Inter-team Coordination in Large-Scale Agile Development: A Case Study of Three Enabling Mechanisms

Finn Olav Bjørnson[1], Julia Wijnmaalen[2], Christoph Johann Stettina[2], and Torgeir Dingsøyr[1,3(✉)]

[1] Department of Computer and Information Science, Norwegian University of Science and Technology, Sem Sælandsvei 9, 7491 Trondheim, Norway
[2] Centre for Innovation, The Hague, Leiden University, Schouwburgstraat 2, 2511 VA The Hague, The Netherlands
[3] Department of Software Engineering, Safety and Security, SINTEF, 7465 Trondheim, Norway
torgeird@sintef.no

Abstract. Agile methods are increasingly used in large development projects, with multiple development teams. A central question is then what is needed to coordinate feature teams efficiently. This study examines three mechanisms for coordination: Shared mental models, communication and trust in a large-scale development project with 12 feature teams running over a four-year period. We analyse the findings in relation to suggested frameworks for large-scale agile development and a theory on coordination, and provide new recommendations for practice and theory.

Keywords: Large-scale agile software development
Multiteam systems · Inter-team coordination · SAFe · LeSS
Project management · Portfolio management

1 Introduction

Agile software development methods are increasingly used in large-scale software development. These projects typically involve multiple teams responsible for the development of numerous features of a solution, and often develop systems that are critical to companies or societies. A study investigating fundamental assumptions within large-scale agile development [1] characterises such projects as having complex knowledge boundaries within them, as well as an interactive complexity and tight coupling with technologies and processes outside the project. A key change from small- to large-scale is that work across boundaries becomes at least as important as work within teams. The topic of inter-team coordination, which has also been included in the research agenda on large-scale agile development [2], is critical in large projects and development programmes.

© The Author(s) 2018
J. Garbajosa et al. (Eds.): XP 2018, LNBIP 314, pp. 216–231, 2018.
https://doi.org/10.1007/978-3-319-91602-6_15

Coordination is often defined as *managing interdependencies* [3]. While there is a growing body of literature on coordination in management science [4,5], in this paper we draw on multiteam system research to increase our understanding of team processes in agile development. Specifically, we refer to three coordinating mechanisms in teamwork proposed by Salas et al. [6]. In line with Scheerer et al. [7] we believe prior work in this field can inform practice in software development, and have therefore chosen this as our theoretical model and for example not theory on coordination modes from sociology [8], or works on agile development such as the model for coordination in co-located development projects [9] or previous empirical studies on agile inter-team coordination [10].

Our findings are based on material from one of the largest software development programmes in Norway, which is described in an exploratory case study [11]. We explore coordination by using three mechanisms proposed by Salas et al. [6], namely Shared Mental Models, Closed-loop Communication and Trust, and by identifying practices that supported the mechanisms in our case. We contrast our findings from the theory and case with the current frameworks for large-scale agile development in order to provide feedback on the current recommendations for practice. We investigate the following research question: *How can knowledge about multiteam systems explain inter-team coordination in a large development programme, and how can this knowledge inform recommendations to practice as expressed in current large-scale development methods?*

The paper starts with outlining present theoretical knowledge on large-scale agile development and coordination. We provide a brief description of research methods, present the findings and discussion structured after the three coordination mechanisms before concluding and providing suggestions for further work.

2 Large-Scale Development and Coordination

Large-scale agile development can be found in practice in at least two distinct forms: (1) in the case of large-scale software development project or program [11] as part of a temporal organisation, and (2) as part of a standing organisation where an IT department engages in ongoing software development embedded in a portfolio management approach [12]. In the following we use the first understanding of large-scale:

2.1 Large-Scale Agile Development: Studies and Practitioner Frameworks

A 15 million-dollar project lasting 28 months to develop a web-based customer booking engine for an American cruise company [13] was one of the first large-scale agile projects studied. The project was distributed and combined Scrum with the Project Management Body of Knowledge Framework. Customers were available but did not work together with developers on a daily basis. Some of the challenges identified in the study were due to the size and involvement of a

high number of internal business sponsors, users, project managers, analysts, and external developers from the UK and India. The communications were mainly formal, and formal documents were needed for changes. However, the project was considered a success, and the study describes the balance between traditional and agile methods as essential in achieving both project control and agility.

