Contents lists available at ScienceDirect

Information and Software Technology

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

# Understanding agile software development practices using shared mental models theory

Xiaodan Yu [a,*], Stacie Petter [b,1]

[a] School of Information Technology and Management, University of International Business and Economics, Boxue Building 1311, No. 10 Huixin Dongjie, Chaoyang District, Beijing 100029, China
[b] College of Information Science & Technology, University of Nebraska at Omaha, 6001 Dodge Street, PKI 173B, Omaha, NE 68182, USA

ARTICLE INFO

*Article history:*
Received 24 May 2013
Received in revised form 24 February 2014
Accepted 25 February 2014
Available online 6 March 2014

*Keywords:*
Agile software development
Agile practices
Shared mental models theory
Extreme programming
Scrum

ABSTRACT

*Context:* Agile software development is an alternative software development methodology that originated from practice to encourage collaboration between developers and users, to leverage rapid development cycles, and to respond to changes in a dynamic environment. Although agile practices are widely used in organizations, academics call for more theoretical research to understand the value of agile software development methodologies.
*Objective:* This study uses *shared mental models* theory as a lens to examine practices from agile software methodologies to understand how agile practices enable software development teams to work together to complete tasks and work together effectively as a team.
*Method:* A conceptual analysis of specific agile practices was conducted using the lens of shared mental models theory. Three agile practices from Xtreme Programming and Scrum are examined in detail, system metaphor, stand-up meeting, and on-site customer, using shared mental models theory.
*Results:* Examining agile practices using shared mental models theory elucidates how agile practices improve collaboration during the software development process. The results explain how agile practices contribute toward a shared understanding and enhanced collaboration within the software development team.
*Conclusions:* This conceptual analysis demonstrates the value of agile practices in developing shared mental models (i.e. shared understanding) among developers and customers in software development teams. Some agile practices are useful in developing a shared understanding about the tasks to be completed, while other agile practices create shared mental models about team processes and team interactions. To elicit the desired outcomes of agile software development methods, software development teams should consider whether or not agile practices are used in a manner that enhances the team's shared understanding. Using three specific agile practices as examples, this research demonstrates how theory, such as shared mental models theory, can enhance our understanding regarding how agile practices are useful in enhancing collaboration in the workplace.

## 1. Introduction

Agile software development is a lightweight, incremental software development method with specific practices that emphasize close interaction with customers. One benefit of agile software development practices is the ability for software development teams to adapt to changing requirements from customers while identifying and reducing certain risks that arise during software development [1]. Yet, the full benefits of agile software development are not fully understood [2], and research examining agile software development as an alternative to traditional methods is sparse [1]. Although agile software development methods were created by software developers in response to frustration with traditional software development approaches, critics argue that "there is little scientific support for many of the claims made by the agile community; and [agile practices] are rarely applicable, and are rarely applied by the book" [3, p. 836]. These issues have been a concern and criticism of agile software development among researchers, and sometimes practitioners, because agile software development was developed out of practice with no theoretical foundation. Only a few studies have explored the theoretical

* Corresponding author. Tel.: +86 10 6449 5140.
*E-mail addresses:* yxd.xiaodanyu@gmail.com (X. Yu), spetter@unomaha.edu (S. Petter).
[1] Tel.: +1 402 554 2077.

http://dx.doi.org/10.1016/j.infsof.2014.02.010
0950-5849/© 2014 Elsevier B.V. All rights reserved.

underpinnings of agile software development [4]. As organizations increase their reliance on agile software development, exploring the tenets of agile software development methods using theory can yield insights to understand how agile software development methods provide value to software development teams, such as explaining how each agile practice contributes to higher levels of collaboration among the software development team and customers.

Therefore, this study seeks to answer the following question: "How can theory be applied to agile software practices to explain how agile practices enable higher levels of collaboration during software development?" To answer this question, we apply a theory from cognitive psychology known as shared mental models. The literature on agile software development and the literature on shared mental models have an overlapping interest in improving team performance. The theory of shared mental models investigates the role of shared understanding on team performance and has prescribed methods for fostering shared mental models. Agile practices seek to enhance the software development process through improved interaction among software development teams and customers. Why is improved interaction needed among development teams and customers? What aspects of interactions should be emphasized among software development teams and customers? How does increased interaction improve collaboration during software development? These questions remain unclear. In this paper, we propose that shared mental models theory provides a theoretical foundation to explain the importance of agile practices for collaboration by explicating how agile practices enhance the team's shared understanding. Through this application of shared mental models theory, we offer an exemplar of how theory can explain one the benefits of using agile practices, higher levels of collaboration.

This paper is organized as follows. First, we explain why the theory of shared mental models is applicable to agile software development by reviewing background information on the related domains: agile software development and shared mental models. Next, in the context of three agile practices, we examine agile practices and their value in the context of shared mental models. This examination of agile practices in the context of shared mental models theory explains how the agile practices offer benefits to the software development team. We conclude by offering recommendations for future research and highlighting the contributions of this research.

## 2. Background

### 2.1. Agile software development

In the late 1990s, agile software development methods emerged as a response to the challenges associated with traditional software development methods, which are often perceived as inflexible and unable to address changing requirements from users [5]. Agile is formally defined by Conboy [6] as "the continual readiness of an ISD (information systems development) method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment" [6, p. 340]. Many agile methodologies have been developed [7], with Scrum and Extreme Programming (XP) considered as two of the most widely used agile methods. Each agile methodology is comprised of its own specified practices with varied focuses. For example, XP consists of practices that focus on software development team activities, while Scrum consists of practices to improve project management by quickly exposing risks within the project [7].

The value of agile software development over traditional methods is a focus on people's interactions (developers and users) within a project as one of the primary drivers of success [8]. Although the Standish Group's reports can be controversial [9], this widely cited group has stated that "software applications developed through the agile process have three times the success rate of the traditional waterfall method and a much lower percentage of time and cost overruns" [10, p. 25]. The promise of agile practices is that these methods offer the potential to enable software development teams to adapt to customer's changing requirements through high levels of interaction and collaboration, which can lead to better project outcomes.

Despite the popularity of agile software development, organizations struggle to adopt agile practices. When organizations choose to adopt agile practices, but encounter challenges in adopting agile practices, the general tendency is to abandon the practice [11]. Yet, when agile practices are abandoned or implemented improperly, organizations tend to struggle to develop and maintain collaboration throughout the project [12]. Further, given the large number of different agile software development methodologies and practices, it is common for organizations to adapt a few agile practices rather than implement a full agile software development methodology. Understanding the virtue of each agile software development practice within a methodology is a necessary step toward a successful implementation of agile software development. Yet empirical research has found the perceived benefits of agile software development are not fully understood in research or in organizations [13]. The primary factors that influence the adoption of agile software development among organizations are subjective norms and training [2]. Studies by agile practitioners suggest that only half of organizations that state that they use agile methods actually meet the criteria that define agile software development [14].

By applying theory to agile practices, one is able to understand the value of agile practices as techniques to increase collaboration within the software development team and among the software developers and customers.

