Empirical Software Engineering (2022) 27:56
https://doi.org/10.1007/s10664-021-10115-0

A teamwork effectiveness model for agile software development

Diane Strode[1] · Torgeir Dingsøyr[2,3] · Yngve Lindsjorn[4]

Accepted: 29 December 2021/Published online: 10 March 2022

Abstract
Teamwork is crucial in software development, particularly in agile development teams which are cross-functional and where team members work intensively together to develop a cohesive software solution. Effective teamwork is not easy; prior studies indicate challenges with communication, learning, prioritization, and leadership. Nevertheless, there is much advice available for teams, from agile methods, practitioner literature, and general studies on teamwork to a growing body of empirical studies on teamwork in the specific context of agile software development. This article presents the agile teamwork effectiveness model (ATEM) for colocated agile development teams. The model is based on evidence from focus groups, case studies, and multi-vocal literature and is a revision of a general team effectiveness model. Our model of agile teamwork effectiveness is composed of shared leadership, team orientation, redundancy, adaptability, and peer feedback. Coordinating mechanisms are needed to facilitate these components. The coordinating mechanisms are shared mental models, communication, and mutual trust. We critically examine the model and discuss extensions for very small, multi-team, distributed, and safety-critical development contexts. The model is intended for researchers, team members, coaches, and leaders in the agile community.

Keywords Agile leadership · Agile methods · Agile teamwork model · Agile teams · Big five model of teamwork · Mutual performance monitoring · Peer feedback · Redundancy · Scrum teams · Teamwork model · Teamwork theory

Communicated by: Vittorio Cortellessa

✉ Diane Strode
diane.strode@whitireia.ac.nz

Torgeir Dingsøyr
torgeir.dingsoyr@ntnu.no

Yngve Lindsjorn
ynglin@ifi.uio.no

Extended author information available on the last page of the article

1 Introduction

Teamwork is crucial in software development. A recent book on teams in software development states that '*the speed, frequency, complexity, and diversity of changes needed for modern software-rich systems means that teams are essential*' (Skelton and Pais 2019, p. 44). Although teamwork is crucial, studies of software development teams report challenges with communication, coordination, learning, prioritizing work tasks, team orientation, and team leadership (Moe et al. 2010; Stray et al. 2011).

Agile software development methods are centred on teams (Baham and Hirschheim 2021; Chow and Cao 2008) that perform knowledge-intensive work (Tiwana 2004). Team members need expertise and problem-solving abilities to translate complex requirements into software solutions. Agile teams are cross-functional to cope with this situation, bringing together experts from different domains who work intensively together to produce a cohesive software system to solve a business or social problem (Project Management Institute and Agile Alliance 2017). In addition, agile teams aim to be empowered, autonomous, self-reflecting, and self-adjusting (Stray et al. 2018). These team characteristics differ from team structures where coaches and leaders are central controlling leadership figures (Hoda et al. 2013).

There is much advice available for agile development teams to help improve their effectiveness. This advice is available in development methods, ongoing discussions in the agile community, and published research on teamwork in general and agile teamwork in particular. However, it can be difficult for a team to navigate all this advice and select the best advice to apply to their situation when they want to become and remain effective. A whitepaper from scrum.org, for example (Overeem 2016), lists 25 characteristics of 'a great development team', including 'trust each other' and 'pursues technical excellence' to 'don't need a definition of done'.

Theoretical models of teams and teamwork are often general and are assumed to be relevant to all types of teams (e.g. (Hoegl and Gemuenden 2001; Mathieu et al. 2008; Salas et al. 2005)). They also tend to include more abstract factors such as 'communication', 'coordination', 'balance of member contribution', and 'cohesion' (Lindsjørn et al. 2016) and do not explain how effective teams should behave or what practices might support effective teams. One well-regarded general model that includes both abstract factors and specifies appropriate behaviours is the Salas et al. (2005) model of effective teamwork. The model specifies the core components of teamwork and the coordinating mechanisms needed for teamwork and describes behaviours that effective teams commonly display, termed behavioural markers by Salas et al. (2005). The specific relevance of these abstract factors, mechanisms, components, and behaviours in the context of agile software development teams is unclear although some studies of agile teamwork indicate that many of them might be relevant (Dingsøyr et al. 2016; Lindsjørn et al. 2016; Moe et al. 2010; Strode 2015).

There is no one comprehensive model that combines both abstract factors and explicit indicators of what constitutes effective agile teamwork. Therefore, this study develops a model of teamwork effectiveness for agile software development that is based on empirical studies and knowledge in the research and practitioner literature and is tailored specifically for the agile practice domain. To develop a comprehensive model of agile teamwork effectiveness, we address the following research question:

> What are the coordinating mechanisms, core components, and behavioural markers for effective teamwork in agile software development?

Section 3 explains how coordinating mechanisms, core components, and behavioural markers form a teamwork effectiveness model.

In addressing the research question, this study draws on the agile teamwork literature and findings from two sources of empirical evidence, a focus group study in Europe and case studies in New Zealand. Based on a review of existing team and teamwork effectiveness models, we propose a revision of the general teamwork effectiveness model, the Big Five model of effective teamwork (Salas et al. 2005), into the Agile Teamwork Effectiveness Model (ATEM).

The article is organized as follows. First, in Section 2, we define teamwork in general and for agile teams specifically. Second, we review the pertinent literature on teamwork and teamwork studies of agile software development. Section 3 describes our research design and the details of the two sources of empirical support, focus groups and case studies, and also describes the theory development. In Section 4, we synthesize the evidence on coordinating mechanisms and core components and revise the behavioural markers. In Section 5, we present the complete ATEM and then critically examine the model, discussing extensions to the core model to encompass very small teams, multi-team projects, distributed development, and safety-critical development. We present limitations and implications for theory and practice and suggest further work. Section 6 concludes by summarizing the contribution and highlighting contributions for various groups of users of this research.

Note: In the remainder of this article, we refer to the Salas et al. (2005) teamwork effectiveness model as the Salas Big Five model; we refer to agile software development teams as agile teams.

2 Teams, Teamwork, and Team Effectiveness

In this section, we explain the scope of our study, define agile teams and teamwork, define team effectiveness, and present three influential team effectiveness models. Finally, we summarize the research on teamwork in agile software development from academic and practitioner sources.

The scope of our study is single agile teams where the teams work full time to develop a single product; work independently within an organization, such as Team A in Fig. 1; and are colocated and making software that is not safety-critical. This is the context for which agile software development methods were originally intended (Williams and Cockburn 2003). We do not address dependencies with other teams (such as Teams B and C), overlapping teams (such as Teams D and E), or teams who are a part of a larger project with external resources (such as Team D). We also do not focus on knowledge sharing between teams through individual participants, such as in a community of practice used in many organizations, or on external roles, such as agile coaches.

The team can be organized as a project or as a standing team (a team that works continuously on one product) responsible for a product.

2.1 Teams and Teamwork

In organizational psychology, Kozlowski and Bell (2012, p. 6) define teams as

(a) composed of two or more individuals,

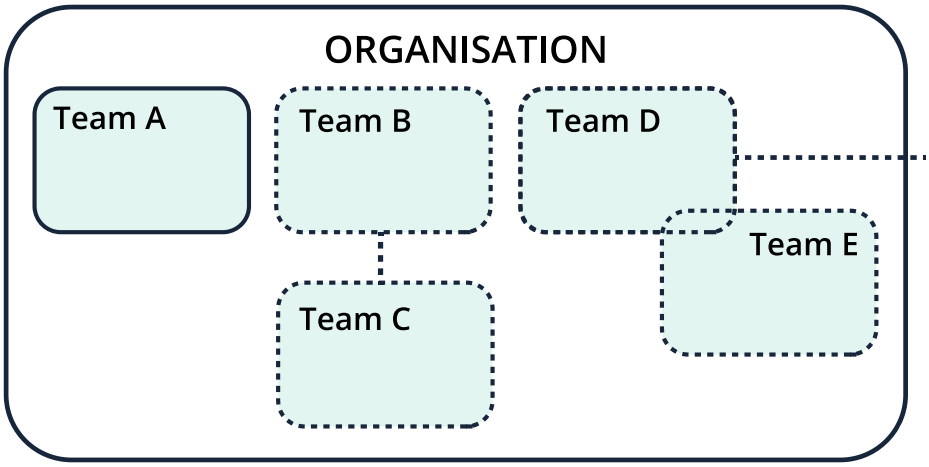

Fig. 1 Independent teams, such as Team A, are the focus of this study

(b) who exist to perform organizationally relevant tasks,
(c) share one or more common goals,
(d) interact socially,
(e) exhibit task interdependencies (i.e., workflow, goals, outcomes),
(f) maintain and manage boundaries, and
(g) are embedded in an organizational context that sets boundaries, constrains the team, and influences exchanges with other units in the broader entity.

These characteristics distinguish teams from groups of people working together. This definition is also used in an extensive review of teamwork and team effectiveness studies by Mathieu et al. (2008).

Agile teams meet each of the seven criteria, (a) to (g). In software development, a team size (a) of 4 to 6 members is described as the 'best size' in Sommerville's textbook on software engineering (Sommerville 2016). Small team size is supported by Rodriguez et al. (2012), who report software team sizes of 9 or less correlate with improved productivity based on a study of 195 software projects' data held in a repository by the International Software Benchmarking Standards Group. Moreover, the agile community typically recommends that teams have from 5 to 9 members (Project Management Institute and Agile Alliance 2017; Sutherland and Schwaber 2020). Small teams are recommended because they make team communication simpler and increase trust among team members.

Agile teams perform organisationally relevant tasks (b) that are usually described in a product backlog where teams either jointly plan work for an iteration or select high-priority work tasks if they use a method such as kanban (Kniberg 2011).

Agile teams have goals (c) and often define them at the iteration level. For example, in a scrum team, sprint goals are used to focus the team on appropriate task work during a sprint (Schwaber and Beedle 2002). The list of features to be included in a product ('product backlog' in scrum) decomposes the goal into work tasks defined and prioritized by a product owner. This list of features usually takes the form of user stories that the team interprets as a goal to achieve during an iteration of development (Cohn 2004).

In agile teams, the degree of social interaction (d) will depend on the development method. Methods with much 'ceremony' (method-specific activities) such as scrum involve more formal social interaction than, for example, a kanban team. Social interaction is supported in scrum ceremonies, including daily stand-up meetings, retrospectives, planning meetings, and product demonstrations (Schwaber and Beedle 2002). In addition, social interaction and information sharing are supported by colocating agile teams in a single workspace (Cockburn 2002).

For software teams, including agile software teams, there will typically be task interdependencies (e), some that can be foreseen when planning the work and some the team discovers during their work. Task dependencies are one of three forms of dependency identified in agile teams, along with knowledge and resource dependencies (Strode 2016). Task interdependencies between software development teams and other teams also occur. For example, a team developing features might depend on one or more teams maintaining a platform (Skelton and Pais 2019).

In agile teams, (f) maintaining and managing boundaries is usually allocated to a particular team member. In scrum, the team member who maintains and manages the boundaries between the team and the organization is the scrum master. The scrum master maintains team boundaries by protecting the team from interruptions from people external to the team (Shastri et al. 2021): '*The Scrum Master helps those outside the Scrum Team understand which of their interactions with the Scrum Team are helpful and which aren't*' (Schwaber and Sutherland 2017, p. 7).

As for organizational context (g), agile teams are typically embedded within organizations and are affected by organizational culture and structure (Iivari and Iivari 2011; Strode et al. 2009). Although agile teams are autonomous, this autonomy may be limited by organizational routines, tool choices, decisions on who are team members, and organizational decisions on the use of physical workspace (Moe et al. 2009a).

Agile teams fit the definition of Kozlowski and Bell (2012), but in management science, other definitions of team emphasize complementary skills. Team members have common performance goals and *'hold themselves mutually accountable'* (Katzenbach and Smith 2005). In software engineering, Sommerville states that '*putting together a group that has the right balance of technical skills, experience, and personalities is a critical management task*' (Sommerville 2016, p. 657). In agile development, the teams are expected to be 'cross-functional', which means that within the team, team members have all the skills needed to accomplish team tasks (Project Management Institute and Agile Alliance 2017). Regarding common performance goals, in practice, individuals in software teams often do not have the same performance goals, as can be seen in teams composed of people from several organizations with different incentive systems, such as in the Perform programme (Dingsøyr et al. 2018). Finally, regarding mutual accountability, in agile development, the team is committed to achieving tasks (Project Management Institute and Agile Alliance 2017), so, in principle, team members in an agile team hold each other mutually accountable.

This analysis of agile teams against recognized team characteristics shows that if agile teams live up to the expectations expressed in the methods, they are true teams, not just groups.

Agile teams employ specific practices during software development that are designed to facilitate effective teamwork. Teamwork is concerned with how a team works together (Crawford and LePine 2013; Salas et al. 2014), and encompasses shared behaviours, attitudes, and cognitions, whereas task work includes the specific tasks teams carry out to achieve their goals (Salas et al. 2014). Task work in agile teams includes the development tasks and may

also include agile practices. Teamwork and task work facilitate one another and are entwined (Crawford and LePine 2013).

2.2 Team Effectiveness and Teamwork Effectiveness Models

In this study, we develop a teamwork effectiveness model. Salas et al. (2005) proposed the Big Five teamwork model, distinguishing between team effectiveness and team performance. Team performance is defined as the outcome of a team's actions regardless of how they accomplish their task, which in a software development context can be meeting project goals, budget, and schedule as well as the quality of the software developed (Dingsøyr et al. 2016). In contrast, team effectiveness is defined more holistically (Salas et al. 2005) by including how the team interacts when accomplishing their task. In our context, such team interaction could be through meetings for planning, review, and retrospectives; pair programming; or artefacts for the coordination of work such as iteration backlogs and product backlogs. This broad definition of team effectiveness includes the team members' motivation to work together, often measured by job satisfaction. See Fagerholm et al. (2015) for further aspects of how software teams see performance, including factors such as communication, team spirit, and team identity.

Many theories, models, and frameworks focus on teams and teamwork. Hollenbeck et al. (2012) identified 42 different team types and more than 130 models and frameworks that explain team effectiveness (Burke et al. 2007). Three influential widely accepted models are commonly referred to in the general teamwork and agile team literature; the 'Big Five' model (Salas et al. 2005), the Teamwork Quality model (Hoegl and Gemuenden 2001), and the Input-Process-Output model (Mathieu et al. 2008).

The three models and examples of their use in studies of software development are described in Table 1. In Section 3, Research Design, we explain the Salas Big Five model in more detail and why we chose this model as a basis for the ATEM rather than either of the other two models.

2.3 Teamwork in Software Development and Agile Methods

Studies of software teams provide detailed models that relate aspects of teamwork quality to team effectiveness. In addition to the models presented in Table 1, we find models in Janz (1999), Moe et al. (2010), and Dingsøyr et al. (2016) and studies on project success and performance of agile teams (Drury-Grogan 2014; Schmidt 2016).

These studies demonstrate the importance of cooperative learning in project success for software development teams (Janz 1999), discuss the challenges with achieving team leadership and team orientation in a scrum development team (Moe et al. 2010), and identify five particularly important factors also identified in prior studies on software teams: team coordination, goal orientation, team cohesion, shared mental models, and team learning (Dingsøyr et al. 2016). In addition, coordination and teamwork/collaboration and process/design were found to be focus areas in a recent mapping study of research on software engineering teams (DeFranco and Laplante 2018).