Since this study, a small body of studies of large-scale agile development efforts have been published. A transition from traditional plan-based development to agile development is reported by Petersen and Wohlin [14], following a case with three large subsystem components developed by 117 people. Another study shows how Ericsson used communities of practice to support process improvement and knowledge transfer in a large development programme with 40 teams [15]. A third study describes chains of Scrum teams in case organisations with 150, 34 and 5 teams [16]. Further, Bass [17] investigates method tailoring in large-scale offshore development and Dingsøyr et al. [11] describe architectural work, customer involvement and inter-team coordination in a large development program. Scheerer et al. [7] describe large-scale agile development as a multiteam system, and discuss theoretically how coordination can be achieved in this domain.

Implementations of large-scale agile development are often supported by practitioner frameworks, most prominently Scaled Agile Framework (SAFe) or Large-Scale Scrum (LeSS). The Scaled Agile Framework (SAFe) [18] was designed by Dean Leffingwell, based in part on his experiences with the Nokia transformation [19]. The framework is based on the idea of an enterprise model dividing a software development organisation into three parts: Team, Program, and Portfolio. While sometimes criticised as too prescriptive, SAFe is the most applied practitioner framework specifically designed for agile methods at large [20]. The Large-Scale Scrum (LeSS) [21] model was predominantly created by Bas Vodde and Craig Larman to scale the original Scrum framework outside of individual Scrum teams. Similarly to SAFe, LeSS proposes an organisational structure based on teams, accompanied by specific practices and principles. Although coordination is mentioned as a challenge, both frameworks address this only parenthetically.

Coordination is crucial in large-scale development. Begel et al. [22] report on how Microsoft engineers coordinate, finding that coordination is mostly focused on scheduling and features. They further point out that *'more communication and personal contact worked better to make interactions between teams go more smoothly'*. Email was the most used tool to keep track of dependencies on other teams, for developers, testers and also program managers. The study emphasised that *'creating and maintaining personal relationships between individuals on teams that coordinate is indicated by many respondents as a way to successfully collaborate with colleagues'* and finally that *'respondents would like more effective and efficient communication between teams to ease their daily work burden'*. A recent study explains ineffective coordination in large-scale development as resulting from a lack of dependency awareness, due to *'misaligned planning activities of specification, prioritization, estimation and allocation between agile team and traditional inter-team levels'* [23].

2.2 Coordinating Multiple Teams

In 2001 Mathieu, Marks and Zaccaro introduced the term 'multiteam system' to indicate *'..two or more teams that interface directly and interdependently in response to environmental contingencies toward the accomplishment of collective goals. Multiteam system boundaries are defined by virtue of the fact that all teams within the system, while pursuing different proximal goals, share at least one common distal goal; and in doing so exhibit input, process and outcome interdependence with at least one other team in the system'* [24, p. 290].

Unlike a traditional team, a multiteam system is too large and specialised to effectively employ direct mutual adjustment among each and every member of the system [25]. An empirical study by Marks and colleagues [26] points out that coordination *'involves synchronising the efforts of multiple teams in a joint endeavour to handle situational demands'* [26, p. 965]. Hence, cross-team processes appear to predict multiteam performance more than within-team processes [26]. The study by Dingsøyr et al. [11] describes practices for inter-team coordination and found that the case program established far more arenas for coordination than what is recommended in agile frameworks, that these arenas changed over time, and that program participants emphasised the importance of an open-office landscape.

The term coordination has been used for a variety of phenomena. Triggered by the impracticality of the variety of definitions, Salas et al. [6] reviewed existing literature and introduced the 'Big Five' in teamwork. The Big Five in teamwork consists of five components and three coordinating mechanisms. The components are found in almost all teamwork taxonomies: *team leadership, mutual performance monitoring, backup behaviour, adaptability, and team orientation.* While the five components have been previously used to understand teamwork in agile teams elsewhere [27,28], in this paper we will focus on the coordination mechanisms. According to Salas et al. [6], the coordinating mechanisms fuse the values of the five components. The Big Five framework is equally relevant inter-team processes in multiteam systems [29,30].

The three coordinating mechanisms are:

Shared Mental Models are crucial for coordinated effective action [31]. *'For teams to effectively work together, teams must have a clear understanding of the work process, the tasks, and of other teams capabilities'* [6, p. 565]. It is important that all teams share the same mental model so they can interpret contextual cues in a similar manner and make compatible decisions regarding their common goal [32,33]. A shared mental model is even more important in times of stress or in a fast-changing environment as the amount of explicit communication decreases in such situations [6]. Moreover, misaligned mental models can cause conflicts and misunderstandings [34, p. 299] [35, p. 99].

