field study. We had access to these companies through an industry-managed research project on teamwork in agile software teams. We investigated a Norwegian consulting company, an International telecommunications company and an International software company, which we denote, respectively, Alpha, Beta and Gamma. Table 1 gives an overview of the investigated teams. In the time period of the interviews conducted in each of the teams, the teams were static (one person we interviewed had handed in his resignation).

Alpha is a company with 350 employees. We studied two distributed, closely collaborating Kanban teams who worked on the same project. One team was located in Norway (Alpha 1) and one in Poland (Alpha 2). The team members located in Poland were employees of the project's client, a Fortune 500 industrial company. The project members worked on maintenance and extensions of a content management system.

Beta has 700 employees in 20 countries. We studied two Scrum teams in Norway and five Scrum teams in Malaysia. The teams were working on different projects, which had iterations lasting from two to four weeks.

Gamma has 150 employees in four organizational units. We observed one team in Norway and two in the United Kingdom. The goal of the Scrum project was to develop an engineering software product for the oil and gas industry and the length of the iterations was usually three weeks.

3.2. Grounded theory

Grounded theory is defined as "a general methodology of analysis linked with data collection that uses a systematically applied set of methods to generate an inductive theory about a

Table 1
The investigated teams.

Company	Team	Location	Team members	Distribution	Interviewed	Observed
Alpha	1	Norway	10	Co-located	√	√
	2	Poland	9	Co-located	√	√
Beta[a]	1	Norway	9.5	Distributed	√	√
	2	Norway	8.5	Co-located	√	√
	3	Malaysia	10	Co-located	√	√
	4	Malaysia	3	Co-located	√	
	5	Malaysia	8	Co-located	√	
	6	Malaysia	6	Co-located	√	
	7	Malaysia	9	Co-located	√	
Gamma	1	Norway	10	Co-located	√	√
	2	UK	9	Distributed	√	√
	3	UK	8	Distributed	√	√

[a] Teams 1 and 2 in Beta shared one team member.

Table 2
Data collection.

Technique	Number	Description
Interviews	8 in Phase 1 52 in Phase 2	We conducted semi-structured interviews with open-ended questions
Observations of DSMs	9 in Phase 1 70 in Phase 2	We made notes from all DSMs we observed. Thirteen of the observations were recorded and transcribed word by word
Questionnaires	19 in Phase 2	Project members in Alpha answered a questionnaire anonymously

substantive area" (Glaser, 1992, p. 16). This methodology aims to develop a theory rather than extend or verify existing theories.

Grounded theory was introduced in 1967 with the publication of Glaser and Strauss's (1967) book, *The Discovery of Grounded Theory*. The method later evolved into two versions with separate terminology and processes as Glaser and Strauss developed different perspectives (Goulding, 1998). These two versions are known as the Glaserian and the Straussian methods of grounded theory, respectively. Among other factors, they differ with respect to the formulation of research problems, data analysis and coding techniques. According to Strauss and Corbin (1990, p. 34), a researcher should begin a grounded theory study by defining a research problem. In contrast, Glaser advises to start investigate an area of interest without establishing such a definition (Glaser, 2001, p. 21). Furthermore, Glaser describes selective coding as occurring early in the analysis, when the core category has been identified, while Strauss and Corbin describe selective coding as occurring towards the end of the analysis, with the purpose of selecting the core category (Glaser 1992, p. 75). Glaser argues that the theory should only explain the phenomenon under study, while Strauss and Corbin describe coding matrices as explaining the phenomenon beyond the immediate field of study (Goulding, 1998). Glaser (1992, p. 62) criticizes the use of these coding matrices, claiming that they lead to theories based on preconceptions because data is "forced" into categories.

Because Glaserian and Straussian grounded theories have substantial differences, it is important that researchers explicitly state which method they use. We decided to follow Glaser's method since we wanted to approach the field with no predefined research questions but rather a general interest in the topic; we wanted to let the concepts and categories emerge from the data in a more flexible way than what is prescribed in the Straussian method.

3.3. Data collection techniques

Throughout the study, data collection and analysis occurred within an iterative process. Table 2 shows the data techniques we used. Fig. 2 depicts our research approach.

3.3.1. Interviews

This study involved 60 interviews across 12 teams profiled in Table 3. Eight of the participants in Alpha 1 were interviewed in Phase 1; all other interviews were conducted in Phase 2. The teams ranged in size from three to eleven persons. Nine different roles were interviewed in total. If interviewees had more than one role, the table shows their main role in the team.

The participants gave their consent for the interviews to be recorded and agreed to the publication of the results subject to anonymity. The interviews varied between 25 and 96 min. In most of the interviews, at least two researchers participated. One asked questions, and one or more others took notes and asked additional questions at the end.

The interview guide used in our semi-structured interviews consisted of four parts. In the first part, we introduced ourselves and assured the interviewee of confidentiality. The topic of investigation presented to the interviewees was "teamwork in agile projects". The second part comprised "warm-up" questions regarding the interviewee's background, experience and current activities. The third part involved the main interview. This part was modified as the research progressed in Phase 2 to comply with the theoretical saturation of the core category (DSM), which is common when using grounded theory. The fourth part included closing questions and provided an opportunity for the interviewee to ask questions and make additional comments. Table A.1 in the Appendix shows how the interview guide looked in the beginning of the data collection in Phase 2. All the interviews were transcribed word by word, mostly by the first author, partly by an MSc student, whose transcripts were validated by the first author.

3.3.2. Participant observation

Our participant observation was guided by a protocol based on Spradley (1980) that contained questions to be answered by the researcher; see Table A.2. The protocol was initially proposed by the first author and reviewed by the two other authors. Information recorded while observing meetings included names and roles of the attendees, start and end time, type of discussions, leadership and facilitation, format (who was sitting or standing), communication channel (phone or video), the number of participants present

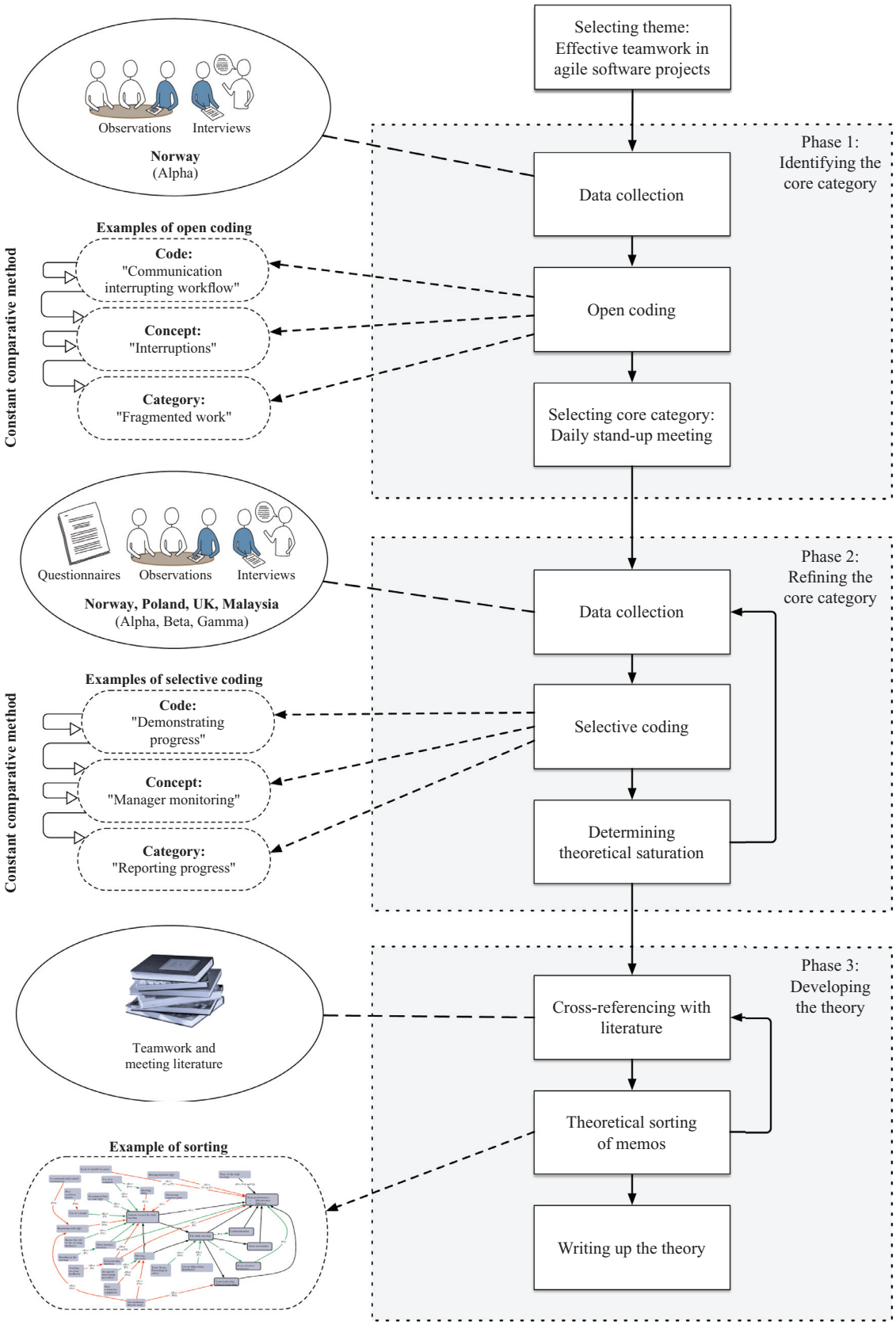


Fig. 2. Our grounded theory research process.

Table 3
Interviewees.

Company	Team	Interviewees (*N*)	Roles of interviewees[a] (*N*)	Age, mean (median)	Years in company, mean (median)
Alpha	1	9	A (3), D (4), PM (1), T (1)	34.7 (35.0)	6.2 (5.5)
	2	6	A (1), D (2), PM (1), T (1), TL (1)	31.5 (31.0)	3.1 (3.3)
Beta	1	4.5	A (1), D (1), SM (1), T (1), TW (0.5)	44.9 (45.0)	10.7 (10.0)
	2	5.5	D (3), SM (2), TW (0.5)	41.5 (40.5)	10.9 (10.5)
	3	9	D (5), PM (1), PO (1), SM (1), T (1)	32.9 (32.0)	2.3 (2.5)
	4	3	D (1), SM (1), TL (1)	32.3 (31.0)	2.3 (1.0)
	5	4	D (1), PM (1), SM (1), T (1)	32.3 (30.5)	5.6 (2.8)
	6	3	A (1), D (1), TL (1)	32.7 (32.0)	1.0 (1.0)
	7	4	D (2), TL (1), PO (1)	35.8 (37.5)	4.4 (3.0)
Gamma	1	9	A (2), D (6), PM (1)	42.0 (38.0)	10.5 (10.0)
	2	2	D (1), PO (1)	44.0 (44.0)	14,5 (14,5)
	3	1	D (1)	59.0 (59.0)	14.0 (14.0)
			Sum: A (8), D (28), PM (5), PO (3), SM (6), T (5), TL (4), TW (1)	Avg: 37.0 (35.0)	Avg: 6.5 (4.5)

[a] A = Architects; D = Developers; PM = Project Managers; PO = Product Owners; SM = Scrum Masters; T = Testers; TL = Team Leaders; TW = Technical Writers.

Table 4
Questionnaire.

Daily stand-up meetings
• I am satisfied with the outcomes of our stand-up meetings
• I look forward to our stand-up meetings
• I feel energized and ready to get down to work after a stand-up meeting
• I think our stand-up meetings improve our development process
• I feel that our stand-up meetings contribute to better teamwork
• I feel that it is worth spending time on stand-up meetings given the results we get from them
• List three positive things about the stand-up meetings
• List three negative things about the stand-up meetings

in each location, language and atmosphere. The observer wrote notes either during the meeting or immediately after. On many occasions, two observers were present to ensure the reliability of the information captured. We sometimes took pictures to document the meeting setting. When we started to observe meetings in Phase 2 in Company Beta, we decided to audiotape the meetings. However, for capacity reasons, we only managed to audiotape and transcribe the first thirteen meetings. Note that transcribing a meeting is more time consuming than transcribing an interview, particularly because keeping track of who is speaking is hard.

3.3.3. Questionnaire

A questionnaire with statements about DSMs (Table 4) was distributed to Alpha 1 and Alpha 2. The respondents scored the statements on a 5-point Likert scale from strongly disagree to strongly agree, except for the last two questions, which had three blank spaces each. Similarly to the development of the interview guide and observation protocol, the questionnaire was proposed by the first author and reviewed by the two other authors.

3.4. Data analysis techniques

3.4.1. Coding

We followed Glaser's two sequential stages of substantive coding: *open* and *selective*. Open coding sets the direction of the research by identifying a core category and serves as the initial step of the theoretical analysis in grounded theory (Glaser, 1992, p. 39). Selective coding continues the process and is limited only to categories related to the core category.

Glaser (2011, p. 3) uses the terms *code, concept* and *category* synonymously and explains that they all "refer to conceptualizing an emergent pattern." For the sake of clarity, we refer to three different levels of data abstraction: *Codes* are assigned to statements at the first level of abstraction; groups of codes are *concepts* at the second level; and groups of concepts are *categories* at the third level; see Fig. 3.

Theoretical coding involves detecting relationships between codes, concepts and categories and may occur throughout the whole study (Hernandez, 2009). Glaser (1978, 1998, 2005) has identified 50 different theoretical coding families to assist researchers in conceptualizing how categories and their properties may relate to each other. The use of these coding families increases completeness and relevance for the grounded theory (Glaser 2005, p. 70).

