Responding to change over time: A longitudinal case study on changes in coordination mechanisms in large-scale agile

Marthe Berntzen[1] · Viktoria Stray[1,2] · Nils Brede Moe[2] · Rashina Hoda[3]

Abstract

Context Responding to change and continuously improving processes, practices, and products are core to agile software development. It is no different in large-scale agile, where multiple software development teams need to respond both to changes in their external environments and to changes within the organization.

Objective With this study, we aim to advance knowledge on coordination in large-scale agile by developing a model of the types of organizational changes that influence coordination mechanisms.

Method We conducted a longitudinal case study in a growing large-scale agile organization, focusing on how external and internal changes impact coordination over time. We collected our data through 62 days of fieldwork across one and a half years. We conducted 37 interviews, observed 118 meetings at all organizational levels, collected supplementary material such as chat logs and presentations, and analyzed the data using thematic analysis.

Results Our findings demonstrate how external events, such as onboarding new clients, and internal events, such as changes in the team organization, influence coordination mechanisms in the large-scale software development program. We find that external and internal change events lead to the introduction of new coordination mechanisms, or the adjustment of existing ones. Further, we find that continuous scaling requires continuous change and adjustment. Finally, we find that having the right mechanisms in place at the right time strengthens resilience and the ability to cope with change in coordination needs in complex large-scale environments.

Conclusions Our findings are summarized in an empirically based model that provides a practical approach to analyzing change, aimed at supporting both researchers and practitioners dealing with change in coordination mechanisms in large-scale agile development contexts.

Keywords Large-scale agile · Software development · Coordination · Organizational change · Continuous improvement · Longitudinal case study

🙂 Springer

1 Introduction

Agile software development welcomes change (Fowler and Highsmith 2001), and large-scale agile is abundant with changes, for example, in customer requirements, technical dependencies, team composition, and tool use. In large-scale agile, defined as software development involving more than six teams or more than 50 developers[1] (Dikert et al. 2016), multiple teams join efforts in developing an overall software system. Large-scale organizations must cope with rapid external disruptions, such as technological innovation, economic and political destabilization, and re-negotiation of workplace arrangements while continuously improving their software engineering practices. Additionally, dependencies between teams represent further challenges to efficient software delivery. Therefore, understanding how change impacts coordination may make the difference between successful and non-successful software development in a large-scale context.

Coordination, often defined as the management of dependencies (Malone and Crowston 1994), is central to agile software development because of dependencies that may impact software delivery efficiency (Strode 2016). Dependencies occur "when the progress of one action relies on the timely output of a previous action or on the presence of a specific thing, where a thing can be an artifact, a person, or a piece of information" that "can be managed well, poorly, or not at all" (Strode 2016, p. 24). Coordination is needed because if dependencies are insufficiently managed, they can cause blockages and bottlenecks that delay the development progress and, ultimately, software delivery (Cataldo and Herbsleb 2012).

When multiple teams work together to develop software, several coordination challenges arise (Dingsøyr et al. 2018a). For example, interfacing between teams becomes problematic because dependencies in one team may delay or hinder the work of other teams (Bick et al. 2018), and achieving and maintaining technical consistency becomes difficult (Dikert et al. 2016). Such challenges require that agile practices are adapted to the large-scale level (Dingsøyr et al. 2018a). However, the self-organizing teams model central to agile can become problematic because of the need to align and coordinate interdependent teams' work practices and outputs (Moe et al. 2021). These challenges must be solved within an ever-changing context, making it essential to use the most effective *coordination mechanisms*, which we define as organizational processes, entities, or arrangements used to manage dependencies between activities to realize a collective performance (Okhuysen and Bechky 2009). Selecting the most suitable mechanisms at any given time and modifying or replacing them in response to change is also a challenge.

Change in coordination is a topic in need of further exploration, especially in the context of large-scale agile, because changes bring new and different coordination requirements that, if insufficiently managed, can cause delays and bottlenecks, and even project breakdown (Cataldo and Herbsleb 2012; Dikert et al. 2016; Bick et al. 2018). In this study, we consider two forms of change relevant to coordination in large-scale agile. On the one hand, change can be understood as *event-based*, that is, as "something specific

---

[1] There is no agreed-upon definition of what exactly constitutes 'large-scale' in the research community (Edison et al. 2022). In line with Dikert et al. (2016), we define large-scale agile as more than six teams or involving more than 50 developers. Our case organization, Entur, eventually had 17 teams and could, therefore, also have been classified as 'very large-scale' according to a much-used definition (Dingsøyr et al. 2014). However, because the size of the program over time grew from 'large-scale' to 'very large-scale' (Dingsøyr et al. 2014), we keep with the general term 'large-scale' to better relate to the literature on large-scale agile overall (Edison et al. 2022; Uludağ et al. 2022).

that happens" through disruptive events or patterns of events (Jarzabkowski et al. 2012). Software engineering studies have focused on 'a change' either by studying the 'before and after' a large-scale agile transformation or while following what happens during the implementation of a large-scale agile framework (e.g., Paasivaara et al. 2018; Russo 2021; Gustavsson et al. 2022). On the other hand, change can be understood as a *continuous process* or flow of activities that are harder to pinpoint but easily observable in retrospect (Van de Ven and Poole 2005; Langley et al. 2013). We believe that both perspectives of change can inform the analysis of how organizational changes influence coordination mechanisms and how the mechanisms themselves change over time.