Closed-loop Communication is more than merely developing and sending messages; it also has to do with creating a shared meaning [36, p. 178]. Communication is the simple exchange of information whereas closed-loop communication adds a feedback loop: Was the information received and interpreted correctly [6, p. 568]? This extra feedback loop is pivotal for successful communication

between multiple teams [37, p. 25]. Communication both between and within teams is needed to share information, synchronise actions and keep the shared mental model updated. Also it avoids the noise generated by teams merely focusing on their own tasks [26,36]. Although communication might be hindered by team boundaries through distrust [34].

Trust is defined as the shared belief that teams will perform their roles and protect the interests of their co-workers. [6, p. 561]. Mutual trust is the teams confidence in the character, integrity, strength and abilities of another team [38, p. 106] or group. Trust moderates the relationship between team performance and various other variables [39,40]. Trust is a crucial team process, however it does not develop easily across group boundaries [34,41].

3 Method

The empirical data in this study was gathered in a previous study, described in detail by Dingsøyr et al. [11]. The case was chosen since it was characterised as the most successful large-scale programme in Norway at the time, with extensive use of agile methods. The research question, *how agile methods are adapted at a very large-scale*, was investigated in an interpretative embedded exploratory case study. A case study approach was chosen as it achieves *'the most advanced form of understanding'* [42, p. 236].

In the original study, data was collected from two sources, group interviews [43] and documents. The original study had three focus areas: Customer involvement, software architecture, and inter-team coordination. Three two-hour group interviews were organized with 24 particpants from the programme. The data from the interviews was transcribed, and the transcriptions along with project documents were imported into a tool for qualitative analysis.

A descriptive and holistic coding [44] was then performed on the topic of inter-team coordination. The results were discussed and presented back to the participants which provided input that led to some small revisions of the first findings.

Our findings in this paper are based on the original transcriptions of the inter-team coordination focus area, as well as on all material coded with 'organisation of the programme'. The material was re-analysed with a starting point in the original coding, but with respect to the coordination mechanisms. In particular we looked for description of practices of shared mental models, closed-loop communication and trust at the inter-team level.

4 Case

In this paper we refer to our case as the Perform programme, which was a programme led by a public department, the Norwegian Public Service Pension Fund, further called 'Pension Fund', which required a new office automation system. The programme ran for four years, from January 2008 to March 2012.

4.1 Context

The Pension Fund is a department with about 380 employees who provide 950,000 customers with several types of services. It integrates heavily with another public department. The Pension Fund initiated Perform due to public reform that required new functionality in their existing office automation system. The content of the reform was not known when the programme started, which was one of the main reasons for choosing agile development practices for the project. The goal of the programme was to enable the Pension Fund to provide timely and accurate services and to ensure a cost-effective implementation of the reform.

At the time, Perform was one of the largest IT programmes in Norway, with a final budget of EUR 140 million. It involved 175 people, 100 of whom were external consultants from five companies. In total the programme used about 800,000 person-hours to develop 300 epics, divided into 2500 user stories. The epics were released through 12 releases, from late 2008 to early 2012. The whole program was co-located on one floor in an open-office landscape, with team members seated together. The programme was considered a success in that it delivered the necessary functionality within time and on budget. In the following sections we focus on the period from around 2011 when the programme organisation was large and had time to adjust its work practices.

4.2 Structure and Organisation

Perform was managed as a matrix programme, with four main projects intersecting, mainly through personnel from the feature teams in the development project. The programme was led by a director, focusing mainly on external relations, and a programme manager who focused on operations. Four main projects had their own project manager: Business, architecture, development and test projects.

The *business project* was responsible for analysing needs, and defining and prioritising epics and user stories. Product owners could be found in this project along with employees from the line organisation in the department and technical architects from the development teams who contributed on a partial basis.

The *architecture project* was responsible for defining the overall architecture. This project was staffed by a lead architect as well as staffed on a part-time basis by technical architects from the development teams.

The *test project* was responsible for testing procedures and approving deliverables. The project had a lead tester and part-time testers from the development teams.

The *development project* was itself split into three sub-projects. One was lead by the Pension Fund, consisting of six teams. The other two were led by consulting companies Accenture and Sopra Steria respectively, each with three teams. These development teams worked according to Scrum with three-week iterations, delivering on a common demonstration day. Each team was staffed with a Scrum master, a technical architect, a functional architect, a test responsible, and 4–5 pure developers, a mixture between junior and senior levels.

The development process was influenced by a national contract standard with four phases: analysis, description, construction and approval. The process was staged and continuous so the programme worked on approving the previous phase while constructing the current phase and analysing and describing features in the coming phase.

