Team Autonomy in Large-Scale Agile

Nils Brede Moe
SINTEF
nils.b.moe@sintef.no

Bjørn Dahl
NTNU
bjornhd@stud.ntnu.no

Viktoria Stray
University of Oslo, SINTEF
stray@ifi.uio.no

Lina Sund Karlsen
NTNU
linaska@stud.ntnu.no

Stine Schjødt-Osmo
NTNU
stinsc@stud.ntnu.no

Abstract

Large-scale software development is increasingly making use of agile practices. In large-scale projects, a team needs to align with other teams and the rest of the organization. This has been shown to threaten team autonomy, which, in turn, reduces responsiveness and flexibility. Hence, agile teams face challenges in adapting to larger-scale development. We conduct a multiple case study of three large-scale projects to investigate barriers to team autonomy in large-scale agile. Two barriers are identified: overall direction and external dependencies. We found that goals are often set by management without involving the teams, that they are often equal to deliverables and deadlines, and that team members often do not know what the goals are. Consequently, teams struggle with setting and communicating goals as well as establishing a shared direction. Organizational dependencies lead to teams having to deal with additional tasks, resulting in specific members shielding the teams from external noise.

1. Introduction

Large software development projects are increasingly adopting agile development practices. Teams in large-scale projects need to reach agreement on many decisions with experts, managers, stakeholders and other teams [28]. Further, quality concerns and the need for frequent and coordinated releases forces companies to govern, control, and standardize multi-team development efforts. Therefore, the agile team working in a large-scale environment needs to be aligned with other teams and the rest of the organization. If the team breaks the quality or functionality or is late, it will affect other teams and deliverables. However, the need for aligning the work, processes, and technology and for coordinating externally is a threat to team autonomy, which is the key to agility.

The notion of autonomy and self- management is not new; research in this area has been around since Trist and Bamforth's study of self-regulated coal miners in the 1950s [34]. We use the label "self-managing teams" as a synonym for "autonomous teams," and for "empowered teams." Guzzo and Dickson [11] describe such teams as

> *employees that typically perform highly related or interdependent jobs, who are identified and identifiable as a social unit in an organization, and who are given significant authority and responsibility for many aspects of their work, such as planning, scheduling and assigning tasks to members, and making decisions with economic consequence.*

While autonomous agile teams promise to increase employee motivation and job satisfaction significantly [15], as well as boost creativity and productivity [9], implementing such teams in a large-scale context is challenging. When many agile teams are working toward the same goal, a lot of coordination and management effort is required [7], and the team cannot have full authority over all aspects of the work as a single one-project team. Further, in large innovative projects, the degree of complexity and uncertainty is high, as the work executed in teams is influenced by the work and inputs from other teams. While there is a need for alignment and coordinated decision-making, Tata and Prasad [33] claim that team members need to affect managerial decisions genuinely in order to benefit from self-management. Otherwise, they will experience only symbolic self-management, and if the managerial decisions are only affected by symbolic input, the team members might hesitate to embrace self-management. Furthermore, for autonomous agile teams to work together in a large-scale project, there is a need for organizational control and alignment, for the teams to be able to collaborate toward achieving the desired objective. Therefore, a single team cannot be entirely autonomous in a large-scale environment.

URI: https://hdl.handle.net/10125/60136
ISBN: 978-0-9981331-2-6

A question is, then, how to align teams without reducing team autonomy? What team external dependencies and managerial decision hinders team autonomy in large-scale agile? What reduces team authority in large-scale agile projects?

Motivated by the importance of team autonomy in agile software development and the need for alignment, the main goal of this paper is to understand the enablers and barriers of autonomy, and to explore the conflict between autonomy on a team level and the need for organizational control in large-scale agile software development. Our research question is:

What are the barriers to team autonomy in large-scale agile?

In this paper, we examine team autonomy in the context of large-scale agile software development. We understand large-scale development as a development effort with many teams—from 3 to 20 teams [21].

The remainder of the paper is organized as follows: In Section 2, we present background information on autonomous agile teams and large-scale agile. In section 3, we describe our research method in detail. In Section 4, we present results from an inductive multiple case study of three distinct large-scale projects across three cases. We discuss our findings in Section 5. Section 6 concludes and presents key findings from the study.

2. Autonomy in large-scale agile

In this section, we present background information on autonomous agile teams and challenges in implementing them. Second, we describe how agile teams are coordinated in large-scale agile.