As research on coordination in agile software development and large-scale agile is maturing (Dingsøyr et al. 2012; Hoda et al. 2018; Berntzen et al. 2022), we believe it is important to focus on understanding how and why coordination practices and mechanisms change over time. In this paper, we aim to explore the relationship between organizational changes and changes in coordination mechanisms in the context of large-scale agile software development. We do this by investigating the following research question:

> *What type of organizational changes influence coordination mechanisms in large-scale agile, and how do these mechanisms change over time?*

We report on findings from a longitudinal case study conducted over one and a half years in a large-scale organization called Entur. The time frame enabled us to follow the case organization as changes were taking place. We spent a total of 62 days at the field site, and observed 118 meetings, such as retrospective and stand-up meetings, client meetings, and board meetings, conducted 37 interviews with practitioners in roles such as team leaders, product owners, and program managers, and collected supplementary material from resources, such as chat logs and documentation pages as the development program continued to scale. The data were analyzed using thematic analysis (Braun and Clarke 2006, 2012) in light of the theoretical framework of Jarzabkowski et al. (2012), who proposed a process theory of how coordination mechanisms are created in light of organizational change. As longitudinal studies of such scope and detail are rare within software engineering (Sharp et al. 2016) this study represents a unique contribution to the literature on large-scale agile coordination over time. Further, in comparison to other studies dealing with change in coordination mechanisms in large-scale agile (Moe et al. 2018; Gustavsson 2019; Dingsøyr et al. 2022), this study also focuses on *change* itself as opposed to the result of a change.

Our study offers the following contributions to software engineering research and practice:

- We advance research on change in coordination mechanisms over time in large-scale agile software development, which is called for in previous works (Moe et al. 2018; Dingsøyr et al. 2022).
- We provide a rich empirical description with a unique level of depth in the data collection.
- We build on existing theoretical work to propose a model of change in coordination mechanisms in large-scale agile (Fig. 7).
- We provide an actionable approach to analyzing change for practitioners who want to deep-dive into understanding and responding to change in coordination mechanisms (Table 4).

The paper is organized as follows. In Section 2, we review some of the existing research on coordination and change in large-scale agile. Section 3 presents the case organization and provides detailed information about our research methodology. Section 4 presents our research findings, which are discussed in Section 5, where we also discuss the practical and theoretical implications of the study. Section 6 concludes the paper.

2 Background

In presenting the background literature informing our study, we first consider different approaches to change in large-scale agile. Next, we present background on coordination and coordination mechanisms before presenting a process theoretical approach to studying change in coordination mechanisms.

2.1 Change in Large-Scale Agile Software Development

Change is central to agile software development. "Responding to change" is part of the agile manifesto (Fowler and Highsmith 2001) and core to agile. Traditionally, studies have focused on how software development teams respond to change (Hoda and Noble 2017; Spiegler et al. 2021). Further, changes are typically understood as "things that happen," that is, events that are more or less beyond the individual developer's or team's control, for example, changes in requirements (Aldave et al. 2019; Madampe et al. 2022), re-organizing of the team (Spiegler et al. 2021) or company structure (Gustavsson et al. 2022; Carroll et al. 2023), or technical issues (Kwan et al. 2011; Cataldo and Herbsleb 2012). In large-scale agile, such events can be even more complex and challenging because of the dependencies that exist between teams that develop an overall product with inter-dependent components or several inter-dependent products.

The notion of *continuous improvement* is also core to agile because agile teams constantly strive to find better ways of solving their day-to-day challenges (Fitzgerald and Stol 2017). In large-scale settings, where inter-team coordination is needed, continuous improvement of coordination practices is vital for teams to successfully keep up with each other (Kalenda et al. 2018; Paasivaara et al. 2018; Dingsøyr et al. 2022). Using an efficient mix of coordination mechanisms appears essential to respond to changes and continuously improve (Strode 2016). However, what constitutes an optimal mix of mechanisms may change over time (Moe et al. 2018; Dingsøyr et al. 2022). This is particularly true in uncertain situations, such as technological organizations, where hierarchies and rules-based systems are less useful (Jarzabkowski et al. 2012). Continuous improvement means continuously making changes. Accordingly, change can be understood as a continuous flow of activities. From this perspective, it is difficult to pinpoint precisely when such changes occur, but they are easily observable retrospectively (Van de Ven and Poole 2005; Langley et al. 2013).

As research and practice are maturing, different terms are used to describe variations of large-scale agile approaches. A recent study separates first- and second-generation large-scale agile development methods (Dingsøyr et al. 2022). First-generation methods are described as development methods that combine agile and traditional methods, typically using agile at the team level and traditional project management practices used at the inter-team, project, or organizational level (e.g., Batra et al. 2010; Bick et al. 2018). Many organizations use this type of mix between agile practices and other project management

practices because large-scale organizations (need to) have other governance structures surrounding the development and other agile business activities (Kalenda et al. 2018; Edison et al. 2022). Second-generation methods use ideas from the agile and lean communities, focusing on the product, collaboration, informal communication, and flexible and evolutionary delivery models and organizations (Dingsøyr et al. 2022). The commercial large-scale agile development frameworks, including SAFe, Large-Scale Scrum (LeSS), and the Spotify model, are examples of second-generation agile development methods (Dingsøyr et al. 2022). These and other scaling frameworks are widely used by practitioners in large-scale agile because they propose specific processes and mechanisms to manage the challenges with dependencies in large-scale software development (Dikert et al. 2016; Edison et al. 2022). Much existing research on large-scale agile transformation, including the studies referenced in the coming section, are case studies of companies implementing one of the large-scale agile frameworks.

However, not all large-scale development projects and programs use large-scale frameworks or methods. Some research critiques against large-scale frameworks are that they are not flexible enough to handle changing coordination needs (Gustavsson et al. 2022). In practice, many organizations take a less rigorous methodological approach to large-scale development. The comprehensive literature review by (Edison et al. 2022) shows that most organizations adapt agile methods to fit their specific contextual needs, regardless of adopting a large-scale framework. Irrespective of the approach taken, both researchers and practitioners agree that coordination is a crucial challenge to the success of large-scale agile development (Dikert et al. 2016; Edison et al. 2022; Uludağ et al. 2022).