### 2.2. Shared mental models

Shared mental models is a theory from cognitive psychology that focuses on the thought processes or activities that occur at a team level. While heterogeneity of team members can strengthen a team by leveraging diversity, shared mental models theory proposes that effective teams need to maintain a shared understanding within the team [15]. Without some commonality among the team members in terms of understanding the tasks and relationships within the team, accomplishing team goals would be nearly impossible.

Shared mental models was developed based on mental model theory; mental model theory was first proposed by Johnson-Laird to account for individual reasoning [16]. Mental model theory proposed that an individual's mind holds representations of anything that an individual has encountered in the physical world. Such representations are abstract and accessible by individuals when learning new knowledge or solving problems [17].

The theory of shared mental models extends the idea of an individual's mental model to conceptualize teams as a unified information processing unit. A shared mental model is defined as the "knowledge structures held by members of a team that enable them to form accurate explanations and expectations for the task, and, in turn, to coordinate their actions and adapt their behavior to demands of the task and other team members" [15, p. 228]. Shared mental models provide the team with an internal knowledge base that allows team members to decide what actions to take when novel events happened. Through shared mental models, teams form compatible expectations of the task and team. These shared mental models help the team to understand current events, consider what may happen in the near future for the team, and understand why events happen [18].

There are two different types of shared mental models developed within a team. A team can share mental models about not only the tasks to be performed, but also how the team will interact. *Taskwork* mental models focus on performing tasks as a team, such as understanding the goals, complexities, challenges, interdependencies, and procedures of accomplishing tasks as a group [15]. *Teamwork* mental models are the shared understanding related to the team's interaction and coordination, such as communication patterns, roles and responsibilities of team members, role interdependencies, and background knowledge of each team member (e.g., individual team member's preferences, skills, and habits) [15].

Ideally, teams will develop mental models that are not only similar among the individual team members, but also accurate about the nature of the task and the manner in which the team will work together [18]. The degree of the team's shared mental model *similarity* is indicated by the degree of consistency among each team members' mental models in content and/or structure. The shared mental model *accuracy* refers to the degree to which a team member's mental model is similar to a "true score," which is an objective view of the task and team interaction usually obtained from asking experts [19]. Previous studies have suggested that shared mental models are especially important in teams that involve intense stress and in teams that are unable to engage in constant communication. Recognizing the important role of shared mental models in enhancing teams' performance, prior research examining shared mental models theory have developed several both theory-driven and empirically-grounded practices, such as planning [20], leader briefings [21], reflexivity, cross-training, and guided self-correction [22] to support the development of shared mental models within a team. Shared mental model practices emphasize enhancing the similarity and accuracy of taskwork mental models and/or teamwork mental models to improve the shared understanding within the team.

#### 2.2.1. Cognitive processes of shared mental models

Teams rely on essential cognitive processes to build shared mental models [23–25]. In this paper, we identify four specific stages for developing a shared mental model within a team: knowing, learning, understanding, and executing. Knowing refers to the stage of shared mental model development in which team members are exposed to information that is relevant for accomplishing team tasks or projects. When teams are first established, teams will engage in various kinds of information exchange activities, ranging from a casual team chat to team building exercise to a formal team orientation meeting [23]. These knowing activities expose team members to information that is relevant to the team for accomplishing the team's goals. During this stage, team members are encouraged to share information that is previously possessed individually by each team member to others in the team. Without intervention, team members will tend to discuss information that is shared among all members of the team, rather than relying on the unique information held by each team member [26]. Therefore, knowing activities are critical to ensure the meta-knowledge of the team (i.e., knowledge about the knowledge of specific team members) is explicitly identified.

Learning is the shared mental model development stage in which team members begin to integrate information obtained in the knowing stage. Transactive memory system theory states that each team possesses a system to store meta-knowledge of the team [27]. Team meta-knowledge includes information about which individuals in the team possess the knowledge needed to guide the functioning of the team. Therefore, in this learning stage, team members integrate information gathered in the knowing stage and to build the team's transactive memory system. Ideally at the learning stage, team members should have a clear knowledge of team members' expertise, skills, knowledge, and background. Further, there should be an accurate knowledge of the important information that is relevant to fulfilling the team's goals. The outcome of the learning stage is essential for knowledge construction at the team level.

Understanding is the third stage of shared mental model development when team members develop shared understanding about tasks and the team by solidifying the views and perspectives that each team member holds individually. In the understanding stage, teams reconcile conflicts and reconfirm goals, values, task strategies, and team interaction strategies through rounds of team communication and ongoing team collaboration [24,25]. The goal of this stage is to reach consensus and build common ground so that teams are ready for the next stage.

The fourth stage, executing, occurs when teams develop shared understanding and execute team goals. When teams experience a novel situation during the executing stage, teams adaptively respond to the novel situation based on the shared understanding of the task environment established before. The novel situations either will serve to reinforce the established shared mental models of the team or will facilitate another round of the shared mental model development process [19–21].

### 2.3. Agile practices and shared mental models theory

Most agile practices were developed using the core values of agile software development [1]. Yet, the lack of a solid theoretical base for agile practices [3] causes difficulty or inconsistency in agile practices adaptation [28] because the essential purpose of agile practices is not well understood or clearly conveyed to agile teams. This section explains the value of examining the benefit of agile practices using the lens of shared mental model theory.

Shared mental models theory was originally used to account for the performance of military teams, such as whether or not a group accomplished a mission successfully [15,29,30]. These studies identified teams that possessed shared mental models were better at predicting other team members' behaviors and needs [31]. Later studies applied shared mental models theory in contexts, such as business, manufacturing, education, and health care [32–36].

Several studies have examined whether or not shared mental models theory is related to the improvement of information systems development team performance [37–40]. These studies have found that shared mental models among information system development teams facilitates information sharing between developers, reconciles conflicts between client representatives and software development project leaders, and improves the overall quality of software. Rai et al. [40] found that having customer representatives on development teams, rather than developers visiting the client site, improved project success. The authors noted that by having customer representatives that were considered a part of the development team, a shared sense of trust, norms, and values was created with the software developers, which proved useful to address issues that arose during the project. In their study, Zhang et al. [41] discovered that developers and testers possess different mindsets and goals. The developers' primary goal was to complete coding as quickly as possible; the testers' primary goal was to ensure the software was of high quality. If these groups had developed shared mental models across the team, not only would there be a greater understanding of the differing goals across these groups, but also these groups could work together to address other issues in this project, such as limited software development standards, negative interactions between the groups, and delayed communication. These types of misunderstandings that can negatively impact the outcome of a software development process can be reconciled by using agile practices to develop shared mental models within a team.

While research suggests that shared mental models improves software development team performance (even among agile teams), there is still a black box regarding how agile practices aid in the creation of shared mental models. Further, prior research has neglected the multi-dimensional nature of shared mental models [18]. This study opens that black box by offering an explanation of how agile practices develop shared mental models, and more specifically, identifies how agile practices create shared taskwork and teamwork mental models with an agile software development team.