The literature on agile teams appears in both academic and practitioner-based outlets. Tables 2 and 3 summarize key studies from each of these sources. Table 2 focuses on post-2000 empirical studies that specifically address aspects of teamwork in agile software development. The studies are mainly taken from DeFranco and Laplante (2018). We have excluded

Table 1 General team effectiveness models used in studies of software engineering teams

Model	Description	Source	Use in software development
Salas Big Five model	The 'Big Five' model has five components that lead to effective teamwork: team leadership, mutual performance monitoring, backup behaviour, adaptability, and team orientation. Each of the 'Big Five' is required for team performance, but each component may be manifested differently across teams because of constraints of team tasks and varying needs of the team. The 'Big Five' are facilitated by three coordinating mechanisms: shared mental models, closed-looped communication, and mutual trust. This framework is based on a literature review of 20 years of literature from 1985 to 2005 on teamwork, team effectiveness, and team performance. More than 20 studies were analysed to develop the 'Big Five' framework.	(Salas et al. 2005)	(Dingsøyr and Lindsjørn 2013; Moe and Dingsøyr 2008; Strode 2015)
Teamwork Quality (TWQ)	The TWQ model focuses on the quality of collaborative work within teams and has six sub-constructs: communication, coordination, the balance of member contribution, mutual support, effort, and cohesion. The TWQ is significantly associated with team performance, measured as project success. The model was tested in a questionnaire administered to 145 German software teams.	Hoegl and Gemuenden (2001)	(Lindsjørn et al. 2016; Poth et al. 2020)
Input-Process-Output (IPO)	The IPO is a framework for studying team effectiveness. *Inputs* are antecedent factors that enable and constrain team member interactions. Inputs include individual team member characteristics; the structure of tasks; and the influence of external leaders, organizational design, and environment complexity. *Processes* are the functions and interactions individual team members must perform to accomplish team tasks. Processes include transition phases (mission analysis, goal specification), action phases (task accomplishments, monitoring progress, coordinating members), and interpersonal processes (conflict management, motivation, and confidence building). *Outputs* are types of team performance and team members' affect and viability. This model was proposed by McGrath (1964) and used by Mathieu et al. (2008) in a thorough literature review.	(Mathieu et al. 2008; McGrath 1964)	(Melo et al. 2013)

findings on team effectiveness models because the most influential are shown in Table 1. The studies indicate that agile methods positively influence team effectiveness through psychological safety, transparency, communication, emphasis on face-to-face communication, and contribution to team wellbeing. Table 3 focuses on seminal sources written by practitioners that we believe are influential in the practitioner community. These sources are not peer-reviewed, they are what is commonly referred to as 'grey literature', but they indicate, for example, that face-to-face communication is seen as the most efficient form of communication, and they also include concepts from the research literature such as psychological safety. As we explain in the research method section on theory development, we synthesize findings from our empirical studies from the research literature on agile software development as well as this grey literature in our arguments and evidence for a revised model of teamwork effectiveness.

3 Research Design

The purpose of this study was to develop a theoretical model of team effectiveness for agile software development. To ensure a broad basis of support for our model, we drew on various sources, which is a recommended strategy for empirically based theory-building research (Eisenhardt and Graebner 2007; Gregor 2006; Sjøberg et al. 2008). The sources were multivocal literature (Garousi et al. 2019) and empirical evidence. The supporting literature was

Table 2 Empirical studies of agile software development teams and teamwork

Study aim	Main findings	Source
Influence of agile team practices on productivity	'Social' agile practices (daily meetings, pair programming, reviews and retrospectives, etc.) positively influence psychological safety, transparency, communication, and ultimately productivity.	(Hennel and Rosenkranz 2020)
Influence of task interdependence on teamwork quality and project performance	Task interdependence is significantly associated with teamwork quality. Teamwork quality (coordination, cohesion, and learning) mediates the relationship between task interdependence and project performance.	(Kuthyola et al. 2017)
Influence of shared mental models and backup behaviour on performance	Agile development practices positively influences the sharedness of team members' mental models as well as degree of backup behaviour. If task complexity is high, backup behaviour has a positive and significant impact on team performance.	(Schmidt et al. 2014)
Influence of pair programming on team performance	Pair programming helps teams establish backup behaviour by improving the shared mental models among the team's developers. Backup behaviour reduces the negative effect of task novelty on performance.	(Kude et al. 2013)
Influence of agile development methods on communication	Scrum and extreme programming (XP) practices improve both formal and informal communication.	(Pikkarainen et al. 2008)
Mental models and developer work habits	Face-to-face communication leads to fewer task switches over instant messaging and email.	(LaToza et al. 2006)
Influence of agile methods on the wellbeing of team members	Agile methodology (XP) has a positive effect on the level of enthusiasm of the software developers in the most dynamic projects.	(Syed-Abdullah et al. 2006)

Table 3 Grey literature on agile software development teamwork

Model or method and source	Main claims regarding teams
The Scrum Guide (booklet) (Sutherland and Schwaber 2020)	'Scrum is a lightweight framework that helps people, teams, and organizations generate value through adaptive solutions for complex problems'. A scrum team includes developers, a product owner, and a scrum master. The qualities and values of a scrum team are: learn through inspection, be transparent about problems and progress, adapt based on learnings, commitment to goals, support and respect teammates, be open about challenges, be capable and independent people, have the courage to do the right thing and work on tough problems, build trust. Scrum team properties are: cross-functional and self--managing, typically 10 or fewer people responsible for all product-related activities from stakeholder collaboration, verification, maintenance, operation, experimentation, research and development, and anything else required, manage their work, work in sprints at a sustainable pace, accountable for creating a valuable, useful increment every sprint.
Team Topologies: Organizing Business and Technology Teams for Fast Flow (book) (Skelton and Pais 2019)	Organizations use four fundamental team types to achieve effective software delivery. A team is defined as 'a stable grouping of five to nine people who work towards a shared goal as a unit' (p. 44). The four team types are a stream-aligned team, an enabling team, a complicated sub-system team, and a platform team. These team types are based on Conway's Law, which proposes that the software architecture of a system will reflect the communication structures of the organization where the system was developed. The team types are viewed as effective organization designs when coupled with appropriate software boundaries and team interactions. Software teams should be small (max. 9); have long lifespans; support stable, trusting relationships; and manage the cognitive load on the team. Teams have three interaction modes: collaborating, facilitating, and X-as-Service.
Agile Practice Guide (booklet) (Project Management Institute and Agile Alliance 2017), pp. 168)	The agile manifesto is a key source for this book published by the PMI (Project Management Institute). The content is developed by volunteers from the agile community. The guidelines are not restricted to the software development context. The book focuses on team composition and team leadership. Key points on teams include the following: • Leadership follows the servant–leader model, and teams should be empowered. • Teams are colocated in a team space, dedicated rather than involved in multiple projects, self-managing, and cross-functional. • Teams include generalists and specialists. • Teams have collective ownership of the work and are collaborative. • Teams have fewer than 10 people.

Table 3 (continued)

Model or method and source	Main claims regarding teams
	• Teams should have stable working environments. • Agile teams include team members, a product owner, and a team facilitator. Guidance for multi-teams and distributed teams extend these founding ideas.
Google re: Work (Rozovsky 2015) (Duhigg 2016)	This model of software teams is based on two articles. Google re: Work proposes five key factors observed to set successful teams apart from other teams at Google: 1. Psychological safety 2. Dependability 3. Structure and clarity (of goals, roles, and plans) 4. Meaning of work (personally meaningful work) 5. Impact of work
Agile Software Development with Scrum (book) (Schwaber and Beedle 2002)	Scrum has a team-oriented philosophy primarily based on studies by Takeuchi and Nonaka (1986) of successful new product development (NPD) projects in Japan and the USA. Teams in scrum must show commitment, focus, openness, respect, and courage. Based on an IPO (input process output model), scrum involves a self-organizing team using the technology, requirements, and multi-learning as input to the development process, called a sprint, which is controlled using observation and adjustment of progress. The output is an increment of the total software product. 'Scrum deals primarily at the level of the team' (p. 2). 'Scrum is about deep social interactions that build trust among team members' (p. 106). Scrum is designed to enable teams, guided by knowledge and experience, to cooperate effectively to produce complex, sophisticated products rather than following a formal project plan. Scrum's empirical process control techniques are designed to enable management to carry out control by observation and incremental adjustment while the team carries out the development unhindered. Team size is recommended at 7+− 2; teams are cross-functional; all members are responsible for analysis, design, coding, testing, and user documentation. All members work on all tasks. Only the team can change the contents or estimates of a sprint backlog during a sprint. Teams must attend a daily scrum meeting. Guidelines for large projects involve setting up multiple teams who work from the same product backlog. Each team has its own scrum master.
Agile Manifesto 2001	Team-related principles include the following: The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. The best architectures, requirements, and designs emerge from self-organizing teams. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Table 3 (continued)

Model or method and source	Main claims regarding teams
Extreme Programming Explained: Embrace Change (book) (Beck 2000)	Extreme programming (XP) assumes that development work is best carried out in small colocated teams to aid with communication. Communication is necessary to project viability because it reduces misunderstandings and problems. High team morale and a good working environment are recommended. Teams must have the courage to make changes they see are necessary to the software they develop.

multi-vocal because we included teamwork research, agile software development teamwork research, and literature published for the practitioner community (grey literature) on agile software development.

To select a basis for an agile teamwork effectiveness model, we reviewed the three models presented in Table 1. After comparing the three models, we selected the Salas Big Five model for the following reasons:

- *Solid basis in literature:* The Salas Big Five model is well-grounded in the team literature. The model is based on a literature review and analysis of research on teams published over 20 years, from 1985 to 2005. The review included both empirical studies and theoretical models of team effectiveness. The TWQ model, although tested on empirical evidence from a study of software engineering teams, has a somewhat less well-grounded basis in the team literature than the Salas Big Five model because the contributing research ranges from 1982 to 1995.
- *Practical applicability:* The Salas Big Five model is practically applicable and potentially measurable because the model describes observable behaviours (referred to as behavioural markers) and includes testable propositions. The TWQ model focuses more narrowly on team internal collaboration and specifically excludes leadership factors, which the Big Five includes. The IPO model lacks practical applicability because it is a framework for categorizing team effectiveness factors rather than studying actual teams in specific contexts.

The Salas Big Five model of effective teamwork has eight interacting factors and propositions that state the relationships between the factors. Three factors are coordinating mechanisms and include shared mental models, closed-looped communication, and mutual trust. These coordinating mechanisms facilitate five teamwork components. The teamwork components are team leadership, mutual performance monitoring, backup behaviour, adaptability, and team orientation.

Each component and coordinating mechanism in the Salas Big Five model has an associated set of behavioural markers. Klampfer et al. (2001, p. 10) defined behavioural markers as '*observable, non-technical behaviours that contribute to superior or substandard performance within a work environment (for example, as contributing factors enhancing safety or in accidents and incidents in aviation); observable behaviours of teams or individuals; usually structured into a set of categories*'. Behavioural markers are used for training and assessing behaviour in the aviation industry (Flin and Martin 2001), intensive care units (Dietz et al. 2015), and crisis management (Gatfield 2008). Behavioural markers are short descriptive

statements derived from the '*analysis of data from multiple sources regarding performance that contributes to successful and unsuccessful outcomes*' (Klampfer et al. 2001, p. 10). A good marker has the following characteristics:

- It describes a specific, observable behaviour, not an attitude or personality trait, with a clear definition (enactment of skills or knowledge is shown in behaviour).
- It has demonstrated a causal relationship to the performance outcome.

– It does not have to be present in all situations.
– Its appropriateness depends on context.

- It uses domain-specific language that reflects the operational environment. It employs simple phraseology.
- It describes a clear concept.

The Salas Big Five model, by including behavioural markers to indicate the observable behaviours associated with a coordination mechanism or component, has immediate value for practitioners who can use the markers to evaluate whether their teamwork is effective.

The empirical evidence for our study extends the findings from two independent sources: a focus group study of teamwork in agile environments (Dingsøyr and Lindsjørn 2013) and case studies of agile teams (Strode 2015). The focus groups provide material that was based on the participants' experience on many teams, whereas the case studies provide in-depth perspectives on single-team contexts. Combining the findings from these two studies increases the international nature of the research and improves the breadth and depth of the evidence for the ATEM. The two studies were carried out independently. Each study had different researchers and was carried out in different parts of the world, at different times, and with different research methods, but both studies focused on evaluating the Salas Big Five team effectiveness model in agile software development.

The focus group study was reported to the Norwegian Centre for Research Data. Whitireia Polytechnic, New Zealand provided ethical approval for the case study.

We present the two studies and discuss their limitations in the following sections. Finally, we describe our approach to theory development.

3.1 Focus Groups

Focus groups enable researchers to quickly obtain detailed information on emerging phenomena through structured, moderated discussions with small groups of practitioners. The researcher can interact directly with respondents to clarify responses or ask follow-up questions. In addition, the focus group participants can react to and build upon responses from other focus group members (Stewart et al. 2007).

The motivation of the focus group study was to investigate what fosters and what hinders effective teamwork in agile software development teams and relate the results of the participants' input to the Salas Big Five team effectiveness model. Before participants were presented with the team effectiveness model, they carried out a brainstorming session, writing stickers (as many as possible), where they wrote down what they thought fostered and what they thought hindered effective teamwork. Each sticker recorded one relevant item.

3.1.1 Participant Selection

In total, 111 persons participated in 22 focus groups, with 4 to 6 participants in each group. In addition to 11 conference-based focus groups conducted at three conferences on agile software development, we conducted focus groups in four companies in Norway. In two of the companies, the participants represented development teams in actual ongoing projects. In the third company, we divided the whole development department into three focus groups. Finally, in the fourth company, participants were recruited for a focus group held after working hours, resulting in another two groups with members from a variety of projects.

3.1.2 Data Collection

We developed a plan for each focus group workshop which included a schedule and a set of exercises for the participants to carry out. Each focus group workshop was planned for 90 min and followed the same schedule (see Appendix 1).

During the focus group activity, each participant filled in a context questionnaire (see Appendix 2). The questionnaire showed that the participants were mainly software developers (39%), followed by scrum masters (18%), team leaders (12%), and project managers (10%). Most of the participants were using the scrum software development method (59%), followed by kanban (22%), lean software development (9%), and XP (8%). As for gender, 65% were male, and 35% were female. The participants worked in teams with 3 to 20 members (average 8.4, standard deviation 3.2). The teams had on average 6.6 full-time members (standard deviation 3.1). The teams the participants worked in were collaborating with up to 35 other teams. However, 55 participants were working in teams that did not collaborate with other teams. The participants had, on average, 11.9 years of experience with software development (standard deviation 8.4) and 4.3 years with experience with agile software development (standard deviation 2.5).

To document our results, we took minutes of all the focus group meetings and pictures of the final results, showing groups of items that foster or inhibit team effectiveness. This was documented for each teamwork component in the Salas Big Five model, and we also documented items that did not fit into the model. Figure 2 shows an example of results from one group at the XP2012 conference workshop.

3.1.3 Data Analysis

In total, the groups found seven items that did not fit into the model. The researchers classified these items. Examples of such items were ‘team size’ and ‘too difficult work tasks’, which were moved to team leadership. All participants received the minutes. Then, we recorded the items and whether they were marked as fostering or hindering teamwork in a spreadsheet. The items on four stickers were unreadable in the minutes, and this left a total of 1426. These items were first read to check that the topics identified were categorized into the correct Salas Big Five teamwork components. Seventeen items were moved by the second and third authors from one component to another.

The analysis was both quantitative and qualitative. The quantitative analysis consisted of counting the number of items marked as fostering or hindering teamwork. The qualitative analysis consisted of a thematic grouping of items allocated to each teamwork component.