3.4.2. Constant comparative method

The constant comparative method is a key part of grounded theory and involves comparing codes and concepts to produce a higher level of abstraction (Glaser, 1992, p. 39). More specifically, codes are compared with other codes to produce concepts, codes are also compared with concepts to produce new concepts, and concepts are compared with other concepts to produce categories.

3.4.3. Memoing

Memos are written notes to record reflections on the data and codes and their relationships as they occur to the analyst while coding and writing (Glaser, 1978, p. 83). Our memos usually consisted of a few statements or questions. An example of a memo written during the analysis had the title "Many participants" and contained the question "Does the number of participants affect the duration?" Fig. 4 gives an overview of the memos we sorted during the last phase of the coding. The approximately 100 memos were written and managed by the first author throughout the study using the tool MacJournal.[1]

3.4.4. Quantitative measurement of meeting attitude

To obtain a quantitative measure of the interviewees' attitude about meetings, the first two authors studied all interview transcripts and independently scored the attitude towards DSMs on a

[1] MacJournal is a registered trademark of Mariner Software, www.marinersoftware.com.

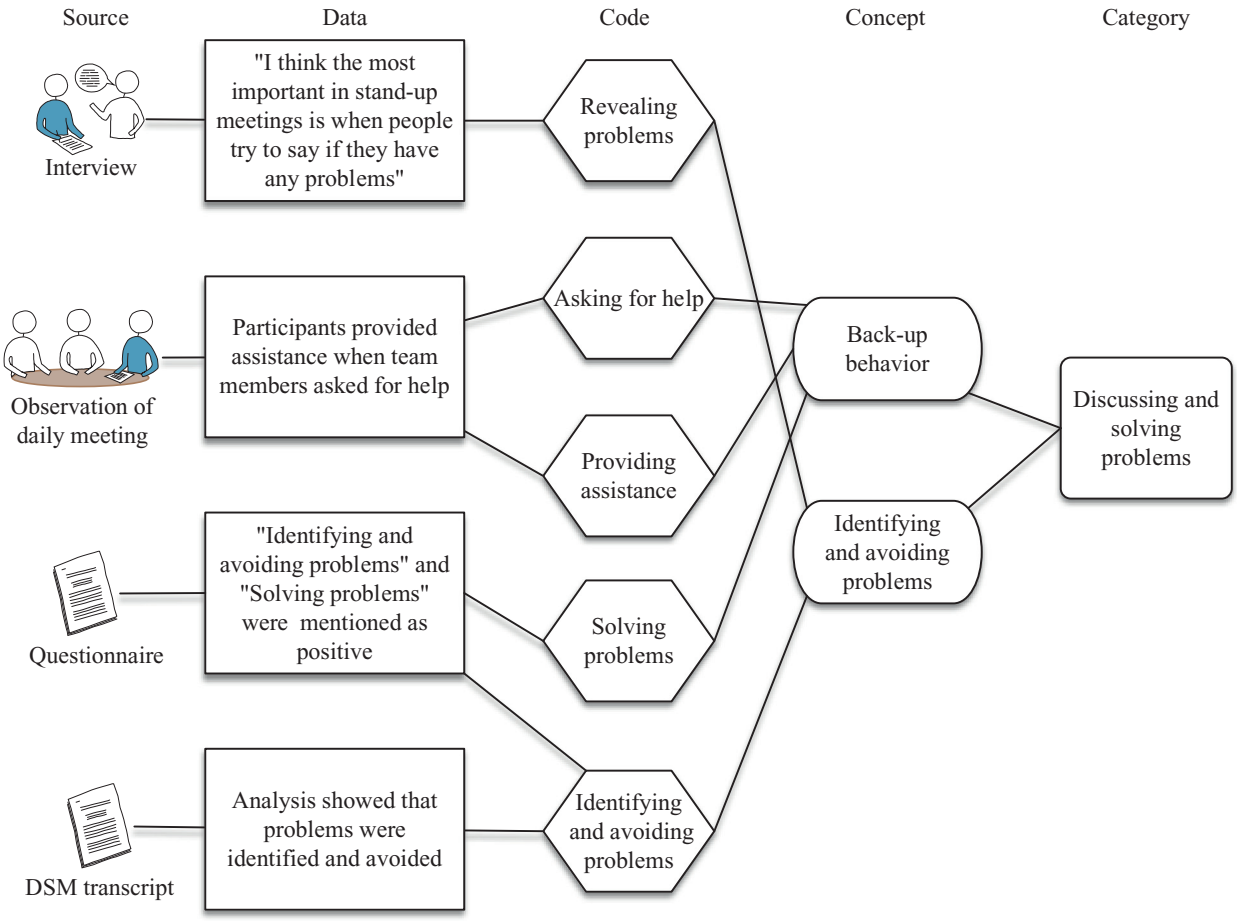


Fig. 3. Three levels of data abstraction.

Likert scale from 1 to 5 (strongly negative to strongly positive). The authors had substantial agreement (weighted kappa = 0.72) (Landis and Koch, 1977).

3.5. Phase 1: identifying the core category

3.5.1. Data collection

The theoretical sampling process in classic grounded theory begins with initial data collection and analysis before any core category has been identified (Glaser, 1978). By adhering to the Glaserian method, we established our general area of interest as "effective teamwork in agile software development projects." As the study progressed, our particular interest (the core category) was identified as "DSMs in agile software development projects." We chose to conduct the first interviews in Alpha 1 since they had been practicing agile methods for almost a decade. The interviews in Phase 1 lasted between 25 and 61 min (average 39 min) and was conducted and transcribed by the first author.

3.5.2. Open coding

With systematic reading and coding of all interview transcripts, initial categories emerged. We started the coding as soon as the first interviews were transcribed using the qualitative research tool NVivo.[2] Adhering to the prescribed method of open coding (Glaser, 1992, p. 48), we had no preconceived codes at this point. Each transcript was coded in its entirety in detail, because at this early stage in the process, we could not know which data would be relevant. We used the constant comparative method, as illustrated on the left side of Fig. 2. We first compared codes within the same interview. We then compared codes from one interview with codes from other interviews, data based on observations, and our memos. Our open coding process generated 46 codes. Table 5 shows examples of codes that were assigned to statements in the open coding process.

3.5.3. Selecting the core category

Open coding comes to an end when a core category is selected (Glaser, 1992, p. 39). Glaser (2001, p. 200) emphasizes that researchers should tolerate confusion in the open coding process, because discovering a core category may be challenging.

If more than one core category is found, a researcher must select one of them, since grounded theory centers on one core category. Glaser (2001, p. 201) states that selecting the core category may seem like a big commitment, but fear of selecting an unsuitable core category may result in the research taking too long: "It is best to test out a core category and then another if the first does not work, than to drift two years in open coding." If other interesting core categories are also identified, they can be investigated in separate studies.

Three potential core categories emerged from our open coding: *Fragmented work*, *Communication* and *The daily stand-up meeting*. We selected the last one as our core category because it related to most other categories in a meaningful way. Furthermore, all participants expressed concerns related to the practice, and they were very eager when they talked about it.

[2] NVivo is a registered trademark of QSR International, www.qsrinternational.com.

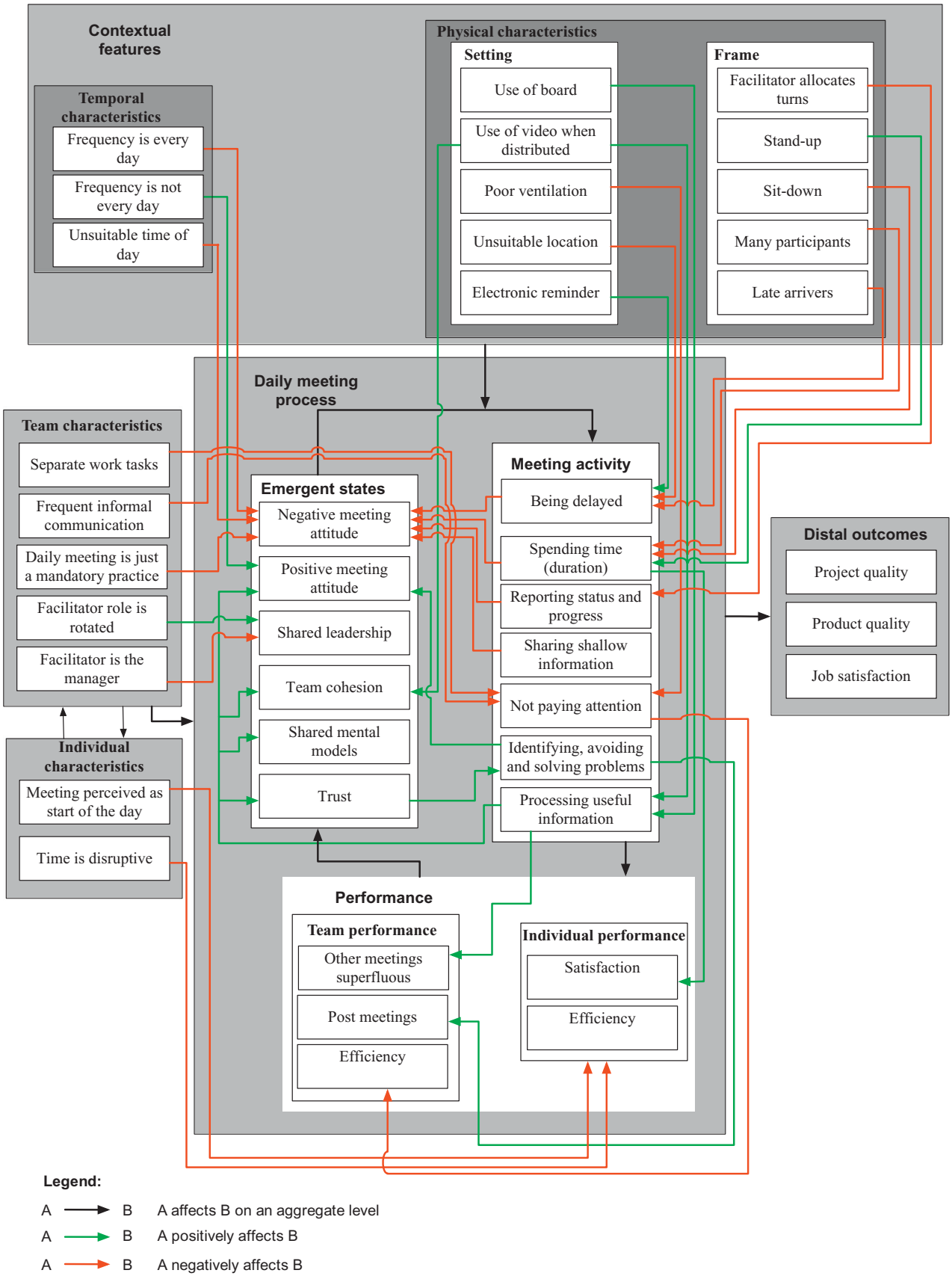


Fig. 4. Snapshot of sorted memos during theoretical sorting. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

Table 5
Examples of open coding.

Role in Alpha	Statements	Codes
Architect	"No, the teamwork was not very effective last week. I received 3-4 phone calls a day from the other site. They sort of interrupt me in the middle of what I am working on. (…) I think I have to switch between tasks a lot, and after an interruption, I have to start over what I was doing."	*Communication interrupting workflow*, *Task switching*, *Start over*
Developer	"Our biggest challenge is communication (…) I really look forward to getting the videoconference equipment so that we can have daily face-to-face meetings with the other site. Often the interruptions caused by all the questions on MSN [chat client] reduce my efficiency and make me frustrated. Another benefit of having the stand-up with the other site will be that we will be in charge of the meeting, and since we have a limit of work tasks in progress, maybe we can affect the other team positively. Now they start too many tasks at once."	*Daily face-to-face meetings*, *Frustrated by interruptions*, *Work in progress*
Project manager	"For many years, we have debated how to run the daily stand-up meeting. Some people think that since we work on different tasks, they don't need an overview of what others are doing. They find the meeting irrelevant and to last too long. So we have tried to reduce the time spent in the meeting, but then others complain that they lose an overview of what is happening in the project."	*How to run the daily stand-up meeting*, *Overview of what others are doing*

Table 6
Examples of selective coding.

Company	Role	Statements	Code	Attitude
Beta	Scrum Master	"Everyone knows that they have to report something. So they need to do something."	Demonstrating progress	Positive
Gamma	Developer	"It motivates you to work because in the meeting, you have to tell what you have done."	Demonstrating progress	Positive
Beta	Developer	"What I don't like with the daily meeting is reporting progress when you haven't had much progress from the day before; it is a short interval to report on."	Demonstrating progress	Negative
Gamma	Developer	"You can feel the pulse go up because you are supposed to talk about the things you have done and show progress."	Demonstrating progress	Negative
Gamma	Project manager	"Developer X always tells a lot of details. I think he wants to prove that he has done a lot of work."	Demonstrating progress	Negative

3.6. Phase 2: refining the core category

3.6.1. Data collection

The aim of theoretical sampling is to ensure that new data contribute to theory development (Glaser, 1992, p. 101). Selecting new interviewees and sites for observation follows from the results of the coding process; it is not based on random selection. Consequently, we did not plan at the start of the study where to collect subsequent data, because we did not know in advance where the research would lead. Once we identified the DSM as the core category, we decided to gather data from Alpha 2 to supplement the data from Alpha 1. We thought it would be valuable to investigate DSMs from the perspective of both sites in the multicultural, distributed project in Alpha. Furthermore, Alpha was about to start using video equipment in DSMs, which we thought could generate interesting insights into distributed DSMs.