2.1. Autonomous agile teams and their barriers

Agile teams usually consist of many—not necessarily rigid—roles [32]. Back and Anders describe the roles typically found in an agile software development team [1]: developer/programmer, tester, architect, interaction designer, and project manager/product owner. To make all of these roles work together in a team, autonomous agile teams often make use of certain team practices or ceremonies. *Stand-up meetings,* which are short, daily meetings in which team members share their work progress and possible impediments, are used to keep track of the progress of the software [30]. *Retrospectives* are another popular practice. These are meetings in which the team members reflect on past work processes: what we did well, what we want to keep doing, and what we want to do more of [18]. During a retrospective, the team members discuss possible measures that can ultimately improve the sustainability of the team. Furthermore, doing retrospectives frequently is associated with the business value in the long run (ibid).

According to Moe, Dingsøyr, and Dybå [22], autonomous agile teams should be responsible for planning and scheduling their work, as letting the individuals participate in these activities will increase their commitment to the team plan. Scrum and Kanban are examples of bottom-up self-determined work designs that Parker and Wall [25] consider a defining feature of autonomous teams. Stand-up meetings and retrospectives are also practices well within the aspect of control and management in the definition given of an autonomous team by Goodman, Devadas, and Hughson [10]. In other words, as Moe et al. [22] allude to, the research itself is not new, it has just found a new area of application.

Understanding how to enable autonomous software development teams requires more than just examining the team's inner workings. We must also understand the barriers at the team and organizational levels. In an international workshop on autonomous agile teams, Stray et al. [29] revealed the top barriers to be not having clear and common goals, lack of trust, too many dependencies to others, lack of coaching and organizational support, and diversity in team norms. Further, Moe et al. [22] identify several team-level challenges in a case study of a single agile team: individual commitment, failure to learn, and individual leadership. Individual commitment is linked to a lack of commitment to the team goals; they found that team members tended to pursue their own individual goals instead of the team goals. Failure to learn concerns process improvement; even though the team members frequently discussed potential changes, they did not implement them. One reason was that the management did not set aside time for process improvement. Moe et al. (ibid) claim that if a team is not given the possibility to improve, it will experience only symbolic self-management, as explained by Tata and Prasad [33].

Additionally, Stray, Moe, and Dingsøyr [31] found that even though agile methods were implemented, critical decisions were, in some cases, made by the project managers without involving the developers. These findings are supported by Moe et al. [22] who found that even though the concept of shared leadership was introduced to the teams in their study, team members did not change their decision-making processes. This behavior led to difficulties in aligning decisions when team members did not know what others were doing. Important decisions were also made without informing the rest of the team, which led to a low level of trust. For autonomous agile teams to be successful, Moe, Aurum, and Dybå [20] argue that team members

need to identify important decisions they should make together to be able to make the shared decisions they are supposed to.

Implementation of autonomous teams is difficult if there are barriers on the organizational level. One of these barriers is organizational control. Moe et al. [22] found that certain forms of detailed control by the management inhibit autonomy because the whole point is that the teams should control themselves. Boehm and Turner [2] argue that this is where the project manager comes in; one of the project manager's primary roles is to be the barrier between the organization and the team, preventing unnecessary interruptions.

The two other challenges on the organizational level are shared resources and specialist culture [22]. Shared resources entail that projects fighting for resources and the most skilled employees rarely build redundancy. In other words, sharing resources across several projects threatens the autonomy. Specialist culture is a result of organizations supporting and incentivizing being the best at what one does rather than creating generalists who can fill each other's functions (ibid).

2.2. Teams in Large-scale agile

Coordinating externally is an issue for autonomous agile teams in large-scale agile. No team possesses all the knowledge needed to solve complex tasks. Therefore, teams need to coordinate work with other teams and experts. Further, as teams learn, products become more mature, and the development process changes, coordination mechanisms in large-scale agile change [21]. According to Boehm and Ross [4], the primary problem with project coordination is that stakeholders such as users, customers, the development team, and the management have to be simultaneously satisfied. This view is supported by Pikkarainen, Haikara, Salo, Abrahamsson, and Still [26], who claim that agile practices do not provide communication mechanisms in situations where many teams are involved in the same development process. Scrum and Kanban are for single teams, and not meant for cross-team communication. As a consequence, according to Pikkarainen et al. (ibid), they are not tools for coordinating multiple teams or projects at the same time. Nyrud and Stray [24] identified 11 coordination mechanisms in a large-scale agile project and concluded that ad hoc conversations were the most important.

In a large-scale setting, the most common strategy for coordination across several teams is Scrum of Scrum. Scrum of Scrum is a scheduled meeting at which one team member acts as "ambassador" to participate in a daily meeting with ambassadors from other teams. However, Scrum of Scrum has been found to be inefficient in larger projects.

Because of challenges coordinating work in large-scale agile, agile consultants have created several frameworks for scaling agile, such as the Large-Scale Scrum (LeSS) [16] and Scaled Agile Framework (SAFe) [17]. The LeSS framework offers less structure and gives suggestions, tools, and tips for practices that can be used for coordination, such as communities of practice and scheduled multi-team meetings. In the LeSS, any team or team member should be able and expected to reach out to another team if there is an issue to be solved (both through scheduled and unscheduled meetings). The LeSS can be understood as a bottom-up approach to coordination and gives the autonomous agile team authority to adjust practices. The SAFe seems to create a structure with more organizational control, which might leave less flexibility for meetings to emerge and for teams to take the initiative for coordination.