2.2 Coordination and Coordination Mechanisms in Large-Scale Agile

Coordination has been studied across a wide range of research fields, including management, organization studies, information systems management, and software engineering (Espinosa et al. 2007; Okhuysen and Bechky 2009). Early studies on management and organization (e.g., March and Simon 1966; Thompson 1967) recognized the need to coordinate interdependent processes and activities, and the topic is still widely studied today (Castañer and Oliveira 2020). Within large-scale agile, research on coordination has dealt specifically with dependency management, as this represents a key challenge to the success of large-scale development (Stray et al. 2022a, b). An analysis of agile teams resulted in three dependency groups: knowledge, resource, and process dependencies (Strode 2016). *Knowledge dependencies* relate to information required for progress, including historical knowledge and task allocation knowledge. *Process dependencies* include activities and business processes that must be in place for tasks to proceed, while *resource dependencies* include technical dependencies and dependencies related to the availability of some resource (i.e., a person, a place, or a thing) (Strode 2016). These dependency categories form a taxonomy of dependencies for agile teams, which has also been applied at the large-scale level by focusing on dependencies between teams (Berntzen et al. 2021).

Dependencies are managed using *coordination mechanisms*, which we define as organizational processes, entities, or arrangements used to manage dependencies between activities to realize a collective performance (Okhuysen and Bechky 2009). Coordination mechanisms have been operationalized differently across fields. Early conceptualizations include work standardization, outputs, skills, and norms as central coordination mechanisms (Mintzberg 1989). Malone and Crowston (1994) propose coordination mechanisms such as priority orders, budgets, sequencing, tracking, and

standardization. Another approach is the three modes of coordination mechanisms introduced by Van de Ven and colleagues (1976), which includes a group mode (i.e., mechanisms based on mutual adjustment through feedback), a personal mode (i.e., mutual adjustment based on feedback between two people), and the impersonal mode (i.e., the use of plans, schedules, and standardized information). Much research on agile development and coordination in software engineering relies on Van de Ven's (1976) coordination modes and Malone and Crowston's (1994) coordination definition. In addition, Strode and colleagues (2012) have developed a theory of coordination in agile development teams, which has recently been applied at the inter-team level (Berntzen et al. 2021; Stray et al. 2022a).

In large-scale agile, teams working on different parts of the overall system need to coordinate, for example, merging of code, testing, and releases. In such situations, teams rely on mechanisms adapted to be used across teams, which we refer to as *inter-team coordination mechanisms*. In recent work, we presented a taxonomy of inter-team coordination mechanisms specific to large-scale agile (Berntzen et al. 2022) consisting of meetings, such as stand-up meetings, retrospectives, and communities of practice (Moe et al. 2018); roles, such as product owners (Bass 2015); and tools and artifacts, including communication and documentation platforms, such as Slack, JIRA, and Confluence (Lin et al. 2016), and goal management tools such as Objectives and Key Results (OKRs) (Niven and Lamorte 2016; Stray et al. 2022b). As changes in the development process, such as changing requirements or change in tool use, team or role composition, or system architecture, for example, are likely to have a multi-team impact with often far-reaching consequences (Cataldo and Herbsleb 2012; Dikert et al. 2016), understanding change in relation to coordination is of great importance.

There are few studies of change in coordination in large-scale agile, but the topic is gaining traction. For example, Gustavsson and colleagues (Gustavsson 2019; Gustavsson et al. 2022) investigated how inter-team coordination and team autonomy change when a large-scale framework (such as SAFe) is implemented. Moe et al. (2018) investigated how meetings changed over time, using Van de Ven's (1976) classification of coordination meetings, i.e., the group mode. Their study showed that scheduled meetings were important coordination mechanisms early in the development process but that more unscheduled meetings could replace these arenas over time. Similar findings were reported by Dingsøyr et al. (2018b), who, in addition to the change in meeting formats, found more use of horizontal coordination and change in tool use early versus late in the development program as the case organization matured. Paasivaara et al. (2018) found that in addition to a change in tools and practices (i.e., coordination mechanisms), a shift in mindset was important when conducting a large-scale agile transformation. In a recent study, Dingsøyr et al. (2022) describe changes in inter-team coordination in a large-scale development program that transitioned from using a combination of traditional and agile development practices (i.e., first-generation development methods) to using cross-functional autonomous teams and continuous delivery (i.e., second-generation development methods). Their research showed that the number of coordination mechanisms went down from 27 to 14 when the program started using a second-generation development method and that coordination effectiveness was perceived as higher by interview participants.

Common to these previous studies is the focus on *one specific change* (i.e., how was coordination before/after implementing a framework (Gustavsson et al. 2022; Carroll et al. 2023) or agile transformation (Paasivaara et al. 2018), transitioning from one phase to another (Dingsøyr et al. 2022) or focus on one type of coordination mechanism (such as group mode coordination mechanisms (Dingsøyr et al. 2018b; Moe et al. 2018)).

2.3 A Process-theoretical Approach to Change in Large-scale Agile

In this study, we focus on the phenomenon of change in relation to coordination in large-scale agile, using a process-theoretical lens. Organizational researchers have studied change for decades, often using process theories to capture the complexities of change and evolution over time (Pettigrew 1990; Langley 1999; Van de Ven and Poole 2005). A process theory aims at explaining and understanding how an entity changes and develops over time (Langley et al. 2013). Process theories are suitable for dealing with change and with time (Van de Ven and Poole 2005) through their ability to explain the temporal order of events based on historical narratives (Gregor 2006; Langley et al. 2013) and have been applied in research on software development and on coordination in several ways to explain how changes in organizations may unfold. For instance, Allison and Merali (2007) proposed a theory of software process improvement, where the interplay between software development and software process improvement continuously informed each other and where both process and product were changing each other, and were changed by their surrounding context over time. In a recent paper, Carroll et al. (2023) applied normalization process theory (Murray et al. 2010) to examine how agile practices were embedded and sustained in a large international company that implemented the Spotify model during a large-scale agile transformation. They found that a failure to normalize new practices led to the unraveling of the transformation within 18 months.