4.3 Inter-team Coordination

In the following section, we show how coordination mechanisms were influenced by the practices of the programme, and how they developed over time. We only focus on practices at the inter-team level:

Shared Mental Models is observed in the solution descriptions where technical and functional architects worked together to specify on a wiki what was to be made in more in-depth detail than epics or user stories. As the project matured, these descriptions decreased in size, leading to a more effective use of resources in the business project. This is an indication that participants in the programme started to get a shared mental model of the solution, so less text was needed for specification.

Several practices can be seen as contributing to establishing this shared mental model. We have focused on practices relating to a shared understanding of the *work process*, the *tasks* to be done and the shared awareness of *who knew what*.

The *work process* was established early on in the project, and contributed to a homogenous view on the work practices and a shared mental model. Many developers had previous experience working with Scrum and together with the matrix organisation, the specialised roles and iterative four-phase development process that many developers had used before, a common understanding of how work was to be done emerged. In the beginning, formal arenas were used for communication, coordination and learning, but as time went on, more emphasis was placed on informal channels. This could indicate that a shared mental model had emerged so people knew who to contact directly instead of going through the more formal channels. In total we identified 14 arenas for coordination.

Concerning the *tasks*, there were several practices that contributed to a shared mental model. One of the most important practices was the specialised roles within those teams which were shared with other projects. In several cases, the team rotated the employee responsible for the solution description in each iteration. This lead to an overall increase in domain knowledge, and a shared mental model between the development project and the business project, which enabled more efficient collaboration. *'As the contractor starts to understand more about the context, the customers real problem is more visible, and this means we can find a solution together. Fast.'* The project also provided opportunities for people to listen in on other teams, through stand-up meetings. This was particularly useful for the leaders of the architecture and testing teams, who needed to know what was going on in other teams.

The final part of building and maintaining a shared mental model is knowing *who knows what*. Several practices contributed here. As the project was evolving, coordination increasingly took place directly between different teams. Getting to the point of knowing who knew what was a combination of formal arenas in the beginning and a willingness to experiment with those arenas and change them as the needs changed. *'Having enough coordination arenas to know that "Oh, we need to talk" and "This is what we need to talk about in detail". The combination of the semi-structured meetings and those that just happened were the most important in my opinion. But then you need enough of the places where you are informed on what is happening so you actually go and talk to the ones you need to talk to.'* In addition to the arenas where people got to know each other, the specialists in the feature team played key roles in knowing who knew what, since they often had a better overview of other parts of the project.

Closed-Loop Communication at the inter-team level was observed between several teams, but a prominent example was the introduction of mini demos, shared between feature teams and the business project. Early on, common demos were held at the end of every iteration so everyone could provide direct feedback on the implemented solution. This delayed feedback as iteration length was three weeks, and teams introduced the concept of mini-demos during iterations where they received rapid feedback from the customer in the business project to ensure they had understood the requirements correctly. The fact that the Pension Fund was able to provide upwards of 30 of their best line operatives to man the business team of the project was a great benefit to the developers who could ask directly for clarifications on user stories.

Many emphasised the importance of informal coordination arenas enabled by the co-location and open landscape when it came to the efficient (close-loop) communication in the programme. Even though there were many official coordination arenas, like the daily stand-up, a Scrum of Scrum between teams in the sub-projects and a meta-scrum covering the entire programme, a lot of the coordination happened directly between members of different teams who had to make sure the information they passed through the formal arenas was received and interpreted correctly.

The formal arenas were necessary, however, as a project manager expressed it: *'Having processes to work with dependencies is important for the maturation of the work, to get an improved understanding. But you need frequent local control and cooperation to really figure it out.'* The formal arenas seemed most important in the beginning as an enabler of later informal communication, as expressed by an architect: *'I imagine these arenas are most important in the beginning, but the importance decreases as you get to know each other. You get used to just walking over to the person you know can fix this thing for you and talking to him directly.'*

There seemed to be a widespread belief from all participants that informal direct communication would improve or enable better communication between teams in the programme. The program sought to improve the informal

communication by physically moving teams around in the landscape. *'And then we had some conscious choices to cover the informal communication. A Pension Fund-team that started working with the GUI-part far into the work-flow, they were moved over there and we switched the placement of those two teams [demonstrates on whiteboard]. We wanted that team and that team to work closer together. That was a conscious choice from the Pension Fund and us.'*