3. Research design and method

The goal of this research is to understand barriers to team autonomy in large-scale agile development. Hence, studying how multiple teams collaborate is important. We designed a holistic multiple case study [35] of 14 teams in three projects in three companies (Table 1). According to Yin, case studies are the preferred research strategy when a "question is being asked about a contemporary set of events over which the investigator has little or no control" (ibid, p. 9).

We collected data through semi-structured interviews and retrospectives in three distinct large-scale projects across three case companies. The companies were chosen because they participate in a research program on autonomous teams. All teams in the study were working according to the Kanban method using some Scrum ceremonies such as the daily stand-up meeting. While Scrum divides work into a series of fixed-length iterations (sprints; whatever is scheduled for a sprint is the team's top priority), Kanban benefits from flexible planning because whatever is on the board is the top priority. Kanban focused on continuous delivery with changing priorities. Information about the companies, the projects, the teams, and the data collected is listed in Table 1. Sand and Grass are banks, while Necker is a Software Service Provider. All interviewed teams are situated in Norway, and all projects have been running for more than 18 months.

In all companies, we interviewed the managers working closely with the teams, and all team-members that were available in the interview period. In Sand, we chose the team that had been working together the longest, in Grass both teams that were involved in the project, in Necker one team that was chosen by the management. Interviewees were split into two

categories: those with assigned leadership responsibilities and those without. Interview guides can be found in Appendices A and B. The interviews lasted 40-50 minutes. Data were collected in two rounds. After transcribing and coding a few of the first interviews, the interview guide was revised based on the feedback. A second round was initiated with a revised interview guide. This round also included some follow-up interviews that explored particularly interesting subjects that were uncovered during the first round.

Retrospectives lasted approximately two hours. The retrospectives were sessions facilitated by the researchers where the development teams or the leadership team of the project reflected on their processes: what the teams thought was working well, what wasn't working well, and what they wanted to change. Afterward, the teams agreed on what measures to take. We took notes during the retrospectives.

Subsequently, we conducted a thematic analysis of our data as we wanted as few restrictions as possible for our inductive study [5]. Interviews were passed around for transcription and coding, ensuring that every researcher had insight into every interview. Researchers also participated in collective sessions where the empirical material was discussed. The analysis resulted in two themes (overall direction and external coordination) with several subthemes.

There is a risk that our findings can be explained by factors that evaded our attention. One reason is that we did not conduct a retrospective with all teams, and we did not interview all team members, and therefore, probably missed some subthemes. However, giving feedback to the observed teams and discussing our interpretation of what was going on with the case companies helped with validating our conclusions.

4. Challenges of autonomy in large-scale agile

Two themes emerged from the data analysis describing the challenges for autonomous development teams in large-scale agile, and how they relate to autonomy: overall direction and external coordination. Overall direction is about creating a shared direction among team member and how goals for the teams are set, what they entail, and how they are communicated to the team. External coordination entails how teams coordinate with their environment. This includes how a team relates to other teams and components, how a team deals with additional tasks (that is, tasks that, for example, are not in the backlog), how teams communicate, and how teams coordinate with an external customer.

First, we will briefly present a general bewilderment regarding the term autonomy as a backdrop for the empirical analysis. A developer from Necker exemplified how the meaning of the term is not well known nor easy to understand, as he had to google it before he came to the interview. However, he was not sure if he captured the essence of it. Several developers from Grass stressed the necessary knowledge and resources to implement their activities as the most important features of autonomy. A team lead from Necker added to this, reflecting on how everyone talks about autonomy without a common definition of it. Further light was shed on this confusion by a product owner from Sand who questioned whether higher-level managers know what autonomy is really about. At the same time, the informants spoke warmly of autonomy. A business representative from Grass considered himself a supporter of autonomous teams, and a line manager in the same company talked passionately about this way of working. A developer from Necker agreed, highlighting the freedom in how to develop the solution and working closely together, as features he appreciates. Furthermore, on the question of whether the informants see their teams as autonomous, the general answers were "yes" or "almost." The highlighted differences in interpretations among the informants, as well as the view on their teams being autonomous to certain degrees, illustrates the difficulty of autonomy.