By applying shared mental models theory, the value of agile practices is clarified by examining how shared understanding among the team is increased by using the agile practice. By explaining *why shared understanding is important, what exactly needs to be shared, and how such shared understanding can be obtained*, shared mental models theory provides the rationale for each agile practice. The application of shared mental models theory re-conceptualizes the understanding of agile practices, which explains the value of each practice in enhancing collaboration as well as offers guidance in the operationalization of agile practices.

The theory of shared mental models explains that the shared mental models about teamwork and taskwork enable teams to adapt to changing task conditions when free communication among team members is difficult [42]. A software development team consists of members with diverse skills who accomplish independent tasks. Studies found that not all teams' develop a shared understanding about teamwork and taskwork over time [43]. Interventions, or practices, are needed to formalize the creation of a shared understanding to create an internal knowledge base for the team. Through the development of shared mental models, teams form compatible expectations of the task and team. These shared mental models help the team to understand current events, consider what may happen in the near future for the team, and understand why events happen [18]. Therefore, to act appropriately when customers change requirements, the agile team needs to develop a shared mental model, and agile practices have the ability to help the team achieve this goal.

Shared mental models are formed through the interdependencies associated with the tasks performed within the team. As each team member creates an individual mental model, interactions with the others on the team help the team members adjust their individual mental models to create compatible mental models that enable team members to form mutual expectations of the task and the team [42].

According to the theory of shared mental models, agile practices should help the software development teams to build two types of shared mental models: the *taskwork* mental model and the *teamwork* mental model. Agile practices should facilitate the software development teams' communication and coordination towards building a shared task goal and appropriate strategies to accomplish that goal. Many software development teams fail because not all team members have the same understanding about the product outcome [43]. The theory of shared mental models also suggests the importance of building a teamwork mental model among the team. Effective use of agile practices will improve the quality of team communication and coordination to ensure that team members have a clear understanding of each member's roles and skills. By developing a teamwork mental model, members will also develop consistent expectations related to the communication patterns and the communication channels used by the team.

## 3. Analysis of agile practices

To analyze agile practices in terms of shared mental models theory, the authors identified multiple agile practices from XP and Scrum in terms of a description, stated benefits from practice, and challenges of implementation. The authors also compiled a separate list of shared mental model practices from literature based on the shared mental model's practice description, intended benefits of the practice, and empirical findings related to the use of the shared mental models practice (see Appendix A for a list of shared mental model practices).

The authors read the descriptions of agile practices and identified which shared mental model practices could explain the value or benefit of the agile practice. The authors pre-determined that there should be no effort to "force" a shared mental model practice on any agile practice. Therefore, if no shared mental model practice was relevant, the authors agreed that there should not be any attempt to require that every agile practice have one or more corresponding shared mental models practice(s). For each agile practice, one author would propose one or more potential shared mental models practices that explain the value of the practice. Another author would serve as a foil to question and challenge the explanation.

Since the purpose of this research is to explain how theory can be used to explain how agile practices create value in a software development effort, we chose to examine three agile practices in depth in the main body of the paper. Therefore, this section examines three agile practices from two popular agile software development methodologies, Xtreme Programming (XP) and Scrum: *system metaphor, stand-up meeting,* and *on-site customer*. These agile practices vary in terms of their use within the development lifecycle and in their perceived difficulty of implementation. Some agile practices, like system metaphor, are often perceived as difficult to implement because of its abstract definition [44]. Other agile practices are adopted by many organizations, such as daily stand-up meetings, are relatively easy to implement. Agile practices, such as the on-site customer, are frequently modified because of constraints (e.g., customer is available for questions, but not present on-site). Brief descriptions of additional agile practices and how they could be related to shared mental models practices are provided in Appendix B to demonstrate that shared mental models theory can be applied to practices beyond the three discussed in detail in this section.

### 3.1. System metaphor

The system metaphor is an agile software development practice in the Xtreme Programming (XP) method that is employed at the beginning of the project to develop "a story that everyone – customers, programmers, and managers – can tell about how the system works" [45]. Though agile approaches have been criticized for a lack of architecture compared with plan-driven methods, the system metaphor practice enables agile software development teams to create a "cheap" architecture design which consists of the main components of the software and their interactions [45]. The system metaphor practice helps developers understand key concepts about the system design and provides an explanation of how the system will work in a simplistic way [46]. A good system metaphor should allow people to raise questions regarding the system they would otherwise not have identified or questioned [44,47].

Although there are stated benefits of using the system metaphor agile practice, the value of this practice is not fully understood among agile software development teams. The system metaphor is a commonly dropped agile practice [46] because practitioners are not clear about how to operationalize this practice [11,44].

Furthermore, some argue that there are more dangers to using a system metaphor than benefits. For example, if the chosen metaphor is too weak to stimulate valuable discussion regarding to the design of the system architecture or if the metaphor misleads or limits the developers' understandings of the system functionalities, the time spent developing the metaphor could be considered as counterproductive [11].

From the lens of shared mental models theory, when an agile development team uses the system metaphor practice, they are developing a shared mental model by naturally employing the shared mental models practice of planning. *Planning* is an intervention used in the early stages of team development to enhance the shared understanding of the task and the team [20]. The planning practice to develop shared mental models encourages teams to discuss on team goals, team roles, and how the team can react to unexpected events.

Teams develop greater shared mental models when engaged in more effective planning behaviors [20]. Teams that made plans in low-workload periods developed greater shared understanding of one another's informational requirements for future high-workload periods [20]. By engaging in planning behaviors, teams are more likely to contribute their unique information and knowledge to the team. Therefore, the planning practice enables each team member to not only have a clear understanding of the task goals and the steps needed to accomplish the goals, but also helps the team identify which information is known only by specific team members [20,48].

The system metaphor practice is consistent with the shared mental models planning practice. The system metaphor practice encourages agile teams to create an open environment and use metaphors or stories to develop shared understandings regarding system goals, key concepts, major system functionalities, and roles and expertise of the agile team members. When viewing the system metaphor agile practice in the context of the shared mental model practice of planning, one can identify that the system metaphor is used as the same role as the shared mental model planning practice for the development of teams' shared mental models. Referring to the four cognitive stages of developing shared mental models (i.e. knowing, learning, understanding, and executing), the system metaphor practice helps agile team developers be more effective in the knowing and learning stages. Specifically, the system metaphor practice allows agile teams to identify unshared information, to enable developers to know the system's key concepts and functionalities, and to quickly communicate with customers the system requirements using common language. Further, the system metaphor practice improves the team's *taskwork* mental model in terms of both similarity and accuracy in that developers and users will develop shared and correct understandings of the goals and the scope of the system. The system metaphor practice also develops the team's *teamwork* mental models for similarity and accuracy in that developers and customers will create a shared and correct understanding of one another's roles, responsibilities, and technical backgrounds as well as develop a pattern of team communication and coordination.