In a longitudinal case study of coordination in a large-scale technology company that underwent an organizational restructuring, Jarzabkowski et al. (2012) proposed a process-theoretical framework to explain how coordination mechanisms are created in practice through five cycles. The process starts with some disruptive event, such as a reorganization, restructuring or transformation, that *disrupts existing ways of coordinating*. In the second cycle, actors are trying and failing to coordinate effectively and thereby *orients to absences in coordinating*. Third, new efforts to coordinate are made to fill the absences, which *creates new elements of coordinating*. In the fourth cycle, *new patterns of coordinating* are formed as links are created between elements of the new coordination mechanism. In the fifth and final cycle, the new coordination *patterns are stabilized* as new mechanisms are formalized (Jarzabkowski et al. 2012).

In our analysis and results, presented in the coming sections, we draw on the theoretical framework by Jarzabkowski et al. (2012). This framework is suitable because it explicitly recognizes that coordination mechanisms are not stable entities but are created over time in response to changes (i.e., disruptions). Such a view of coordination mechanisms appears highly compatible with large-scale agile software development, where responding to change is vital to succeeding. However, this study focuses specifically on disruptive events, which we understand as events of a certain magnitude, such as shutting down organizational departments, mergers and acquisitions, or changing an organization's technological platform (Jarzabkowski et al. 2012). Additionally, similar to the examples from large-scale agile literature cited in the previous section, Jarzabkowski et al. (2012) also focused on *one specific* event. Because continuous improvement is core to agile, we wanted to capture how the many events, large and small, as well as the more subtle, ongoing changes, shape coordination practices over time. Moreover, Jarzabkowski et al. (2012) limit their discussion to how coordination mechanisms are created. Because changes are omnipresent in large-scale software development, we wanted to understand not only how mechanisms are created but also how they change in

response to internal and external changes, and whether the five-cyclical model can also apply to ongoing changes in coordination mechanisms. To this end, we also draw on the notion of continuous improvement (Fitzgerald and Stol 2017) and an understanding of change as a continuous process of activities (Langley and Truax 1994; Langley 1999) introduced in the above sections.

3 Research Methods

In this study, we explore what types of organizational changes influence coordination mechanisms over time through a case study conducted over 1.5 years in a large-scale agile program in an organization called Entur. The case study approach allowed for a deep, situational understanding of the research topic (Flyvbjerg 2006), and the longitudinal format provided the opportunity to investigate research questions of temporal character ((Van de Ven and Poole 2005; Langley et al. 2013). In software engineering, case studies are valuable for understanding software development activities in context (Runeson and Höst 2008; Wohlin and Aurum 2015). We chose a single case to investigate our research question because we had the opportunity to conduct longitudinal fieldwork in an interesting organization where we were given access to many and varied data sources, as described below. Single-case studies should be motivated by opportunities for learning about a phenomenon of interest (Flyvbjerg 2006), and the case needs to be relevant enough to provide such opportunities. Flyvbjerg (2006) defines a *critical case* as a case that has "strategic importance in relation to a general problem" (p. 229), in our case, coordination in large-scale agile software development. Entur can be considered a critical case because of their size and number of teams with varying levels of interdependence, making them likely to experience many of the coordination challenges outlined in Section 2. The features described in the following are, however, not unique to this organization. Therefore, it can be argued that what is valid (or not) for this case would also be valid (or not) for many cases (Flyvbjerg 2006). Single-case studies are further suitable for process-theoretical research because of the level of detail required for understanding the intricacies of processual change in organizations that are not easily captured using formal variance theories with higher levels of abstraction (Van de Ven and Poole 2005; Ralph 2018).

In the following, we rely on two sensemaking strategies to present the case and our findings. We use a narrative strategy (Pettigrew 1990) to convey the rich textual information and to highlight the identified change themes. Additionally, we rely on the visual mapping strategy (Langley and Truax 1994; Langley 1999), as presented in Fig. 4, to visually represent the findings.

3.1 Case Description

Our case, Entur, is a public sector IT organization established in 2016 in response to a public transportation reform initiated by the Norwegian Ministry of Public Transportation. Entur aims to make public transportation an easier and more viable option for the Norwegian public. To fill this mandate, Entur develops several products related to public transport in Norway. One central product is a multi-platform travel planner that aims to allow travelers to plan and manage their entire trip within one single application. Another is a
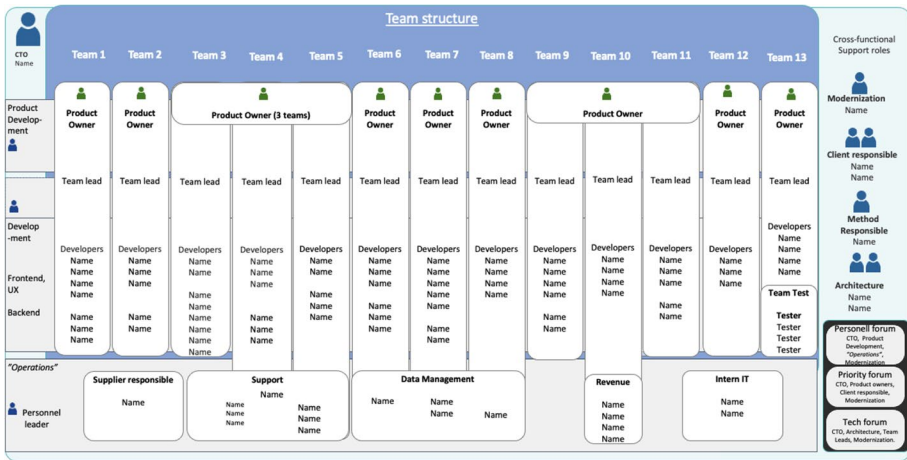
Fig. 1 The team organization of the development program in 2018

new sales platform and API that railway operators and other public transportation operators can use to distribute their products to the public through the Entur app or their own channels. Finally, they collect, refine, and share public transportation data through open APIs.[2] Because they used agile methods right from the beginning, Entur can be considered a mature agile development program. They have also experienced substantial organizational growth within few years and have, at the same time, been successful in delivering on the goals of the public transportation reform. The company is recognized within its national context, for example, by the Digitalization Council of Norway.[3]