When analyzing the data collected from Alpha with regards to the emerging theory of DSMs, we did not know the theoretical concepts that were specific to this company. Thus, we had to collect data from other companies. We chose Beta and Gamma, introduced in Section 3.1, because they were easily accessible and conducted DSMs regularly.

3.6.2. Selective coding

In the selective coding, we coded only the transcript passages that were pertinent to DSMs. The coding involved extensive use of the constant comparative method. After *DSM attitude* emerged as a category, all statements were also coded as either positive or negative (towards DSMs). As an example, Table 6 shows statements that were coded as *Demonstrating progress*.

Table A.3 gives an overview of the positive and negative opinions (codes) stated by the interviewees on each of the concepts. The number in the parenthesis after the opinions indicates how many statements expressed that opinion. For example, the two statements that constitute "Demonstrating progress (2)" in the second column under category "Reporting progress" in Table A.3 can be found in the first two rows of Table 6.

Table A.4 gives an overview of the positive and negative aspects of DSMs as expressed in the questionnaires. This table was constructed in the same way as Table A.3 but was based on text responses to the questionnaires rather than oral quotes from the interviews.

3.6.3. Determining theoretical saturation

The selective coding continues until the researcher has sufficiently elaborated the core category and its connections to other relevant categories (Glaser, 1978, p. 53). When our analysis showed no more new important points or aspects from new interviews, the three authors agreed on a common belief that additional data collection would not generate any new results; that is, we believed that the point of theoretical saturation was reached. We then moved to Phase 3, as illustrated in Fig. 2.

3.7. Phase 3: developing the theory

3.7.1. Cross-referencing with literature

We consulted meeting and teamwork literature for relevant theories. The theory that we found most relevant was the Adaptive Structuration Theory (AST). AST examines organizational change facilitated by different types of structures for social action

(DeSanctis and Poole, 1994). Although AST links structure primarily to the structure provided by advanced information technologies, structure can also be provided by processes, procedures and organizations (DeSanctis and Poole, 1994). Structures consist of *structural features* and a *spirit*. The *spirit* is the researchers' current interpretive account (based on multiple sources of evidence) regarding the values and goals of the practice. AST posits that how a practice is used is impacted by the *faithfulness* to which a team uses the practice in keeping with the spirit in which it is meant to be used, the team's *attitudes* toward the practice, and the team's *level of consensus*.

We also consulted other theories and frameworks, including the Theory of Activity Regulation (Zijlstra et al., 1999), Conservation of Resources Theory (Hobfoll, 2001), Collaborative Performance Framework (Bedwell et al., 2012), Group Effectiveness Model (Cohen and Bailey, 1997), Meeting Satisfaction Model (Reinig, 2003), Shared Mental Models Theory (Cannon-Bowers et al., 1993) and Team Situation Awareness (Endsley, 1995). Consulting this literature helped us perform a well-founded analysis of the data in the last phase of the grounded theory process.

3.7.2. Theoretical sorting of memos

Theoretical sorting is an essential step in the grounded theory process in which one is supposed to do most of the theoretical coding (Section 3.4.1). The theoretical sorting of memos is the key to presenting the theory to others in words and writing (Glaser, 1992, p. 109). At the beginning of the theoretical sorting of memos, we printed all memos on paper, and we looked for similarities and connections among the ideas. For each memo we asked, as Glaser (1978, p. 123) suggests, "where does it fit in?"

We did most of the theoretical coding in this stage of the analysis. (We identified a few relationships in earlier stages.) We explored the coding families *Six C's, Social arena, Dimensions, System parts* and *Model*. We found that the *Model* family was the most suitable one to explore our data. When using this method, the researcher models the "theory pictorially by either a linear model or a property space" (Glaser, 1978, p. 81). To model the memos and their relationships, we used the diagramming tool Omnigraffle Pro. [3] An example of a cycle of memo sorting using the Model coding family is depicted in Fig. 4, in which each box represents the title of a memo, and the arrows represent relationships between memos. For example, at this point in time, a memo was titled "Many participants". The red arrow from this memo to the memo "Spending time (duration)" illustrates the proposition that many participants in a meeting cause the meeting to last longer. New memos were created based on theoretical sorting and cross-referencing with the literature and included in the next cycle of sorting.

3.7.3. Writing up the theory

A grounded theory should use the fewest possible concepts to explain as much variation as possible in the phenomenon under study with the greatest possible scope (Glaser, 1978, p. 125). Based on our analysis of interviews and observational data, we identified codes, concepts and categories (three levels of abstractions). The six categories we arrived at became constructs in our proposed theory. For each category, we included propositions that state important interactions among the constructs. The DSM theory is proposed a starting point for understanding the key constructs and relationships of DSMs in agile software development projects, adhering to the principle that a grounded theory is readily modifiable (Glaser, 1978, p. 129).

[3] Omnigraffle Pro is a registered trademark of The Omni Group, www.omnigroup.com.

4. Results: an initial theory of DSM

This section describes the principal results of the study in the form of an initial theory of DSMs, shown in Fig. 5. The theory is summarized in Table 7 using the four-component structure outlined in Sjøberg et al. (2008). The *constructs* are the basic elements of the theory, the *propositions* are the interactions among the constructs, the *explanations* describe why the propositions are as specified and the *scope* of the theory describes the universe of discourse in which the theory is applicable. The scope of our theory is agile software development projects. This section describes each of the constructs and the explanations for the propositions that relate to that construct.

4.1. C1 team characteristics

We found two team characteristics that affected the DSM process: how the team took responsibility for their work (C1a, self-management) and how much knowledge they shared (C1b, knowledge redundancy).

4.1.2. Explanation 1a

A premise in agile development is that software teams are self-managed. In the present study, all teams had applied Scrum for several years, but the degree of self-management was generally low. A low level of self-management in the team made it easy for authoritative managers to use the DSM to obtain status information that was mainly useful only to themselves. One developer stated: "We have stand-up meetings only when the project manager is here. He probably feels that he needs an overall picture. So, I think the stand-up meeting means a lot to him at least."

One indicator of low self-management in several teams was the Scrum Master's control of the allocation of speaking turns in the meeting. In contrast, in a few teams, high self-management was indicated by turn-taking following a round-robin approach; that is, the person standing at one point of the semicircle started, and the other participants then continued in a counter-clockwise sequence. Another indication of high self-management in these teams was the rotation of the Scrum Master role.

The leadership style of the Scrum Master sometimes made it harder for the team to be completely self-managed. For example, a Scrum Master, who was also a project manager, was aware that he affected the DSMs: "I don't think they would conduct the daily meetings in the same way if I had not been present. They need someone who has the overview and asks the unpleasant questions. I don't believe there is such a thing as a completely self-organized team. There must be someone who is firm and gets things done." Interestingly, we observed that when this project manager was relocated and a developer took on the Scrum Master role, the level of self-management increased and the average meeting duration decreased from 27 min (median 28) to 19 min (median 17). We observed that the meetings became less formal and that team members shared more information among each other instead of reporting to the Scrum Master. Our observations indicate that the reduction in meeting time was mainly due to less reporting of status.

4.1.3. Explanation 1b

All teams promoted knowledge redundancy, which means that the same knowledge is shared among more team members (Rindfleisch and Moorman, 2001). However, the teams found it difficult to achieve redundancy in practice because many team members had expertise in different technical areas and, therefore, had specialized roles. Low knowledge redundancy in the team negatively affected the DSM process in that the team members did not pay attention, which resulted in less interaction among them in the DSM. Twelve interviewees stated that they were not interested

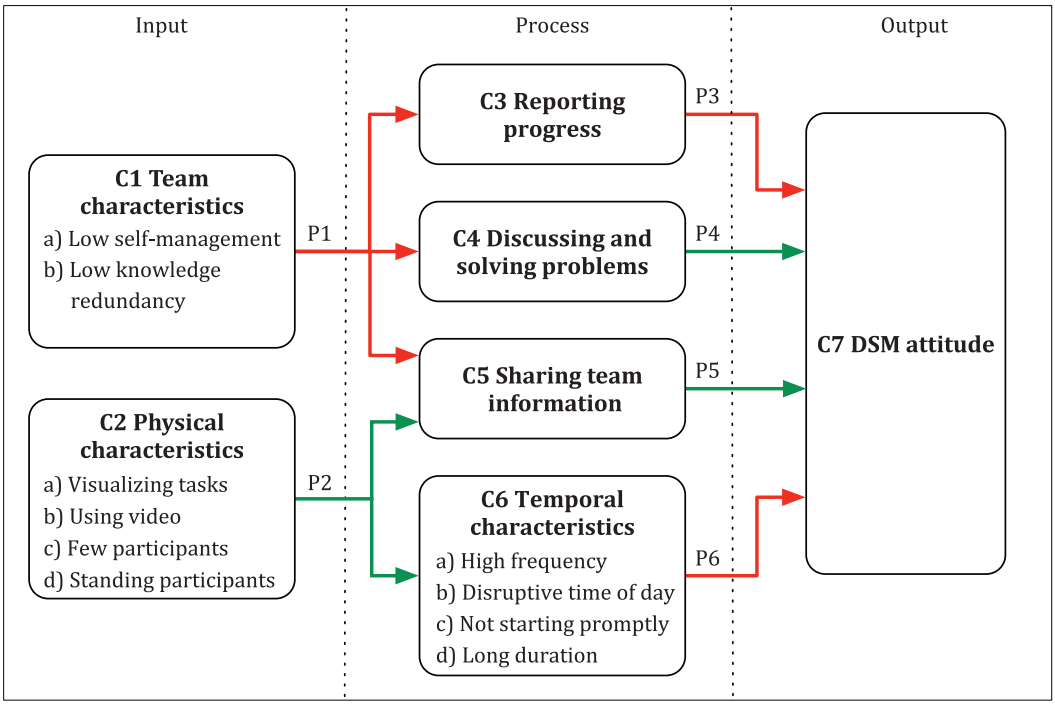


Fig. 5. A theory of DSM.

in what others were saying in the meeting because their roles or tasks were not affected by the information that was shared. One developer stated, "In theory, everyone in the team should be able to solve every task, but in practice, you work in the area you know. […] Personally, I am not very interested in what others are doing." A technical writer said, "The daily meetings do not work that well because they take too much time. We can spend a lot of time talking about solutions, and I think that is boring because it is irrelevant to me." Our findings confirm a previous study that found that over time, as the team members' roles became increasingly specialized, the extent of shared mental models and communication decreased (Levesque et al., 2001).

4.2. C2 physical characteristics

There were mainly four characteristics that affected the DSM process: use of board (C2a), use of video (C2b), number of participants (C2c) and proportion of team standing (C2d). Table 8 shows the physical characteristics of the observed meetings, such as use of board and meeting set-ups (Fig. 6). Table 9 shows the number of participants and proportion of team standing.

4.1.4. Explanation 2a

We observed that using a board to visualize tasks had a positive effect on the DSM process because people could relate what participants said to the tasks for the iteration. Interviewees also indicated that what people said in the meeting was then more relevant to all of the participants. One developer explained an additional benefit: "Now we focus on how to get the tasks to move across the board, and, as a consequence, the tasks get done faster. It is very positive that the stand-up meetings now feel useful." Another developer said, "I think it is important that we view the backlog in the meeting because then the team don't forget the direction of the user stories and the sprint objective. It also reminds us that there are still a lot of tasks waiting to be done."

4.1.5. Explanation 2b

Alpha had both co-located and distributed meetings. In the beginning, Alpha 1 had co-located meetings with only Alpha 1 team members attending (Set-up 1). When the team from Poland (Alpha 2) visited Norway, these two teams had co-located meetings. After Alpha 2 went back to Poland, the two Alpha teams had distributed meetings with the use of video (Set-up 2). When Beta and Gamma had distributed meetings, they did not use video but several team members attended using phone (Set-up 3).

Alpha used video in many of the distributed DSMs (Set-up 2), which positively affected the participation and communication because people paid more attention than when using a phone. The team members in Alpha stated that it was important to use video in the distributed project because they appreciated to see each other's faces during the DSMs. One developer explained, "When I heard that we would start using video, I thought it wouldn't change anything because I thought it was good enough to use phones. Now I can say that it actually did change a lot because if you see people's reactions and their gestures, you immediately know whether they are listening or are bored, and whether they don't understand what you said. Then you have to use other words. Using video also helps because you immediately see who is talking and who is going to say something more." Dorairaj et al. (2012) also found that distributed teams preferred video conferencing to telephone conferencing in distributed DSMs.

4.1.6. Explanation 2c

A low number of participants positively affected the DSM process because then the participants found the information that was shared to be more relevant and therefore paid more attention to what was said. They also tended to be more active in the discussions. The average number of participants in the observed meetings varied among the teams from 7 to 17 with an overall average of 11. We found a negative, nonsignificant correlation between the average number of participants (4th column in Table 9) and the average score of these participants' attitude towards DSMs (Fig. 8) (Spearman $\rho = -0.33$, $p = 0.29$). This finding is consistent

Table 7
Constructs, propositions, explanations and scope of the theory.