Integrating shared mental models practices with the system metaphor agile practice offers two benefits. First, it provides a systematic approach for agile teams to generate system metaphors by ensuring the important aspects relevant to the task and the team are discussed among the users and developers. Second, it enhances the communication effectiveness within the development team by enabling more correct anticipations of one another's informational requirements [18,20]. Anticipating informational requirements of other team members is especially relevant and important in agile teams because of the presence of cross-functional members that have differing technical specialties (i.e., Java, database management, security, among others). When applying the planning practice to the system metaphor practice, each development team member is now aware of one another's specialization. This can improve time and resource allocation for more efficient software development. This practice allows team members to understand the needs and interests across the development team to potentially save time in testing and refactoring code, which is necessary due to the rapid iterations in agile software development.

## 3.2. Stand-up meeting

The stand-up meeting is one of the most basic and most frequently used Scrum practices [49]. Stand-up meetings are conducted daily and are short meetings, usually no more than 15 min. In this meeting, the entire team (developers, scrum masters, and on-site customers) discuss the completed work, identify current bottlenecks or dependencies, and talk about next steps [6]. Pre-planned questions often guide the meeting and are centered on sharing the team's progress towards completing the current tasks within the project [50,51]. Stand-up meetings offer many benefits, such as daily monitoring and control of the project's progress [52]. Researchers have also suggested that stand-up meetings are helpful for building trust among agile team members [53,54].

However, research and practice has acknowledged that the stand-up meeting practice may suffer from challenges such as extended meetings [53], inactive participants [49,54], interference due to interruptions [51], or inappropriate information shared during the meeting [51]. Furthermore, while researchers and managers suggest that stand-up meetings help in developing trust within the team, developers tend to disagree with this view and believe that distrust can arise as a result of the meetings [54]. Stand up meetings have also encouraged stronger levels of commitment to failing courses of action within a project [52].

The stand-up meeting can aid in the creation of a shared mental model within the team. Using shared mental models theory as a lens, the stand-up meeting agile practice provides an opportunity to increase teams' shared understandings about taskwork through daily monitoring and control of the project's progress [52]. The stand-up meeting agile practice also provides a chance for teams to learn more about the team, such as the other team members' progress, roles, and skills. The daily stand-up meeting incorporates the shared mental models practices of *leader briefings* and *reflexivity*.

*Leader briefings* are a form of leader communication within teams. Leader briefings should include: (1) statement of the goals for the task, (2) identification of significant risks and how to address them, (3) specification of opportunities, and (4) prioritization of actions [21]. Supporting the need for short meetings, research has identified that the effectiveness of leader briefings is based on the content, not the length of the meeting [55]. Effective leader briefings conducted prior to the execution of a task enhance the similarity and accuracy on the members' mental models and increase the team's ability to adapt to changing task demands [21,55,56].

A second related shared mental models practice is *reflexivity*, which is "the extent to which group members overtly reflect upon the group's objectives, strategies, and processes and adapt them to current or anticipated endogenous or environmental circumstances" [57]. Reflexivity can be stimulated within a team by asking members to reflect on their performance to date, to consider potential improvements, to identify how to implement the improvements, and finally, to implement the new improvements [58]. Using the reflexivity shared mental models practice enhances the similarity of teams' interaction models through developing a shared understanding with regard to the role of the leader in coordinating the team [58].

In the context of the shared mental models practices of leader briefing and reflexivity, the stand-up meeting agile practice improves similarity and accuracy of the taskwork mental model and teamwork mental model. Referring to the four stages of shared mental models development (i.e. knowing, learning, understanding, and executing), the stand-up meeting plays significant roles in the last two stages: understanding, and executing. In the stand-up meeting, under the facilitation of the scrum master, important updates and progress relating to the agile project are shared among the agile team. With effective reflective discussion during the stand-up meeting, the agile teams' understanding about the project goals, challenges, limitations, opportunities, and system requirements are consolidated and enhanced to the extent that agile team members freely and openly share their opinions. The stand-up meeting helps to identify cognitive differences and goals between the scrum master and the stand-up meeting participants. Specifically, the stand-up meeting improves the taskwork mental model's similarity in that the team should have shared knowledge of the task priorities and therefore only discusses the tasks that have the highest priorities. The stand-up meeting also increases the taskwork mental model's accuracy because developers can offer suggestions on how to resolve obstacles within the project.

Stand-up meetings that follow the agile practice will incorporate the shared mental model practice of leader briefings, which allows for additional discussion related to identifying potential solutions. Through these interactions, team members develop shared understandings of the consequences of the current approach and future steps. Should the consequence be manifested, the team is more likely to be able to respond quickly to the situation. Finally, through stimulating reflexivity within team through the stand-up meeting, team members will have an opportunity to develop introspective practices to better adapt and address changing circumstances.

### *3.3. On-site customer*

The on-site customer agile practice states that a customer should be present with the development team on a full time basis [45]. Specifically, this practice requires both the developers and customers to interact daily [59,60]. The on-site customers should spend most of their time participating in planning game sessions, acceptance testing, and retrospective sessions [61]. In addition, the on-site customer may participate in the daily stand-up meeting. The role of the customer is to specify functionality, prioritize requirements, perform acceptance testing for each release, and make final business decisions [61]. Prior studies have suggested on-site customer is one of the most beneficial agile practices to the team performance [62,63]. Benefits of adopting on-site customers include increased communication and collaboration between development teams and customers and ensuring customer acceptance of the final software product.

Though the ideal way of implementing the on-site customers is to have the customer physically present full-time with developers, modifications are made with the on-site customer agile practice when the on-site customers are too busy to participate fully in the agile project. For example, agile teams may chose "substitutes" for the customer by obtaining information from the project manager, salespersons, or even the programmers themselves [11,63]. Sometimes teams partially adopt this practice by exchanging emails extensively with customers and inviting customers for weekly face-to-face meetings [47].

From the lens of shared mental models theory, our understanding about values of on-site customers can be enhanced. Specifically, on-site customers help agile teams develop greater taskwork mental models by implicitly implementing the shared mental models practice, cross-training. Cross-training is "an instructional strategy in which each team member is trained in the duties of his or her teammates" [64, p. 87] with the intent to increase the accuracy and similarity of the team's shared mental models. The depth of cross training can vary based on the needs of the team [65]. For example, <u>positional clarification</u> is simply a meeting to verbally discuss each team members' job; <u>positional modeling</u> adds observation as another way of learning about a team member's job, which could be direct observation or watching videos of different daily activities; and <u>positional rotation</u> provides team members with first-hand experience of other members' duties. Research has found that with effective cross-training, team members develop greater taskwork mental models and are able to compensate for teammates' limitations. Moreover, empirical studies show that teams that were given cross-training showed greater adaptability by anticipating the other teammates' informational requirements accurately and collaborating with teammates smoothly [34,65].