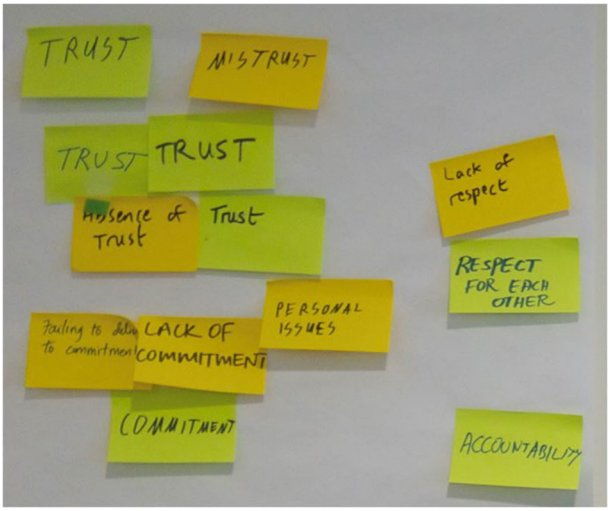

Fig. 2 Partial results of a focus group for the teamwork component 'trust'. Items that foster teamwork are in green and those that hinder effective teamwork in yellow

The limitations of this focus group study include the limited depth of the material collected because the individual terms included no explanatory text. Another limitation was a potential issue with external validity. There is a potential bias in participants as most responded to the invitation to the focus group meeting and might be more interested in teamwork than the average person working with software development. Most respondents were also working in a Scandinavian country, which might influence their opinion of teamwork factors. A potential threat to construct validity is that we rely on first the participants' understanding of the teamwork components and then the researchers' interpretation of the items when checking consistency. We have sought to ensure reliability through detailed descriptions of data collection and analysis and through showing traceability to the data material in the results section.

3.2 Case Studies

A positivist case study method was used to evaluate the applicability of the Salas Big Five model of effective teamwork to the agile context. Positivist case study research is a genre of qualitative research (Eisenhardt 1989; Eisenhardt and Graebner 2007; Keutel et al. 2014; Paré 2004; Yin 2018). Sarker et al. (2018, p. 761) characterize positivist case studies as representing a fact-based shared reality, using inductive or deductive data analysis, and often used to validate or refute an existing theory. Positivist case studies are distinct from interpretive, ethnographic, or grounded theory forms of qualitative research (Sarker et al. 2018). Further, the case study method provides detailed and current information on participants' experiences, knowledge, activities, and understandings in a specific bounded context (Yin 2018). The specific bounded context for each case was ongoing agile software development project teams.

The cases reported in this study are part of a broader multi-case study of coordination and dependencies in agile software development projects. Interview data containing material that focused on team issues was identified and analysed and contributed to this study.

3.2.1 Case Selection

Three cases from a commercial bank in New Zealand were selected for this study. The unit of analysis was an individual case of agile software development. The teams were all in the same physical location in a single large room. Each project had a stable team over most of the lifetime of the project. Because the teams were all in the same organization and location, they were all working within the same organizational culture. All project team members had volunteered to use agile software development, and all had received the same training in agile values and practices from the same trainers. All teams had the same agile coach and supervisory manager. All projects were resourced similarly. Furthermore, each project was supported by the same IT infrastructure unit, used the same underlying mainframe system, and complied with the same interface design guidelines and quality expectations.

When selecting cases for a multi-case study, the advice is to select cases showing variation (Yin 2018). The cases were selected from a pool of nine ongoing and independent projects carried out on the same floor of a large open-plan office. All agile development projects had been moved into this shared space. Before the cases were selected, the coach was asked to identify suitable cases that were technically complex, had several stakeholders, and varied in how they adopted agile software development practices. From this set, the researcher selected project teams that were creating a high-value business application where the software products under development were distinctly different from each other. Each project was a typical agile software development project with about 10 team members and was at least one-third complete and ongoing at the time of data collection. Finally, the project team members had to be willing to participate in a research project. The cases were code-named Globe, Tech, and Rock. Table 4

Table 4 Project profiles and interview information

Case	Globe	Tech	Rock
Organization-level interviews	Senior manager of all agile projects, including Globe, Tech, and Rock Agile coach for all agile projects (external consultant working on site)		
Roles of interviewees	Business lead/product owner [EC01] Agile lead/scrum master [EP01] Business analyst/agile lead [ET01] Joint interview: • Senior analyst programmer [ET02] • Tester [ET03]	Online banking business leader (customer proxy) [FP01] Test analyst [FT01] Mainframe architect (domain specialist) [FT02] Mainframe developer [FT03] Senior programmer/agile lead [FT04]	Senior business analyst [GP01] Senior analyst programmer [GT01] Test analyst [GT02] Technical designer [GT03]
Team size	10	10	6 to 12 varying over the project duration
Software product profile	Foreign exchange service	Transaction notification service	Online statements
Agile method adopted	Scrum	Scrum	Kanban-scrum hybrid

Key—The codes identify a specific person who was interviewed. These codes are used in the quotes in Section 4. For example [EC01] represents project Globe (E), product owner (C), and an interview (01).
• Project code: E = Globe; F = Tech; G = Rock
• Role code: T refers to a team member engaged primarily in development, C and P refer to people with other roles in the team such as product owner, agile lead, and customer proxy.
• 01–05 indicates a single individual.

shows the details of the cases and interviews. Each interviewed person received an information sheet explaining the research project and signed a consent form to agree to be interviewed and have their interview recorded.

3.2.2 Data Collection

Data collection followed a snowball approach. In each project team, the team leader was interviewed and asked to nominate other project team members. These team members were selected because they took different roles within the project and agreed to be interviewed. Up to five people from each project were interviewed using a semi-structured interview schedule (reproduced in Appendix 3 and published in Strode (2016). The data collection was restricted to five people to minimize interruptions to the project's work while providing an adequate depth of information about the project and the team. Interviews were carried out over a week for each case at times that suited the interviewees and in meeting rooms at the workplace. The interviews were recorded and transcribed in full. Additional case data were collected by observation and by taking notes at selected stand-up meetings and product demonstrations. Further data included photographs of scrum wall boards showing stories; tasks and task allocations; photographs of burn-down charts; publicly available data from the organization's website; and system and project documentation such as organization charts, interface designs, and example kanban cards.

3.2.3 Data Analysis

Qualitative content analysis was used to analyse the data, as described by Schreier (2013). This form of content analysis is a systematic method for analysing qualitative data and is carried out by assigning segments of source material (where a segment is a phrase, sentence, or paragraph in the interview transcripts) to the categories of a coding frame. Directed content analysis was used, which is appropriate when the coding frame is based on an existing theory (Hsieh and Shannon 2005). Following Schreier (2013), the Salas Big Five teamwork effectiveness theory was the coding frame but with one adjustment. The coding frame was adjusted before use to better align the component 'team leadership' with the philosophy of agile methods (Moe et al. 2009b) and autonomous teams as defined by Hollenbeck et al. (2012). These sources argue that autonomous teams (agile teams aim for autonomy) require 'shared team leadership'. So the coding frame was adjusted to include shared team leadership rather than team leadership. Schreier (2013) also advises that the qualitative content analysis method should include a pilot analysis to assess the applicability of the coding frame. The pilot involves the coding frame being tried out on a part of the material. The coding frame was found to be suitable because the analysis provided evidence to support or refute each of the categories in the coding frame. The coding was applied by a single coder over two points in time some weeks apart, which is a mechanism to improve validity when multiple coders are not available. The coding frame was applied to all of the interview data with the aid of the HyperResearch™ tool.

In a case study design, triangulation improves the validity of the study findings (Yin 2018). Data triangulation is the use of different data sources to provide evidence on which to base findings. Firstly, in this study, data were collected on the same topic from different sources of the same type. The three data sources comprised three cases of agile software development. Then, within each case, there were four or five data sources because up to five people in each project were interviewed. The semi-structured interview technique followed a pre-designed

schedule, including closed and open-ended questions, allowing for further probing questions when necessary. Secondly, data were collected from different sources. The photographs, observations, and collection of project documents were not analysed but confirmed the project details, the use of agile practices, and teamwork in the teams. Reliability was achieved using a transparent research process, as described here, so the findings can be traced from initial theory to conclusions (Yin 2018). Reliability was also improved by applying an existing conceptual framework (i.e. the coding frame) in the same manner to each case to identify instances of evidence for teamwork concepts.

The case study method has limitations. The findings from the set of cases are based on data from three cases within a single organization with a restricted number of interviews and observations collected over a short time period for each case. More cases and extensive data collection may have identified additional or disconfirming evidence. Another potential limitation is the use of snowballing to identify interviewees. Although this technique was necessary to locate knowledgeable and interested people on the team, this necessarily excluded some people and therefore could have provided a biased or incomplete view of team activities. We mitigated this issue somewhat by interviewing at least half of the project team members in each case and by collecting additional non-interview data on the team activities. As a case study, we generalize through use of theory (Yin 2018). The limitation of external validity due to case context is addressed by combining evidence from these three cases, the focus group study, and the research and grey literature in drawing final conclusions.

3.3 Theory Development

The ATEM was developed from independent analyses of the focus group and case study material using the Salas Big Five model as an initial framework. We compared the original description of each Salas Big Five coordinating mechanism and component and its behavioural markers with our empirical findings and with the recent empirical studies of others and the grey literature. This was done in a series of meetings where one of the two first authors drafted a section on each coordinating mechanism and component, which was then discussed. We sought to ensure that our revisions to the Salas Big Five model were primarily based on research findings with the grey literature as secondary support. We used terminology common in the software and agile practice field to ensure that the ATEM has practical relevance. We also sought to comply with the characteristics of the behavioural markers listed at the start of this section.

4 Revising the Salas Big Five: Components, Coordinating Mechanisms, and Behavioural Markers

This section presents our arguments for the ATEM based on revisions to the Salas Big Five teamwork effectiveness model. We explain our revisions of the three coordinating mechanisms and then the five core components. Each mechanism and component has behavioural markers, and we also revised these markers where we found evidence that this would better reflect agile teamwork.

Each subsection starts with a definition of the mechanism or component, followed by 1) empirical findings from the focus groups, case studies, and literature and 2) a discussion of how the empirical studies support or refute the behavioural markers for each mechanism or

component. For some mechanisms or components, we both redefine behavioural markers and also rename the mechanism or component. We argue that the new names better reflect agile team research findings and the guidance in the grey literature. These names are also more recognizable for software engineering practitioners. Where we have changed the name, the section heading indicates this change by first stating the Salas Big Five model name followed by 'becomes', then the new name. The arguments for the change are given at the end of the discussion of each mechanism or component. The full ATEM is summarized and discussed in Section 5.

4.1 Shared Mental Models

Shared mental models are defined as '*An organizing knowledge structure of the relationships among the task the team is engaged in and how the team members will interact*' (Salas et al. 2005, p. 560).

4.1.1 Empirical Findings on Shared Mental Models

The focus groups resulted in 200 items (of 1426; 14%) allocated to shared mental models. This was the third most frequently occurring team effectiveness factor. The main sub-component was 'common understanding of goals' (97). 'Common understanding of tasks', 'common understanding of process', and 'common understanding of the product' were also reported along with just 'common understanding'. Table 5 shows the results for this component. Note that in Table 5 and all of the focus group tables, items reported to foster and hinder team effectiveness are shown along with the counts of items allocated to each sub-component (e.g. 97 items were categorized into the sub-component 'common understanding of goals' in the analysis of the shared mental model component).

Table 5 Shared mental model sub-components with selected items that foster and hinder team effectiveness

Sub-component	Items	
	Foster (total=126)	Hinder (total=74)
Common understanding of goals (97)	Agreement on goals Common goals Clear vision Everyone understands goal	Lack of common goal No clear common goal Unclear goals Unclear mission
Common understanding of tasks (18)	Clearly defined tasks Clear tasks Well-defined needs	Disagreement on the distribution of tasks Unclear tasks Unnecessary work because needs are misunderstood
Common understanding of the process (10)	Good process Team rules Work agreement	Rules and standards No process Working in personal caves
Common understanding of the product (8)	Knowledge of domain Knowledge of technology Understanding of what is to be delivered	Mismatching expectations No common understanding of deliverable Poor specification
Other:	'Common understanding of roles', 'knowledge of customer needs', 'colocation', and 'knowledge of scope'	

The case studies showed that certain agile development practices support shared mental models, including 1) specification meetings, 2) planning meetings, and 3) stand-up meetings. For example, shared mental models (1) were created during specification meetings, as one team member explained: '*the spec workshop, although I find it boring as hell sometimes, it's interesting, everybody's getting an understanding of what's supposed to be happening. I don't necessarily understand what they're all talking about, but there's clarity...between everybody. So that nobody, theoretically, is thinking when I say "I want to change a rule in the host system" that I'm saying I want to change a rule somewhere else. Because we've talked about it, everybody has an understanding of what I want*' [EC01]. This quote indicates that a shared mental model, in the form of a common understanding of the work process and tasks, has formed among the team members. In another case, planning (2) and stand-up meetings (3) increased the team's shared mental model. A team member explained, '*We've gone back to our desks, and we're gearing up for the real start of the coding and the developing, and it just occurred to us that we just said it's better for you to do these, it's better for me to do these, and you and you and you. So it's like a Gentleman's Agreement that the pieces of work are grouped in. So in our minds, is not set or written, or anything, but what we have is in our inner mind we already kind of talk about this piece of work is more suited for you*' [FT03]. This could indicate a shared understanding of who knows what in the team or a shared understanding of the skills and expertise of others on the team, which is one facet of a shared mental model reported by Levesque et al. (2001).

The lack of a shared mental model was clear in Rock when a project leader said, '*Another thing that we probably could have done better, is had a bit more of an overview of ... what we're trying to do. But [new team members are] sort of thrust in and said "do this" with no context...I know for some of the other [system] developers that came on, it would have probably been really good for me to sit down with them and talk about the project and give them a bit more context about what they're actually doing, and why*' [GP01]. In this same case, another team member reflected that stand-up meetings supported a shared mental model, '*So the stand-up meeting I think contributed a lot to it. Because you'd know during those meetings what people were working on, and who was working on what*' [GT01].

Studies of shared mental models in software development teams indicate that a shared mental model has a positive impact on software team performance (Dingsøyr et al. 2016; Schmidt et al. 2014) and can have a larger effect on performance than age, tenure, or gender (Kang et al. 2006). In agile software development, Yu and Petter (2014) argue that shared mental models result from adopting common agile practices, including backlogs, sprints, meetings (e.g. daily stand-up retrospective, planning), and having a customer on site.

4.1.2 Discussion of Behavioural Markers for Shared Mental Models

When a team establishes a shared mental model, team members can anticipate each other's needs and adjust their work strategies to adapt to changes in the team or team tasks (Salas et al. 2005). 'Anticipating and predicting each other's needs' is the first behavioural marker for shared mental models in the Salas Big Five model (see Table 6). Because evidence from the case studies on shared mental models in agile teams supports this marker, we retained this marker in the model.

We revised the second marker from 'identifying changes in team, tasks, and teammates and adjusting strategies as needed' to explicitly focus on five common understandings that we

Table 6 Behavioural markers for shared mental model

Behavioural markers for shared mental models (Salas et al. 2005)	New behavioural markers for shared mental models
Anticipating and predicting each other's needs	Existing marker supported
Identify changes in the team, task, or teammates and implicitly adjusting strategies as needed	Existing marker removed
	Common understanding of goals
	Common understanding of tasks
	Common understanding of the work process
	Common understanding of the product
	Common understanding of individual skills and expertise

found contribute to a shared mental model in agile teams. The common understandings are about goals, tasks, work process, product, and skills.

Many focus group participants associated a common understanding of goals as a part of a 'shared mental model'. This finding is supported by the results from a study of systems development teams, where 'clarity of mission' was positively related to team effectiveness (Lu et al. 2011). Therefore, we added 'common understanding of goals' as a new marker.

General studies on teams describe shared mental models as composed of a shared understanding of tasks, work processes, and who knows what in the team (Converse et al. 1993). The case study provides insight into team practices which can lead to a shared mental model and evidence for a shared understanding of each team member's expertise. Still, the focus group studies provided support for markers that focus on the factors software teams perceived as important; that is, teams exhibit a common understanding of goals, tasks, work process, and product. From general studies and the case study, we included 'individual skills and expertise'. To keep the markers short, we did not include 'adjusting strategies as needed' as in the original second marker, but we emphasize that the understanding should be updated when there are changes. Table 6 shows the proposed new behavioural markers for the shared mental model component.