Table 1: companies, the projects, the teams and the data collected

Company	Industry	Project Description	No. of teams	Data Collection
Sand	Finance/ banking	Change program for internal IT services and routines	5	9 interviews, including product owners, a project manager, a business representative and a domain architect. Retrospective on one team and on the project level
Grass	Finance/ banking	Web program for external end-users	2	8 interviews, including tech leads, developers, a business representative and a line manager. Retrospectives with two teams
Necker	ICT	Software development program for a large, public customer	7	6 interviews, including team leads and a project manager. Retrospectives on one team and the project level

4.1. Overall direction

4.1.1. The Importance of a Shared Direction

Several informants emphasized the importance of creating a shared direction for the teams in their large-scale project. A business representative from Sand emphasized that finding common ground and clarity of goals is necessities when implementing autonomous teams:

> *We as a team needed to try to establish what we had to accomplish. We had to lay that fundament and make team members familiar with the goals and vision. (…) We spent many sessions on finding a common ground.*

We found that finding common ground is also about understanding the tasks the team needs to do and the order in which the tasks should be developed. A domain architect from A explained:

> *Before, we didn't have a plan (…) but now we are in the process of making one (…) the overall goal is clear, but it doesn't say anything about the order of the tasks. The question is whether we should split them, or if we should do them as one.*

Further, to understand the tasks and goals of the team, the team has to understand the needs of the end-user of the system under development. However, in large-scale agile projects, the team is seldom in direct contact with the end-user like in a single-team project. We found team members discussed end-user needs and problems without having the same end-user type in mind, which resulted in misunderstandings in the team. One team decided to create a set of personas (description of different end-user types and their behavior), to be able to better discuss and understand customer needs in the team.

Having the right competence, network and experience in the team is an enabler for a shared direction. We found that experienced team members with an overview of the work can help other team members and pull the team in the right direction. A team lead from Necker stated that a team needs someone who possesses an overview of the team's assignment. A developer from Grass argues in line with the team lead, expressing that experienced developers are important not only because they understand the direction of the work but also understand how to do the work and who knows what in the large-scale project.

While understanding the direction for the team is important, we found that it is likewise important to understand the direction of the large-scale project which the team is a part of. A project manager from Necker explained that if the team is missing the larger picture, it is difficult to relate the team's tasks to it.

We found that within a specific large-scale project, the project consisted of the same roles. However, often members within the same large-scale project did not have an identical understanding of the roles. One example was the team lead role in Sand. We found that team leads, the manager of the program, and team members all had different understanding of the role. Some team leads did not even know they were really responsible for leading and developing the teams. As a consequence, it became difficult to pull the teams and the project in the same direction within the large-scale project. Some teams had members from different departments, and in one team, the members did not understand or accept their leader's authority. A team lead from Sand described: *"I have no control over what they do all the time. It is not like I need that, but I have no power to get things I see as important through in the team."*

A tech lead from the same company explains how the tech lead role was formed during the reorganization of the company. The informant explained how he got a description of his new role but did not remember all of the defined responsibilities. The complexity of the tech lead role is supported by a project manager from Necker, who said that it takes a lot of time to understand the nature of the tech lead role.

4.1.2. Setting and Communicating Goals

As explained in the previous section—shared goals are important for a shared direction. In the investigated large-scale projects, we found that goals are often set by management without involving the teams, the goals are often equal to deliverables and deadlines, and team members are not always sure what the goals are.

For several teams, goals appeared to be equivalent to delivery deadlines. A developer from Grass explained: *"Lately, our only goal is related to deliveries. It is about finishing something at a given time."*

When questioning whether the team has any goals other than specific deadlines, most answered that, if additional goals existed, they were not known to the team. A developer from Grass explained that the departments recently set some new ones, but he was not exactly sure what the new goals were. While the goals were unclear seen from the team members, the business and management side working with the team had a different opinion. A business representative from Grass explained how the goals are communicated orally, and that his impression is that the developers have a holistic picture of what they are doing. However, he acknowledged that there is no arena for creating a shared goal between the teams working on the large-scale project. However, this is something they are aiming for,

as the goals of the departments involved in the large-scale project are not aligned today.

One explanation for why goals were perceived as unclear to the teams might be that they were often identified by someone outside the team, not involving the team members. A developer from Grass explained that goal-setting is done mainly by the business department in the large-scale project. A tech lead supported this, adding that, after the goals are set, they are given to the team. When the team sees such goals as unrealistic, they do not commit to them. One developer from Grass explained:

> *I think they* [the management] *set the deadline with the intention of giving us something to work towards. And then we just have to see if we reach the deadline, or if we have to postpone the date or reduce the scope of our work. In my opinion, the deadlines do not always make sense*.