Constructs

C1 Team characteristics
C2 Physical characteristics
C3 Reporting progress
C4 Discussing and solving problems
C5 Sharing team information
C6 Temporal characteristics
C7 DSM attitude

Propositions

P1 Some team characteristics negatively affect the DSM process
  (a) Low level of self-management negatively affects the DSM process
  (b) Low level of knowledge redundancy negatively affects the DSM process
P2 Some physical characteristics positively affect the DSM process
  (a) Visualizing tasks positively affects the DSM process
  (b) Using video instead of phone positively affects the DSM process
  (c) Having few participants positively affect the DSM process
  (d) Standing in DSMs decreases duration
P3 Reporting progress negatively affects the attitude towards DSMs
P4 Discussing and solving problems positively affect the attitude towards DSMs
P5 Team information sharing positively affects the attitude towards DSMs
P6 Some temporal characteristics negatively affect the attitude towards DSMs
  (a) Too high frequency negatively affects the attitude towards DSMs
  (b) Conducting DSMs at a disruptive time negatively affects the attitude towards DSMs
  (c) Not starting DSMs promptly negatively affects the attitude towards DSMs
  (d) Too long duration of DSMs negatively affects the attitude towards DSMs

Explanations for each of the propositions

E1 (a) When team members are not in control of the DSMs, it is easy for authoritative managers to use the meeting to obtain status information that is mainly useful to themselves
  (b) Team members with specialized expertise or roles are uninterested in hearing about what others are doing because it does not concern them
E2 (a) When using a board or other visualization tools, it is easier for the participants to relate what the other participants say to the tasks for the iteration
  (b) In a distributed DSM, the participants pay more attention and communicate better when using video than when using phone
  (c) When the number of DSM participants is low, the information shared is perceived as being more relevant to all of them
  (d) Sit-down meetings last longer than stand-up meetings, probably because standing is less comfortable than sitting
E3 When participants do not pay attention or are uncomfortable in DSMs caused by reporting progress, their attitudes towards the meeting are negatively affected
E4 When participants are supported in identifying, avoiding and solving problems in DSMs, the attitudes of the participants are positively affected because spending time on these activities is perceived as useful
E5 Sharing information within a team makes the participants obtain an overview of who is doing what
E6 (a) Frequency as high as every workday is perceived as too often
  (b) DSMs are perceived as disruptive when the development work is interrupted and the time it takes to resume the work is considered a waste
  (c) When DSMs do not start promptly, time is wasted for those who arrive on time
  (d) When DSMs are perceived to have a too long duration participants feel that their time is wasted

Scope

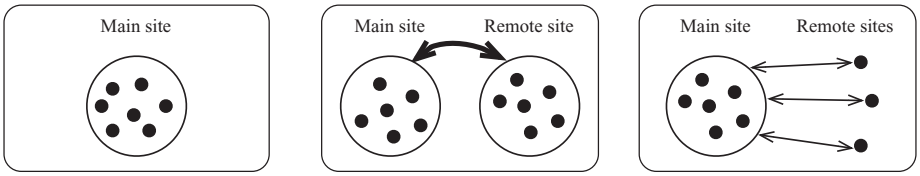
S Agile software development projects

Table 8
Physical characteristics of the observed DSMs.

Company	Team	Location	Language	Board	Set-up[a]	Main turn-taking procedure	Roles present[b]
Alpha	1	Office space	Norwegian	Interactive whiteboard	1	Task-oriented	A, D, PM, T
	1 and 2 same site	Office space	English	Interactive whiteboard	1	Task-oriented	A, D, PM, T, TL
	1 and 2 different sites	Office space	English	Interactive whiteboard	2	Task-oriented	A, D, PM, T, TL
Beta	1	Meeting room	Norwegian	Physical board	3	Facilitator allocated	D, SM, T, TW
	2	Meeting room	Norwegian	None	3	Facilitator allocated	D, SM, T, TW
	3	Meeting room	English	Backlog on projector	1	Round-robin	D, PO, SM, T
Gamma	1	Meeting room	Norwegian	None	1	Round-robin	A, D, PM
	2	Meeting room	English	None	3	Speaker selected next	A, D, PO, SM
	3	Meeting room	English	None	3	Facilitator allocated	A, D, SM

[a] See Fig. 6.
[b] A = Architects; D = Developers; PM = Project Managers; PO = Product Owners; SM = Scrum Masters; T = Testers; TL = Team Leaders; TW = Technical Writers.

with research on group size, which indicates that the smaller the group, the more likely are the participants to be satisfied with the group meeting (Hare, 1952). Sharp et al. (2014) also found that team members often lost focus and interest, and participated less in the DSMs as the size of the team grew. A general recommendation is that the number of participants in a meeting should be as low as possible, but high enough to represent many viewpoints (Romano and Nunamaker, 2001). Doyle and Straus (1993) suggest that a meeting of seven to fifteen participants is ideal for decision-making and problem solving because the meeting is then large enough to allow for a facilitator but small enough to be informal and enable all participants to be involved.



Set-up 1: Fully local

Set-up 2: Distributed over two sites with video

Set-up 3: Distributed over several sites with phone

Fig. 6. Meeting set-ups of DSMs.

Table 9
Number of participants and proportion standing.

Company	Team	*N* meetings	Avg. number of participants (SD)	Avg. number of participants main site (SD)	Avg. number of remote participants (SD)	Avg. proportion of team standing, %
Alpha	1	6	10.3 (2.2)	10.3	0	100
	1 and 2 same site	3	16.7 (2.1)	16.7	0	100
	1 and 2 different sites	10	16.3 (1.8)	8.7 (3.1)	7.6 (2.4)	100
	Team mean	**7**	**14.4 (3.6)**	**11.9 (4.2)**	**2.5 (4.4)**	**100**
Beta	1	18	8.6 (2.2)	6.7 (1.8)	1.9 (1.5)	58
	2	24	6.9 (1.3)	5.8 (1.7)	1.0 (1.0)	52
	3	15	10.4 (2.1)	10.4 (2.1)	0	100
	Team mean	**19**	**8.6 (1.8)**	**7.7 (2.4)**	**1.0 (0.9)**	**69**
Gamma	1	1	9.0	9.0	0	100
	2	1	9.0	3.0	6.0	0
	3	1	9.0	5.0	4.0	0
	Team mean	**1**	**9.0 (0.0)**	**5.7 (3.1)**	**3.3 (3.1)**	**33**
All	**Mean**	**9**	**10.7 (3.2)**	**8.4 (3.2)**	**2.1 (1.4)**	**67**

4.1.7. Explanation 2d

A standing format decreased the duration of the DSM. The primary rationale for standing in a meeting is that its duration is supposedly reduced because standing is less comfortable than sitting (Bluedorn et al., 1999). In our study, one developer said, "It is absolutely necessary that we stand. The few times we tried to sit, the meetings lasted for 20–30 minutes." The teams we observed that had all participants standing had meetings with an average duration of 12 min, while the teams that had all or some of the participants sitting had meetings that lasted 19 min on average; that is, we found sit-down meetings to be approximately 60% longer than stand-up meetings (independently of whether the meetings were distributed or not).

4.3. C3 reporting progress

When participants spent a large proportion of the DSM on reporting progress, their attitudes towards the meeting were negatively affected because spending time on these activities was perceived as uncomfortable or a waste of time (E3). The interviewees reported that they became disengaged when responses to the question "What have you done since we last met?" tended to become progress information to the manager. Nine interviewees stated that when the meeting was about reporting progress, many team members tuned out. An architect said, "I don't think people listen to what is said in the stand-up meeting. Everyone wants to be finished with the round as quickly as possible to get back to work." Some participants said that they often did not pay attention because they were focused on remembering what they had accomplished since the last meeting and preparing what they were going to say. Seven of the interviewees stated that it was unpleasant to report what they worked on. One developer explained that the previous Scrum Master made them feel as though team members were being called to account for being behind schedule: "The feeling I got in the meeting was 'Why haven't you managed it? You have to pull yourself together!' The meeting was more like blaming us for not having done what had been planned. It felt like an oral exam." Another said: "The bad thing about Scrum is that you have to do daily updates. I don't like those meetings. It is kind of reporting. You have to report every single day."

In Beta 3, after a while, the DSMs were more about informing team members about what was relevant for the whole team rather than reporting status to the Scrum Master, which was the case to begin with. One manager explained how they went from reporting to informing each other: "When I came into the organization, I was asked, "Can you show us how to do the Daily Scrum the right way?" They conducted the Daily Scrums very robotic. They did it just because the process said so without understanding the essence. It is very different now. They discuss their own problems and what is important to them; they don't report to the product owner or Scrum Master and don't care whether the managers are there." In contrast, a manager from Beta 1 found the reporting useful to herself: "I find the meeting as a daily source of inspiration because of the information sharing about progress and the opportunity to make corrective actions. As a manager, I think it is a joy to attend because there is always someone in the team who shares positive progress."

4.4. C4 discussing and solving problems

When participants were supported in identifying, avoiding and solving problems in a DSM, the attitudes of the participants were positively affected because spending time on these activities was perceived as useful (E4). Team members helping each other and showing willingness to provide and seek assistance is a critical component of teamwork that is called *back-up behavior* (Dickinson and McIntyre, 1997). Thirteen interviewees stated that they appreciated sharing information about problems and getting help in the meeting. One developer stated why he perceived the last question as more important, "What you did yesterday and what you plan to do today are part of the plan, whereas the impediments faced are unexpected and could mean that our planning has missed something or we did not fully understand the user story." When

problems were discussed in the DSMs, the team members tended to support each other in finding a solution. One architect said: "There were some people in the beginning that didn't like Scrum at all. They felt that if they would share obstacles, it would have consequences for them, and they might be seen as incompetent. But when we got over that, people became more open and would share questions and problems."

The Scrum Guide (Sutherland and Schwaber, 2013a) states that DSMs highlight and promote quick decision-making. We did observe quick decision-making when trying to solve problems, but one should not ignore the negative effects of the short time allowed for making decisions, including lack of time to consider alternatives and document the decisions made. A further discussion of this topic can be found in (Drury et al., 2012; Stray et al., 2012a).

4.5. C5 sharing team information

Sharing information within a team makes the participants obtain an overview of who is doing what and positively affected the attitude towards the DSMs (E5). Obtaining an overview of what other team members were doing was the most frequently mentioned positive outcome of attending DSMs. A technical writer stated, "I don't think the daily meeting works that well, but it is nice to attend even though not everything is directly related to me because I get the bigger picture. I know what others are doing, and I feel more involved, even though I cannot contribute with much." Other interviewees also said that attending DSMs help erase boundaries between roles and sites, and increase team spirit, cohesion and trust.

One developer said, "The information flow in the project is not very good; the stand-up meeting is the main source of information." However, one developer in another team stated, "The way the meeting is today is not worth much. It is a difficult balance how much you should say. Some share too many details, while others just say, 'I work on this.' Something in between would be the best. But it is not easy to know what to say from day to day."

Several DSMs in the studied teams resulted in post meetings that involved the subset of participants for whom the follow-up topic was relevant. In Alpha, the post meetings were seldom initiated by the Scrum Master, while in the other companies the meetings were usually initiated by the Scrum Master. We found no pattern that the duration of the DSMs affected the likelihood of having post meetings.

Herbsleb (2007) found that distributed developers tend to have little knowledge of what people at other sites are doing from day to day, and this lack of awareness makes it more difficult to initiate contact. To help overcome this challenge, the teams in Alpha held post meetings after DSMs while the video was still on. One developer explained: "It is much easier to ask someone at the other site to stay a couple of minutes after a meeting than to invite them later. It might be tricky to invite someone to a separate meeting because you don't know when they are available." Paasivaara et al. (2008) also found that DSMs resulted in informal communication across sites after the meeting.

4.6. C6 temporal characteristics

Four temporal characteristics negatively affected the DSM attitude: too high frequency (C6a), having the meeting at a disruptive time (C6b), not starting promptly (C6c) and too long duration (C6d). Table 10 shows temporal characteristics of the observed meetings.

4.6.1. Explanation 6a

The participants felt that their time was wasted when the DSMs were conducted too frequently. In several teams, a high degree of informal communication outside DSMs reduced the necessity and benefits of the meeting. Members in nine of the teams often talked to each other during the workday because they were co-located in an open-space office. The three teams that had distributed team members also communicated frequently by using chat and e-mail. One developer said, "I can say that today there was nothing interesting at the stand-up, because everybody said something that I already knew." An architect stated, "If someone faces an obstacle, we are supposed to flag this at the stand-up, but my experience is that if people are stuck, they don't wait until the next day but try to find someone who can help right away." A project manager said, "Team members have asked me why we have stand-up meetings, because they feel that they already know what is happening in our team."

Although we refer to the investigated meetings as "daily" stand-up meetings, not all teams conducted them daily; three teams conducted meetings three or four times a week. The average over the companies was 4.6 times a week. Eight of the interviews stated that conducting the meeting every day was too frequent. A Scrum Master said DSMs were worth the time but would appreciate conducting them less frequently. He explained, "Company policy says it is a Daily Scrum, so we have to have it often, but I think twice a week would be fine, and many of the team members agree with me." Another Scrum Master described why they held the meeting every day: "It will require more effort to meet more seldom. Then you have to remember that it is not a meeting today, it is tomorrow. Additionally, you have to remember what you did two days ago when you inform about what you have done."

One project manager said DSMs increased the level of stress because he had to attend these meetings in addition to other mandatory meetings. This statement reflects Luong and Rogelberg's (2005) findings, which showed that a high number of meetings was associated with increased feelings of fatigue and subjective workload.

There might be phases in the iteration in which it would be beneficial to conduct the DSM more frequently than in other phases. One developer stated, "I guess the importance of the meeting might be higher in the middle of the sprint, when the team has already developed part of the code, which is where new problems or potential problems are discovered."