4.2 Mutual Trust

Mutual trust is defined as the '*shared belief that team members will perform their roles and protect the interests of their teammates*' (Salas et al. 2005).

4.2.1 Empirical Findings on Mutual Trust

The focus groups resulted in 197 items (of 1426; 14%) allocated to trust. This was the fourth most frequently occurring team effectiveness factor. Many focus group participants stated simply that 'trust' is important (67). Table 7 shows the sub-components for mutual trust, which were 'respect', 'social climate', 'conflict', 'openness', and 'other'.

The trust sub-components show a diverse understanding of trust, focusing on the respect of team members, a good social climate in the team, few conflicts, openness, and that team members should feel safe when working in the team.

Table 7 Mutual trust sub-components with selected items that foster and hinder team effectiveness

Sub-component	Items	
	Foster (total=115)	Hinder (total=82)
Respect (28)	Respect for the team members' competence	Hostility between team members
	Respect for each other	Intolerance
	Politeness	Lack of respect
Social climate (20)	Beers on Friday	Negative climate
	Good atmosphere between team members	No empowerment
	Social activities	
Conflict (18)	Few conflicts	Many conflicts within the team
		Someone's need to have control
		Large cultural differences
Openness (17)	The team accepts diverging views	Dread to be open/honest
	Open to others' views	Blame game
	Feeling safe	Insecurity
Other	'Safety', 'engagement', 'belonging', 'stress', 'balance in team', 'collaboration'	

The case studies provide support for mutual trust as a factor in team interactions. Trust in the cases included 1) trusting team members, 2) trusting information, and 3) trusting that the team had the skills and knowledge needed for the work. Trust that the team does their best (1) was commented on by two participants: '*But we have such a good team that everybody is just communicating…and never says something bad about another person, we generally believe that everybody does his best to develop something …*' [ET02], and '*Best job to their abilities and …we don't really question people's abilities. Everyone does their best*' [ET03]. The following statement shows there was trust that the team would provide accurate information (2): '*So the negotiation in our team is possibly raising technical stories that have little or no business benefit, even though it directly contributes to the end product, and is required. And you negotiate with the business lead…I talk to [her] about it, obviously [she] trusts the development people to give [her] accurate information about why it's important and it's very easy and simple conversation*' [FT04]. Finally, one informant expressed trust in team members having suitable skills (3): '*The team dynamic. The core team members all understood, once we'd got into the project, what was needed. The people doing all the coding were all on the same page and all competent people with good skills*' [GT02].

4.2.2 Discussion of Behavioural Markers for Mutual Trust

Organizations have increased their focus on trust because of the emergence of self-directed teams and reliance on empowered workers (Costigan et al. 1998). This change is emphasized in the practitioner literature. For example, a principle behind the agile manifesto states, '*Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done*' (Williams 2012, p. 72). Trust and respect are organizational conditions needed for a team's psychological safety (Edmondson et al. 2004). Psychological safety '*refers to a climate in which people are comfortable being (and expressing) themselves*' (ibid, p. 1) (see also Table 3 where psychological safety is a factor in the Google teamwork model). The Salas Big Five model has two behavioural markers for trust: 1) information sharing and 2) willingness to admit mistakes and accept feedback. We found evidence for the first marker in the case studies and the second marker in the focus groups. In addition, we

Table 8 Behavioural markers for mutual trust

Behavioural markers for mutual trust (Salas et al. 2005)	New behavioural markers for mutual trust
Information sharing	Existing marker supported
Willingness to admit mistakes and accept feedback	Existing marker supported
	Supportive team social climate

added a third, new marker for mutual trust named 'supportive team social climate' due to evidence originating from the focus group responses. Table 8 shows the new markers for mutual trust.

4.3 Closed-Loop Communication Becomes Communication

Closed-loop communication is defined as '*the exchange of information between a sender and a receiver irrespective of the medium*' (Salas et al. 2005, p. 561).

4.3.1 Empirical Findings on Communication

The focus group resulted in 244 items (of 1426; 17%) allocated to communication. This was the second most frequently occurring team effectiveness factor. Many items simply described 'communication' (60). The sub-components that many focus group participants noted were 'colocation', 'openness', 'communication infrastructure', 'visualising status and progress', and a 'friendly atmosphere'. Table 9 shows the sub-components of closed-loop communication.

While the Salas et al. (2005) definition of 'closed-loop communication' focuses on information exchanged between 'a sender and a receiver', the case study findings suggest that communication in an agile team setting is oriented towards the whole team and not a single receiver. As one case study participant said, '*I send communications to... the developers as a whole*' [FP01]. The focus group findings show the perceived importance of colocation for achieving close-loop communication.

The case study material supported the focus group findings on the link between colocation and communication. '*Because we help each other, we communicate a lot, and we always can jump in and say, "Oh can you please have a quick look here for me", or approach [Carlos], like this morning and say, "I think I've got it right. Can you just come and have a look to make sure that I've got it right?" And he just comes and look over my shoulder and just says, "Yep, yep, yep"*' [ET02]. Further, communication for achieving understanding was supported by colocation: '*I send communications to... the developers as a whole. I'll also talk to the host developers...or else I'll go and talk to...the BA. There might be something I don't fully understand in the acceptance criteria. Or the...business lead if I...have a question there. I receive communications from everyone on the team... I communicate with everyone on the team and I receive communications from everyone on the team*' [FP01]. Communication for problem-solving was also supported by colocation: '*If I had an issue or a problem, then I would go straight to, he was just sitting right in front of me, [Valor]. And [Kate] was...on my left...So we did have those conversations ... face to face always helps. We were able to explain right away and the turnaround was quick. .... The tester was beside me as well. So you had the test analyst, you had the three components people...right beside each other*' [GT01].

Table 9 Sub-components of closed-loop communication with selected items that foster and hinder team effectiveness

Sub-component	Items	
	Foster (total=142)	Hinder (total=102)
Colocation (48)	Physical presence	People are distributed
	Colocation	Distance
	Physically placed together	Not colocated
Openness (21)	Open communication	Secrecy
	Openness in the team	Retaining information
	Open dialogue	
Communication infrastructure (15)	Process support tools	Bad tools
	Suitable office spaces	Bad office facilities
	Tools that work	
Visualizing status and progress (10)	Informative workspace	No whiteboards
	Visualize things that go well	
	Whiteboard/task board	
Friendly atmosphere (14)	Good atmosphere	Scolding
	Fun	Antisocial environment
	Friendly tone	Bad atmosphere
Other	'frequent communication', 'absence of conflicts', 'absence of interruptions', 'absence of introverted team members', 'customer available', 'common language and culture', 'team leadership', 'slow response', and 'follow-up'	

Agile teamwork studies of communication report that face-to-face communication can lead to fewer task switches for developers when compared to communication over instant messaging and email (LaToza et al. 2006) and that agile practices can have a positive effect on communication and team productivity (Hennel and Rosenkranz 2020; Pikkarainen et al. 2008).

4.3.2 Discussion of Behavioural Markers for Communication

We argue that communication is key to efficient software development in agile teams. We decided, however, to reduce the focus on one-to-one communication by renaming the Salas Big Five model component 'closed-loop communication' as 'communication'. We removed 'Following up with team members to ensure the message was received' and replaced it with a marker that focuses on the whole team's responsibility: 'The team follows up on the progress of tasks'. We also incorporated findings from agile development practice in the behavioural markers. We replaced the focus on one-to-one communication with what many practitioners express as important for communication, namely to 'visualise project information' by using agile task boards or making sketches of architecture or work processes during team interaction. This change is also supported by studies of collaboration in agile development teams which show how agile teams use artefacts such as stickers and physical boards to facilitate coordination and communication (Sharp and Robinson 2006). Therefore, we added the behavioural marker, 'facilitate informal communication' to emphasize the need for physical infrastructure such as a team room and easy physical access to other team members and business experts. Table 10 shows the behavioural markers for communication.

Table 10 Behavioural markers for communication

Behavioural markers for closed-loop communication (Salas et al. 2005)	New behavioural markers for communication
Following up with team members to ensure the message was received	The team follows up on the progress of tasks
Acknowledging that a message was received. Clarifying with the sender of the message that the message received is the same as the intended message	Visualize project information
	Facilitate informal communication

4.4 Team Leadership Becomes Shared Leadership

Team leadership is defined as the '*Ability to direct and coordinate the activities of other team members, assess team effectiveness, assign tasks, develop team knowledge, skills, and abilities, motivate team members, plan and organize, and establish a positive atmosphere*' (Salas et al. 2005, p. 560).

4.4.1 Empirical Findings on Shared Leadership

Leadership was a primary concern in the focus groups, with 289 items (of 1426; 20%) concerned with aspects of leadership. In total, 79 of the 1426 items simply reported 'leadership'. Table 11 shows the top five sub-components of leadership identified in the analysis: 'planning', 'shielding from interruptions', 'work process', 'adequate resources', and 'infrastructure'. Table 11 shows the results for the team leadership component.

The leadership items concerned leadership within the team, across the team-organization boundary, and project planning. Intra-team leadership is indicated by the sub-component 'work processes' because the items in this category focus on functions within a team. The sub-components concerned with boundary-spanning include 'shielding from interruptions', 'adequate resources', and 'infrastructure'. Shielding involves a leader removing interruptions

Table 11 Eight main sub-components of team leadership with items that foster and hinder team effectiveness

Sub-component	Items	
	Foster (total=178)	Hinder (total=111)
Planning (24)	Good planning	Bad planning
	Participative planning	Too-thorough planning
	Adequate planning	Short-sighted planning
Shielding from interruptions (24)	Reduce unnecessary interruptions	Interruptions
	Shielding the team	Workday split up
	Someone protecting the team	Change the agreed content
Work processes (15)	Slack to think big	Heavy process
	Responsibility process in place	Unnecessary processes
Adequate resources (16)	Full-time members	Part-time resources
	Capacity	Lack of resources
	Availability	Resource allocation
Infrastructure (14)	Working infrastructure	Lack of tools
	Good work conditions	Lack of technical infrastructure
	Access to tools	Unnecessary tools

to the team by non-team actors. Provisioning adequate resources and infrastructure requires a leader or team member to interact with the organization. The project planning sub-component of leadership included 'good planning', 'adequate planning', and 'participative planning'.

The case studies showed that the participants viewed leadership from an intra-team perspective. In these colocated teams, leadership was shared with no single acknowledged team leader; that is, the team members would allocate themselves to tasks rather than being assigned to tasks by a specific leader. This was shown in comments from Globe and Tech, such as, *'... not peer pressure as such, but it's almost team responsibility and someone has to do it ... normally what happens is someone will say, "Oh yeah, I can do that". They will delegate themselves*' [ET01]. '*Well, it's not assigned, you take it, ...we'll have a bunch of stories that are mainframe centred, and a bunch of tasks that are all related to those stories, and obviously there's got to be some order in the tasks because you can't test before you build. But nevertheless, we'll decide just between ourselves, and we might say, "[Nick], you take that one, [Livia], you take that, and I'll do this one'*" [FT02]. Team members in Rock tended to rely more on a central figure to ensure that work progressed and tasks were allocated and completed by team members.

4.4.2 Discussion of Behavioural Markers for Shared Leadership

The Salas Big Five model has six behavioural markers for team leadership (see Table 12). We found, however, no evidence for three of the Salas Big Five behavioural markers but evidence for six new markers. Firstly, we found no evidence that leaders in agile teams synchronize and combine individual team member contributions. This can be explained because in agile teams, team member contributions are managed with task boards, the self-assignment of tasks, and continuous integration of the software code (Fitzgerald and Stol 2017; Sharp and Robinson 2010).

Secondly, we found no evidence for leadership involved in clarifying team roles. We propose that this is because roles tend to be defined by the chosen methodology (e.g. scrum or XP) in agile project teams, and role differentiation is not the aim. In addition, among the team members, there must be an appropriate assemblage of skills necessary for project and product

Table 12 Behavioural markers for shared leadership

Behavioural markers for team leadership (Salas et al. 2005)	New behavioural markers for shared leadership
Facilitate team problem-solving	The agile team facilitates team problem-solving
Provide performance expectations and acceptable interaction patterns	The agile team determines performance expectations and acceptable interaction patterns
Synchronize and combine individual team member contributions	The agile team synchronizes and combines individual team member contributions using agile practices combined with automated tools
Seek and evaluate information that affects team functioning	The agile team seeks and evaluates information that affects team functioning
Clarify team member roles	Agile values and methodologies determine team member roles
Engage in preparatory meetings and feed-back sessions with the team	Agile values and methodologies determine the frequency and type of preparatory meetings and feedback sessions
	A servant leader facilitates a boundary-spanning function
	Agile team practices provide a planning function

completion, and specific skills are more important than roles (Project Management Institute and Agile Alliance 2017).

Thirdly, we found no evidence that leaders in agile teams engage in preparatory meetings and feedback sessions with the team. We propose that this is because agile teams have a shared form of leadership with no designated leader, and all team members are involved in preparatory meetings and giving feedback to one another, such as in planning meetings, sprint reviews, retrospective meetings, and daily stand-up meetings (Project Management Institute and Agile Alliance 2017).

We renamed the Salas Big Five team leadership component based on these arguments and supporting evidence from our material and the agile team literature. The revised name is 'shared leadership' because this name more accurately reflects the form of leadership adopted by agile teams and better encompasses the six new behavioural markers we identified. These new markers are shown in Table 12.

The lack of fit between the Salas Big Five model's behavioural markers and our findings can be explained by two characteristics of the agile approach. Firstly, specific practices in the agile approach substitute for many of the leadership behaviours proposed in the Salas Big Five model. Agile project teams aim to be self-directing (i.e. self-organizing or autonomous) rather than directed and coordinated by a designated leader (Hoda and Murugesan 2016), and leadership tends to be shared among the team members (Moe et al. 2009b). Rather than relying on a leader, agile teams evaluate and plan changes to their performance in retrospective sessions, self-assign tasks based on a prioritized product backlog, and synchronize and combine individual contributions using software tools and task boards (Project Management Institute and Agile Alliance 2017). In addition, agile teams develop and share team knowledge, skills, and abilities using techniques such as pair programming and practices such as colocation, and teams aim for generalist skill sets (Beck and Andres 2005; Kude et al. 2013).

The second characteristic of the agile approach that differs from the Salas Big Five model is the role of a 'servant leader' in an agile team (Greenleaf 2003; Project Management Institute and Agile Alliance 2017; Sutherland and Schwaber 2020; Van Dierendonck 2011). This role facilitates team empowerment and motivation and performs boundary spanning between the organization and the team; however, there is limited research into this aspect of agile teams (Holtzhausen and de Klerk 2018). The key idea is that initially, a servant leader empowers the team and then steps back once this team-level state has been achieved. Due to the lack of research on servant leadership in agile teams, we have not focused on servant leadership as a factor but on shared leadership which evolves in mature agile teams.

Because team leadership has changed within agile teams to become shared leadership, we redefined this component as follows. Shared leadership is the '*ability of the team to direct and coordinate their activities, assess team performance, assign tasks, develop team knowledge, skills, and abilities, motivate one another, plan and organise, and establish a positive atmosphere*'.

4.5 Mutual Performance Monitoring Becomes Peer Feedback

Mutual performance monitoring is defined as the '*ability to develop common understandings of the team environment and apply appropriate task strategies to accurately monitor teammate performance*' (Salas et al. 2005, p. 8).

4.5.1 Empirical Findings on Peer Feedback

In the focus group material, this component had the smallest number of items (76 items of 1426; 5%) indicating that there is little awareness of its importance. The results from the focus groups include items related to reflecting on work practice, giving each other feedback in the team, that tasks have joint responsibility, and that the team is aware of their work effectiveness. Table 13 shows the main sub-components of mutual performance monitoring.