The Large-Scale Agile Team Environment Entur started working with agile methods from day one and, therefore, never underwent an agile transformation as part of their organizational scaling journey. Since the outset in 2016, the development organization has grown rapidly, from five teams in 2016 to 17 in January 2020, and the growth has continued. Figure 1 displays the team organization in 2018 when we started our fieldwork. Despite being a large-scale agile organization from the outset, Entur has chosen not to adopt any scaling framework (such as SAFe or LeSS). Instead, they use software development best practices and gain inspiration from companies such as Spotify and Google, and tailor any new approach to their specific needs. For example, they established a biweekly tech lead forum modeled from Spotify's guilds and experimented with using OKRs (used and popularized by Google from around 2016 (Niven and Lamorte 2016)).

The development teams were cross-functional but focused on different parts of the overall deliveries, such as sales, ticketing, pricing, and web and app. Team size and composition varied slightly over time; the largest team had more than 16 members, while the smallest had about five. All teams had a team leader and a product owner, and a tech lead after this role emerged during 2019. All additional team members were developers that in principle could work on any part of their teams' code through shared repositories and
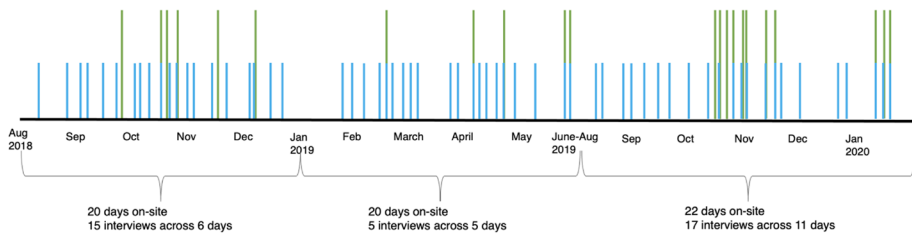
Fig. 2 Data collection timeline. Short, blue bars represent unique observation days on-site, while long, green lines indicate that interviews were conducted that day

documentation. In addition, a dedicated test team worked with the different teams to ensure appropriate testing of the different parts of the system. Some teams, such as the sales and product teams worked primarily with backend, whereas web and app teams worked with front-end and UX. Because all teams were connected at some level in delivering the overall product, all teams needed to coordinate. However, some teams had more dependencies to other teams than others. For example, almost all other teams had dependencies to the team that wrote the tickets and on-board products, and the UX and web team were dependent on almost all other teams, in that they needed the output of other teams to be ready in order to deliver visual output to travelers using the Entur app. The teams were autonomous in the choice of development methods, and most teams used practices from Scrum and Kanban. As sfat thiuch, there was a culture of testing new approaches, both within teams and at the inter-team level. At the same time, the need for inter-team coordination and some level of alignment across teams was high as they grew from five to 17 teams. The size and complexity of both the technology and the large-scale organization came with the consequence of many and complex knowledge, technology, and process dependencies. As such, coordination was an ongoing and evolving challenge. More details on the evolution and scaling of the organization and the coordination practices and mechanisms are provided in Section 4.

3.2 Data Collection

Data was collected from August 2018 to January 2020. During the one and a half years of fieldwork, the first author conducted observations and interviews and collected supplementary material such as meeting minutes, Slack (a collaborative communication tool) logs, e-mails, and Confluence (a software documentation tool) pages. The second and third authors participated in some of the data collection. Figure 2 displays a timeline of the data collection. Data collection ended in January 2020, when the pre-planned fieldwork period of the first author had come to an end. We chose an ethnographic approach to data collection (Sharp et al. 2016), which includes researcher immersion in the case context and longer periods of fieldwork with detailed observation and extensive notetaking following an observational protocol (Crang and Cook 2007). In software engineering, ethnographic approaches provide opportunities for a detailed understanding of the development practice, including both social and technical aspects of the development process (Sharp et al. 2016). Our ethnographic approach included long-term non-participant observation, conducted in face-to-face settings, undertaken to understand and capture coordination in large-scale agile, and conducted with process theoretical underpinnings (Sharp et al. 2016).

Table 1. Data collection details by type of data material

Observations	Type of observation	Number
	Internal meetings:	
	Prioritization meetings	10
	Tech lead forum (Community of practice)	7
	Weekly program demos	7
	Product owner weekly meetings	6
	Inter-team stand-up meetings	6
	Inter-team retrospectives	4
	OKR workshops	2
	Ad hoc inter-team meetings	26
	Intra-team meetings	26
	External meetings:	
	'Change workshop'	5
	Client meetings	6
	Other meetings	3
	Total number of meetings observed	118
	Unique days on-site	62
Interviews	Roles interviewed (Gender/Mean tenure IT/Mean tenure company)	
	Product owners (5 male, 4 female, IT tenure 11.5 years, company 1.8 years)	9
	Program managers (4 male, 1 female, IT tenure 18 years, company tenure 1.6 years	5*
	Program architects (4 male, IT tenure 19 years, company tenure 1.4 years)	4*
	Tech leads (3 male, 1 female, IT tenure 7 years, company tenure 2.4 years)	4
	Team leaders (2 male, IT tenure 9 years, company tenure 1.5 years)	2
	Agile methods specialist (male, IT tenure 15 years, company tenure 4 years)	1**
	Unique individuals	25
	*Six participants were interviewed twice or more (4 managers and 2 architects	6
	**Recurring interviews with the agile methods specialist	6
	Total number of interviews	37
Supplementary documentation	Slack logs, JIRA and Confluence documentation, e-mails, internal and external documents (e.g., presentations, reports, minutes)	