4.6.2. Explanation 6b

When the time of the DSMs was perceived as disruptive, this negatively affected the attitude towards them. One developer stated, "The daily meeting interrupts my workflow and it takes time to resume the work after the meeting." Another stated that the meeting was particularly disruptive during intensive programming. This is consistent with what Solingen et al. (1998) found: The recovery time was longer when the interruption occurred during programming because it requires deep concentration. However, a few interviewees appreciated being interrupted. One developer said, "I believe the stand-up meetings intend to break the developers out of their zone, so they can take a breather to evaluate their situation and update the team." A break from a task may give the subconscious time to process complex problems (Jett and George, 2003).

All the teams held their DSMs in the morning, but they struggled to find a time that satisfied everyone. The challenge was to conduct the meeting as early as possible, but not before all team members had arrived at work. A Scrum Master described the situation as follows: "We have team members who arrive before 7 a.m. and team members who prefer to arrive at 10 a.m. I think the interruption caused by the stand-up meeting is what people find negative, so I try to conduct the meeting as early or late as possible so that the team members get the largest blocks of time to work, but this block is different for everyone." A tester explained

Table 10
Frequency, time of day and duration of the observed DSMs.

Company	Team	Total number of meetings observed	Frequency, times a week	Time of day	Avg. duration, min (SD)
Alpha	1	6	5	09:00 a.m.	11.7 (2.6)
	1 and 2 co-located	3	5	09:00 a.m.	7.0 (3.0)
	1 and 2 distributed	10	5	09:00 a.m.	11.9 (3.7)
	All teams	**19**	**5.0 (mean)**		**10.2 (2.8)**
Beta	1	18	4	10:30 a.m.	20.2 (4.4)
	2	24	4	10:00 a.m.	24.8 (9.9)
	3	15	5	10:00 a.m.	19.0 (9.2)
	All teams	**57**	**4.3 (mean)**		**21.3 (3.0)**
Gamma	1	1	3	09:45 a.m.	9.0
	2	1	5	09:30 a.m.	13.5
	3	1	5	09:45 a.m.	18.0
	All teams	**3**	**4.3 (mean)**		**13.5 (4.5)**
Total: company level ($N=3$)			4.5 (mean)		15.0 (5.7)
Total: team level ($N=9$)			4.6 (mean)		15.0 (5.8)
Total: all meetings ($N=79$)			NA		18.9 (8.9)

that conducting meetings in the morning was important so that everyone would know what they needed to do for the rest of the day. A developer said, "The daily meeting is a way to start the day. Ten o'clock has sort of become the 'clock-in' time although some managers are not happy with that. According to company policy, all staff is supposed to arrive at work at 8:30."

We observed that the time before the meeting was often spent on tasks that did not require full concentration. One developer said, "Normally I will use the time before the meeting to settle down, read new e-mails, update Scrum Works Pro if I forgot to update it last night, and get coffee." A project manager said the meeting could be a disruptive interruption for the team members, but it could also enforce discipline because people want to check their e-mail and read the newspaper anyway; the time before the meeting may be a natural block of time to do these things.

4.6.3. Explanation 6c

When the DSMs did not start promptly, time was felt wasted for those who arrived on time, and search for attendees disturbed other people. Several teams had some kind of punishment for late arrivers. We identified three reasons for a meeting starting late: waiting for people to arrive, waiting for a "critical mass" and waiting for connection to be established. Often, the meeting did not start promptly because not all team members had arrived at work. One developer said, "We start it about 10:30 every morning. But if someone is late, we give a buffer. So, once every team member is here, we'll start the standing meeting." Another reason for delay was that the participants that were present waited for a "critical mass" before they started the meeting. Furthermore, meetings were also delayed due to time spent connecting with distributed team members by phone. In Alpha, the team members were reminded of the meeting by a signal given automatically at 9 a.m. by an electronic board. This forced everyone at work to attend the meeting on time since it was hard to ignore the loud signal. A project manager in Alpha described an additional benefit: "I feel it is more nagging if you as a Scrum Master or project manager go around and say 'Let's have the morning meeting.' The alarm is a mechanism that makes the job easier."

4.6.4. Explanation 6d

The participants felt that their time was wasted when the DSMs lasted longer than planned. The duration of the meeting was noted as too long by thirteen interviewees, five of whom belonged to Beta 2, which had the longest average duration. Table 10 shows the average duration of the observed meetings at the team level, company level and total. All teams planned 15 min for the meeting, and the observed DSMs actually lasted, on average at the company level, exactly 15 min. However, there were great variations. In Beta, they struggled with adhering to the 15-min time limit. Their average meeting length was more than 50% longer than that in Alpha (21.3 versus 10.2 min). At the team level, the average duration varied between 7.0 and 24.8 min.

Fig. 7 shows the meeting duration over time in Alpha and Beta (Gamma is not shown because of a few observations in this company). There was a sharp drop in Beta 2 at Meetings 5 and 6. These meetings were shorter because the regular Scrum Master (project manager) was absent, which is illustrated by an excerpt from a transcribed meeting:

> Developer X: "On Monday the Scrum Master is back and will lead the meetings again."
> Developer Y: "Oh, no!" [Everybody laughs] "Then we will be back to half an hour."

Meetings 12 and 13 in Beta 2 were kept brief because the team had other meetings scheduled (a demo meeting and a company meeting). The graph shows a sharp increase in duration after Meeting 10 in Beta 3. The extended length of Meeting 13 was due to an intense discussion in which disagreements occurred between the testers and the developers. They argued about adding test cases late in the sprint, and one of the testers started crying. Meetings 14 and 15 required extra time for recovering activities after Meeting 13.

We found a positive, mostly nonsignificant correlation between the meeting duration and the number of participants in Beta (B1: $\rho=0.35$, $p=0.23$; B2: $\rho=0.32$, $p=0.17$, B3: $\rho=0.60$, $p=0.02$). In Alpha (all meetings in both teams), we found a negative correlation ($\rho=-0.3$, $p=0.32$). The reason for shorter meetings in Alpha when more people attended was that they focused the discussion on tasks instead of each individual to save time.

The teams tried different practices to keep the meetings short. One practice was to have the meeting facilitator remind the participants to keep the discussion on target. Another practice was to use a countdown timer. A developer explained, "Our updates have stretched a little because sometimes we go straight into technical discussions of our roadblocks instead of waiting for the daily update to finish first. Before we start the daily update, the Scrum Master sets the clock timer to 15 min. If we have not finished when the timer goes off, the team can walk away to continue their work without the need to wait for the rest to finish updating."

4.7. C7 DSM attitude

The last construct in the theory is the attitude towards the DSM. This construct is either directly or indirectly affected,

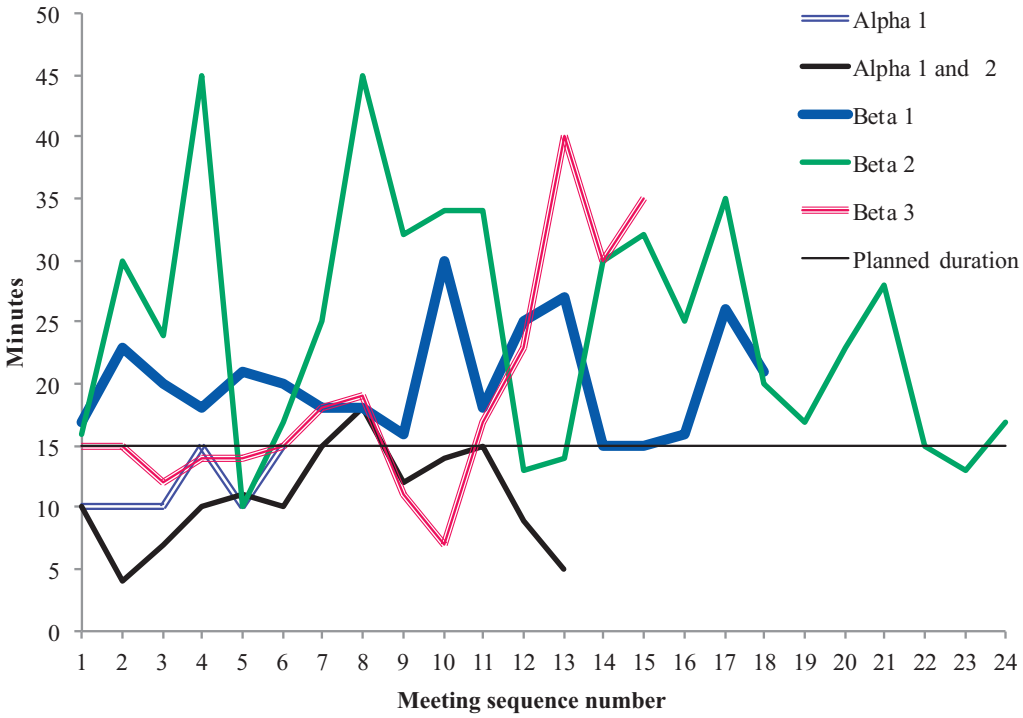


Fig. 7. Duration of DSMs in Alpha and Beta.

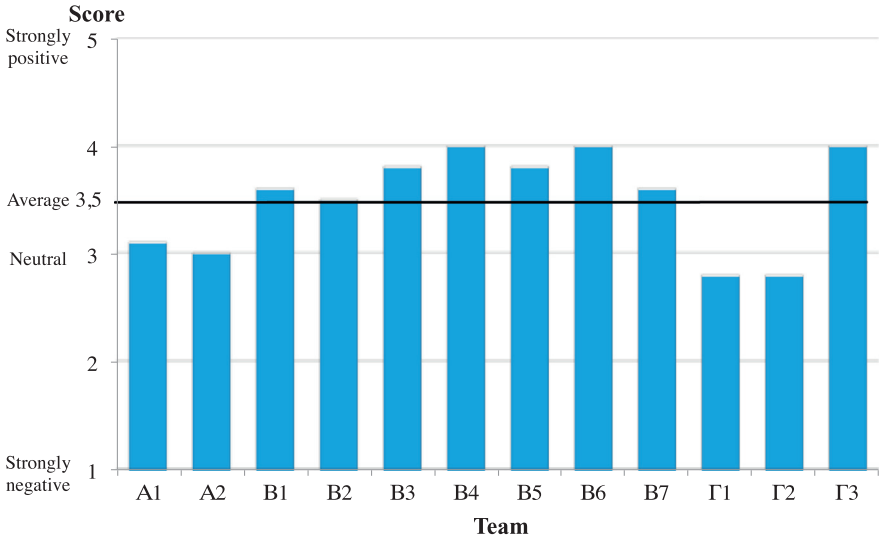


Fig. 8. Average meeting attitude for each team.

positively or negatively, by the six constructs described above. In this section, we report the interviewees' overall attitude towards the DSM. In Section 3.4.4, we described how we arrived at the individual scores. Fig. 8 shows the individual scores aggregated to team level. The score for each team varied from 2.8 to 4.0 with an average of 3.5; that is, the teams were slightly more positive than negative on average. Considering the popularity of the DSM, it was unexpected that the attitudes towards the meeting were not more positive.

The product owners, Scrum Masters and team leaders were the roles that were most positive towards the DSM; see Fig. 9. One explanation is that they had roles in which *receiving* progress information was perceived as highly valuable, while most of the time they did not have to *provide* such information themselves. One product owner explained: "I attend the daily Scrum meetings because as a product owner I need to know the progress of how the team has implemented the sprint backlog." In contrast, the team members had to report detailed progress information, which contributed negatively to their attitude towards the DSM.

Furthermore, the Scrum Masters and team leaders were often the facilitators of the DSMs, which is another explanation for why they were slightly more positive than the team members. A survey conducted by Cohen et al. (2011) found that individuals who considered themselves as meeting facilitators perceived the meeting quality to be significantly higher than did the other participants.

Additionally, in Alpha 1 and 2, we had the opportunity to investigate the attitude towards certain aspects of DSMs by giving team members the questionnaire described in Table 4. Fig. 10 shows the respondents' scores on specific statements. The average overall scores in Alpha 1 and 2 were 3.4 and 3.0, respectively. Although

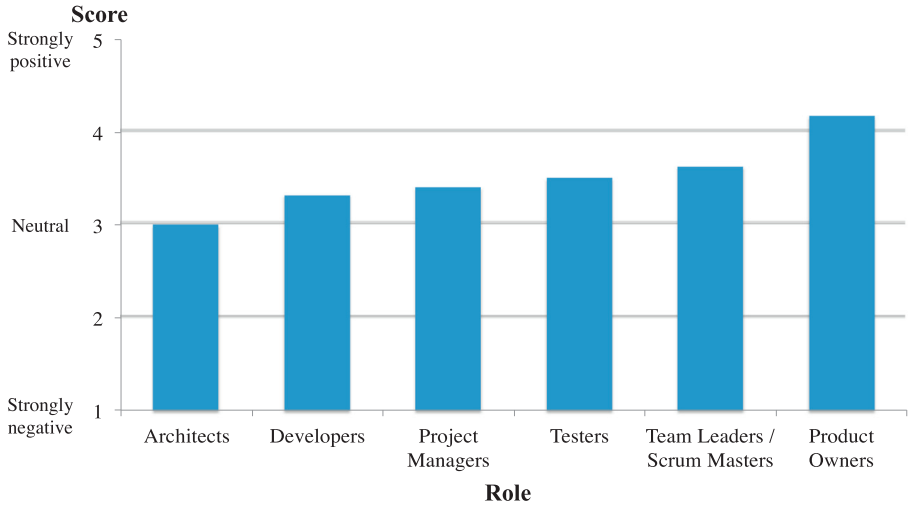


Fig. 9. Average meeting attitude for each role.