The Salas Big Five model cites studies finding that mutual performance monitoring is important in stressful situations where team members are more likely to make errors. Team members are also often not aware of the errors they make, which can be remedied with feedback. In the Salas Big Five model, mutual performance monitoring affects team performance through effective backup behaviour. Further, mutual performance monitoring requires an understanding of what others in the team are doing (i.e. part of a shared mental model) and that the 'monitoring' is not perceived negatively by team members (i.e. there needs to be trust within the team).

Mutual performance monitoring was evident in Globe and Tech but not in Rock. In Globe, the team members *'keep an eye on each other'* and on the task board to ensure that the agile process is followed to avoid an unexpected backlog of testing. A team member explained, '*We have learnt from mistakes with sprints … say developers working on stories for 8 days out of 10 and then deliver everything to us in the last two days and we would have massive testing tasks in a short time so that was like a mini Waterfall within the sprint and that's not how it is meant to happen. We are meant to finish the first story, then move on to the next one, and so on or at least start a couple of stories but finish them before we move towards the bottom of the wall. So, we keep an eye on each other and if we see that we tend to [take action]*' [ET02]. In Tech, the team members were able to quickly assess whether a lack of communication about changes in the software was affecting a team member and could take action. A team member

Table 13 Main sub-components of mutual performance monitoring with selected items that foster and hinder team effectiveness

Sub-component	Items	
	Foster (total=53)	Hinder (total=23)
Reflect on practice (22)	Joint reflection exercises Know own strengths and weaknesses Retrospectives	Long feedback loops Lack of self-awareness
Joint responsibility of result (20)	Ownership Responsibility for result Result orientation	Conflicting incentives Focus on individual achievements Individual rewards
Feedback (9)	Clear feedback Concrete feedback Constructive feedback	Negative team members Poor feedback mechanisms
Measuring effectiveness (7)	Regular check on progress Measurable effectiveness Intention to improve	No measure of effectiveness
Other	'Status', 'clear expectations', 'turnover', 'openness', 'recognition', 'demonstration', 'joint review'	

said, '*So the communication within the team I think is fundamental. And you see it when someone forgets to tell someone something, that it interrupts what they were actively doing, and they go "what's just gone on here?" And quickly do a run around the team and find out someone's done something that's impacted another person. So, communication is definitely the biggest one. That doesn't mean that it's perfect … you still get small break downs that cause interruptions*' [FT04].

4.5.2 Discussion of Behavioural Markers for Peer Feedback

The term 'mutual performance monitoring' is, to our knowledge, not used in studies of agile software teams other than those explicitly using the Salas Big Five model. However, we find many studies of practices that foster joint work and feedback, such as in pair programming (Hannay et al. 2009), regular demonstrations (Schmitz et al. 2019), and retrospectives (Lehtinen et al. 2017). One could also argue that a practice such as continuous build and test-driven development would lead to feedback and awareness of errors. Also, daily meetings (Stray et al. 2016) could function as a mechanism to provide feedback on work by others. We believe there is good support in agile practices in achieving mutual performance monitoring, which might explain the limited awareness of this factor. We do not think there is evidence for recommending specific practices as behavioural markers, and we decided to keep the behavioural markers in the Salas Big Five model (2005) for agile software development teams. However, we decided to rename the factor to 'peer feedback' because we believe agile team members will find this name more understandable and relevant to their team environments. In addition, the term 'peer feedback' does not have the negative connotations of 'monitoring', which implies that team members are constantly checking on teammates' work or actions to correct them in some way. We also propose that feedback be given 'regularly' and have rephrased the second marker accordingly.

Because mutual performance monitoring is not evident in agile project teams, and we propose the new factor 'peer feedback', we also developed a definition for peer feedback that varies from the Salas Big Five model (2005). Peer feedback is *'the ability to develop common understandings of the team environment and based on those understandings to give accurate peer feedback to team members'*. Table 14 shows the behavioural markers for peer feedback.

Table 14 Behavioural markers for mutual performance monitoring/peer feedback

Behavioural markers for mutual performance monitoring (Salas et al. 2005)	New behavioural markers for peer feedback
Identifying mistakes and lapses in other team members' actions	Existing marker supported
Providing feedback regarding team member actions to facilitate self-correction	Regular feedback regarding team member actions to facilitate self-correction

4.6 Backup Behaviour Becomes Redundancy

Backup behaviour is defined as the '*ability to anticipate other team members' needs through accurate knowledge about their responsibilities. This includes the ability to shift workload among members to achieve balance during high periods of workload or pressure*' (Salas et al. 2005, p. 560).

4.6.1 Empirical Findings on Redundancy

The focus groups resulted in 108 items (of 1426; 8%) allocated to backup behaviour. This was the seventh most mentioned teamwork component. Table 15 shows the top six sub-components of backup behaviour. These sub-components include 'the right competence', 'distribution of tasks', 'time to work together', 'specialization', 'joint commitment to tasks', and 'experience sharing'. Notably, 'workload distribution' is a sub-component that exactly matches one of the Salas Big Five model behavioural markers. Table 15 shows the results for this component.

The case study findings support each of the behavioural markers for backup behaviour proposed in the Salas Big Five model (see Table 16). Workload distribution was a concern, and the team members took action to shift work to underutilized team members. The team members were able to take on the tasks of their teammates because of their broad skill sets. For example, developers could take on testing tasks. In Tech, team members would ask their team members if they could help. For example, a team member said, '*When we are say, finished with a task and there's nothing more to do, then we'll go and ask if there's a sequence of things that one person is intending to do if there's some backlog or we ask the person if he is alright or if it's alright for us to jump into one of the tasks*' [FT03]. In Globe and Rock, the task board provided a mechanism to identify when backup behaviour might be appropriate. In Globe, a team lead explained, '*there's a bit of cross-work. Sometimes if there's a lot of testing work*

Table 15 Main sub-components of backup behaviour with selected items that foster and hinder team effectiveness

Sub-component	Items	
	Foster (total=58)	Hinder (total=50)
The right competence (49)	Everyone can contribute	Lack of competence
	Need for other people's knowledge	Skewed competence profile in team
	Skilled team members	Too-similar competence
Distribution of tasks (15)	Equal status of team members	Bottlenecks
	Right task to right team member	Too many tasks in progress at once
	Tandem working	Wrong people on tasks
Time to work together (9)	Have time to work together	Little pair work
	Pair programming	'Loners' in team
	Pair working	Team members isolated
Specialization (8)		Large diversity in skills
		Little spread of knowledge
		'Experts'
Joint commitment to tasks (7)	Empowerment	No helping each other
	Give/take responsibility	One responsible per task
	Tasks given to pairs or group	
Experience sharing (6)	Time to learn for all team members	Little overlap on tasks

Table 16 Behavioural markers for backup behaviour/redundancy

Behavioural markers for backup behaviour (Salas et al. 2005)	New behavioural markers for redundancy
Recognition by potential backup providers that there is a workload distribution problem in their team	Existing marker supported
Shifting of work responsibilities to underutilized team members	Existing marker supported
Completion of the whole task or parts of tasks by other team members	Existing marker supported

outstanding, the developers will help with some testing. And there are some tasks [on the task board] that everyone on the team will work on, depending on who is free and who is available' [EP01]. In Rock, a team member stated, '*And some of it was at the wall [*i.e. *task board] …, "Okay when are you going to be able to finish all that testing?" and I would say, "Well I've got this and this and this to do", and [Madhup] would say, "Well I'm just about finished this, and I haven't got anything coming for a week, so I can do something*" [GT02].

Backup behaviour is recognized in agile team studies, but the focus is on how pair programming supports backup behaviour; other factors or practices that might contribute to backup behaviour are not considered. In one study, Kude et al. (2013) measured agile team effectiveness based on role-focused questionnaire data from 62 colocated scrum teams. The results showed that pair programming helps teams establish backup behaviour by improving the shared mental models among the team's developers and that backup behaviour reduces the negative effect of task novelty on performance. Our findings also support the result that shared mental models contribute to backup behaviour. Many sub-components in the focus groups that fostered backup behaviour were activities that promote shared mental models, including knowledge sharing, team or individual empowerment (which allows all members to contribute), having time to work together, and work in paired modes.

A study by Coman et al. (2014, p. 125) defines backup behaviour in agile teams simply as '*the extent to which team members help each other perform their roles*'. They identified collaboration and cooperation as forms of backup behaviour occurring during pair programming in a study of three software development projects. Cooperative backup behaviour was of short duration and occurred when a developer helped a teammate on a small or isolated subtask, whereas collaborative backup behaviour was of longer duration, occurred less often, and arose when teammates worked together because they share the same goal when solving an entire issue.

4.6.2 Discussion of Behavioural Markers for Redundancy

The findings in our focus groups mention pair programming as a mechanism to support backup behaviour, whereas this is not mentioned in the case studies. However, the focus group findings indicate a wider range of sub-components related to backup behaviour, including skill sets (consisting of competence, specialization, and experience of sharing), task commitment, and time to allow for helping behaviours. The case findings indicate that mechanisms for initiating backup behaviour are to ask teammates if they need assistance and to use the information displayed on task wallboards. Therefore, based on the focus group and case study evidence, we agree with the behavioural markers in the Salas Big Five model and found no evidence for new behavioural markers.

We decided to rename backup behaviour as 'redundancy'. We made this change because the redundancy of skill sets in agile software development is necessary to enable backup behaviour. Without suitable skills, there can be no effective backup behaviour. In agile teams, the team members aim to be cross-functional, which means the team contains all of the skills needed to achieve their goal (Project Management Institute and Agile Alliance 2017). Team members also aim to be generalizing specialists (see Table 3), which means an individual team member has not only a speciality but also a breadth of experience and multiple skills. This generalizing specialist characteristic is necessary so that team members '*can routinely help each other*' or, in other words, provide backup behaviour (Project Management Institute and Agile Alliance 2017, p. 42). We also renamed this component because the idea of redundancy is more familiar to those involved in software development than the term backup behaviour. Table 16 shows the behavioural markers for backup behaviour, now renamed 'redundancy'.

4.7 Adaptability

Adaptability is defined as the *'ability to adjust strategies based on information gathered from the environment through the use of backup behaviour and reallocation of intra-team resources. Altering a course of action or team repertoire in response to changing conditions (internal or external)'* (Salas et al. 2005, p. 560).

4.7.1 Empirical Findings on Adaptability

The focus groups resulted in 118 items (of 1426; 8%), indicating that 'adaptability' is of relatively low perceived importance as this component was ranked sixth of the eight teamwork components. The main sub-components were 'organizational constraints', 'team environment', 'collaboration

Table 17 Main sub-components of adaptability with examples of items that foster and hinder team effectiveness

Sub-component	Items	
	Foster (total=58)	Hinder (total=60)
Organizational constraints (37)	Choosing own tools	Dependencies outside team
	Environment	Limited resources
	Suitable tools	Bad technological choices
Team environment (30)	Good atmosphere	Bad atmosphere in team
	Fun at work	Egoism
	Humour	Internal competition
Collaboration culture (9)	Rolling pair work	Customer uncertainty
	Willingness to solve problems	Instability
	Willingness to collaborate	Lack of collaboration
Team focus (8)	Good team dynamics	Other or new tasks
	Daily synchronization	Disturbances
	Continuous improvement	
Right competence (6)	Competence	Team not proper for task
		Lack of competence
		Important persons outside team
Conflict (4)		Internal conflicts
		Conflicts
Other (24)	'Joy of work', 'engaged team members', 'little priority to team tasks', 'team composition', 'openness'	

culture', 'team focus', 'right competence', and 'conflict'. Table 17 shows the main sub-components of adaptability with examples of items that foster and hinder team effectiveness.

In Globe and Tech, the teams readily adapted to change when they encountered a problem. In these cases, the teams experienced problems with the way their work was organized. They recognized the problem and jointly decided to change how they were working to improve the situation. For example, a team leader in Globe explained, '*When we were first doing our Agile Sprints, we would do … spec workshops, which is where you go through the stories to understand the business intent of upcoming stories…we used to…have these meetings once a week for a couple of hours…But…we found that when we got to the sprint and we decide what we've going to do, sometimes the Business Lead will have changed her mind on what her priorities were, or sometimes you couldn't do what we thought we were going to do. And… all that work had been wasted. So, we decided that rather than do the… workshop…two weeks in advance, we would do it immediately before our Sprint planning. And that's worked out better*' [EP01]. In Tech, the team acted when they recognized a problem with the product demonstration; the team leader explained, '*So we failed…when we went to show it the test environment wasn't up and running. So, we couldn't actually demo it…we just had to talk about it… everyone in the team was very upset about that because it … looks bad … So …one of the decisions we made was that for the show and tell we're going to move it to a different day, to a Tuesday, and rearrange all those other meetings slightly to make it work. And … we were going to put one person in charge of the show and tell and we'd rotate that. And…that person was…responsible for checking things and chasing up if the environments were down, … making sure the rooms are booked, and the video conferencing is working*' [FP01].

In Rock, the team found it challenging to adapt their work practices to incorporate more agile practices although team members had had prior positive experiences with agile practices. A team member explained, '*Before this project, I was already involved in another agile project where we had... Scrum… – but that was also more of a hybrid because we started off with requirements that were meant to be from waterfall sense of development… But we ended up making it an agile project. But the difference with that one and this one is that this one, there we had the planning sessions … we had stories, and we had estimations for the stories, and we kept to the how many hours you have within the Sprint, and then we did the actual Sprints, and we did the actual show and tells. So, I had, in comparison to this one, it had more Agile components in it*' [GT01].

4.7.2 Discussion of Behavioural Markers for Adaptability

The case findings support the behavioural markers proposed in the Salas Big Five model, particularly the findings from Globe and Tech. In both cases, the teams encountered change, specifically, changes in priorities and a failed product demo, and they developed a new plan to deal with the changes, namely, reduced preplanning and rotating responsibility for show-and-tell sessions. Therefore, we accept the behavioural markers proposed in the Salas Big Five as appropriate to explain adaptability in agile software development project teams. Table 18 shows the behavioural markers from the Salas Big Five model.

Organizations adopt team structures to facilitate organizational adaptability (Burke et al. 2006). Burke et al. (2006) argue that team adaptability is emergent and '*is manifested in the innovation of new or modification of existing structures, capacities, and/or behavioral or cognitive goal-directed actions*' (Burke et al. 2006, p. 1190). Although agile project teams are explicitly designed to adapt to changing circumstances, research addressing agile team

Table 18 Behavioural markers for adaptability

Behavioural markers for adaptability (Salas et al. 2005)	New behavioural markers for adaptability
Identify cues that a change has occurred, assign meaning to that change, and develop a new plan to deal with the changes	Existing marker supported
Identify opportunities for improvement and innovation for habitual or routine practices	Existing marker supported
Remain vigilant to changes in the internal and external environment of the team	Existing marker supported

adaptability is scant. We found two studies. The first, by Salo et al. (2004), was a single case study of an XP team reporting that adaptability can be achieved with little effort by adopting post-iteration reviews (more commonly known as retrospectives) to reflect and plan adjustments to practices and processes. The second study, by Grass et al. (2020), based on 44 interviews across three organizations, found that empowerment was the focal factor in team adaptability. They found that leadership grants empowerment and that the agile team accepts the empowerment, and this interaction underpins team adaptability. Based on the practitioners' understandings, we see the practice of holding retrospective meetings at the end of each development sprint or iteration as the formal mechanism by which agile teams reflect and then adapt (Project Management Institute and Agile Alliance 2017).

Based on our findings and the limited support from the literature, we propose that team adaptability is a factor in agile project teams as posed in the Salas Big Five.

4.8 Team Orientation

Team orientation is defined as the *'Propensity to take other's behaviour into account during group interaction and the belief in the importance of team goal's over individual members' goals'* (Salas et al. 2005, p. 561).