This study extends our previous research, and parts of the data material have been analyzed for other studies. Specifically, we analyzed 12 interviews and observations from 17 meetings from September – November 2018 in Berntzen et al. (2019). In another study (Berntzen et al. 2021), we used data from August 2019 to January 2020, including 12 interviews and observations from 26 meetings. Finally, 31 interviews, observation notes from 94 meetings, and supplemental material such as Slack logs were analyzed in developing the taxonomy of inter-team coordination mechanisms and the TOPS framework (Berntzen et al. 2022). The study presented in this paper adds to previous studies with new analyses that shed light on previously unreported change-related processes and events, with a unique focus on studying them over time. In approaching the research question for this study, new data was added, including the full range

of observation notes from 62 days on-site and 14 new meetings, including six client meetings, five 'change workshops' where organizational and structural changes were discussed with internal and external representatives and three other meetings including two external client preparation meetings and a board meeting. We also included six additional interviews conducted with Entur's agile methods specialist. While confidentiality clauses prevent us from sharing original data material, we share examples from all data sources throughout the manuscript. Table 1 provides details of the underlying data material supporting the study.

Observations Observations were conducted on a regular basis, as shown by the short bars in Fig. 2. The first author conducted all observations. Additionally, the third author was present on a few occasions, for example, during the 'change workshops'. Because we wanted a broad and detailed data material, we observed both meetings and the everyday work at the office. The observation days varied somewhat across the weekdays during the 1.5 years. Our presence varied as we wanted to observe the broad range of inter-team meetings conducted across the week. For example, we could be present one week on Monday and Thursday, the next on a Wednesday, and yet another week we could be absent. For the meeting observations, we observed meetings on all organizational levels, primarily inter-team meetings such as the product owner prioritization meeting and the tech lead forum, and team-level meetings such as team retrospectives and daily stand-up meetings. We were also able to observe client meetings and a board meeting. Because of our ongoing presence, we were able to join in on spontaneous ad hoc meetings as well as planned meetings. We used an observation protocol detailing, for example, the physical setting, people present, and tools and artifacts used. These observations left us with a rich data material with detailed descriptions of the observation setting, including, but not limited to, a focus on the coordination of development activities (see Table 1). The protocol template is included in Appendix 1.

Interviews In addition to the field observations, we conducted 37 interviews to gain a deeper understanding of the case. Some interviews were conducted on the same day. Interview days are illustrated by the long bars in Fig. 2. Interviews were held as open conversations in a semi-structured format. We used the same interview guide throughout the data collection (see Appendix 2). The interview guide was slightly modified to focus on the disciplinary area of each role (for example, product owners were asked more about clients and products, whereas architects were asked more about technical architecture). Concerning our focus on change and coordination, the questions remained the same. Example questions include: "*What challenges do you see now and in the future in the development program?*", "*How has your role changed over time?*" and "*What do you think have been the biggest developments here in relation to coordination across teams?*" We interviewed 25 individuals in total. Six participants, four program managers, and two architects, were interviewed twice. One person, the agile methods specialist, was interviewed six times. These follow-up interviews, held approximately bi-monthly, were more conversational and did not follow the same interview guide as the other interviews. We included these interviews as they contributed to understanding change and coordination over time in the program. The interviews lasted 50 min on average. The first author conducted 29 interviews, the second author conducted two interviews, and the six interviews with the agile method specialist were group interviews conducted by the first, second, and third authors. The first author translated interview quotes from Norwegian to English, and all authors checked the quality of the translation.
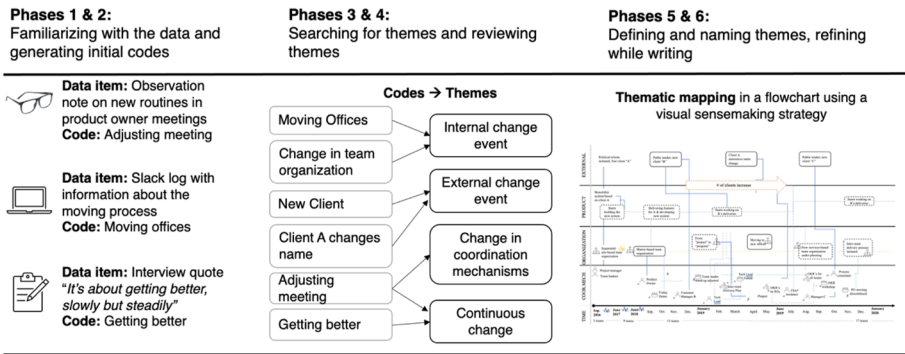
Fig. 3 Illustration of the Thematic Analysis

Supplementary Material During the fieldwork, we were given access to Entur's internal digital communication tool, Slack, and much of the development process documentation through JIRA and Confluence. The supplemental material allowed us to follow the use of these coordination mechanisms in real-time and record particularly interesting written conversations or documentation pages and provided the ability to search in past conversations and records. For example, we were able to follow in detail the introduction of the new Slack guidelines described in Section 4.3, not only through interviews and observations but also to see the actual implementations in the Slack channels as they took place. In addition to these major sources of information, we collected company presentations, and meeting invitations and agendas sent to us via e-mail.