Trust. An example came early in the programme, when there was a delay in delivery. Some in the management in the Pension Fund wanted to monitor the programme more closely. But the director of the Pension Fund stated: *'Let the people who know how to work, work!'* A project manager states the outcome of this decision: *'We had a delivery which we missed, and it was touch and go if we should be allowed to continue. [..] The fact that we were allowed to mature was important. I feel the suppliers learned from each other and used the others to grow.'*

There are several other examples in the material illustrating that trust was given between teams. For example, the manager team trusted the feature teams to take the right actions. *'They [feature team] told us [manager team] that they exchanged tasks, more like a checkup that if you don't mind, we've done so and so, because its better. -OK Go!'* This was also a result of a shared mental model: The teams identified each other as better suited to their respective tasks. In another example, the feature teams trusted the other teams to respect their need for shielding during hectic periods. *'You know that my team needs to be shielded for a time, and then you just walk over to (sub-project manager) and tell him.'*

Some of the trust seems to be built on the informal coordination that was enabled through the more formal arenas, as we have described previously. The matrix structure and specialised roles in the feature teams also contributed. The feature teams had to trust the business team to deliver enough work for an iteration and the business team had to trust the feature teams to swap tasks if they identified that other teams were more suited. There was much openness between teams and between suppliers who were otherwise competitors. The open workspace and common lunch area also seem to have contributed to the relaxed and trusting atmosphere between the teams. The mix of informal and formal arenas provided developers from different teams the opportunity to discuss problems and suggest solutions informally as they cropped up, but the final decision to implement was made in a formal arena. This practice allowed developers to handle problems autonomously, and made sure that there was a proper way to disseminate the solution to other teams.

5 Discussion

We return to our research question: *How can knowledge about multiteam systems explain inter-team coordination in a large development programme, and how can*

this knowledge inform recommendations to practice as expressed in current large-scale development methods?

Rentsch and Staniewisz [45, p. 226] hypothesise that 'coordination mechanisms need to function at the team, inter-team and system levels in order for the multiteam system to utilise their available capacity fully'. As a program delivering on time and cost, we know that the Perform program utilised their capacity well, and we also know that the program continuously worked on improving work practice, which led to changes in coordination practices. The feature teams followed Scrum with practices such as iteration planning, daily meetings, demonstrations and retrospectives on team level. In the following section, we discuss our two research questions for each of the coordination mechanisms:

5.1 Shared Mental Model

A Shared Mental Model includes a clear understanding of the work process, the tasks, and of other teams capabilities [6, p. 565]. In the results, we showed how the solution descriptions were reduced in size, which is an indication of a shared mental model.

We believe the Scrum development method can be seen as a powerful shared mental model, as it is easy to understand with few roles, work practices and artefacts. The programme added new roles and projects on top of this, building on the contract model which was widely known amongst participants. All teams demonstrated developed functionality every three weeks. Adding more roles, work practices and artefacts risks limiting the function of a development method as a shared mental model. With the new frameworks SAFe and LeSS, we believe LeSS is closest to the model used in the Perform case. SAFe is a more complex model, and we believe this model will require more effort to function as a shared mental model. Building on a lightweight framework such as Scrum could help develop a shared mental model.

The fact that all the teams in the project are located on the same floor means that project members walk by the progress boards of other teams and this aids the formation of a shared mental model. The matrix model meant that projects such as business and development worked together to develop a shared understanding of tasks prior to actual development. Furthermore, joint responsibility from the development and testing projects led to a shared understanding of quality requirements on the codebase. Placing a team together physically is common advice in the agile practitioner literature, but having a matrix organisation is more controversial. Scrum describes only the role of a facilitator and team members for a development team. Our case suggests that adding roles on the teams and organising the programme in a matrix structure is important to develop a shared mental model. Additional roles could help develop a shared mental model of tasks.

The open-office landscape led to overall insight in the work across teams as the progress boards were visible, and it was easy to see which teams were having discussions after daily stand-up meetings. We also described the number of meetings in the beginning of the programme as something that established

knowledge of who knows what. Developing a shared mental model of who knows what requires extra effort in projects with people new to a domain and the project organisation.

5.2 Closed-Loop Communication

Agile development methods have led to shorter feedback loops in development projects. In the Perform program, the iteration length was three weeks. *Closed-loop communication* is addressed by the frequent review and coordination practices in and across teams through planning, demonstrations or product backlog refinement, as well as through frequent discussions with the teams and relevant stakeholders in the preparation and review of partial results. This is also described in the SAFe and LeSS frameworks. Perform introduced the new practice of mini-demos to improve the closed loop communication between the development and business projects.