While teams were often not involved in defining the goals, a line manager from the same company expressed that taking part in identifying goals is an essential feature of autonomous teams. While involvement was desired, team members acknowledged that they could not be a part of all goal-related processes in the large-scale project. A tech lead from Sand explained: *"There are things you have no influence over as a team, because they happen on a higher level in the organization, or during a release process in which other teams are involved."*

4.2. External Coordination

4.2.1. Organizational Dependencies

In a large-scale setting, the teams are dependent on other teams, projects, departments, and/or systems within the organization, and vice versa. This is exemplified by a domain architect from Sand, "*We do not live in our own world (...) one has to coordinate with other teams who share components with your team."*

The domain architect explained that co-locating with teams who share the same system components is helpful when dealing with this matter. The challenge of depending on others is also present at Grass. A developer stated that the team frequently needs to clarify different issues with the business department. Discussing unclear specifications and the need for confirming decisions are examples of when a team needs to make contact before moving on. Further, a tech lead from the same company stated that since the workflow in the project was not synchronized, his team needed to clarify issues frequently with other departments. Problems with the synchronization also resulted in teams needing to wait for other resources and other teams to finish their part of the job. A team lead from Necker explained how they were unable to move on even if they had finished their own work.

> *We have external dependencies. We had some cases when integrating with systems made by external teams, and they were either not ready or it was not documented well enough, or we could not even access it (...) And you always have a lot of cases going on that you cannot finish because you have to wait for others.*

Because of the dependencies, the teams were also approached by others. When describing how the business department at Grass communicates directly with individuals in the team, a tech lead said, "*They talk directly to those having the task at hand. Sometimes this is fine, but ideally, they should involve the whole team, so that everyone knows what is going on."*

Depending on others to do part of the job reduced the team autonomy because decision making is limited, the team cannot fully control how tasks are conducted, and they need to adjust their processes to other teams and actors.

4.2.2. Dealing with Additional Tasks

Several informants state that additional tasks (tasks not prioritized in the spring backlog or prioritized by the team) delay the teams in doing their initial work. One of the interviewees (head of development) from Grass said that the team receives a stream of such additional tasks. A team lead in Necker stated that these unrelated tasks might even postpone entire sprints (the team was using Scrum). The team lead. Therefore. paid attention to external actors trying to get the team to do such additional tasks. A domain architect from Sand stated that even though tasks are seemingly unrelated, one can argue that they align with the goals of the team, as the goals are often very general. A developer in the same company stated that such tasks can be, for example, errors in previously developed products that need to be fixed right away. Another reason for such tasks emerging is the scale of the development effort. Because of the size and complexity of the large-scale project, it seems impossible to plan everything that needs to be done. Because of all the dependencies, sometimes a team needs to stop what it is doing and solve new tasks for other teams before being able to move on.

To reduce the challenge of having to deal with too many additional tasks, most of the teams have one team member responsible for communicating with the rest of the project and organization. In most teams, we found it to be the team lead or the tech lead. According to several informants across the companies, there was one particular reason why this was often the team or the tech

lead's responsibility: Individuals may find it difficult to know what to prioritize, and when they are approached, they are interrupted. A tech lead at Grass described his role as a link between the team and the rest of the organization. If anyone wants to talk to the team, they often approach him. A developer from Grass stated that being interrupted when writing code makes his tasks much more time consuming because his work requires deep concentration.

Several team and tech leads stated that they try their best to shield their team from externalities, filtering out what they consider as unnecessary for their team to know or take part in. Another team lead stated that the shielding responsibility is the single most important task he has.

5. Discussion

In the previous section, we described two themes that emerged from the data analysis describing the challenges for autonomous development teams in large-scale agile, and how they relate to autonomy: Overall direction and external coordination. We now discuss our research question: *What are the barriers to team autonomy in large-scale agile?*

According to Guzzo and Dickson [11], the autonomous team is given "significant authority and responsibility for many aspects of their work, such as planning, scheduling and assigning tasks to members, and making decisions with economic consequence."

When discussing the barriers, we will focus on those that reduce the responsibility or authority of the autonomous team in the large-scale project.

5.1 Shared Goals and Direction - to Where?

For a team in a large-scale agile context, we found that there is a need to understand the shared direction. There is also a need to have a shared understanding of the work processes, the tasks, and the roles and responsibilities. Having experts in the team, as well as setting goals and implementing them in the organization, are key activities for achieving a shared direction.

Also, we presented how higher-level managers set goals for large-scale projects, how such goals are not always seen as relevant, and how they are not always successfully communicated to the team. We also found that goals are often the same as deadlines or deliverables and that they are not aligned with other teams or the rest of the organization. In this section, we will discuss committing to shared goals and direction.

5.1.1. Commitment to Goals.

The empirical analysis shows that team members in the large-scale agile projects are often or always excluded from goal-setting processes. Instead, the goals are set by higher-level managers and are given to the teams. This contrasts Manz and Sims' [19] view that external leaders should allow the teams to set their own goals to facilitate autonomy. Participation in goal-setting is also associated with an increase in motivation [13] and increased meaningfulness for those who are trying to achieve them [12] since the goals are less trivial. Despite the benefits of letting the teams participate, this is evidently not the approach in the teams we have studied.