Similar to the shared mental models practice of cross-training, the on-site customer agile practice improves the development of shared mental models, specifically, the taskwork mental model. The on-site customer agile practice offers developers a greater opportunity to learn the needs of the customers. Referring to the four stages of shared mental models development, the on-site customer agile practice enhances agile teams' understanding and executing stages. Through quality, frequent, and various types of communication with the customers, developers have more occasions to identify if the system's functionality meets the customers' needs and to ask questions about the system being developed. Thus, the agile team developers acquire a shared and accurate understanding of the task which enables the team to execute tasks efficiently. Implementing the on-site customer practice help developers of the agile teams build similar and accurate taskwork mental models (e.g., have accurate and shared understanding on the project goals and customer needs).

Integrating the cross-training shared mental models practice with the on-site customer agile practice offer two benefits. First, it offers a guideline for agile team developers when communicating with the on-site customers. This approach allows the developer to obtain first-hand knowledge about the customer's organization and day-to-day activities. This cross-training between the customer and developer helps each party anticipate and respond to changing customer requirements. This approach proactively enhances the developer's adaptability to user's changing requirements when on-site customers are not available or if "substitute" customers are being used in the project. Second, integrating the cross-training shared mental models practice offers alternative approaches to learn about the customers' needs. Developers have the option to use all three types of cross-training methods to learn more about the customer. Positional clarification about the customer's activities may be presented during the system metaphor practice. Positional modeling could be incorporated via videos demonstrating customers' activities during planning sessions. Positional rotation could be achieved during a one-day (or longer) visit by the developer with the customer.

## 4. Discussion

Agile software development methods have rapidly become a mainstream software development methodology as organizations search for different methods to improve software development outcomes. Trends examining software development methods show that agile practices tend to be adapted to the context of the workplace as organizations adopt more agile-like software development practices. However, agile practices are often undervalued, in part due a lack of theoretical foundations. This paper offers a new viewpoint of agile software development methods by focusing on the agile team's cognitive approach in developing a shared understanding of the task and team interaction using shared mental models theory. We have described three agile practices and demonstrated how these agile practices enable collaboration among software development teams and customers in the context of shared mental models theory. Specifically, we identified the shared mental models practices that agile practices employ or integrate to develop taskwork mental models and/or teamwork mental models' similarity and accuracy.

Prior empirical research has examined agile practices and explained the role of these practices in agile team outcomes [66–70]. Using the lens of shared mental models theory, we explore how three agile practices, system metaphor, stand-up meeting, and on-site customer, create shared understanding as part of the software development effort. Prior research supports our assertions that shared mental models can be a useful theoretical lens to understand the value of agile practices.

For example, the iterative use of the system metaphor practice elicits users' requirements by allowing users to focus on describing the business complexities without concerning themselves with the technical challenges [28]. In addition, with short and abstract descriptions in each system metaphor, project managers can estimate project costs and make schedules properly [69]. Fruhling and Vreede [47] noted that "user/developer-shared scenarios (system metaphors) gave the developers and users a common understanding of how the system should function" [p. 52]. These studies are consistent with the idea that the system metaphor practice enables developers and users to develope a shared taskwork mental model for the software development effort.

Daily stand-up meetings were deemed as crucial by agile team managers to delivering a quality software product through facilitating and structuring the communication in agile teams [28,69,71]. Stand-up meetings are considered an important practice for developing trust and enhancing agile teams' motivation [66,67]. The practice of stand-up meetings was found as the best agile practice in a global agile teams because it enabled transparency between sites at different countries [49]. Through stand-up meetings, agile team members developed understanding not only about the progress on the projects, but also learned about the different technical knowledge and the mutual knowledge of who knows what [28]. Engaging in daily stand-up meetings enable agile team members regularly reflecting on their activities [72]. To take full advantage of daily stand-up meetings, open environment was found to be effective in creating a sense of "togetherness" among agile team members, which in turn enhance the effective collaboration in agile teams [73]. These studies are consistent with the earlier explanation of how both teamwork mental models and taskwork mental models can be enhanced through the adoption of the stand-up meeting agile practice.

On-site customers have been deemed as critical to improving customer satisfaction [69]. When it is not feasible to have an on-site customer, high engagement between the customer and development team lead to a shared understanding of the functionality that the users require from the system [47]. Frequent communication with customers can reduce conflict that occurs due to diversity within the agile team and increases the opportunity for agile teams to manage change successfully [73,74]. These prior studies that focus on the role of the on-site customer aid in the development of a strong taskwork mental model.

Prior empirical studies often employ a bottom-up approach to identify important agile practices and understand the impact these agile practices have on project outcomes (i.e., team communication, trust, motivation and de-motivation) [66–70]. Alternatively, this article demonstrates how agile practices can be understood and valued using a theoretical lens. Specifically, our analysis identified how shared mental models practices explain how agile practices develop shared mental models to improve collaboration and reduce misunderstanding within software development teams. Since prior research has found a relationship between shared mental models and software development team performance [37–40], this article explains how agile teams can obtain shared mental models through the implementation of agile practices.

By explaining how agile practices help software development teams progress through the four stages of shared mental model development, knowing, learning, understanding, and executing, the value of using agile practices for improving collaboration within the team and among the developers and customers is identified. Further, by discussing the role of agile practices improve the development of taskwork mental models and/or teamwork mental models in terms of similarity and accuracy demonstrate the importance of using agile practices as intended.

In contrast to agile practices that are proposed by practitioners based on work experience, shared mental models practices are developed based on team cognition and organizational behavior theory as well as empirical evidence. The effectiveness of shared mental models practices in facilitating the development of high quality of shared mental models (i.e., similarity and accuracy) has been tested and verified in the academic literature. Therefore, examining agile practices in the context of shared mental models practices provides an exemplar of how to understand the value of agile practices using theory.

While there are numerous agile practices across various agile software development methodologies, we specifically examined three practices that are well-known from two different agile methodologies. The three agile practices discussed in this manuscript range from practices that are difficult to understand and implement, such as system metaphor, to practices that are easier to understand and apply, such as stand-up meetings. The variation in these practices demonstrates the capabilities of the shared mental model theoretical lens to explain the value of each of these agile practices within their respective agile method.

### *4.1. Future research*

One option for future research is to examine other agile practices using the lens of shared mental models theory to develop additional recommendations for agile practices. Appendix B demonstrates that other agile practices can be understood from the lens of shared mental models theory and practices. Further research could also apply different theoretical lenses to agile software development methodologies or specific practices to identify how each practice creates value to the organization, customer, and software development team. Another future research direction can be focused on optimizing the implementation of agile practices by explicitly integrating shared mental models practices in a software development project. Although the effectiveness of agile practices in small development teams are acknowledged, the implementation of agile methods in large-scale development teams is perceived as challenging [75]. Considering the theoretical base of shared mental models practices and the specifications of the operationalization of the shared mental models practices, shared mental models practices offer opportunities for agile practitioners to improve the effectiveness of agile practices. To successfully integrate shared mental models practices into agile practices, we suggest the following steps. First, one should consider the agile practice, the challenges of implementing the agile practice, and the content of the shared mental model that is most important for the agile practice. Contents of the shared mental model that should be considered are taskwork mental model accuracy, taskwork mental model similarity, teamwork mental model accuracy, and teamwork mental model similarity. Clearly identifying the cognitive dimension to be improved will deepen one's understanding of the agile practice. Second, as the rationale of each practice is understood, practices developed to create shared mental models can be adapted and applied to the agile context.