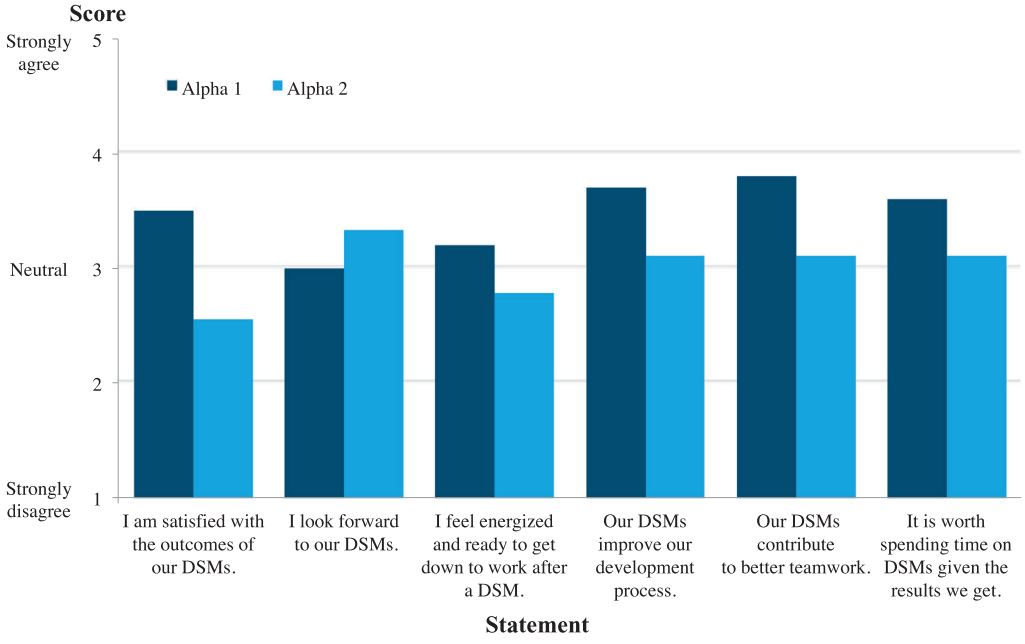


Fig. 10. Meeting attitude of teams Alpha 1 and Alpha 2 from questionnaire.

not directly comparable, the overall attitude in these teams accorded with the scores from the interviews, which were 3.1 and 3.0, respectively. These results indicate consistency between our interview scores and what the participants themselves indicated in the questionnaire.

5. Discussion

On the basis of our findings, this section provides a normative definition of what we now will denote a *stand-up meeting* and a set of guidelines for how to conduct such a meeting. This section also discusses limitations of our study and issues for future research.

5.1. Normative definition based on the empirical findings

Throughout this article the practice investigated is called "DSM", where the letter D stands for "daily". However, the findings indicate that the meeting should not necessarily be held daily but still regularly to distinguish it from an ad hoc meeting. Teams may decide to have it three or four days a week, as practiced and appreciated by many teams in this study. Consequently, it may be sensible to remove the term "daily" from the name of the practice. (We still use the term DSM in the remainder of the article to avoid confusion.)

The Scrum Guide (Sutherland and Schwaber, 2013a) says nothing about standing in the DSMs. However, we found the issue of standing to be important. The teams that had all participants standing had considerably shorter meetings than those that had some people, especially the Scrum Master, sitting. This is consistent with the results of a previous experiment that found that the sit-down meetings lasted 34% longer than the stand-up meetings, but both types of meetings produced decisions with the same quality (Bluedorn et al., 1999). Consequently, we support the recommendation from XP that people should stand in DSMs.

The main purpose of DSMs is to enable team members to obtain an overview of what other team members are doing. This phenomenon is called *team awareness*, which is explained as "... an understanding of the activities of others, which provides a

context for your own activities" (Dourish and Bellotti, 1992). It is not at static state, but the result of recurrent processes of information sharing within a team (Salas et al., 1995).

In summary, based on the empirical investigation we define a stand-up meeting as follows:

> A *stand-up meeting* is a brief communicative event that involves two or more people in a team; it is regularly scheduled with a pre-arranged time and place; the participants stand; it is organized and managed by the team; and its primary purpose is to increase team awareness.

5.2. Empirically based guidelines for DSM

In addition to the recommendations given when introducing our theory in Section 4, this section provides an additional set of guidelines on how to conduct the DSM based on our collected data.

5.2.1. Focus on the future

Team members often respond to the three questions presented in Section 2.2 during DSMs. However, our findings suggest that the team members should only respond to Questions 2 and 3, which we suggest to reformulate as follows: "What will I do until we meet again to help our team achieve its goals?" and "What problems do I know of that may prevent progress?". Scrum-specific formulations of these questions can be found in Sutherland and Schwaber (2013a). We suggest to leave out Question 1 ("What have I done since we last met?") since the answers tend to become dominated by status reporting. Sutherland and Schwaber (2013b) state that the DSM is too often seen as a status event. This statement is also confirmed in our study: reporting progress was the greatest pitfall of DSMs because it often resulted in disengaged or uncomfortable participants. Stray et al. (2012b) observed that a substantial amount of time for Question 1 was spent on self-justification; that is, participants used the time for detailed explanations of what they had done and why they had not achieved as much as expected, if that was the case. Other means than sequentially asking each team member what was done yesterday are more efficient than DSMs for providing information about what has been accomplished. For example, visual workplaces make most status inquiries unnecessary because the project status is constantly displayed, updated and accessible to all (Bell and Orzen, 2010). If need be, additional oral information about what has been accomplished could be provided on the fly when answering Questions 2 and 3.

In addition to the negative aspects of reporting that have already been described, Question 1 consumes a large proportion of the total time in DSMs at the cost of the time available for the two other more important questions. In Stray et al. (2012a), we reported that the proportions of the statements concerning the DSM questions were distributed as follows: Question 1: 53%, Question 2: 25% and Question 3: 22%.

Question 2 is important because informing each other about what one plans to do increases team awareness. The information provided also makes it easier to coordinate work.

Regarding Question 3, Scrum guidelines suggest that DSMs should not be used for discussing solutions to problems raised in the meetings (Schwaber and Beedle, 2002). However, our interviewees stated that spending time on this activity was one of the most valuable aspects of the DSMs. Our observations of the meetings also confirmed that a major part of the communication resulted in identifying problems that were solved or simply avoided. Identifying and avoiding problems may entail major cost savings because finding problems earlier is more cost effective than finding them later (Boehm, 1984). Rising and Janoff (2000) and Pikkarainen et al. (2008) also reported that solving and clearing obstacles were the best parts of DSMs. Of course, solving complex problems will usually take longer than 15 min. In those cases, a post meeting should be held, typically with a subset of the team.

5.2.2. Avoid the facilitator allocating turns

When the facilitator decided the order of speakers, the meeting tended to shift towards reporting status. Consistent with what Boden (1994, p. 89) stated, we also found that when the facilitator allocated the turns, the meeting often became conversations between the facilitator and the respondents, and some roles or persons tended to be given more attention than others, as also reported in Stray et al. (2013). Consequently, we recommend speaking in turns by using a round-robin approach and having a designated first speaker position in the room. It improves the communication if the team members know without intervention who speaks next. It is also a good idea to rotate the role of meeting facilitator. This person does not have to be the Scrum Master, particularly not if the Scrum Master is also a manager.

5.2.3. Strive to find the least disruptive time

If team members all arrive at the same time in the morning, starting the workday with a DSM is a good idea. However, in our study, people arrived at different times, and few of the team members began working on development tasks before the DSM was held. These findings are consistent with what Sharp and Robinson (2004) observed in XP teams: the DSM "heralded the real start of the day." They also observed that the time between developers arriving at work and the DSM being held was used to eat breakfast, check e-mail and read the newspaper. Consequently, if people do not arrive at the same time in the morning, we recommend conducting the meeting just before lunch to avoid the DSM becoming the start of the actual workday. There would then be no additional interruption, and discussions could continue during lunch. The proximity to lunch may also serve as a motivation to end the meeting on time. Keep in mind that DSMs also have an indirect cost called resumption lag, which is the time it takes for a person to collect their thoughts and return to a task after an interruption (Parnin and Rugaber, 2011).

5.2.4. Keep the meeting brief

Viewing a DSM as a "huddle," a brief gathering of team members, instead of a meeting might decrease the duration and reinforce the informality of the meeting. Even though the meetings in our study lasted 15 min on average, perceiving the meeting as lasting too long was one of the most frequently mentioned negative aspect, both in the interviews and in the questionnaires. The DSMs should be as brief as possible and not last more than 15 min. The DSM should always start on time. Do not wait for late arrivers or postpone the meeting if some people are unable to attend on time. Furthermore, ending the meeting on time is important because participants tend to stop paying attention when the time limit is reached.

5.3. Limitations

As in any empirical study, there are some limitations that need to be discussed. Regarding the data collection, the presence of researchers in a DSM may be intrusive and alter the behavior of the meeting attendees. Nevertheless, we believe this effect was small because most of the teams were observed over a long period of time, which meant that the participants became used to being observed. Tape recorders used in meetings may also bias the social situation, although we observed no difference in the behavior of the meeting attendees when the meeting was audio-recorded.

To address potential limitations, we used the seven principles for conducting an interpretive field study that were proposed by

Table 11
Application of Klein and Myers principles for interpretive field research.

The seven principles (Klein and Myers, 1999)	Practiced in this study
1. The Hermeneutic Circle	We observed and interviewed teams over a period of 26 months, which allowed us to study DSMs from different viewpoints and in different phases of the projects. We conducted a large number of interviews and observations, moving back and forth between the three companies. Both the data collection and analysis involved multiple researchers, who analyzed the data systematically in an iterative process, adding more interviews and different data sources to the analysis through theoretical sampling
2. Contextualization	We describe in detail the characteristics of the observed DSMs and the interviewees. We also describe the size and industry of the companies
3. Interaction between researchers and subjects	We had lunch, dinners and informal conversations with the team members. This socializing made it easier for us to be trusted and gave us valuable insight in the teams. Our findings were presented and discussed in all three companies and led to feedback
4. Abstraction and generalization	Investigating DSMs in different national and organizational cultures helped us understand the general perceptions and nature of the practice. By using grounded theory, we abstracted our results into a theory
5. Dialogical reasoning	Our analysis generated DSM constructs that we combined to form a theory. We frequently referred to the data to ensure that codes were representative and checked relationships among codes and themes. In the third phase of the research process, we cross-referenced our results with literature on teamwork and meetings
6. Multiple interpretations	We collected data from 12 teams in four countries. We interviewed 60 persons with various roles and were able to document multiple viewpoints and their reasons. The large amount of data collected reduced the chance of biases and systematic distortions
7. Suspicion	We had a critical approach when analyzing the data; both positive and negative aspects of DSMs were studied. The researchers were external to the organizations, having no interest or agenda beyond creating an unbiased view of DSMs

Klein and Myers (1999), as presented in Table 11. In addition, we undertook data triangulation to reduce researcher bias. We conducted interviews, observed meetings and collected questionnaire data. Such triangulation may yield more reliable data than the use of only one data source because what people report is not always consistent with reality; for example, people tend to underestimate the amount of time they spend in face-to-face meetings when they are asked how much time they spend in various activities (Panko and Kinney, 1995). Interviews yield subjective data; however, the more interviewees stating the same opinion, the less likely bias is associated with that opinion (Diefenbach, 2009). The high number of interviews in our study enabled us to cross-check and compare the data, which produced results that are more convincing than would be possible with fewer interviews.

We studied three quite different organizations. Still, we do not know how well our results generalize to other organizations. For example, the teams that we studied had used DSMs for more than two years. Our results may not necessarily be applicable to teams that just have started using DSMs.

5.4. Future work

Our premise is that practitioners will be better able to conduct DSMs in a way that is beneficial to the organization if they increase their understanding of how the practice affects the overall productivity. However, obtaining such an understanding is difficult. Future work should investigate DSMs more specifically from a cost-benefit perspective. Do the benefits of DSMs justify the time spent in them and the cost of interruption? This complex problem may be divided into several research questions; for example, "How do DSMs affect team performance?" To answer such a question, researchers may investigate how DSMs affect teamwork quality, defined by Hoegl and Gemuenden (2001) in terms of the subconstructs *communication, coordination, balance of member contributions, mutual support, effort* and *cohesion.* Addressing the greater challenge of cost-benefit of DSMs in a solid way requires empirical data from a larger number of software projects. A larger number of projects may also enable analyses of the effect of various context factors on the DSM process. Context factors may include, for example, aspects of company and team culture. Such factors should also be included in a refined version of the DSM theory.

6. Conclusion

Most views and claims about DSMs reported in the literature are based on anecdotal evidence. In contrast, we conducted a grounded theory study of 12 agile teams in three companies. An initial theory of DSM was proposed that consists of seven constructs and six propositions.

We identified factors that affected the DSM attitude positively and negatively. Considering the popularity of DSMs, it is surprising that we found participants to be almost neutral about DSMs on average, although slightly more satisfied than dissatisfied. The two most prominent positive factors were that the team members obtained an overview of what others were doing and were given an opportunity for discussing and solving problems. The two most prominent negative factors were reporting status progress in the meeting and that DSMs were often considered to occupy too much time relative to the gains from the meetings.

The findings show that DSMs may not necessarily have to be held daily, and focus in the meetings should be on discussing and solving problems, and planning for the future rather than reporting what has been done. Furthermore, it is beneficial to be standing in the DSMs and to conduct the meetings by a task board. If the meeting participants are distributed over several sites, it is better to use video than phone. Additionally, the number of participants should be kept small. Teams where not all team members start working at the same time in the morning, should defer the DSMs until the time that is least disruptive for the participants, for example, just before lunch.