Further, a business representative states that the goals are communicated well, and his impression is that the team members understand where the team is headed. However, the empirical analysis shows that this is not always the case. Even though both business representatives and developers regard goals and direction as important, there is not necessarily a shared understanding of them. This is illustrated by informants having different views on goals. One view is that goals are closely related to deliveries, for example, finishing something on time. One explanation of why deliverables are so important in the large-scale agile project is the dependencies between the teams. If one team is delayed, it might affect other teams. A second view is that goals are set so that teams have something to strive for, not necessarily something to achieve in a certain time. A third view is that higher-level goals are not communicated in such a way that they reach the team level. This is apparent from an informant not knowing what the higher-level goals are.

In other words, there is an incongruence; while the impression among leaders is that goals are communicated and understood by the team, statements from team members indicate that this is not the case. There seems to be a lack of shared understanding of the goals set by the management and what direction to take. According to Moe et al. [22], a challenge that follows from this is individual commitment; team members will pursue their own goals if they have no reason to commit to the shared ones. Hence, if higher-level managers do not let team members partake in setting goals, the team members might create and pursue their own goals instead.

Therefore, not letting the team partake in goal setting is a barrier to team autonomy in large-scale because it will likely impact the team's autonomy in two ways. First, if not participating in the goal-setting causes the team members to set their own goals, the individual autonomy will increase because individuals are working independently toward their own objectives [15]. Second, according to Hoegl and Parboteeah [14], the external autonomy will decrease because the team does

not have the authority to decide its own goals. The goals are set by higher-level managers deciding what is important for the team. Finally, if the team is involved, it might be more likely that the goals are not equal to deliverables and deadlines because such goals do not motivate the team.

Team goals not being aligned with the rest of the organization or the large-scale project goals not being aligned with the team reduces the understanding of the shared direction and where the team and the project are heading.

Hence, the lack of a shared understanding of goals and direction makes it hard for the team to schedule, assign tasks, and make decisions that are aligned with the rest of the large-scale project.

5.2 Shielding the Team

The teams in our study are all part of larger-scale agile settings, and we have described how the teams need to coordinate and communicate with their external surroundings because of all their dependencies. This section explores this topic further.

5.2.1 External Dependencies

The empirical analysis shows that the teams in large-scale agile communicate and coordinate interdependently with other teams and departments within their companies or with a customer and other external teams. This seems necessary for two reasons. The first is that the specifications of the product, such as new features, are subject to change over time and, therefore, need to be communicated to the team. The features are also seldom understood before the team starts developing, and therefore, there is a need for constant dialogue with the business side. Secondly, resources often have to be synchronized between multiple development teams as the product or service can depend on components from many of them.

However, Pikkarainen et al. [26] state that agile practices do not provide the communication mechanisms in situations where many teams are involved in the same development process. In practice, a common solution seems to be that higher-level managers assign the responsibility of the external coordination to a leadership role within the team. From this way of dealing with the external environment, we draw a parallel to what Boehm and Turner [3] refer to as a project manager, a role operating as a barrier between the organization and the team. Even though the case companies have different titles for the role responsible for the external coordination, all of them seem to have one aspect in common; they assign the responsibility to one designated team member with leadership responsibilities. Depending on the case company, this responsibility is assigned to either the team lead, tech lead, or product owner.

According to informants, the person responsible for the external coordination is tasked with shielding the team. This involves protecting the team from unnecessary interruptions and deciding which pieces of information are important enough to put forward to the team. Empirically, those who have such a role consider themselves as links between the team and other departments of the organization. In cases where the team relates to an external customer, they take care of the communication and information flow between the customer and the team. The general need for coordination is addressed by Nerur, Cannon, Balijepally, and Bond [23] who state that software development teams need to interact with an ever more diverse set of stakeholders who have different expectations and needs than the team. In other words, shielding the team is a complex task. When the most experienced developer (tech lead) is the only interface to the rest of the project, he or she might become a bottleneck. Further, the team loses a key resource that could contribute to the development work and thereby support the rest of the team. Since this person is the one who is most suited for setting a shared direction for the team, he or she needs to balance the work of handling external dependencies and setting the direction for the team.

Furthermore, the empirical analysis also shows that teams frequently receive additional tasks from their external surroundings. These tasks are often outside the scope of what the team is assigned to do, such as tasks concerning errors in previously delivered products or tasks that emerge because of the complexity and the number of dependencies in the large-scale project. Agile teams are supposed to be flexible and respond rapidly to complex and ever-changing problems [6, 8]. However, the empirical analysis reveals how the additional tasks delay the teams in their work since they are forced away from what they initially were doing. In that sense, the adaptability and flexibility may itself impede the team's progress, as considerable capacity is used to solve unrelated tasks.