A final future research direction is the study of using information technology to support the development of shared mental models in distributed (or virtual) agile teams. Prior research has shown that information technology reduces communication cost within teams and enhances team collaboration [76,77]. Distributed agile teams must rely on information technology for communication as opposed to face-to-face interaction [49,78,79]. Yet, challenges arise when implementing agile practices in distributed teams. Delayed communication [41] and the potential for cultural differences [49] introduce the possibility of conflict in distributed agile teams. The use of the collaborative technology can aid in overcoming these challenges in distributed agile teams [78]. Therefore, future research could explore the how collaborative technology used to support agile practices in distributed agile teams supports the development of shared mental models.

## 5. Conclusion

Through this analysis of several agile practices, we demonstrate that shared mental models is a theoretical lens that is not dependent upon a specific agile software development methodology, but one that can be applied across agile methods and practices. We also specifically chose depth, rather than breadth in this conceptual paper. While we could explain how shared mental models explains all potential benefits of each agile practice in one or more agile methods, we purposefully wanted to demonstrate the application of theory to specific agile practices for a specific benefit – enabling better collaboration within the team. Therefore, this paper is an exemplar of how theory could be applied to the understanding and use of agile practices, rather than a comprehensive discussion of how shared mental models theory applies to each agile practice.

Agile software development methods are criticized for its lack of theoretical foundation. However, this research demonstrates how to apply a theory, shared mental models, to understand the value of agile practices in developing higher levels of collaboration in a software development effort. By leveraging research of shared mental models practices, our analysis demonstrated how specific agile practices facilitate the four stages of shared mental model development (knowing, learning, understanding, executing). This research highlights how agile practices contribute to building two important types of shared mental models within a team: teamwork mental models and taskwork mental models. For researchers and practitioners, the application of shared mental models theory to agile software development methods demonstrates how agile practices provide value to the software development team.

## Acknowledgements

We thank the associate editor and two anonymous reviewers for their insightful comments that aided in the development of this article.

## Appendix A. Review of shared mental models practices

The following section presents an adapted list of shared mental models practices [18].

| Shared mental models practice | Description | Benefits |
|---|---|---|
| Planning [20] | A practice implemented in the beginning of a team project. A team leader or a facilitator guides the team in clarifying team goals and team assignments. This practice also requires the team to discuss on possible actions when unexpected events happen | • Increases the similarity of information shared among team members<br>• Develops a more efficient communication strategy to prepare the team when the team's workload increases |
| Reflexivity [58] | A practice that requires teams to review tasks accomplished and to reflect on the process of accomplishing the tasks in terms of how well the task has been done and if there are possibilities to improve the task performance | • Increases the similarity of teams' interaction models<br>• Creates a more efficient team interaction strategy by developing a shared understanding of the role of the leader in coordinating the team |
| Leader briefing [21] | A practice used by team leaders when he/she announces the task goals and priorities of the steps to accomplish the tasks | • Helps team members translate the advantages of heterogeneity, such as the variety of professional backgrounds, knowledge, skills, and abilities, into significant processes of questioning, reviewing, and exploring |
| Team-interaction training [21] | A practice used for training task information embedded in the necessary teamwork skills for effective team task execution. | • Increases similarity of team interaction mental models<br>• Improves similarity of the team members' knowledge organization |

**Appendix A** (*continued*)

| Shared mental models practice | Description | Benefits |
|---|---|---|
| Self-correction training [22] | A practice that requires each team member reflects on their accomplished tasks. The reflections are guided by a facilitator. | • Develops a more accurate mental models of teamwork and greater shared task expectation |
| Cross-training [80,81] | Consists of three practices (i.e., positional clarification, positional modeling, and positional rotation) that aim to enable the trainees a shared feeling of their teammates' work. | • Obtains accurate and shared taskwork and teamwork models |
| | | • Acquires more accurate team-interaction models |

## Appendix B. Review of additional agile practices

This appendix provides a brief explanation of the benefits of other agile practices in the context of shared mental models. For each agile practice, a description of the agile practice is identified along with any respective shared mental model practices, as well as how shared mental models would be developed in the team when the agile practice is leveraged.

| Agile practice (agile method) | Description | Shared mental models practice | Four Stages to building shared mental models | Specific type of shared mental models |
|---|---|---|---|---|
| Product backlog (Scrum) | List of prioritized requirement items provided by the product owner[82]. As new requirements are identified by the product owner, these are added to the product backlog | Planning, leader briefing | Enhances the knowing stage by clearly presenting all requirements to the team. Facilitates the information sharing process among the team | Establishes shared accurate taskwork mental models about the task requirements |
| Effort estimation (Scrum) | Estimate the most likely effort to take for the product [83]. | Self-correction | Occurs in the understanding stage when teams integrate knowledge from customers and developers to make estimations on the effort required for the tasks in the product backlog | Accurate effort estimation leads to accurate shared taskwork mental model on how much amount of time and effort the project will take |
| Sprints (Scrum) | Rapid series of iterations that deliver a working product at its conclusion; each sprint lasts only 3–5 weeks [82]. | Team-interaction training | Influences all four stages of building shared mental models by iteratively asking team members to know, learn, understand, and execute on the project | Enhances the similarity and accuracy of the taskwork mental models on the task requirements |
| | | | | Improves the teamwork mental models because teams are able to learn how to work through successes and failures quickly through a series of rapid development cycles |
| Sprint planning meeting (Scrum) | Meeting that plans only for the next sprint (3–5 weeks of work) based on the requirements in the product backlog[82] | Planning | Enhances the development of learning and understanding stages by facilitating information sharing and integration | Improves the teamwork mental models through structure and forced communication |
| Sprint retrospectives (Scrum) | At the conclusion of each sprint, the team identifies and discusses challenges and opportunities within last sprint[84] | Self-correction training, reflexivity | Enhances the development of learning and understanding stages by facilitating the information sharing and integration. This practice also improves the teams' executing capability in the next sprint | Improves both the taskwork mental models and teamwork mental model accuracy |