4.8.1 Empirical Findings on Team Orientation

The focus groups resulted in 182 items allocated to team orientation (of 1426; 13%). Team orientation was of relatively low perceived importance as this component was ranked fifth of the eight teamwork components. The main sub-components were 'team cohesion', 'team environment', 'prioritisation of team tasks', 'team member respect', 'responsibility', and 'conflict'. Table 19 shows the main sub-components of team orientation with example items that foster and hinder team effectiveness.

Two case studies provided evidence of a focus on team goals over individual goals. In Globe, the team showed commitment to the tasks in the iteration backlog. The team lead explained, '*The Business Lead will want ten stories done and the project team will only want to do five and there'll be a bit of to-ing and fro-ing and they might agree to six, or they might agree to seven, or they might only want to do five. But it's up to the team. The Business Lead will always try and put a bit of pressure on, which is her job. But it's up to the team to commit to something*' [EP01]. In Tech, the team members committed to the task they had selected, as explained by a team member: '*But there's a sense of ownership, so if you pick something up [a task] you tend to see it through'* [FT04].

Table 19 Main sub-components of team orientation with example items that foster and hinder team effectiveness

Sub-component	Items	
	Foster (total=101)	Hinder (total=81)
Team cohesion (42)	Common goals Good and healthy culture Members have ownership of the plans Understanding of the importance of all team members	Lack of team cohesion Lone wolves Regular changes in direction Unclear goals/demands
Prioritization of team tasks (20)	High motivation level Narrow focus Time for group processes (retrospective …)	Failure to follow agreement Lack of focus among team members Private agenda Not prioritizing the team
Team member respect (19)	Know the team People solution-oriented Respect for individuals	Large ego Strong individualists 'My way is best'
Other	'Common understanding of roles', 'knowledge of customer needs', 'colocation', 'knowledge of scope'	

The Salas Big Five model describes team orientation as an attitudinal dimension, unlike the other dimensions that are described as behavioural. The authors point out that team cohesion is different from team orientation; team cohesion is a desire to work in a particular team, while team orientation is '*a general preference to work in team settings*' (Salas et al. 2005). However, teamwork studies on software development teams identify team cohesion as a factor that strongly improves performance (Dingsøyr et al. 2016). This was also found in a study of agile development teams (Kuthyola et al. 2017). Focus group participants described cohesion with items such as 'common goals' and 'members have ownership of the plans'. In the teamwork literature, team cohesion is often defined as the team '*sticks together and remains united in the pursuit of its goals and objectives*' (Mudrack 1989, p. 781). Hoegl and Gemuenden (2001) use team cohesion as a teamwork quality factor in their model and cite a study by Mullen and Copper (1994) that distinguishes three types of cohesion: 1) interpersonal attraction of team members, 2) commitment to the team task, and 3) group pride or 'team spirit'. Agile practices that involve frequent meetings, such as in daily meetings, joint planning, demonstration, and retrospective meetings, and practices such as pair programming and shared code ownership are likely to make the team members more 'united' and cohesive. Further, agile methods emphasize team commitment to the product backlog. The backlog is a project artefact that aids team cohesion as it provides the task work that leads to the goals and objectives for the teamwork.

4.8.2 Discussion of Behavioural Markers for Team Orientation

There is broad agreement that team orientation is important for team effectiveness. The Salas Big Five model explains that team orientation affects team effectiveness through team members' 1) willingness to engage in mutual performance monitoring and 2) acceptance of feedback and/or assistance through backup behaviour.

The Salas Big Five model has two behavioural markers for team orientation. The first is '*Taking into account alternative solutions provided by teammates and appraising that input to determine what is most correct*' (Salas et al. 2005, p. 561). We find that the sub-component

Table 20 Behavioural markers for team orientation

Behavioural markers for team orientation (Salas et al.)	Proposed new behavioural markers
Taking into account alternative solutions provided by teammates and appraising that input to determine what is most correct	Existing marker supported
Increased task involvement, information sharing, strategizing, and participatory goal setting	Existing marker supported
	The team sticks together and remains united

'team member respect' in the focus group material supports this marker. The second is '*Increased task involvement, information sharing, strategizing, and participatory goal setting*' (Salas et al. 2005, p. 561). Here, we find that the sub-components 'team cohesion' and 'prioritisation of team tasks' in the focus group material support this marker. Globe underscores that the team is committing to tasks in the product backlog, while Tech shows commitment to tasks at an individual level.

A significant obstacle to team orientation is conflict in a team. One of the focus group sub-components was 'conflict', and prior studies have identified relational conflicts in the team contributing to negative effectiveness (Dingsøyr et al. 2016). In contrast, Dingsøyr et al. (2016) found that conflict over tasks positively influences effectiveness. Therefore, collaborative management of conflicts can signify that a team is performing well.

We interpret the first behavioural marker as a sign of willingness to manage task conflicts and that team cohesion is unlikely if there are relational conflicts in a team. We, therefore, propose to keep the first behavioural marker. The second marker involves the following four distinct features: Firstly, increased task involvement was central to the focus group member perceptions of team effectiveness in prioritizing team tasks. Secondly, information sharing was identified in focus groups but placed 'experience sharing' in backup behaviour. We suggest keeping 'information sharing' here as well. Lastly, the focus group material on 'common goals' and 'ownership of plans' is reflected in strategising and participatory goal setting. We, therefore, decided to keep the second behavioural marker. Still, because prior studies of software team effectiveness and perceptions from focus group participants identify team cohesion as an important factor, we decided to add the cohesion marker '*The team sticks together and remains united*'. Table 20 shows the behavioural markers for team orientation.

5 The Agile Teamwork Effectiveness model (ATEM)

This study sought to answer the research question '*What are the coordinating mechanisms, core components, and behavioural markers for effective teamwork in agile software development?*' We drew on a previously published teamwork effectiveness model, the Salas Big Five model, which we carefully evaluated and then revised. We named this new model the 'Agile Teamwork Effectiveness Model', or ATEM. The revision was based on insight from case study and focus group material and support from multi-vocal sources that included the research and grey literature on agile software development. Therefore, the ATEM is research-based and incorporates widely accepted practitioner knowledge for software development teams, as expressed, for example, in the agile practice guide (Project Management Institute and Agile Alliance 2017).

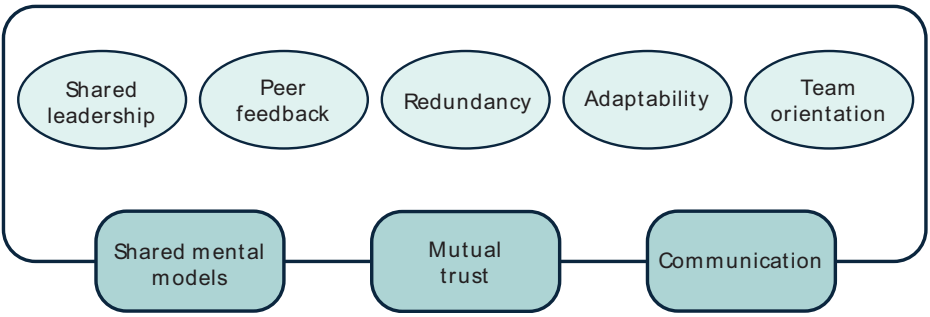

Fig. 3 The ATEM with its five teamwork components and three coordinating mechanisms

Section 4 discussed each coordinating mechanism and component and their associated behavioural markers. This resulted in the ATEM consisting of the following five core components that can explain and predict agile team effectiveness: shared leadership, peer feedback, redundancy, adaptability, and team orientation. Three additional mechanisms coordinate the ATEM components: shared mental models, mutual trust, and communication. The ATEM is depicted in Fig. 3.

The ATEM includes all of the components and coordinating mechanisms in the Salas Big Five model, but we have revised and renamed some to better fit the knowledge garnered from agile team research and practice and reflect common terms in the software engineering domain. The complete list of behavioural markers for the coordinating mechanisms and components of the ATEM, as defined in Section 4, are shown in Tables 21 and 22, respectively.

ATEM is an explanatory model because it describes and explains the factors that make up effective agile teamwork. The model also has a predictive aspect because teams can use it to guide their actions (Gregor 2006). In particular the behavioural markers can guide agile teams as to the behaviours they should support and encourage, whether by adopting particular agile practices or by other means, because when these behaviours are at high levels then teamwork is more likely to be effective.

In the following sections, we critically examine the ATEM. First, we discuss the model, how we believe it can be adapted to different contexts, and how different groups can use it. We end the discussion by discussing limitations and implications and suggesting further work.

Table 21 Coordinating mechanisms with behavioural markers in the ATEM

Teamwork coordination mechanism	Behavioural markers
Shared mental models	Anticipating and predicting each other's needs
	Common understanding of goals
	Common understanding of tasks
	Common understanding of the work process
	Common understanding of the product
	Common understanding of individual skills and expertise
Mutual trust	Information sharing
	Willingness to admit mistakes and accept feedback
	Supportive team social climate
Communication	The team follows up on the progress of tasks
	Visualize project information
	Facilitate informal communication

Table 22 Teamwork components with behavioural markers, ATEM

Teamwork component	Behavioural markers
Shared leadership	The agile team facilitates team problem-solving
	The agile team determines performance expectations and acceptable interaction patterns
	The agile team synchronizes and combines individual team member contributions using agile practices combined with automated tools
	The agile team seeks and evaluates information that affects team functioning
	Agile values and methodologies determine team member roles
	Agile values and methodologies determine the frequency and type of preparatory meetings and feedback sessions
	A servant leader facilitates a boundary-spanning function
	Agile team practices provide a planning function
Peer feedback	Identifying mistakes and lapses in other team members' actions
	Regular feedback regarding team member actions to facilitate self-correction
Redundancy	Recognition by potential backup providers that there is a workload distribution problem in their team
	Shifting of work responsibilities to underutilized team members
	Completion of the whole task or parts of tasks by other team members
Adaptability	Identify cues that a change has occurred, assign meaning to that change, and develop a new plan to deal with the changes
	Identify opportunities for improvement and innovation for habitual or routine practices
	Remain vigilant to changes in the internal and external environment of the team
Team orientation	Taking into account alternative solutions provided by teammates and appraising that input to determine what is most correct
	Increased task involvement, information sharing, strategising, and participatory goal setting
	The team sticks together and remains united

5.1 A Critical Examination of the ATEM

Comparing the ATEM to previously published team and teamwork effectiveness models, one could note that learning is part of previous models. Learning is necessary for teams to be effective because they must learn both the application domain and technology during the team's lifespan (Tiwana 2004). Dingsøyr et al. (2016) show a direct relationship between team learning and team performance, and Janz (1999) has 'cooperative learning' as a factor that influences work outcome. Other team effectiveness models conceptualize 'learning' differently; for example, Hoegl and Gemuenden (2001) treat learning as an outcome in their model as a part of 'team member success'.

Although we have not included learning as an independent component in the models, learning is encompassed within shared mental models, redundancy, and peer feedback. We argue that learning is a factor that influences many aspects of teamwork; therefore, we decided not to include learning as a separate component in the model.

Second, why do we not focus on how a team develops teamwork over time? Prior studies have investigated how agile teams mature over time (Gren et al. 2017). We have focused on defining components, mechanisms, and behavioural markers and not critically discussed advice on which factors to focus on first in teamwork development. In the original Salas Big Five model, Salas et al. (2005) state that the 'importance and prominence' of the components will vary through different stages of team development. They argue that team leadership and team orientation will be important in the early stages, while peer feedback and redundancy will be more important over time. Communication might change over time as studies have found that experienced teams tend to communicate less than inexperienced teams (Salas et al.

2005). We do not have sufficient material in our data to add to this discussion but suggest this as a topic for further work in studies of teamwork effectiveness in software development.

Third, why do we look only at colocated teams who develop non-safety critical software? Colocated teams are the 'original' home ground for agile development methods (Williams and Cockburn 2003), and the case study material is from colocated teams as well as most of the focus group material. A team effectiveness model for distributed agile teams might need to consider additional factors such as reducing the impact of temporal, geographic, and socio-cultural distance (Ågerfalk and Fitzgerald 2006). In the following, we discuss what modifications we see for other types of agile teams.

Fourth, could the model apply to groups as well as teams? Although many organizations label 'groups' as 'teams', they are different, and our model applies to agile teams rather than groups. In the background section, we defined teams as typically having shared goals and task interdependencies, whereas groups tend to have separate goals and tasks, although group members might interact socially and be part of the same organizational structure. Groups, with fewer or no interdependencies, might not need to develop a shared mental model nor need the same level of trust, and communication may be less important. A group, however, would need to establish effective leadership to distribute tasks to competent group members, whereas in knowledge-based teamwork with a shared goal (i.e. to develop a single cohesive product such as software), this is most effectively done through shared leadership. In conclusion, a group might function well with a simplified model with less focus on the coordinating mechanisms.

Fifth, would this model be relevant to a non-agile team? The ATEM is based on a more general model of teamwork effectiveness, and we believe non-agile teams would find the factors in the Salas Big Five model more relevant, such as when they have a formal team leader. A related question is, if you work in an agile team, would the degree of usage of agile methods influence which factors are important in teamwork? An agile method with many ceremonies, such as scrum, could both lead to and require more teamwork than a team using, for example, kanban (Stray et al. 2011). For a kanban team, the flow of work tasks is important, and special emphasis should be put on building redundancy to be able to give priority to the most important tasks. A scrum team might need to focus on team orientation to ensure that the team meets its goals during an iteration. Each team might need to decide on these questions based on their context.

Finally, the Salas Big Five model states propositions about the relationships between the components and the coordination mechanisms. For example, one proposition is that '*effective adaptability requires the existence of shared mental models*' (Salas et al. 2005, p. 583). Why do we not include such relationships in our model? In revising the Salas Big Five model, we rely on support from three data sources, including focus groups, case studies, and the literature. The focus group material does not provide insight into relationships and the case studies only limited evidence. Therefore, we have limited the scope of the work in this article to building a strong case for the components, coordination mechanisms, and behavioural markers based on all three sources. We suggest focusing on the relationships between components and coordination mechanisms in future research.

5.2 The ATEM Extensions – Outside the Comfort Zone

We have focused on a teamwork effectiveness model for collocated teams working on a single product. What would be different if the model were to be used for small teams or in a multi-team, distributed, or safety-critical setting?

A *very small team* with three to four members would not face the same challenges in achieving effective teamwork, as a larger team with, for example, nine members might. Communication, establishing trust, and shared mental models would be easier with fewer team members. Agile practices such as daily meetings and conducting retrospectives could be less time-consuming in smaller teams. However, a small team would have trouble engaging on larger tasks which would take more time. There might also be limited opportunity for peer feedback because of a smaller variation in competence and soft skills in a small team. A small team might rely more on external participation to ensure high-quality peer feedback.

In a *multi-team setting,* we believe all the components and coordinating mechanisms in the model will also be important. If a team has dependencies with other teams, studies of multi-team systems (Marks et al. 2005) suggest that coordinating across teams is needed for the overall success of the set of teams (Bjørnson et al. 2018; Shuffler et al. 2014). Further, good coordination within the team is important for overall coordination between teams (Firth et al. 2015). However, teamwork components have been found to affect team performance differently in the multi-team setting than in single-team settings (Lindsjørn et al. 2018) which used the TWQ model (Hoegl and Gemuenden 2001). Which factors are particularly important in multi-team settings is a topic we currently do not know enough about.

Distributed development provides particular challenges to agile development when teams cannot rely on oral communication and tacit knowledge sharing (Ågerfalk and Fitzgerald 2006). Trust can be difficult to develop without physical meetings in a development team (Moe and Smite 2007), and communication can be more prone to misunderstandings if it is text-based and asynchronous. A shared mental model can be harder to develop if meeting points are few. Team orientation might suffer due to different priorities in parts of an organization or among subcontractors. We think that all of the components and mechanisms are relevant for distributed development but suggest that a team would have to work differently with each component or mechanism in a distributed team or in a virtual agile team, which is a common way of working during the recent global pandemic. Shared mental models, for example, would normally be developed in synchronous meetings in a colocated team. If this is to be developed asynchronously, it might require tasks to be added to the team backlog that focus specifically on learning about the goals and tasks, processes and product, skills and expertise of others, or issues affecting the work. Global development teams need to give particular attention to additional challenges to effective teamwork due to temporal, geographic, and sociocultural distance (Ågerfalk and Fitzgerald 2006; Smite et al. 2010).