3.3 Data Analysis

We used thematic analysis (Braun and Clarke 2006, 2012) to analyze the data. We chose this analytical approach because it is a flexible method that allows the researcher to handle a large data set like ours. Thematic analysis is also well-suited for interpretive research because it recognizes the active role of the researcher in shaping the analysis and the findings (Braun and Clarke 2012). Thematic analysis allows researchers to work systematically to identify and analyze commonalities across large and varied project data. The method is flexible because the six phases are iteratively conducted. This means the data can be analyzed while new material is added, which was suitable for our longitudinal fieldwork. Typically conducted following a six-phased, iterative process, the method allows for a deep analysis where results are grounded in the data (Braun and Clarke 2012). Within software engineering, the method is widely used to provide a deeper understanding of the content and meaning of data (e.g., Wohlin and Aurum 2015; Munir et al. 2016; Berntzen et al. 2022; Hussain et al. 2022; Ågren et al. 2022). The following sections provide more information about how we conducted the analysis. Figure 3 illustrates the thematic analysis process.

A thematic analysis is ideally both inductively and deductively guided, thereby ensuring strong links to the data material and the existing literature on the subject. In practice, the analysis is often more strongly guided by one of the approaches (Braun and Clarke 2006, 2012). In our case, while we ensured covering both approaches across the phases of the analysis by using Jarzabkowski et al. (2012)'s model of how coordination mechanisms are created from disruptive events, it was our data that most strongly guided the analysis and

the resulting themes in that we did not limit our analysis to only look for themes related to this model. Instead, we kept an open mind as to what other change aspects were present in the data. This made us look beyond change events only to notice how continuous change over time influenced the coordination mechanisms in the case.

Phases 1 and 2: (Re-)Familiarizing with the Data and Generating Initial Codes During these phases, we reviewed the complete data material from the perspective of change, transition, and evolution of the program and its coordination mechanisms over time. Because the data had been used previously for other studies, we already had a collection of existing coordination mechanisms used in the case. The first author re-read the interview transcripts, field notes, meeting observations, and supplementary material and made analytical notes along the way. These were shared and discussed with the rest of the author team. Specifically, the first three authors discussed the opportunity to describe change in coordination over time during the fieldwork. Following this, the first and fourth authors coined the idea to describe and analyse how coordination mechanisms evolved over time during in-person analysis workshops. Following these initial activities, we proceeded to generating initial codes. As illustrated in the left-hand side of Fig. 3, at this stage, we used broad, descriptive codes. For example, a Slack log containing the discussion of the office moving was assigned the code 'moving offices'. The initial coding phases ended when we had assigned relevant codes to all data items.

Phases 3 and 4: Searching for Themes and Reviewing Themes During the third phase, we shifted from generating codes to searching for themes. Themes are defined as prevalent patterns within the data, that is, recurring instances of similar types (Braun and Clarke 2006), for example, types of changes that happen outside the organization. We searched for such patterns by grouping and re-grouping the codes from the first and second phases. We identified several lower-level change themes, including changes related to public tenders and clients, changes in the organization of the product, changes in the internal team structures, changes in meeting practices, roles, and tools and artifacts (i.e., coordination mechanisms), changes in the physical location, and much more. Shifting to the fourth phase, we examined the change themes in detail and combined themes that could be grouped under larger themes in light of the data and the definitions of change presented in Section 2 (Jarzabkowski et al. 2012; Langley et al. 2013). We ended up with three high-level change themes, namely external and internal *change events* and *continuous change* (see Table 2), as well as *changes in coordination mechanisms* (see Table 3). Here, we used the categories from the taxonomy of inter-team coordination mechanisms (Berntzen et al. 2022), that is, *meetings, roles,* and *tools and artifacts*, but focused this analysis on changes in the mechanisms. The middle part of Fig. 3 provides a simple representation of how we arranged codes into themes.

Phases 5 and 6: Defining and Naming Themes and Producing the Report The final two phases of a thematic analysis tend to intertwine as findings are often put to scrutiny through writing up the final report (Braun and Clarke 2012), which was also the case in our analysis. We selected interview quotes, field note passages, and supplemental material for presentation and related the findings back to the research question. Themes were refined during the writing process as all authors wrote, read, and discussed the material. As illustrated by the right-hand side of Fig. 3, we organized the results using a visual mapping strategy (Langley 1999) (see Fig. 4), which helped us further refine the presentation of our findings. At this point in the analysis, we conducted member checks by providing Entur representatives with the draft to receive their input to ensure that the findings also held practical relevance.

4 Findings

In presenting our findings, we first describe change events identified from the longitudinal data. Next, we provide detailed examples of changes in the coordination mechanisms used in the program over 1.5 years. We explain how each example relates to the *disruption of existing ways of coordinating*, *orienting to absences in coordinating and making* new efforts to coordinate, which *creates new patterns of coordinating* (Jarzabkowski et al. 2012). Tables 2 and 3 summarizes the change events and changes in coordination mechanisms. Figure 4 presents a process flowchart consisting of five lanes. The first lane represents the organization's external environment, and the second and third lanes represent the product and the internal organizational environment. To illustrate the organization's growth, we have included a representation of the number of teams between the second and third lanes. The fourth lane shows the coordination mechanisms used at Entur that changed over time in relation to changes identified from the analysis and showcased in the above lanes. Finally, the fifth lane, 'time,' displays the timeline for the changes. In Fig. 4, each change event is indicated by a circle containing a letter and a number (e.g., E1, I2) corresponding to the sub-sections in Sections 4.1 and 4.2. In addition, the arrows with dotted lines that run alongside the lanes represent continuous or ongoing changes in the product or the organization, described in Section 4.3. Arrows are drawn from each box to the relevant mechanisms to symbolize the relationship between a change event and a coordination mechanism. In a large-scale development program like Entur, there are more changes than can be described in a report or presented in a flowchart. We, therefore, selected the most compelling examples related to our research questions.

4.1 External Change Events

The first theme is related to changes that took place in Entur's external environment. We consider them 'events' because they happened at a specific point in time. These are presented in the upper lane of Fig. 4. One of the product owners explained the importance of context surrounding the organization: "*Our whole external context, with the public reform and all it entails, moving from one software system to another, it has all been decided by external circumstances. Our maneuverability is shaped by it*" [I01, Product Owner]. In the following, we present three notable external change events.