## References

[1] T. Dingsøyr, S. Nerur, V. Balijepally, N.B. Moe, A decade of agile methodologies: towards explaining agile software development, J. Syst. Softw. 85 (2012) 1213–1221.
[2] L. Vijayasarathy, D. Turk, Drivers of agile software development use: dialectic interplay between benefits and hindrances, Inf. Softw. Technol. 54 (2012) 137–148.
[3] T. Dyba, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (2008) 833–859.
[4] J.B. Barlow, J.S. Giboney, M.J. Keith, D.W. Wilson, R.M. Schuetzler, Overview and guidance on agile development in large organizations, Communications of the Association for Information Systems, vol. 29, 2011 (Article 2).
[5] R. McCauley, Agile development methods poised to upset status quo, SIGCSE Bull. 33 (2001) 14–15.
[6] K. Conboy, Agility from first principles: reconstructing the concept of agility in information systems development, Inf. Syst. Res. 20 (2009) 329–354.
[7] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, Agile Software Development Methods: Review and Analysis, VTT Publications, 2002.
[8] J. Highsmith, A. Cockburn, Agile software development: the business of innovation, Computer 34 (2001) 120–122.
[9] R.L. Glass, The Standish report: does it really describe a software crisis?, Commun ACM 49 (2006) 15–16.
[10] S. Group, 2011 CHAOS Report, 2011.
[11] B. Rumpe, A. Schröder, Quantitative Survey on Extreme Programming Projects, Proceedings of XP2002, 2002.
[12] T. Murphy, D. Norton, in: G. Group (Ed.), Predicts 2010: Agile and Cloud Impact Application Development Directions, 2010.
[13] V.G. Stray, N.B. Moe, T. Dingsøyr, Challenges to teamwork: a multiple case study of two agile teams, in: Agile Processes in Software Engineering and Extreme Programming, Springer, 2011, pp. 146–161.
[14] S. Ambler, in: A. Inc. (Ed.), How Agile Are You? 2010 Survey Results, 2010.
[15] J.A. Cannon-Bowers, E. Salas, Shared mental models in expert decision making, in: N.J. Castellan (Ed.), Individual and Group Decision Making, Lawrence Erlbaum Associates, Hillsdale, NJ, 1993, pp. 221–246.
[16] P.N. Johnson-Laird, Mental models in cognitive science, Cognitive Sci. 4 (1980) 71–115.
[17] P.N. Johnson-Laird, Mental models: towards a cognitive science of language, inference and consciousness, 1986.
[18] S. Mohammed, L. Ferzandi, K. Hamilton, Metaphor no more: a 15-year review of the team mental model construct, J. Manage. 36 (2010) 876–910.
[19] J.R. Rentsch, D.J. Woehr, Quantifying congruence in cognition: social relations modeling and team member schema similarity, in: E. Salas, S.M. Fiore (Eds.), Team Cognition: Understanding the Factors that Drive Process and Performance, American Psychological Association, Washington, DC, 2004, pp. 11–31.
[20] R.J. Stout, J.A. Cannon-Bowers, E. Salas, D.M. Milanovich, Planning, shared mental models, and coordinated performance. an empirical link is established, Hum. Factors 41 (1999) 61–71.
[21] M.A. Marks, S.J. Zaccaro, J.E. Mathieu, Performance implications of leader briefings and team-interaction training for team adaptation to novel environments, J. Appl. Psychol. 85 (2000) 971–986.
[22] K.A. Smith-Jentsch, J.A. Cannon-Bowers, S.I. Tannenbaum, E. Salas, Guided team self-correction: impacts on team mental models, processes, and effectiveness, Small Group Res. 39 (2008) 303–327.
[23] S.A. McComb, Mental model convergence: the shift from being an individual to being a team member, in: F. Dansereau, F.J. Yammarino (Eds.), Multi-Level Issues in Organizations and Time, Emerald Group Publishing, 2007, pp. 95–147.
[24] N. Warner, M. Letsky, M. Cowen, Cognitive model of team collaboration: macro-cognitive focus, in: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, Sage Publications, 2005, pp. 269–273.
[25] P. Van den Bossche, W. Gijselaers, M. Segers, G. Woltjer, P. Kirschner, Team learning: building shared mental models, Instr. Sci. 39 (2011) 283–301.
[26] J.R. Larson, P.G. Foster-Fishman, C.B. Keys, Discussion of shared and unshared information in decision-making groups, J. Pers. Soc. Psychol. 67 (1994) 446.
[27] J.R. Austin, Transactive memory in organizational groups: the effects of content, consensus, specialization, and accuracy on group performance, J. Appl. Psychol. 88 (2003) 866.
[28] R. Vidgen, X. Wang, Coevolving systems and the organization of agile software development, Inf. Syst. Res. 20 (2009) 355–376.
[29] W.B. Rouse, J.A. Cannon-Bowers, E. Salas, The role of mental models in team performance in complex systems, IEEE Trans. Syst., Man, Cybern. 22 (1992) 1296–1308.
[30] B.C. Lim, K.J. Klein, Team mental models and team performance: a field study of the effects of team mental model similarity and accuracy, J. Organ. Behav. 27 (2006) 403–418.
[31] D.L. Kleinman, D. Serfaty, Team performance assessment decision making, in: G.e. al. (Ed.), Proceedings of the Symposium on Interactive Networked Simulation for Training, University of Central Florida, Orlando, FL, 1989, pp. 22–27.
[32] B. Choi, A.M. Jong, Assessing the impact of knowledge management strategies announcements on the market value of firms, Inf. Manage. 47 (2010) 42–52.
[33] Y. Zhang, The impact of task complexity on people's mental models of MedlinePlus, Inf. Process. Manage. 48 (2012) 107–119.
[34] R. Espevik, B.H. Johnsen, J. Eid, Outcomes of shared mental models of team members in cross training and high-intensity simulations, J. Cognitive Eng. Decis. Making 5 (2011) 352–377.
[35] X. Fan, J. Yen, Modeling cognitive loads for evolving shared mental models in human-agent collaboration, IEEE Trans. Syst., Man, Cybern., Part B: Cybern. 41 (2011) 354–367.
[36] I.-H. Jo, Shared mental models on the performance of e-Learning content development teams, Educ. Technol. Soc. 15 (2012) 289–297.
[37] P.M. Bach, J.M. Carroll, Characterizing the Dynamics of Open User Experience Design: The Cases of Firefox and OpenOffice. org, JAIS Special Issue on Empirical Research on Free/Libre Open Source Software Part 2, vol. 11, 2010, pp. 902–925.
[38] J.A. Espinosa, R.E. Kraut, F.J. Lerch, S. Slaughter, J.D. Herbsleb, A. Mockus, Shared Mental Models and Coordination in Large-Scale, Distributed Software Development, ICIS, 2001, pp. 513–518.
[39] J.S. Hsu, J.Y. Chang, G. Klein, J.J. Jiang, Exploring the impact of team mental models on information utilization and project performance in system development, Int. J. Project Manage. 29 (2011) 1–12.
[40] A. Rai, L.M. Maruping, V. Venkatesh, Offshore information systems project success: the role of social embeddedness and cultural characteristics, MIS Quart. 33 (2009) 617.
[41] X. Zhang, T.F. Stafford, J.S. Dhaliwal, M.L. Gillenson, G. Moeller, Sources of conflict between developers and testers in software development, Inf. Manage. 51 (2014) 13–26.
[42] J.A. Cannon-Bowers, E. Salas, S. Converse, Shared mental models in expert team decision making, in: N.J. Castellan (Ed.), Individual and Group Decision Making, Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.
[43] L.L. Levesque, J.M. Wilson, D.R. Wholey, Cognitive divergence and shared mental models in software development project teams, J. Organ. Behav. 22 (2001) 135–144.
[44] W.C. Wake, What is the system metaphor?, in: Extreme Programming Explored, Addison-Wesley, 2002
[45] K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley Professional, 1999.
[46] W.C. Wake, S.A. Wake, The system metaphor explored, in: Extreme Programming Perspectives, Addison-Wesley, 2003.
[47] A. Fruhling, G.-J.D. Vreede, Field experiences with eXtreme programming: developing an emergency response system, J. Manage. Inf. Syst. 22 (2006) 39–68.
[48] J. Orasanu, Shared Mental Models and Crew Decision Making, DTIC Document, 1990.
[49] M. Paasivaara, S. Durasiewica, C. Lassenius, Using Scrum in a globally distributed project: a case study, Softw. Process Improv. Pract. 13 (2008) 527–554.
[50] K. Schwaber, M. Beedle, Agile Software Development with Scrum, Prentice Hall, 2001.
[51] J. Yip, Its not Just Standing up: Patterns of Daily Stand-up Meetings, 2006. <http://martinfowler.com/articles/itsNotJustStandingUp.html>.
[52] K. Dinakar, Agile development: overcoming a short-term focus in implementing best practices, in: Proceeding of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, 2009, pp. 579–588.
[53] A. Hands, It's Not Just Standing Up: Patterns for Daily Stand-up Meetings.
[54] E. Hasnain, T. Hall, Preliminary investigation of stand-up meetings in agile methods, Int. J. Comput. Sci. 35 (2009).
[55] G. Klein, A Script for the Commander's Intent Statement, Klein, Fairborn, OH, 1993.
[56] J.M. Orasanu, Shared mental models and crew performance, in: The Meetings of the Human Factors Society, Orlando, FL, 1990.
[57] M.A. West, Reflexivity and work group effectiveness: a conceptual integration, in: M.A. West (Ed.), Handbook of Work Group Psychology, John Wiley, Chichester, 1996, pp. 555–579.
[58] A. Gurtner, F. Tschan, N.K. Semmer, C. Nagele, Getting groups to develop good strategies: effects of reflexivity interventions on team process, team performance, and shared mental models, Organ. Behav. Hum. Decis. Process. 102 (2007).
[59] A. Martin, R. Biddle, J. Noble, An ideal customer: a grounded theory of requirements elicitation, communication and acceptance on agile projects, in: Agile Software Development, Springer, 2010, pp. 111–141.
[60] A. Martin, R. Biddle, J. Noble, The XP customer role in practice. Three studies, in: Agile Development Conference, 2004, IEEE, 2004, pp. 42–54.
[61] J. Koskela, P. Abrahamsson, On-site customer in an XP project: empirical results from a case study, in: T. Dingsøyr (Ed.), Software Process Improvement Proceedings, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2004, pp. 1–11.
[62] D. Karlstrom, Introducing eXtreme programming—an experience report, in: Proceedings of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, Sardinia, Italy, 2002, pp. 24–29.
[63] W.A. Wood, W.L. Kleb, Exploring XP for scientific research, IEEE Softw. 20 (2003) 30–36.
[64] C.e. Volpe, J.A. Cannon-Bowers, E. Salas, P.E. Spector, The impact of cross-training on team functioning: an empirical investigation, Hum. Factors 38 (1996).
[65] E. Blickensderfer, J.A. Cannon-Bowers, E. Salas, Cross-training and team performance, in: J.A. Cannon-Bowers, E. Salas (Eds.), Making Decisions under