The DSM may seem like a simple practice, but in reality, it is not easy to implement it successfully. In order to overcome the negative factors identified, a set of recommendations and guidelines on how to conduct the DSM was proposed to support software companies in realizing its potential benefits. Companies should evaluate our proposed recommendations and guidelines in their own context and continuously evaluate and improve the way they conduct DSMs to make the practice as valuable as possible.

Relatively few grounded theory studies have been conducted in software engineering. An additional contribution of this paper is that it thoroughly describes how such a study may be undertaken in this field.

Acknowledgments

We are grateful to Yngve Lindsjørn, Nils Brede Moe and Øystein Ingebrigtsen for assisting with data collection and to the many participants who shared their experiences with us and generously welcomed us for observations. We thank the anonymous referees for valuable comments. This work was supported by the TeamIT project funded by the Research Council of Norway through Grant no. 193236/I40.

Appendix

Tables A.1–Table A.4.

Table A.1
Interview guide.

Part	Question
Introduction	Present ourselves
	Say thank you for participating
	Assure confidentiality
	Ask permission to tape record
Warm-up	How long have you been working for this company?
	How long have you been on this project?
	Is it okay to ask: "How old are you?"
	What is your role in the team?
	What are you working on now?
	Whom do you see as your team members?
	Do you collaborate with other teams?
Teamwork and meetings	How does the team make decisions?
	Do team members show interest in other individuals' tasks?
	How are tasks allocated?
	How easy is it to complete someone else's task?
	Do you get feedback on your work?
	• How?
	• When?
	Tell me about your daily stand-up meetings.
	• What is working?
	• What is not working?
	• How many attendees?
	• How many attendees?
	Tell me about your retrospective meetings.
	Tell me about the planning meetings.
	How do you solve problems?
	What do you think of the information flow in the project?
	How do you perceive the teamwork in the project?
	Do you have an overview of what others are doing?
	What are some challenges of working with distributed teams?
	What can you think of that could improve the effectiveness of the teamwork or the project in general?
	How do you think agile methodologies and practices are working in the project?
Closing	Do you have any questions for me?
	Is there anything else you would like to discuss that was not covered by the questions asked?

Table A.2
Observation protocol.

Topic	Question
Space	What is the layout of the physical room?
	How are the actors positioned?
Participants	What are the names and relevant details of the people involved?
	Is someone acting as a leader or facilitator?
Activities	What are the various activities and discussions?
Objects	Which physical elements are used?
Acts	Are there any specific individual actions?
	What are the ways in which all actors interact and behave toward each other?
Events	Are there any particular occasions or anything unexpected?
Time	When does the meeting start?
	What is the sequence of events?
	When does the meeting end?
Goals	What are the actors attempting to accomplish?
Feelings	What are the emotions in the particular contexts?
	How is the atmosphere?
Closing	How is the meeting ended?
	Is there a post meeting?

Table A.3
Positive and negative opinions of DSMs from interviews.

Category, *Concept*	Code (# positive statements assigned)	Code (# negative statements assigned)
Team characteristics		
Interaction	Manager obtaining status information	High within-team interaction making DSM superfluous (3)
Knowledge redundancy		Being uninterested in what others are saying (12)
Self-management		Reporting status to the manager (7)
Purpose of conducting DSM		Conducting just because guideline or policy says so (6)
		Not being aware of the purpose of the meeting (3)
Physical characteristics		
Format	Standing in the meeting (2)	Many don't like standing
	Sitting in the meeting	Combination of people standing and sitting
	Fixed seating	
Setting	Seeing each other on video (7)	Poor conference equipment (2)
	No distributed meetings	Lacking a suitable location (5)
Visualization tools	Using boards in the meeting (9)	Poor synchronization between boards (5)
	No board in the meeting	
Participants	Observers acquiring good understanding of the team (3)	Observers speaking when not supposed to
	Good attendance	People working from home not attending
	Speaking English in the meeting (2)	Too many people attending
Reporting progress		
Demonstrating progress	Demonstrating progress (2)	Demonstrating progress (3)
Paying attention		Being disengaged or uncomfortable because of reporting of progress (15)
Discussing and solving problems		
Identifying and avoiding problems	Identifying and avoiding problems (6)	
	Uncovering dependencies (2)	
Back-up behavior	Solving problems (getting help) (13)	
	Asking for help	
Making decisions	Making decisions (2)	
Team information sharing		
Team monitoring	Obtaining an overview of what others are doing (22)	Not understanding what others are saying (3)
	Sharing information (5)	Irrelevant and shallow information (6)
	Sharing knowledge (3)	Long technical discussions (5)
	Praising (2)	
	Learning	
	Showing interest	
Communication	Forcing individuals to communicate (2)	Too formal communication
	Answering the three questions (4)	Answering the three questions (4)
	Mutual trust	No mutual trust
Team orientation	Team spirit and being involved (4)	
	Feeling like a part of a team	Not feeling like a part of a team (2)
	Socializing (seeing each other) (2)	
	Erasing boundaries (2)	
	Being more open	
Coordination	Coordinating work (3)	
	Inviting distributed members to post meeting (2)	No common goal (3)
	Being aware of the need of a post meeting (3)	
Turn-taking procedure	Round-robin (4)	Round-robin (2)
	Facilitator deciding the tasks to discuss (2)	Facilitator deciding next speaker or task to discuss (2)
		No one taking the role as a facilitator
Temporal characteristics		
Frequency	Conducting the meeting every day	Every day is too frequent (8)
	Making other meetings superfluous (5)	Attending two DSMs each day
	Attendance is a habit	Conducting DSMs only when the manager is present (3)
Time of day	A break from work (2)	Stress caused by time spent in addition to other meetings
	Conducting the meeting at start of the day (3)	Interruption of workflow (6)
	Don't need to be held in the morning	The meeting is held too early in the morning
Starting promptness	Using an alarm to signal start	Relying on facilitator to signal start
	Punishing latecomers (2)	Waiting for latecomers before starting
	Agreeing on importance of being on time	People arriving late to the meeting (3)
Duration	Keeping the meeting short (7)	Duration sometimes too long (13)
	Using a countdown-timer (2)	Duration sometimes too short (2)
	Being allowed to leave	
	Being efficient (3)	

Table A.4
Positive and negative opinions of DSMs from questionnaires.

Category, *Concept*	Code (# positive statements assigned)	Code (# negative statements assigned)
Physical characteristics		
Setting	Seeing each other on video	Poor conference equipment (2)
Visualization tools		Tools out of sync with real status
Participants		Too many people attending
		Not all team members attending (2)
Reporting progress		
Manager monitoring		Project manager not giving status
Attention		Being disengaged
Purpose		Not being aware of the purpose of the meeting
Discussing and solving problems		
Identifying and avoiding problems	Identifying and avoiding problems (3)	
	Revealing problems	
Back-up behavior	Solving problems (receive help) (6)	Being unprepared (2)
	Asking for help (2)	
Team information sharing		
Team monitoring	Obtaining an overview of what other team members are doing (6)	Repetitive (2)
	Obtaining a better understanding of who does what	Irrelevant and vague information (3)
	Setting focus	
	Knowing the current situation (3)	
	Sharing task information	
	Sharing knowledge	
Communication	Focusing on getting things done	Forcing individuals to talk
	Being flexible	Not everyone talking
Team orientation	Being involved	
	Socializing (seeing each other) (4)	
Coordination	Making it easier to contact team members later	Assigning tasks
	Knowing that team members are available for discussion	
Temporal characteristics		
Duration	Being efficient	Duration sometimes too long (7)

.

References

Allen, J.A., Sands, S.J., Mueller, S.L., Frear, K.A., Mudd, M., Rogelberg, S.G., 2012. Employees' feelings about more meetings: an overt analysis and recommendations for improving meetings. Manag. Res. Rev. 35, 405–418.

Amin, Z., Guajardo, J., Wisniewski, W., Bordage, G., Tekian, A., Niederman, L., 2000. Morning report: focus and methods over the past three decades. Acad. Med. 75, 1–4.

Anderson, A.H., McEwan, R., Bal, J., Carletta, J., 2007. Virtual team meetings: an analysis of communication and context. Comput. Hum. Behav. 23, 2558–2580.

Anderson, D.J., 2003. Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results. Prentice Hall, Upper Saddle River, NJ.

Aston, J., Shi, E., Bullot, H., Galway, R., Crisp, J., 2005. Qualitative evaluation of regular morning meetings aimed at improving interdisciplinary communication and patient outcomes. Int. J. Nurs. Pract. 11, 206–213.

Bedwell, W.L., Wildman, J.L., DiazGranados, D., Salazar, M., Kramer, W.S., Salas, E., 2012. Collaboration at work: An integrative multilevel conceptualization. Hum. Resour. Manag. Rev. 22, 128–145. doi:10.1016/j.hrmr.2011.11.007.

Bell, S.C., Orzen, M.A., 2010. Lean IT: Enabling and Sustaining Your Lean Transformation. Productivity Press, New York, NY.

Bluedorn, A.C., Turban, D.B., Love, M.S., 1999. The effects of stand-up and sit-down meeting formats on meeting outcomes. J. Appl. Psychol. 84, 277–285.

Boden, D., 1994. The Business of Talk: Organizations in Action. Polity Press, Cambridge, MA.

Boehm, B.W., 1984. Verifying and validating software requirements and design specifications. IEEE Softw. 1, 75–88. doi:10.1109/MS.1984.233702.

Cannon-Bowers, J.A., Salas, E., Converse, S., 1993. Shared mental models in expert team decision making, In: Castellan, N.J. (Ed.). Current Issues in Individual and Group Decision Making, pp. 221–246, Lawrence Erlbaum Associates.

Carmel, E., Agarwal, R., 2001. Tactical approaches for alleviating distance in global software development. IEEE Softw. 18, 22–29. doi:10.1109/52.914734.

Cohen, S., Bailey, D.E., 1997. What makes teams work: group effectiveness research from the shop floor to the executive suite. J. Manag. 23, 239.

Cohen, M.A., Rogelberg, S.G., Allen, J.A., Luong, A., 2011. Meeting design characteristics and attendee perceptions of staff/team meeting quality. Group Dynamics: Theory, Research, and Practice 15, 90–104. doi:10.1037/a0021549.

DeSanctis, G., Poole, M.S., 1994. Capturing the complexity in advanced technology use: adaptive structuration theory. Organ. Sci. 5, 121–147.

Dickinson, T.L., McIntyre, R.M., 1997. A conceptual framework of teamwork measurement. In: Brannick, M.T., Salas, E., Prince, C. (Eds.), Team Performance Assessment and Measurement: Theory, Methods, and Applications. Psychology Press, NJ, pp. 19–43.

Diefenbach, T., 2009. Are case studies more than sophisticated storytelling? Methodological problems of qualitative empirical research mainly based on semi-structured interviews. Qual. Quant. 43, 875–894. doi:10.1007/s11135-008-9164-0.

Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B., 2012. A decade of agile methodologies: towards explaining agile software development. J. Syst. Softw. 85, 1213–1221. doi:10.1016/j.jss.2012.02.033.

Dorairaj, S., Noble, J., Malik, P., 2012. Understanding team dynamics in distributed agile software development. In: Wohlin, C. (Ed.), Proceedings of the 13th International Conference on Agile Processes in Software Engineering and Extreme Programming. Springer, Berlin, Heidelberg, pp. 47–61. doi:10.1007/978-3-642-30350-0_4.

Dourish, P., Bellotti, V., 1992. Awareness and coordination in shared workspaces. In: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work (CSCW'92). ACM, New York, NY, pp. 107–114. doi:10.1145/143457.143468.

Doyle, M., Straus, D., 1993. How to Make Meetings Work: The New Interaction Method, Reprint edition Berkeley Trade.

Drury, M., Conboy, K., Power, K., 2012. Obstacles to decision making in Agile software development teams. J. Syst. Softw. 85, 1239–1254. doi:10.1016/j.jss.2012.01.058.

Dybå, T., Sjøberg, D., Cruzes, D.S., 2012. What works for whom, where, when, and why? on the role of context in empirical software engineering. In: Presented at the: Proceedings of the ACM–IEEE international symposium on Empirical software engineering and measurement (ESEM '12). ACM, New York, NY, pp. 19–28. doi:10.1145/2372251.2372256.

Elsayed-Elkhouly, S., Lazarus, H., Forsythe, V., 1997. Why is a third of your time wasted in meetings? J. Manag. Dev. 16, 672–676. doi:10.1108/02621719710190185.

Endsley, M.R., 1995. Toward a theory of situation awareness in dynamic systems. Hum. Factors 37, 32–64.

Glaser, B.G., Strauss, A.L., 1967. The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine, New York, NY.

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

Glaser, B.G., 1992. Basics of Grounded Theory Analysis: Emergence vs Forcing. Sociology Press, Mill Valley, CA.

Glaser, B.G., 1998. Doing Grounded Theory: Issues and Discussions. Sociology Press, Mill Valley, CA.

Glaser, B.G., 2001. The Grounded Theory Perspective: Conceptualization Contrasted with Description. Sociology Press, Mill Valley, CA.