We believe that closed-loop communication was enhanced in the Perform case by frequent feedback as prescribed by agile practices, by the physical proximity of teams situated together on one floor, and by the tailored work methods to improve communication such as with the new practice of 'mini-demos'.

5.3 Trust

In the theory section we stated that closed-loop communication can be hindered by team boundaries through distrust. The Perform program involved a number of companies with their own work cultures, yet participants seem to have developed trust (as explained in our results section) both from top to bottom (as illustrated by the Programme Director's *"Let the people who know how to work, work!"*, and trust between teams as described in understanding that teams needed to be shielded during an iteration. Agile methods with focus on transparency and feedback loops are well suited to develop trust.

Other practices such as common work spaces and shared lunches also created a form of identity and stimulated contact, both of which increased the level of inter-team trust. The amount of open and informal communication and decision-making has been shown to be indicators of how trust develops [46]. Additionally, trust literature shows that trust develops if there is more interpersonal contact [47]. There is a negative relationship between team size and interpersonal contact. This relation between team size and trust might explain why the agile methods seem to work better in smaller organisations compared to larger ones.

However, relationships within teams and the context around teams are not static. Hence the static nature of the practical models does not align with the reality of how relations develop in multiteam systems.

Reflective sessions such as retrospectives used in Perform and described in both SAFe and LeSS will likely influence the amount of *trust*. Practices, artefacts and roles facilitate and stimulate coordination, yet one very influential factor hardly receives attention: the human. For example, literature on trust tells us that it does not develop easily across team boundaries [34] and that it influences

the amount of communication [36]. Trust is critical in large-scale development and is challenging to develop. Co-location and the agile practices of frequent delivery seem to develop trust.

5.4 Limitations

The main limitations of this study were that the group interviews were performed after the programme was completed, and that we were not able to follow the case over time. Second, the data collection was not particularly targeted at the mechanisms identified in the multiteam systems literature, but rather structured around broad questions of how the programme organised inter-team coordination. See [11] for further discussion of the limitations of our study.

6 Conclusion

In this paper we used current multiteam systems literature to help understand inter-team coordination processes in large-scale agile project teams. In particular, we use the three coordinating mechanisms proposed by Salas et al. [6] to understand inter-team coordination in practice.

Our results indicate that using the multiteam systems perspective on large-scale agile teams is useful as it provides reasons for practices that are described in agile development frameworks. A shared mental model, closed-loop communication and trust have been identified as important coordination mechanisms in teamwork in multiteams. The findings from our case show the relevance of these three coordinating mechanisms for large-scale agile development and underline the importance of inter-team coordinating mechanisms compared to intra-team coordination.

The three mechanisms are interrelated and their combined effect influences the project's success. The practices suggested in large-scale frameworks indicate that many practices contribute to or influence more than one coordination mechanism at the same time. From the discussion, the following conclusions follow on the coordination mechanisms:

- Building on a lightweight framework such as Scrum helps develop a shared mental model of the development process.
- Additional roles could help develop a shared mental model of tasks.
- Developing a shared mental model of who knows what requires extra effort in projects with people new to a domain and the project organisation.
- Closed-loop communication was developed due to a combination of (1) frequent feedback as prescribed by agile practices, (2) co-location on one floor, and (3) tailoring of work methods to improve communication such as the practice of 'mini-demos'.
- Trust is critical in large-scale development and more challenging to develop than in small-scale scenarios.
- Co-location and the agile practices of frequent delivery seem to develop trust.

There is a growing number of studies on multiteam systems which we believe are relevant for practitioners and researchers in large-scale agile development. In particular it would be interesting for future research to further explore how human aspects influence coordination, following up on Begel et al. [22] who found that creating and maintaining personal relationships was critical to good coordination. In the future, researchers should draw further on findings in the multiteam systems field to provide better advice on how these aspects can be fostered in development methods.

Acknowledgement. This work was in partial supported by strategic internal projects at SINTEF on large-scale agile development and the project Agile 2.0 supported by the Research council of Norway through grant 236759 and by the companies Kantega, Kongsberg Defence & Aerospace, Statoil, Sopra Steria, and Sticos.