Finally, in *safety-critical development,* there is more focus on quality assurance (Kasauli et al. 2018), which would involve more work on peer feedback and possibly more redundancy in competencies to ensure quality in decision making. Such teams will have stricter requirements for precise and accurate communication to ensure that team members do not breach safety requirements. Shared mental models in the team may be more critical and could be supported through the use of quality assurance tools to provide a safety net for developers or through more frequent discussions of the exact definitions of 'done'.

5.3 Limitations

We have described the main limitations of the focus groups and case studies in Section 3. There are further limitations related to the model. We do not claim that the model is generalisable to all contexts although this is mitigated somewhat because the evidence is drawn from two contributing empirical studies that used different research methods, and the evidence was collected in different

countries and continents. We have also supported the arguments for our model by drawing on evidence from multi-vocal literature whenever possible. Further studies of agile teams and teamwork using different research methods might lead to changes to the factors in the model, and we suggest that for future research. When merging our empirical evidence with findings on teamwork in general and teamwork in agile software development in particular, we had to interpret the meanings of a range of concepts. A lack of a shared understanding of concepts in teamwork theory might have led to the inaccurate coding of the data material. We have sought to reduce the impact of potential errors by showing the connection between the data material and our interpretations in Section 4. Further, although we have built the ATEM on solid foundations by drawing on a range of sources, we have not tested whether this model can explain agile team effectiveness better than other models might do. We suggest comparing the ATEM with other models as further work. ATEM might not account for all teamwork factors in agile teams, although we have attempted to identify a 'core' set of factors. Therefore, future adjustments to the model may be needed if new factors emerge. A further limitation of the model is that we have not argued for the potential relationships between components and coordinating mechanisms. The development of testable propositions would complete the model, making it suitable for large-scale field testing.

5.4 How the Model Can be Used: Implications and Future Work

For theory development, the ATEM highlights the core components and coordinating mechanisms for agile software development teams and teamwork. These components and mechanisms can be further examined to better understand why practices in agile development methods are important for effective teamwork. For researchers, it can be easier to relate the effects of development methods or practices to one or more ATEM components or mechanisms than directly to team effectiveness. Further, the ATEM links the software engineering literature to the teamwork literature, which is a source of relevant new insight. Using the ATEM as a basis, we hope researchers will focus studies on topics clearly relevant to effective teamwork and thus enable the research community to better understand the factors which explain and predict agile team effectiveness. As described in the limitations, the model should be tested to see whether it can explain what contributes to team effectiveness better than previous models. Another line of future research would be to gain insight into relationships between components and coordination mechanisms, such as building on propositions developed in the Salas Big Five model. Finally, another line of research could focus on team development over time; for example, how do agile teams develop a shared mental model, or how does an agile team achieve team orientation?

For practitioners, the ATEM could be used by agile teams to help them understand what comprises effective teamwork in an agile context. For example, a prior study, using parts of the focus group material appearing in this current study, found that many teams are unaware of the importance of peer feedback for team effectiveness (Dingsøyr and Lindsjørn 2013). The model and its components, coordinating mechanisms, and behavioural markers could be used by agile team members or team facilitators in retrospectives as a set of topics to address over time or by leaders or agile coaches as topics in internal competence development programmes. In addition, the behavioural markers can be used to evaluate whether a team is acting effectively while its work is underway. Finally, we think the ATEM could help agile teams understand why some of the practices prescribed in the agile development methods are needed and useful, or not, by understanding how a practice contributes to effective agile teamwork.

6 Conclusion

Teams and teamwork are central tenets of agile software development, which is the most widely used approach to developing software systems. However, there is no comprehensive model to explain or predict agile software development teamwork effectiveness. No model sets out the behaviours that effective agile teams display during their teamwork. To address this problem, we have revised a widely accepted general teamwork effectiveness model to fit the needs of agile software development teams. We base our revision on findings from a focus group study, a multi-case study, and a multi-vocal literature review. Our team effectiveness model for agile teams, the ATEM, includes three coordinating mechanisms – shared mental models, communication, and mutual trust – and five components – shared leadership, team orientation, redundancy, adaptability, and peer feedback. We argue that these coordinating mechanisms and components strongly influence team effectiveness.

We have shown how this model can be used by team members, agile coaches, and leaders in software organizations. Further, this model opens new research directions for empirical software engineering, enabling researchers to investigate why and how agile practices and methods contribute to agile team effectiveness.

Appendix 1 – Agenda Workshop Focus Groups

1. *Introduction:* The researchers explained the purpose of and motivation for the workshop and gave an overview of the planned activities.
2. *Group exercise 1:* This exercise included a brief introduction of all group members, then each participant completed a context questionnaire. Next, participants carried out brainstorming sessions focusing on 'What fosters effective teamwork' (documented on green stickers) and then on 'What hinders effective teamwork' (documented on yellow stickers).
3. *Presentation of a team effectiveness model:* The group presented and explained the research-based Salas Big Five model.
4. *Group exercise 2:* Groups presented the results of the brainstorming sessions and categorized the stickers according to the model of team effectiveness. The researchers moderated the placement of stickers in discussion with the group members.
5. *Summary*: The focus groups concluded with a presentation of the group's results, feedback on the workshop, the preparation of a meeting summary, and a discussion of further research on teamwork.

Room setup: The rooms were set up with one table per group. Walls were covered with flip-over charts with numbered areas for grouping stickers from the brainstorming session. Groups were given stickers in the correct colour at the start of each task to avoid participants confusing the colours. At stage 4, groups were given a sheet explaining the teamwork model.

Moderation: The second author moderated the focus groups during agenda items 1 and 3 and the second and third authors during items 2 and 4. The moderation discussion mainly involved deciding where to classify items in the team effectiveness model.

Appendix 2 – Context Questionnaire in the Focus Group Study

Gender	Woman	Man	
Group no			
Age (in years)			
Education (specify the highest degree)	Bachelor's	Master's	PhD
	Other:		
Experience (in years) with software engineering			
Experience (in years) with agile methodology			

How long have you worked in the team?	
How much time do you spend on work in this team?	Specify %
Specify your role in this team.	
How many members are there in your team?	
How many team members are working full time?	
How many other teams are working with the same product?	

Which agile methodology are you using? (possible to have more than one answer)	Scrum		XP	Kanban	Lean
	Other:				
Specify average time (in months) between each release.					
Specify interval time between each iteration.					
Are the team members in your team working in the same location?	To a very small extent	To a small extent	To a moderate extent	To a great extent	To a very great extent
To what extent does the team have daily meetings (stand-up, etc.)?	To a very small extent	To a small extent	To a moderate extent	To a great extent	To a very great extent
To what extent is the whole team part of the planning of every iteration?	To a very small extent	To a small extent	To a moderate extent	To a great extent	To a very great extent
To what extent does the team have reviews/retrospectives after each iteration?	To a very small extent	To a small extent	To a moderate extent	To a great extent	To a very great extent
Specify which collaboration tools you are using in the team	No tool	Task board on the wall	Electronic task board	Spread-sheet (excel…)	Jira
	Jira with Green-hopper	Bug-tracking	Project management tools	Scrum special tools (Scrumworks, etc.)	
	Other:				

Appendix 3 – Case Studies – Interview Schedule

The semi-structured interview schedule for the case studies, (Strode 2016).

1. Background questions

    - Name, job title, years of IT experience, Educational background
    - Describe your job (what is your role on this project) and its goals

2. What are your main work activities (3–5 activities) IN THIS PROJECT

    - For each activity (this will depend on specialization/generalization of work)

    – Main purpose of the activity
    - Depending if work is broken down into distinct activities or not…

– How is work assigned to you?
– How do you know when to start the work/activity?
– Who do you work with to complete the activity [for stakeholder identification]

– Who do you send communications to?
– Who do you receive communications from?

– How do you share out/or delegate the work?
– How do you decide who to share out or delegate the work to?
– What resources (things or information) do you need to complete the activity?
– What technologies do you use to help you carry out the activity? (e.g., email, configuration management tools, wiki, project database, on-line project plan, on-line specification)
– What forms or documents do you need to perform the activity? Examples?
– What is the product or partial product of the activity? (documents, information, software)

– Who relies on the product of this activity?

– Do any of your work products need to be integrated or fit in with other peoples' work or applications?

– Who waits for your work to be completed? [Ask for an example]

• Individual activities:

– How do you know when the activity is complete?
– How do others know when the activity is complete?
– What things hinder this activity

– What do you wait for?
– Negotiate for?
– Bid for?

- What would happen if this activity was not carried out?
- What alternative ways could the outputs or goal of this activity be achieved?

3. Dependency prompts:

    - Who is your vendor(s) [for stakeholder identification]
    - Who is your customer(s) [for stakeholder identification]
    - What other business units do you interact with? [for stakeholder identification]
    - Lifecycle (only used when multiple teams are working on same product for different platforms) – how is this managed/organized?
    - Big picture - how do you achieve an overview of how all of the parts of the system fit together?
    - Testing – test versions - how do you manage/organize this? What types of testing do you do?
    - Parallel development – when two or more people work on the same code module – how do you manage/organize this?
    - Change – when changes are made how does this impact other code modules, documentation, testing?
    - Expertise

        - How do you all know what the product must do?
        - How do you know what the other people on the team are doing on a daily basis within the code?
        - How do you know what the other people on the team are doing on a daily basis within the documentation?
        - How do you know what the other people on the team are doing on a daily basis within the test bank?
        - How do you come to know what the skills and capabilities of the other team members are – how do you know who to ask about things?
    - Historical understanding – how do you find out about previous decisions made that impact

        - The code?
        - This project?
    - Integration – how do you integrate the code and other components?

        - Regularly, randomly, at the end of the project. To a schedule.
    - What is the main source of bugs in the system?

Acknowledgements This work was partially supported by the project Agile 2.0 supported by the Research Council of Norway through grant 236759 and by the companies DNV GL, Equinor, Kantega, Kongsberg Defence & Aerospace, Sopra Steria, and Sticos. We are very grateful to the reviewers, who provided significant advice in developing this article. In addition, we are very grateful to Anastasiia Tkalich at SINTEF Digital and Marit Larsen at TechnipFMC Norway, who provided comments on earlier versions of the manuscript. We would also like to thank everyone who contributed to the research's focus groups and case studies.

Authors' contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Diane Strode, Torgeir Dingsoyr, and Yngve Lindsjorn. The first draft of the manuscript was written by Diane Strode and Torgeir Dingsoyr and all authors commented on subsequent versions of the manuscript. All authors read and approved the final manuscript.

Funding This work was partially supported by the project Agile 2.0 supported by the Research Council of Norway through grant 236759 and by the companies DNV GL, Equinor, Kantega, Kongsberg Defence & Aerospace, Sopra Steria and Sticos.

Data availability The data from the case studies in New Zealand is from a prior conference paper which had ethical approval from the relevant institute and copyright was with the single author (D.E. Strode) under the Creative Commons Attribution. The data in the current submission is publically available in a prior conference paper.

The data from the focus groups is available on request, but most of the material is in Norwegian.

Code availability There is no software or custom code associated with this submission.

Declarations

Conflicts of interest/competing interests There a no known conflicts or competing interests for any of the three authors.

References

Ågerfalk P, Fitzgerald B (2006) Flexible and distributed software processes: old petunias in new bowls? Commun ACM 49(10):27–34

Baham C, Hirschheim R (2021) Issues, challenges, and a proposed theoretical core of agile software development research. Inf Syst J. https://doi.org/10.1111/isj.12336

Beck K (2000) Extreme programming explained: embrace change. Addison-Wesley, Boston

Beck K, Andres C (2005) Extreme programming explained: embrace change, 2nd edn. Addison-Wesley, Boston

Bjørnson FO, Wijnmaalen J, Stettina CJ, Dingsøyr T (2018) Inter-team coordination in large-scale agile development: a case study of three enabling mechanisms. In: Garbajosa J, Wang X, Aguiar A (eds) Agile processes in software engineering and extreme programming XP2018. Porto, Portugal

Burke CS, Stagl KC, Salas E, Pierce L, Kendall D (2006) Understanding team adaptation: a conceptual analysis and model. J Appl Psychol 91(6):1189

Burke CS, Sims DE, Lazzara EH, Salas E (2007) Trust in leadership: a multi-level review and integration. Leadersh Q 18(6):606–632

Chow T, Cao DB (2008) A survey study of critical success factors in agile software projects. J Syst Softw 81(6): 961–971. https://doi.org/10.1016/j.jss.2007.08.020

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

Cohn M (2004) User stories applied: for agile software development. Addison-Wesley Professional, Boston

Coman ID, Robillard PN, Sillitti A, Succi G (2014) Cooperation, collaboration and pair-programming: field studies on backup behavior. J Syst Softw 91:124–134

Converse S, Cannon-Bowers J, Salas E (1993) Shared mental models in expert team decision making. In: Individual and group decision making: current issues, pp 221–246

Costigan RD, Iiter SS, Berman JJ (1998) A multi-dimensional study of trust in organizations. J Manag Issues: 303–317
Crawford ER, LePine JA (2013) A configural theory of team processes: accounting for the structure of taskwork and teamwork. Acad Manag Rev 38(1):32–48
DeFranco JF, Laplante P (2018) A software engineering team research mapping study. Team Perform Manag Int J. https://doi.org/10.1108/TPM-08-2017-0040
Dietz AS, Rosen MA, Wyskiel R, Mendez-Tellez PA, Dwyer C, Salas E (2015) Development of a behavioral marker system to assess intensive care unit team performance. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting
Dingsøyr T, Lindsjørn Y (2013) Team performance in agile development teams: findings from 18 focus groups. In: Baumeister H, Weber B (eds) Agile processes in software engineering and extreme programming, vol 149. Springer, Berlin, pp 46–60. https://doi.org/10.1007/978-3-642-38314-4_4
Dingsøyr T, Fægri TE, Dybå T, Haugset B, Lindsjørn Y (2016) Team performance in software development: research results versus agile principles. IEEE Softw 33(4):106–110. https://doi.org/10.1109/MS.2016.100
Dingsøyr T, Moe NB, Fægri TE, Seim EA (2018) Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. Empir Softw Eng 23(1):490–520. https://doi.org/10.1007/s10664-017-9524-2
Drury-Grogan M (2014) Performance on agile teams: relating iteration objectives and critical decisions to project management success factors. Inf Softw Technol 56:506–515
Duhigg C (2016) What Google learned from its quest to build the perfect team. The New York Times https://www.nytimes.com/2016/02/28/magazine/what-google-learned-from-its-quest-to-build-the-perfect-team.html. Accessed 18 Jan 2022
Edmondson AC (2004) Psychological safety, trust, and learning in organizations: a group-level lens. In: Kramer RM, Cook KS (eds) Trust and distrust in organizations: dilemmas and approaches. Russel Sage Foundation, New York, pp 239–272
Eisenhardt KM (1989) Building theories from case study research. Acad Manag Rev 14(4):532–550. https://doi.org/10.2307/258557
Eisenhardt KM, Graebner ME (2007) Theory building from cases: opportunities and challenges. Acad Manag J 50(1):25–32
Fagerholm F, Ikonen M, Kettunen P, Münch J, Roto V, Abrahamsson P (2015) Performance alignment work: how software developers experience the continuous adaptation of team performance in lean and agile environments. Inf Softw Technol 64:132–147
Firth BM, Hollenbeck JR, Miles JE, Ilgen DR, Barnes CM (2015) Same page, different books: extending representational gaps theory to enhance performance in multiteam systems. Acad Manag J 58(3):813–835
Fitzgerald B, Stol K-J (2017) Continuous software engineering: a roadmap and agenda. J Syst Softw 123:176–189. https://doi.org/10.1016/j.jss.2015.06.063
Flin R, Martin L (2001) Behavioral markers for crew resource management: a review of current practice. Int J Aviat Psychol 11(1):95–118
Garousi V, Felderer M, Mäntylä MV (2019) Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Inf Softw Technol 106:101–121. https://doi.org/10.1016/j.infsof.2018.09.006
Gatfield D (2008) Behavioural markers for the assessment of competence in crisis management Southampton Solent. University [validated by Nottingham Trent University]
Grass A, Backmann J, Hoegl M (2020) From empowerment dynamics to team adaptability—exploring and conceptualizing the continuous agile team innovation process. J Prod Innov Manag 1–28. https://doi.org/10.1111/jpim.12525
Greenleaf RK (2003) The servant-leader within: a transformative path. Paulist Press, New York
Gregor S (2006) The nature of theory in information systems. MIS Q:611–642
Gren L, Torkar R, Feldt R (2017) Group development and group maturity when building agile teams: a qualitative and quantitative investigation at eight large companies. J Syst Softw 124:104–119
Hannay JE, Dybå T, Arisholm E, Sjøberg DI (2009) The effectiveness of pair programming: a meta-analysis. Inf Softw Technol 51(7):1110–1122
Hennel P, Rosenkranz C (2020) Investigating the "socio" in socio-technical development: the case for psychological safety in agile information systems development. Proj Manag J. https://doi.org/10.1177/8756972820933057
Hoda R, Murugesan LK (2016) Multi-level agile project management challenges: a self-organizing team perspective. J Syst Softw 117:245–257
Hoda R, Noble J, Marshall S (2013) Self-organizing roles on agile software development teams. IEEE Trans Softw Eng 39(3):422–444. https://doi.org/10.1109/TSE.2012.30