Glaser, B.G., 2005. The Grounded Theory Perspective III: Theoretical Coding. Sociology Press, Mill Valley, CA.
Glaser, B.G., 2011. Getting Out of the Data: Grounded Theory Conceptualization. Sociology Press, Mill Valley, CA.
Goulding, C., 1998. Grounded theory: the missing methodology on the interpretivist agenda. Qual. Mark. Res 1, 50–57.
Hannay, J.E., Sjøberg, D.I.K., Dybå, T., 2007. A systematic review of theory use in software engineering experiments. IEEE Trans. Softw. Eng. 33, 87–107. doi:10.1109/TSE.2007.12.
Hare, A.P., 1952. A study of interaction and consensus in different sized groups. American Sociological Review 17, 261–267.
Hasnain, E., Hall, T., Shepperd, M., 2013. Using experimental games to understand communication and trust in Agile software teams. Coop. Hum. Asp. Softw. Eng. 117–120. doi:10.1109/CHASE.2013.6614745.
Herbsleb, J., Mockus, A., 2003. Formulation and preliminary test of an empirical theory of coordination in software engineering,. In: Proceedings of the 9th European Software Engineering Conference held jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE-11). ACM, New York, NY, pp. 138–147. doi:10.1145/940071.940091.
Herbsleb, J., 2007. Global software engineering: the future of socio-technical coordination. In: Proceedings of International Conference on Future of Software Engineering, pp. 1–10. doi:10.1109/FOSE.2007.11.
Hernandez, C.A., 2009. Theoretical coding in grounded theory methodology. Ground. Theory Rev. 8, 51–59.
Hobfoll, S.E., 2001. The influence of culture, community, and the nested-self in the stress process: advancing conservation of resources theory. Appl. Psychol.: Int. Rev. 50, 337–421.
Hoegl, M., Gemuenden, H., 2001. Teamwork quality and the success of innovative projects: a theoretical concept and empirical evidence. Organ. Sci. 12, 435–449. doi:10.1287/orsc.12.4.435.10635.
Jett, Q.R., George, J.M., 2003. Work interrupted: a closer look at the role of interruptions in organizational life. Acad. Manag. Rev. 28, 494–507. doi:10.5465/AMR.2003.10196791.
Kauffeld, S., Lehmann-Willenbrock, N., 2012. Meetings matter: effects of team meetings on team and organizational success. Small Group Res. 43, 130–158. doi:10.1177/1046496411429599.
Klein, H., Myers, M., 1999. A set of principles for conducting and evaluating interpretive field studies in information systems. MIS Q. 23, 67–93. doi:10.2307/249410.
Kniberg, H., Skarin, M., 2010. Kanban and Scrum making the most of both. InfoQ http://www.infoq.com/minibooks/kanban-scrum-minibook (accessed October 2015). .
Landis, J.R., Koch, G.G., 1977. The measurement of observer agreement for categorical data. Biometrics 33, 159–174.
Levesque, L.L., Wilson, J.M., Wholey, D.R., 2001. Cognitive divergence and shared mental models in software development project teams. J. Organ. Behav. 22, 135–144. doi:10.1002/job.87.
Luong, A., Rogelberg, S.G., 2005. Meetings and more meetings: the relationship between meeting load and the daily well-being of employees. Group Dyn.: Theory Res. Pract. 9, 58–67. doi:10.1037/1089-2699.9.1.58.
McHugh, O., Conboy, K., Lang, M., 2012. Agile practices: the impact on trust in software project teams. IEEE Softw. 29, 71–76. doi:10.1109/MS.2011.118.
Merisalo-Rantanen, H., Tuunanen, T., Rossi, M., 2005. Is extreme programming just old wine in new bottles: a comparison of two cases. J. Database Manag. 16, 41–61. doi:10.4018/jdm.2005100103.
Moe, N.B., Dingsøyr, T., Dybå, T., 2010. A teamwork model for understanding an agile team: a case study of a Scrum project. Inf. Softw. Technol. 52, 480–491. doi:10.1016/j.infsof.2009.11.004.
O'Neill, T.A., Allen, N.J., 2012. Team meeting attitudes: conceptualization and investigation of a new construct. Small Group Res. 43, 186–210. doi:10.1177/1046496411426485.
Paasivaara, M., Durasiewicz, S., Lassenius, C., 2008. Using Scrum in a globally distributed project: a case study. Softw. Process: Improv. Pract. 13, 527–544. doi:10.1002/spip.402.
Paez, O., Salem, S., Solomon, J., Genaidy, A., 2005. Moving from lean manufacturing to lean construction: toward a common sociotechnological framework. Hum. Fact. Ergon. Manuf. 15, 233–245. doi:10.1002/hfm.20023.
Panko, R., Kinney, S., 1995. Meeting profiles: size, duration, and location. In: Proceedings of the 28th Hawaii International Conference on System Sciences, pp. 1002–1011.
Parnin, C., Rugaber, S., 2011. Resumption strategies for interrupted programming tasks. Softw. Qual. J. 19, 5–34. doi:10.1007/s11219-010-9104-9.
Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J., 2008. The impact of agile practices on communication in software development. Empir. Softw. Eng. 13, 303–337. doi:10.1007/s10664-008-9065-9.
Reinig, B.A., 2003. Toward an understanding of satisfaction with the process and outcomes of teamwork. J. Manag. Inf. Syst. 19, 65–83.
Rindfleisch, A., Moorman, C., 2001. The acquisition and utilization of information in new product alliances: a strength-of-ties perspective. J. Mark. 65, 1–18.
Rising, L., Janoff, N., 2000. The Scrum software development process for small teams. IEEE Softw. 17, 26–32. doi:10.1109/52.854065.
Rising, L., 2002. Agile meetings: putting frequent, short meetings to work for your team. STQE Mag. – Softw. Test. Qual. Eng. 4, 42–46.
Rogelberg, S.G., Leach, D.J., Warr, P.B., Burnfield, J.L., 2006. Not another meeting!" Are meeting time demands related to employee well-being? J. Appl. Psychol. 91, 86–96. doi:10.1037/0021-9010.91.1.83.
Romano, N.C., Nunamaker Jr., J.F., 2001. Meeting analysis: findings from research and practice. In: Proceedings of the 34th Hawaii International Conference on System Sciences. Maui, Hawaii, pp. 1–13. doi:10.1109/HICSS.2001.926253.
Salas, E., Prince, C., Baker, D.P., Shrestha, L., 1995. Situation awareness in team performance: implications for measurement and training. Hum. Factors 37, 123–136. doi:10.1518/001872095779049525.
Schwaber, K., Beedle, M., 2002. Agile Software Development with Scrum. Prentice Hall, Upper Saddle River, NJ.
Scott, C.W., Shanock, L.R., Rogelberg, S.G., 2012. Meetings at work: advancing the theory and practice of meetings. Small Group Res. 43, 127–129. doi:10.1177/1046496411429023.
Sharp, H., Sharp, H., Robinson, H., 2004. An ethnographic study of XP practice. Empir. Softw. Eng. 9, 353–375. doi:10.1023/B:EMSE.0000039884.79385.54.
Sharp, J.H., Ryan, S.D., Prybutok, V.R., 2014. Global agile team design: an informing science perspective. Inf. Sci.: Int. J. Emerg. Transdiscipl. 17, 175–187.
Sjøberg, D.I.K., Dybå, T., Jorgensen, M., 2007. The future of empirical methods in software engineering research. In: Proceedings of International Conference on Future of Software Engineering, FOSE'07. IEEE; Minneapolis, MN, pp. 358–378. doi:10.1109/FOSE.2007.30.
Sjøberg, D., Dybå, T., Anda, B., Hannay, J., 2008. Building theories in software engineering. In: Shull, F., Singer, J., Sjøberg, D. (Eds.), Guide to Advanced Empirical Software Engineering,. Springer, London, pp. 312–336. doi:10.1007/978-1-84800-044-5_12.
Sjøberg, D., Johnsen, A., Solberg, J., 2012. Quantifying the effect of using Kanban versus Scrum: a case study. IEEE Softw. 29, 47–53. doi:10.1109/MS.2012.110.
Solingen, R., Berghout, E., Latum, F., 1998. Interrupts: just a minute never is. IEEE Softw. 15, 97–103. doi:10.1109/52.714843.
Sonnentag, S., Volmer, J., 2009. Individual-level predictors of task-related teamwork processes: the role of expertise and self-efficacy in team meetings. Group Organ. Manag. 34, 37–66. doi:10.1177/1059601108329377.
Spradley, J.P., 1980. Participant Observation, 1st ed. Holt, Rinehart and Winston, Austin, TX.
Strauss, A., Corbin, J.M., 1990. Basics of Qualitative Research: Grounded Theory Procedures and Techniques, 2nd ed. Sage Publications, Thousand Oaks, CA.
Stray, V.G., Moe, N.B., Dingsøyr, T., 2011. Challenges to teamwork: a multiple case study of two agile teams. In: Sillitti, A., Hazzan, O., Bache, E., Albaladejo, X. (Eds.), Proceedings of International Conference on Agile Processes in Software Engineering and Extreme Programming, Madrid, Spain. Springer, Berlin, Heidelberg, pp. 146–161. doi:10.1007/978-3-642-20677-1_11.
Stray, V.G., Moe, N.B., Aurum, A., 2012. Proceedings of the 38th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, pp. 274–281. doi:10.1109/SEAA.2012.16.
Stray, V.G., Moe, N.B., Dybå, T., 2012. Escalation of commitment: a longitudinal case study of daily meetings. In: Wohlin, C. (Ed.), Proceedings of the 13th International Conference on Agile Processes in Software Engineering and Extreme Programming, Malmö, Sweden. Springer, Berlin, Heidelberg, pp. 153–167. doi:10.1007/978-3-642-30350-0_11.
Stray, V.G., Lindsjørn, Y., Sjøberg, D.I.K, 2013. Obstacles to efficient daily meetings in agile development projects: a case study. In: Proceedings of the 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. IEEE, Baltimore, MD doi:10.1109/ESEM.2013.30, 105–102.
Sutherland, J., Schwaber, K., 2013a. The Scrum Guide, pp. 1–16. http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf (accessed October 2015).
Sutherland, J., Schwaber, K., 2013b. Changes Between 2011 and 2013 Scrum Guides http://www.scrumguides.org/revisions.html (accessed October 2015).
VersionOne, 2007. 2nd Annual State of Agile Development Survey. http://www.versionone.com/pdf/2007-state-of-agile-survey.pdf (accessed October 2015).
VersionOne, 2008. 3rd Annual State of Agile Development Survey. http://www.versionone.com/pdf/2008-state-of-agile-survey.pdf (accessed October 2015).
VersionOne, 2009. 4th Annual State of Agile Development Survey https://www.versionone.com/pdf/2009-state-of-agile-survey.pdf (accessed October 2015).
VersionOne, 2010. 5th Annual State of Agile Development Survey. https://www.versionone.com/pdf/2010-state-of-agile-survey.pdf (accessed October 2015).
VersionOne, 2011. 6th Annual State of Agile Development Survey. https://www.versionone.com/pdf/2011-state-of-agile-survey.pdf (accessed October 2015).
VersionOne, 2013. 7th Annual State of Agile Development Survey. https://www.versionone.com/pdf/2012-state-of-agile-survey.pdf (accessed October 2015).
VersionOne, 2014. 8th Annual State of Agile Survey.https://www.versionone.com/pdf/2013-state-of-agile-survey.pdf (accessed October 2015).
VersionOne, 2015. 9th Annual State of Agile Survey. https://www.versionone.com/pdf/state-of-agile-development-survey-ninth.pdf (accessed October 2015).
Yu, X., Petter, S., 2014. Understanding agile software development practices using shared mental models theory. Inf. Softw. Technol. 56, 911–921. doi:10.1016/j.infsof.2014.02.010.
Zijlstra, F.R.H., Roe, R.A., Leonora, A.B., Krediet, I., 1999. Temporal factors in mental work: effects of interrupted activities. J. Occup. Organ. Psychol. 72, 163–185. doi:10.1348/096317999166581 .

Viktoria Stray received the MSc degree in computer science from the Norwegian University of Science and Technology (NTNU) in 2007 and the PhD degree in software engineering from the University of Oslo in 2014. She has three years industry experience from working as a consultant at Accenture and is a certified NLP Master Business Coach from the Norwegian Coaching and NLP Academy. Currently she is a postdoctoral fellow at the University of Oslo. Her research interests include software development methods and socio-technical factors influencing software project success.

Dag I.K. Sjøberg received the MSc degree in computer science from the University of Oslo in 1987 and the PhD degree in computing science from the University of Glasgow in 1993. He has five years of industry experience as a developer and group leader. In 2001, he formed the Software Engineering Department at Simula Research Laboratory and was its leader until 2008, when it was number 1 in a ranking by the Journal of Systems and Software. Since 1999 he has been a full professor of software engineering at the University of Oslo. Sjøberg was an Associate Editor of *Empirical Software Engineering* from 2002 to 2009 and an Associate Editor of *IEEE Transactions on Software Engineering* from 2010 to 2014. His main research interests are the software life cycle, including agile and lean development processes, skill assessment, and empirical research methods in software engineering. Sjøberg is a member of IEEE and ACM.

Tore Dybå received the MSc degree in electrical engineering and computer science from the Norwegian Institute of Technology in 1986, and the Dr. Ing. degree in computer and information science from the Norwegian University of Science and Technology in 2001. He has eight years of industry experience from Norway and Saudi Arabia. He is a chief scientist and research manager at SINTEF ICT and a professor at the University of Oslo. For the period 2001–2012, he was ranked as the top scholar worldwide in agile software development by the Journal of Systems and Software. He was on the editorial board of Empirical Software Engineering from 2007 to 2013. Since 2011 he has beeneditor of the Voice of Evidence column in IEEE Software, and since 2013 he has been on the editorial board of Information and Software Technology. His research interests include evidence-based software engineering, software process improvement, and agile software development.