References

1. Rolland, K.H., Fitzgerald, B., Dingsøyr, T., Stool, K.J.: Problematizing agile in the large: alternative assumptions for large-scale agile development. In: International Conference on Information Systems (2016)
2. Dingsøyr, T., Moe, N.B.: Towards principles of large-scale agile development. In: Dingsøyr, T., Moe, N.B., Tonelli, R., Counsell, S., Gencel, C., Petersen, K. (eds.) XP 2014. LNBIP, vol. 199, pp. 1–8. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14358-3_1
3. Malone, T.W., Crowston, K.: The interdisciplinary study of coordination. ACM Comput. Surv. (CSUR) **26**(1), 87–119 (1994)
4. Mintzberg, H.: Mintzberg on Management: Inside Our Strange World of Organizations. Simon and Schuster (1989)
5. Jarzabkowski, P.A., Le, J.K., Feldman, M.S.: Toward a theory of coordinating: creating coordinating mechanisms in practice. Organ. Sci. **23**(4), 907–927 (2012)
6. Salas, E., Sims, D.E., Burke, C.S.: Is there a big five in teamwork? Small Group Res. **36**(5), 555–599 (2005)
7. Scheerer, A., Hildenbrand, T., Kude, T.: Coordination in large-scale agile software development: a multiteam systems perspective. In: 2014 47th Hawaii International Conference on System Sciences, pp. 4780–4788. IEEE (2014)
8. Dingsøyr, T., Brede Moe, N., Amdahl Seim, E.: Coordinating Knowledge Work in Multi-Team Programs: Findings from a Large-Scale Agile Development Program. ArXiv e-prints, January 2018
9. Strode, D.E., Huff, S.L., Hope, B.G., Link, S.: Coordination in co-located agile software development projects. J. Syst. Softw. **85**(6), 1222–1238 (2012)
10. Sharp, H., Robinson, H.: Three CS of agile practice: collaboration, co-ordination and communication. In: Dingsøyr, T., Dybå, T., Moe, N. (eds.) Agile Software Development, pp. 61–85. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12575-1_4
11. Dingsøyr, T., Moe, N.B., Fægri, T.E., Seim, E.A.: Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. Empirical Softw. Eng. **23**(1), 490–520 (2018)
12. Stettina, C.J., Hörz, J.: Agile portfolio management: an empirical perspective on the practice in use. Int. J. Project Manage. **33**(1), 140–152 (2015)

13. Batra, D., Xia, W., VanderMeer, D., Dutta, K.: Balancing agile and structured development approaches to successfully manage large distributed software projects: a case study from the cruise line industry. Commun. Assoc. Inf. Syst. **27**(1), 21 (2010)
14. Petersen, K., Wohlin, C.: The effect of moving from a plan-driven to an incremental software development approach with agile practices. Empirical Softw. Eng. **15**(6), 654–693 (2010). ISI Document Delivery No.: 653OB Times Cited: 2 Cited Reference Count: 46 Petersen, Kai Wohlin, Claes. Springer, Dordrecht
15. Paasivaara, M., Lassenius, C.: Communities of practice in a large distributed agile software development organization - case ericsson. Inf. Softw. Technol. **56**(12), 1556–1577 (2014)
16. Vlietland, J., van Vliet, H.: Towards a governance framework for chains of scrum teams. Inf. Softw. Technol. **57**, 52–65 (2015)
17. Bass, J.M.: How product owner teams scale agile methods to large distributed enterprises. Empirical Softw. Eng. **20**(6), 1525–1557 (2015)
18. Leffingwell, D.: SAFe 4.0 Reference Guide: Scaled Agile Framework for Lean Software and Systems Engineering. Addison-Wesley Professional (2016)
19. Laanti, M., Salo, O., Abrahamsson, P.: Agile methods rapidly replacing traditional methods at nokia: a survey of opinions on agile transformation. Inf. Softw. Technol. **53**(3), 276–290 (2011)
20. VersionOne: 11th annual survey. the state of agile (2016)
21. Larman, C., Vodde, B.: Large-Scale Scrum: More with LeSS. Addison-Wesley Professional, Boston (2016)
22. Begel, A., Nagappan, N., Poile, C., Layman, L.: Coordination in large-scale software teams. In: Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering, pp. 1–7. IEEE Computer Society (2009)
23. Bick, S., Spohrer, K., Hoda, R., Scheerer, A., Heinzl, A.: Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. IEEE Trans. Softw. Eng. (2017)
24. Mathieu, J., Marks, M.A., Zaccaro, S.J.: Multi-team systems. Int. Handb. Work Organ. Psychol. **2**, 289–313 (2001)
25. Davison, R., Hollenbeck, J.: Boundary spanning in the domain of multiteam systems. In: Multiteam systems. An Organization Form for Dynamic and Complex Environments, pp. 323–362. Routledge (2012)
26. Marks, M.A., DeChurch, L.A., Mathieu, J.E., Panzer, F.J., Alonso, A.: Teamwork in multiteam systems. J. Appl. Psychol. **90**(5), 964 (2005)
27. Moe, N.B., Dingsøyr, T.: Scrum and team effectiveness: theory and practice. In: Abrahamsson, P., Baskerville, R., Conboy, K., Fitzgerald, B., Morgan, L., Wang, X. (eds.) XP 2008. LNBIP, vol. 9, pp. 11–20. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68255-4_2
28. Stettina, C.J., Heijstek, W.: Five agile factors: helping self-management to self-reflect. In: O'Connor, R.V., Pries-Heje, J., Messnarz, R. (eds.) EuroSPI 2011. CCIS, vol. 172, pp. 84–96. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22206-1_8
29. Shuffler, M.L., Rico, R., Salas, E.: Pushing the boundaries of multiteam systems in research and practice: an introduction. In: Pushing the Boundaries: Multiteam Systems in Research and Practice, pp. 3–16. Emerald Group Publishing Limited (2014)
30. Wijnmaalen, J., Voordijk, H., Rietjens, B.: MTS coordination in practice: micro-level insights to increase MTS performance. Team Perform. Manage. Int. J. **24**(1/2), 64–83 (2017)