Hence, having someone shielding the team from external surroundings seems to be important. The empirical analysis reveals that developers find the shielding role relieving, since getting interrupted while focusing on the work makes tasks more time-consuming than they need to be. However, the empirical analysis shows that even though the teams have someone to shield them, the information and distribution of additional tasks coming from the surroundings do not always go through this contact point. Sometimes, representatives from various organizational departments and customers approach developers in the teams

directly. This is similar to what Moe, Dingsøyr, and Dybå [22] explain as stealing resources: external stakeholders, such as customers, approach and occupy developers with unrelated tasks. In other words, external stakeholders approach team members directly, despite members expressing that it disrupts the work they are originally assigned to do.

Based on the analysis, we see two possible reasons why the contact point is bypassed. The first reason is that the contact point might be a bottleneck of information. One of the holders of the shielding role stated that he spends most of his time in meetings and on administrative tasks. He might not be available or simply be overloaded with information. By having only a single point of external information in the team, the contact point may be exposed to what Schick, Gordon, and Haka [27] describe as information overload, too much information to handle. This means that relevant messages might disappear in the overflow of information. As a result, external stakeholders might see it as more reliant to approach the team members directly. The second reason is delivery-focus. As the empirical analysis shows, the teams have tight deadlines and many intervals of work. They are, therefore, time-sensitive, and external stakeholders who have their own deadlines might not be willing to wait for a response when they can just approach the team members directly to get what they want.

Thus, the discussion reveals contradictory interests; the external stakeholders want to make use of the team's resources and make ongoing clarifications while team members prefer being shielded from external noise as it interrupts their work. If the shielding role is bypassed, the team's control over their work is limited by the involvement from the external surroundings. According to Hoegl and Parboteeah [14], the autonomy is, therefore, reduced. Also, if individuals are assigned tasks directly by the external environment, their freedom and control in carrying out their own tasks are impeded. Therefore, the individual autonomy is reduced [15].

6. Conclusion

This paper presented data from a multiple case study of three large-scale projects. We have focused the description of the projects on the barriers that reduce the responsibility or authority of the autonomous team in the large-scale project. From the described large-scale project, we identified two main barriers: overall direction and external dependencies. We found that goals are often set by management without involving the teams, that they are often equal to deliverables and deadlines, and that team members often do not know what the goals are. Consequently, teams struggle with setting and communicating goals as well as establishing a shared direction. Organizational dependencies lead to teams having to deal with additional tasks, resulting in specific members shielding the teams from external noise.

For practitioners, we think this paper illustrates the importance of working on a shared understanding of goals and the difficulties of balancing the need for flexibility and the need to shield the team in large-scale agile. This is an issue which is not well-described in the agile literature and is the most important contribution of this paper.

Further work in this direction should focus on investigating other barriers with autonomous teams in large-scale agile, for example, related to reducing dependencies between teams.

7. Acknowledgments

This work was supported by the Research Council of Norway (grant 267704) and the companies Kantega, Knowit, Sbanken and Storebrand through the research project Autonomous teams.

8. References

[1] Beck, K. and Anders, C., Extreme Programming Explained: Embrace Change. Addison-Wesley, 2004.

[2] Boehm, B. and Turner, R., Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley 2003.

[3] Boehm, B. and Turner, R., "Management Challenges to Implementing Agile Processes in Traditional Development Organizations," *IEEE Software,* vol. 22, no. 5, pp. 30-39, 2005.

[4] Boehm, B. W. and Ross, R., "Theory-W software project management principles and examples," IEEE Transactions on Software Engineering, vol. 15, no. 7, pp. 902-916, 1989.

[5] Braun, V. and Clarke, V., Successful qualitative research: A practical guide for beginners. sage, 2013.

[6] Cockburn, A. and Highsmith, J., "Agile software development: The people factor," Computer, vol. 34, no. 11, pp. 131-133, 2001.

[7] Dikert, K., Paasivaara, M., and Lassenius, C., "Challenges and success factors for large-scale agile transformations: A systematic literature review," Journal of Systems and Software, vol. 119, pp. 87-108, 2016/09/01/ 2016.

[8] Dybå, T., "Improvisation in Small Software Organizations," IEEE Software, vol. 17, no. 5, pp. 82-87, September/October 2000.

[9] Fenton-O'Creevy, M., "Employee involvement and the middle manager: evidence from a survey of organizations," Journal of Organizational Behavior, vol. 19, no. 1, pp. 67-84, Jan 1998.

[10] Goodman, P. S., Devadas, R., and Griffith Hughson, T. L., "Groups and productivity; analyzing the effectiveness of self-managing teams," 1988.

[11] Guzzo, R. A. and Dickson, M. W., "Teams in organizations: Recent research on performance and effectiveness," Annual Review of Psychology, vol. 47, pp. 307-338, 1996.