Stress: Implications for Individual and Team Training, American Psychological Association, Washington, DC, 1998.
[66] O. McHugh, K. Conboy, M. Lang, Using agile practices to influence motivation within IT project teams, Scand. J. Inf. Syst. 23 (2011) 85–110.
[67] O. McHugh, K. Conboy, M. Lang, Using agile practices to build trust in an agile team: a case study, in: Information Systems Development, Springer, 2011, pp. 503–516.
[68] B. Ramesh, K. Mohan, L. Cao, Ambidexterity in agile distributed development: an empirical investigation, Inf. Syst. Res. 23 (2012) 323–339.
[69] B. Ramesh, L. Cao, R. Baskerville, Agile requirements engineering practices and challenges: an empirical study, Inf. Syst. J. 20 (2010) 449–480.
[70] J.H. Sharp, S.D. Ryan, Best practices for configuring globally distributed agile teams, J. Inf. Technol. Manag. 22 (2011) 56.
[71] N.B. Moe, A. Aurum, T. Dybå, Challenges of shared decision-making: a multiple case study of agile software development, Inf. Softw. Technol. 54 (2012) 853–865.
[72] L. Williams, What agile teams think of agile principles, Commun. ACM 55 (2012) 71–76.
[73] C. de O Melo, D.S. Cruzes, F. Kon, R. Conradi, Interpretative case studies on agile team productivity and management, Inf. Softw. Technol. 55 (2013) 412–427.
[74] G. Lee, W. Xia, Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility, MIS Quart. 34 (2010) 87–114.
[75] K. Petersen, C. Wohlin, A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case, J. Syst. Softw. 82 (2009) 1479–1490.
[76] G. DeSanctis, M.S. Poole, Capturing the complexity in advanced technology use: adaptive structuration theory, Organ. Sci. 5 (1994) 121–147.
[77] A. Davis, J. Murphy, D. Owens, D. Khazanchi, I. Zigurs, Avatars, people, and metaverses: foundations for research in metaverses, J. Assoc. Inf. Syst. 10 (2009) 99–117.
[78] J.S. Persson, L. Mathiassen, I. Aaen, Agile distributed software development: enacting control through media and context, Inf. Syst. J. 22 (2012) 411–433.
[79] L.M. Maruping, V. Venkatesh, R. Agarwal, A control theory perspective on agile methodology use and changing user requirements, Inf. Syst. Res. 20 (2009) 377–399.
[80] C.E. Volpe, J.A. Cannon-Bowers, E. Salas, P.E. Spector, The impact of cross-training on team functioning: an empirical investigation, Hum. Factors: J. Hum. Factors Ergon. Soc. 38 (1996) 87–100.
[81] M.A. Marks, M.J. Sabella, C.S. Burke, S.J. Zaccaro, The impact of cross-training on team effectiveness, J. Appl. Psychol. 87 (2002) 3–13.
[82] M. Paasivaara, S. Durasiewicz, C. Lassenius, Using scrum in a globally distributed project: a case study, Softw. Process: Improv. Pract. 13 (2008) 527–544.
[83] S. Grimstad, M. Jørgensen, K. Moløkken-Østvold, Software effort estimation terminology: the tower of Babel, Inf. Softw. Technol. 48 (2006) 302–310.
[84] N.B. Moe, T. Dingsøyr, T. Dybå, A teamwork model for understanding an agile team: a case study of a Scrum project, Inf. Softw. Technol. 52 (2010) 480–491.