31. Zaccaro, S.J., Rittman, A.L., Marks, M.A.: Team leadership. Leadership Q. **12**(4), 451–483 (2002)
32. Cooke, N.J., Salas, E., Cannon-Bowers, J.A., Stout, R.J.: Measuring team knowledge. Hum. Fact. J. Hum. Fact. Ergon. Soc. **42**(1), 151–173 (2000)
33. Mathieu, J.: Reflections on the evolution of the multiteam systems concept and a look to the future. In: Multiteam Systems: An Organization Form for Dynamic and Complex Environments, pp. 511–544 (2012)
34. Hinsz, V.B., Betts, K.R.: Conflict multiteam situations. In: Multiteam Systems: An Organization Form for Dynamic and Complex Environments, pp. 289–322 (2012)
35. DiazGranados, D., Dow, A.W., Perry, S.J., Palesis, J.A.: Understanding patient care as a multiteam system. In: Pushing the Boundaries: Multiteam Systems in Research and Practice, pp. 95–113. Emerald Group Publishing Limited (2014)
36. Keyton, J., Ford, D.J., Smith, F.L., Zacarro, S., Marks, M., DeChurch, L.: Communication, collaboration, and identification as facilitators and constraints of multiteamsystems. In: Multiteam Systems: An Organization Form for Dynamic and Complex Environments, pp. 173–190 (2012)
37. McIntyre, R.M., Salas, E.: Measuring and managing for team performance: emerging principles from complex environments. In: Team Effectiveness and Decision Making in Organizations, pp. 9–45 (1995)
38. Earley, P.C., Gibson, C.B.: Multinational Work Teams: A New Perspective. Routledge, Mahwah (2002)
39. Costa, A.C., Roe, R.A., Taillieu, T.: Trust within teams: the relation with performance effectiveness. Eur. J. Work Organ. Psychol. **10**(3), 225–244 (2001)
40. Dirks, K.T.: The effects of interpersonal trust on work group performance. J. Appl. Psychol. **84**(3), 445 (1999)
41. Williams, M.: In whom we trust: group membership as an affective context for trust development. Acad. Manag. Rev. **26**(3), 377–396 (2001)
42. Flyvbjerg, B.: Five misunderstandings about case-study research. Qual. Inq. **12**(2), 219–245 (2006)
43. Myers, M.D., Newman, M.: The qualitative interview in is research: examining the craft. Inf. Organ. **17**(1), 2–26 (2007)
44. Saldaña, J.: The Coding Manual for Qualitative Researchers. Sage (2015)
45. Rentsch, J.R., Staniewicz, M.J.: Cognitive similarity configurations in multiteam systems. In: Multiteam Systems: An Organizational Form for Dynamic and Complex Environments, pp. 225–253 (2012)
46. Currall, S.C., Judge, T.A.: Measuring trust between organizational boundary role persons. Organ. Behav. Hum. Decis. Process. **64**(2), 151–170 (1995)
47. Mayer, R.C., Davis, J.H., Schoorman, F.D.: An integrative model of organizational trust. Acad. Manag. Rev. **20**(3), 709–734 (1995)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.