Hoegl M, Gemuenden HG (2001) Teamwork quality and the success of innovative projects: a theoretical concept and empirical evidence. Organ Sci 12(4):435–449
Hollenbeck JR, Beersma B, Schouten ME (2012) Beyond team types and taxonomies: a dimensional scaling conceptualization for team description. Acad Manag Rev 37(1):82–106. https://doi.org/10.5465/amr.2010.0181
Holtzhausen N, de Klerk JJ (2018) Servant leadership and the scrum team's effectiveness. Leadersh Org Dev J 39:873–882
Hsieh H-F, Shannon SE (2005) Three approaches to qualitative content analysis. Qual Health Res 15(9):1277–1288
Iivari J, Iivari N (2011) The relationship between organizational culture and the deployment of agile methods. Inf Softw Technol 53(5):509–520. https://doi.org/10.1016/j.infsof.2010.10.008
Janz BD (1999) Self-directed teams in IS: correlates for improved systems development work outcomes. Inf Manag 35(3):171–192
Kang H-R, Yang H-D, Rowley C (2006) Factors in team effectiveness: cognitive and demographic similarities of software development team members. Hum Relat 59(12):1681–1710
Kasauli R, Knauss E, Kanagwa B, Nilsson A, Calikli G (2018) Safety-critical systems and agile development: a mapping study. In: 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)
Katzenbach JR, Smith DK (2005) The discipline of teams. Harv Bus Rev 83(7):162
Keutel M, Michalik B, Richter J (2014) Towards mindful case study research in IS: a critical analysis of the past ten years. Eur J Inf Syst 23(3):256–272
Klampfer B, Flin R, Hausler R, Sexton B, Fletcher G (2001) Group interaction in high risk environment (GIHRE) project. Behaviours Markers Workshop, Zurich, Switzerland
Kniberg H (2011) Lean from the trenches: Managing large-scale projects with Kanban. Pragmatic Bookshelf
Kozlowski SW, Bell BS (2012) Work groups and teams in organizations. In: Handbook of psychology, vol. 12: Industrial and organizational psychology, 2nd edn. Wiley, London
Kude T, Mithas S, Schmidt C, Heinzl A (2013) How pair programming influences team performance: the role of backup behavior, shared mental models, and task novelty. Inf Syst Res 30(4):1145–1163. https://doi.org/10.1287/isre.2019.0856
Kuthyola KF, Liu JYC, Klein G (2017) Influence of task interdependence on teamwork quality and project performance. In: Abramowicz W (ed) Business Information Systems BIS 2017 Lecture Notes in Business Information Processing vol 288. Springer, Cham. https://doi.org/10.1007/978-3-319-59336-4_10
LaToza TD, Venolia G, DeLine R (2006) Maintaining mental models: a study of developer work habits. In: Proceedings of the 28th International Conference on Software Engineering
Lehtinen TO, Itkonen J, Lassenius C (2017) Recurring opinions or productive improvements—what agile teams actually discuss in retrospectives. Empir Softw Eng 22(5):2409–2452
Levesque LL, Wilson JM, Wholey DR (2001) Cognitive divergence and shared mental models in software development project teams. J Organ Behav 22(2):135–144
Lindsjørn Y, Sjøberg DIK, Dingsøyr T, Bergersen GR, Dybå T (2016) Teamwork quality and project success in software development: a survey of agile development teams. J Syst Softw 122:274–286. https://doi.org/10.1016/j.jss.2016.09.028
Lindsjørn Y, Bergersen GR, Dingsøyr T, Sjøberg DIK (2018) Teamwork quality and team performance: exploring differences between small and large agile projects. In: Garbajosa J, Wang X, Aguiar A (eds) Agile processes in software engineering and extreme programming XP2018. Porto, Portugal
Lu Y, Xiang C, Wang B, Wang X (2011) What affects information systems development team performance? An exploratory study from the perspective of combined socio-technical theory and coordination theory. Comput Hum Behav 27(2):811–822
Marks MA, DeChurch LA, Mathieu JE, Panzer FJ, Alonso A (2005) Teamwork in multiteam systems. J Appl Psychol 90(5):964
Mathieu J, Maynard MT, Rapp T, Gilson L (2008) Team effectiveness 1997-2007: a review of recent advancements and a glimpse into the future. J Manag 34(3):410–476
McGrath JE (1964) Social psychology: a brief introduction. Holt, Rinehart and Winston, New York
Melo CDO, Cruzes DS, Kon F, Conradi R (2013) Interpretative case studies on agile team productivity and management. Inf Softw Technol 55(2):412–427
Moe NB, Dingsøyr T (2008) Scrum and team effectiveness: theory and practice. In: Lecture notes in business information processes XP2008. Limerick, Ireland
Moe NB, Smite D (2007) Understanding lacking trust in global software teams: a multi-case study. Lecture Notes in Computer Science [Product-focused software process improvement, proceedings]. In: 8th International Conference on Product-Focused Software Process Improvement, Riga, Latvia

Moe NB, Dingsøyr T, Dybå T (2009a) Overcoming barriers to self-management in software teams. IEEE Softw 26(6):20–26. https://doi.org/10.1109/MS.2009.182
Moe NB, Kvangardsnes Ø, Dingsøyr T (2009b) Understanding shared leadership in agile development: a case study. In: 42nd Hawaii International Conference on System Sciences, Hawaii, USA
Moe NB, 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. https://doi.org/10.1016/j.infsof.2009.11.004
Mudrack PE (1989) Defining group cohesiveness—a legacy of confusion. Small Group Behav 20(1):37–49. https://doi.org/10.1177/104649648902000103
Mullen B, Copper C (1994) The relation between group cohesiveness and performance—an integration. Psychol Bull 115(2):210–227. https://doi.org/10.1037//0033-2909.115.2.210
Overeem B (2016) Characteristics of a great scrum team. https://www.infoq.com/articles/great-scrum-team/. Accessed 18 Jan 2022
Paré G (2004) Investigating information systems with positivist case research. Commun Assoc Inf Syst 13(1):18
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(3):303–337. https://doi.org/10.1007/s10664-008-9065-9
Poth A, Kottke M, Riel A (2020) Evaluation of agile team work quality. In: Paasivaara M, Kruchten P (eds) Agile Processes in Software Engineering and Extreme Programming – Workshops, XP 2020, Lecture Notes in Business Information Processing, vol 396 (pp. 101-110). Springer, Cham. https://doi.org/10.1007/978-3-030-58858-8_11
Project Management Institute and Agile Alliance (2017) Agile practice guide. Project Management Institute, Pennsylvania, USA
Rodriguez D, Sicilia MAEG, Harrison R (2012) Empirical findings on team size and productivity in software development. J Syst Softw 85(3):562–570
Rozovsky J (2015) Five keys to a successful Google team. Retrieved 25 December from https://rework.withgoogle.com/blog/fivekeys-to-a-successful-google-team/
Salas E, Sims DE, Burke CS (2005) Is there a "big five" in teamwork? Small Group Res 36(5):555–599
Salas E, Shuffler ML, Thayer AL, Bedwell WL, Lazzara EH (2014) Understanding and improving teamwork in organizations: a scientifically based practical guide. Hum Resour Manag 54(4):599–622
Salo O, Kolehmainen K, Kyllonen P, Lothman J, Salmijarvi S, Abrahamsson P (2004) Self-adaptability of agile software processes: a case study on post-iteration workshops. In: Eckstein J, Baumeister H (eds) 5th International Conference on Extreme Programming and Agile Processes in Software Engineering, XP 2004, Garmisch-Partenkirchen, Germany (vol. LNCS 3092, pp 184–193). Springer, Cham
Sarker S, Xiao X, Beaulieu T, Lee AS (2018) Learning from first-generation qualitative approaches in the IS discipline: an evolutionary view and some implications for authors and evaluators (PART 1/2). J Assoc Inf Syst 19(8):1
Schmidt C (2016) Agile software development teams: the impact of agile software development on team performance. Springer International Publishing. https://doi.org/10.1007/978-3-319-26057-0
Schmidt C, Kude T, Heinzl A, Mithas S (2014) How agile practices influence the performance of software development teams:the role of shared mental models and backup. ICIS 2014 Proceedings
Schmitz K, Mahapatra R, Nerur S (2019) User engagement in the era of hybrid agile methodology. IEEE Softw 36(4):32–40. https://doi.org/10.1109/MS.2018.290100623
Schreier M (2013) Qualitative content analysis. In: Flick U (ed) The SAGE handbook of qualitative data analysis. Sage, London, pp 170–184
Schwaber K, Beedle M (2002) Agile software development with scrum. Prentice Hall, Upper Saddle River
Schwaber K, Sutherland J (2017) The scrum guide. Scrum, Org and ScrumInc https://scrumguides.org/.
Sharp H, Robinson H (2006) Distributed cognition account of mature XP teams. In: 7th International Conference on Extreme Programming and Agile Processing in Software Engineering, Oulu, Finland
Sharp H, Robinson H (2010) Three 'C's of agile practice: collaboration, co-ordination and communication. In: Dingsøyr T, Dybå T, Moe NB (eds) Agile software development: current research and future directions. Springer Verlag, Berlin, p 13
Shastri Y, Hoda R, Amor R (2021) Spearheading agile: the role of the scrum master in agile projects. Empir Softw Eng 26(1):1–31
Shuffler ML, Rico R, Salas E (2014) Pushing the boundaries of multiteam systems in research and practice: an introduction. In: Pushing the boundaries: multiteam systems in research and practice. Emerald Group Publishing Limited, Bingley
Sjøberg DI, Dybå T, Anda BC, Hannay JE (2008) Building theories in software engineering. In: Guide to advanced empirical software engineering. Springer, London, pp 312–336
Skelton M, Pais M (2019) Team topologies: organizing business and technology teams for fast flow. IT Revolution Press. https://teamtopologies.com/book. Accessed 18 January 2022

Smite D, Wohlin C, Gorschek T, Feldt R (2010) Empirical evidence in global software engineering: a systematic review. Empir Softw Eng 15(1):91–118. https://doi.org/10.1007/s10664-009-9123-y
Sommerville I (2016) Software engineering, 10th edn. Boston, Pearson Education Limited
Stewart DW, Shamdasani PN, Rook D (2007) Focus groups: theory and practice. Sage Publications, Thousand Oaks
Stray VG, Moe NB, Dingsøyr T (2011) Challenges to teamwork: a multiple case study of two agile teams. In: Lecture Notes in Business Information Processing 12th International Conference on Agile Software Development (XP2011), Madrid, Spain
Stray V, Sjøberg DI, Dybå T (2016) The daily stand-up meeting: a grounded theory study. J Syst Softw 114:101–124
Stray V, Moe NB, Hoda R (2018) Autonomous agile teams: challenges and future directions for research. In: Proceedings of the 19th International Conference on Agile Software Development XP, pp 1–5. ACM. https://doi.org/10.1145/3234152.3234182
Strode D (2015) In: Burstein F, Scheepers H, Deegan G (eds) Applying adapted big five teamwork theory to agile software development. Proceedings of the 26th Australasian conference on information systems, ACIS 2015, 30 Nov–4 Dec, Adelaide, Australia
Strode D (2016) A dependency taxonomy for agile software development projects. Inf Syst Front 18(1):23–46
Strode DE, Huff SL, Tretiakov A (2009) The impact of organizational culture on agile method use. In: Proceedings of the 42nd Hawaii International Conference on System Sciences, pp 1–9. https://doi.org/10.1109/HICSS.2009.436
Sutherland J, Schwaber K (2020) The scrum guide: the definitive guide to scrum—the rules of the game. https://scrumguides.org/. Accessed 18 Jan 2022
Syed-Abdullah S, Holcombe M, Gheorge M (2006) The impact of an agile methodology on the well being of development teams. Empir Softw Eng 11(1):143–167
Takeuchi H, Nonaka I (1986) The New New Product Development Game. Harvard Business Review, pp. 137-146
Tiwana A (2004) An empirical study of the effect of knowledge integration on software development performance. Inf Softw Technol 46(13):899–906
Van Dierendonck D (2011) Servant leadership: a review and synthesis. J Manag 37(4):1228–1261
Williams L (2012) What agile teams think of agile principles. Commun ACM 55(4):71–76. https://doi.org/10.1145/2133806.2133823
Williams L, Cockburn A (2003) Agile software development: It's about feedback and change. IEEE Comput 36(6):39–43
Yin RK (2018) Case study research and applications: design and methods, 6th edn. Sage Publications, Thousand Oaks
Yu X, Petter S (2014) Understanding agile software development practices using shared mental models theory. Inf Softw Technol 56(8):911–921

Diane Strode is a senior lecturer in the School of Information Technology, Whitireia Polytechnic, New Zealand. She is also a Research Fellow at The Open University in the United Kingdom. Diane has experience as a software developer for Mobil Oil Australia. Her research focus is agile software development and coordination, and she publishes in the domains of information systems and software engineering. Diane has a PhD from Victoria University of Wellington, New Zealand.

Torgeir Dingsøyr is a professor in software engineering – agile at the Department of Computer Science, Norwegian University of Science and Technology. He is further adjunct chief research scientist at the SimulaMet research laboratory. His research has focused on teamwork and learning in software development, as well as development methods for large software projects and programs. He has published in the software engineering, information systems and project management fields.

Yngve Lindsjørn is an associate professor at the Department of Informatics, University of Oslo. He worked for 10 years as a researcher at Norwegian Computing Center from 1987 to 1997. He has 13 years of industry experience as project manager and as a manager (CEO) of a software company within the IT industry. From 2009 to 2014 he was project manager for a research project investigating the effect of teamwork within and across software development teams. His research include software development methods and teamwork factors influencing software project success.

Affiliations

Diane Strode[1] · **Torgeir Dingsøyr**[2,3] · **Yngve Lindsjorn**[4]

[1] School of Information Technology, Whitireia Polytechnic, Wellington, New Zealand

[2] Department of Computer Science, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway

[3] Department of IT Management, SimulaMet, P.O. Box 134, 1325 Lysaker, Norway

[4] Department of Informatics, University of Oslo, Oslo, Norway