[12] Hackman, J. R., "The design of Work Teams," in Handbook of organizational behavior, J. Lorsch, Ed. Englewood Cliffs, N. J. : Prentice-Hall, 1987.

[13] Hackman, J. R. and Oldham, G. R., "Work redesign," 1980.

[14] Hoegl, M. and Parboteeah, P., "Autonomy and teamwork in innovative projects," Human resource management, vol. 45, no. 1, p. 67, 2006.

[15] Langfred, C. W., "The paradox of self-management: Individual and group autonomy in work groups," Journal of Organizational Behavior, vol. 21, no. 5, pp. 563-585, Aug 2000.

[16] Larman, C. and Vodde, B., Large-scale scrum: More with LeSS. Addison-Wesley Professional, 2016.

[17] Leffingwell, D., SAFe® 4.0 Reference Guide: Scaled Agile Framework® for Lean Software and Systems Engineering. Addison-Wesley Professional, 2016.

[18] Linders, B., Getting Value out of Agile Retrospectives-A Toolbox of Retrospective Exercises. Lulu. com, 2014.

[19] Manz, C. C. and Sims Jr, H. P., "Leading workers to lead themselves: The external leadership of self-managing work teams," Administrative science quarterly, pp. 106-129, 1987.

[20] Moe, N. B., Aurum, A., and Dybå, T., "Challenges of Shared Decision-Making: A Multiple Case Study of Agile Software Development," Information and Software Technology, Microsoft Word Document pp. 1-38, 2012.

[21] Moe, N. B. and Dingsøyr, T., "Emerging Research Themes and updated Research Agenda for Large-Scale Agile Development: A Summary of the 5th International Workshop at XP2017," in Proceedings of the Scientific Workshop Proceedings of XP2017, 2017: ACM.

[22] Moe, N. B., Dingsøyr, T., and Dybå, T., "Overcoming Barriers to Self-Management in Software Teams," IEEE Software, vol. 26, no. 6, pp. 20-26, 2009.

[23] Nerur, S., Cannon, A., Balijepally, V., and Bond, P., "Towards an understanding of the conceptual underpinnings of agile development methodologies," in Agile Software Development: Springer, 2010, pp. 15-29.

[24] Nyrud, H. and Stray, V., "Inter-Team Coordination Mechanisms in Large-Scale Agile," in Proceedings of the Scientific Workshop Proceedings of XP2017, 2017: ACM.

[25] Parker, S. K., Parker, S., and Wall, T. D., Job and work design: Organizing work to promote well-being and effectiveness. Sage, 1998.

[26] Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., and Still, J., "The impact of agile practices on communication in software development," in Empirical Software Engineering vol. 13, ed, 2008, pp. 303-337.

[27] Schick, A. G., Gordon, L. A., and Haka, S., "Information overload: A temporal approach," Accounting, Organizations and Society, vol. 15, no. 3, pp. 199-220, 1990.

[28] Šmite, D., Moe, N. B., Šāblis, A., and Wohlin, C., "Software teams and their knowledge networks in large-scale software development," Information and Software Technology, vol. 86, pp. 71-86, 2017.

[29] Stray, V., Moe, N. B., and Hoda, R., "Autonomous agile teams: Challenges and future directions for research," presented at the Proceedings of XP'18 Companion, Porto, Portugal, 2018.

[30] Stray, V., Moe, N. B., and Sjøberg, D. I. K., ", "The Daily Stand-Up Meeting: Start Breaking the Rules,," IEEE Software, 2018 (in press).

[31] Stray, V. G., Moe, N. B., and Dingsoyr, T., "Challenges to Teamwork: A Multiple Case Study of Two Agile Teams," in Agile Processes in Software Engineering and Extreme Programming, vol. 77, A. Sillitti, O. Hazzan, E. Bache, and X. Albaladejo, Eds. (Lecture Notes in Business Information Processing, 2011, pp. 146-161.

[32] Susman, G. I., Autonomy at work: A sociotechnical analysis of participative management. praeger Publishers, 1976.

[33] Tata, J. and Prasad, S., "Team Self-management, Organizational Structure, and Judgments of Team Effectiveness," Journal of Managerial Issues, vol. Vol. 16 no. Issue 2, pp. p248 - 265, 2004.

[34] Trist, E. and Bamforth, K. W., "Some Social and Psychological Consequences of the Longwall Method of Coal-Getting," Human Relations, vol. 4, no. 1, pp. 3-38, February 1, 1951 1951.

[35] Yin, R. K., Case study research: design and methods. Thousand Oaks, Calif.: Sage, 2002, pp. xiv,219 s.

Appendix

Appendix A and B are available here:
https://figshare.com/s/37ad61b642581e890c58