External Change Event 1 (E1): New Client When Entur was established in 2016, "Client A" was the only railway operator in Norway. Therefore, much of the development of the new software system during Entur's early development phases were based on Client A's needs and prioritizations. The situation changed in October 2018 when an international railway operator, "Client B," won a public tender following the public transportation reform. Client B would now establish in Norway and use Entur's sales system. Following the announcement was a period of preparation before Entur started working with Client B's requirements in early 2019. These requirements were added on top of other priorities. "*There will certainly be tough deadlines towards Client B, too! Not preparing for that would be naïve*" [I03, Program Manager].

Adding a new client disrupted existing ways of coordinating in that more dependencies was added, and the need for overview across the teams increased. In relation to the onboarding of Client B, new coordination mechanisms were introduced as a response to

orienting to the new coordination needs, and new patterns of coordinating were formed by the introduction of new coordination mechanisms. These included a new customer manager role to complement the customer manager of Client A and two artifacts, an inter-team backlog, and an inter-team delivery plan to track which teams worked on what deliveries. The new role and artifacts can be seen in the 'Coordination mechanisms' lane at the bottom of Fig. 4, following the lines from the upper lane.

External Change Event 2 (E2): Client Name Change and Rebranding Another important change event occurred in March 2019, when Client A announced that they intended to rebrand, changing their name, logo, and visual appearance. This strategic rebranding was a huge change event for Client A, who had had their previous name for almost two centuries. It also directly impacted Entur, who had to adapt both the old and the new systems, as the old name was hardcoded in the legacy code throughout the old system. During a team leader stand-up meeting in late March 2019, the agile method specialist informed the team leaders of the change: "*In a month, Client A's web pages will close, and a new web page with the new brand will launch. To us, this means that everything that is visible externally needs to be renamed and visually appear as Client A's new name. […] the teams need to implement changes in the code. For example, the names of all product IDs in the system must be updated.*" [Meeting observation, March 2019]. The event is presented in the middle of the 'external' and 'product' lanes of Fig. 4. This change event illustrates how an uncontrollable external environment had implications for the teams and the system development. While no new coordination mechanisms were added, we observed how existing mechanisms were updated to accommodate the change in coordination needs and the extra work associated with the name change. For instance, we observed that new lanes were added to delivery plans and roadmaps (see Fig. 6 for an example of a physical roadmap) and that the name change was discussed regularly at inter-team stand-up meetings.

Exernal Change Event 3 (E3): Another New Client A third change event took place in mid-2019, when "Client C" won another public tender, resulting in a change process in late 2019 similar to that of Client B's entrance. The onboarding of Client C started in early 2020, about at the time when our data collection period ended. Concerning changes in coordination mechanisms, a new customer manager was added to support Client C. Existing inter-team backlogs and delivery plans were updated to make room for incoming requirements and deadlines from Client C, and in January 2020, a workshop was held looking back at lessons learned from onboarding Client B to further adjust practices in preparation for the third major client.

4.2 Internal Change Events

The second change theme is what we refer to as *internal change events*. As opposed to the external change events, these changes originated within the boundaries of the organization. In Fig. 4, these change events can be seen in the 'Organization' lane. We present four notable internal change events.

Internal Change Event 1 (I1): Changes in Team Organization When Entur was established in 2016, there were five sequentially organized teams. Each team worked on developing their own part of the system, and there was little to no communication between the teams. "*It was truly bad! But we have worked our way forward little by little. First,*

Table 2 The major change themes, based on Jarzabkowski et al. (2012) and Langley et al. (2013)

	Description	Examples
External change event	Time-specific changes taking place in the company's external environment, and the control of which are beyond the organizational boundaries but has implications for actions within the organization	-Transportation reforms leading to the onboarding of new operators/clients (E1, E3) -Client makes name change and rebrands which impacts the development (E2)
Internal change event	Time-specific changes initiated within the organization, controlled by the organization, and based on pre-planned assessments. Has implications for inter-team and team-level coordination	-Reorganization of team or organization structure (I1, I2) -Moving offices (I3) -Implementing shared delivery routines (I4)
Continuous changes	These changes have no set dates but occur on an ongoing and ad hoc basis. Can be both internally and externally driven	-Adjusting meeting practices based on retrospectives (internal) -Picking up "best practices" such as new technology and development methods (external)

we got the priority boards, and then that didn't work so well. To begin with. But then we started to do something, and things got a little better. Also… the [software] modules were maturing, so we had to start talking across teams, and so we have also moved forward in an agile way, sort of" [I13, team leader]. In 2017, a new team matrix-based team organization was established, with nine teams organized according to product delivery areas and inter-team roles [Company presentation, November 2017]. By September 2018, there were thirteen development teams organized under nine delivery areas; examples include Pricing, Sales, Ticketing, and On-board services. This new organization was designed to allow for more and better inter-team coordination in response to the coordination needs

Table 3 Changes in coordination mechanisms. Categories based on Berntzen et al. (2022)

| Coordination mechanism category	Description	Examples of changes
Coordination roles	Roles are coordination mechanisms performed by people coordinating with other people that contribute to managing dependencies within or across teams	-Introducing the product owner role -Discontinuing project manager -Introducing tech lead role -Introducing chief architect role -Adding customer managers
Coordination meetings	Time-boxed or ad hoc arrangements where dependencies are managed by enabling people to discuss, share knowledge and negotiate shared understandings	-Shortening meetings -Changing meeting scope -Removing meetings -Adjusting meeting focus -Increasing unscheduled meetings
Coordination tools and artifacts	Tools manage dependencies by supporting the development process, while artifacts are by-products of the development process	-New Slack communication guidelines -Adjusting meeting agendas -Adding Inter-team backlog -Adding inter-team delivery plan
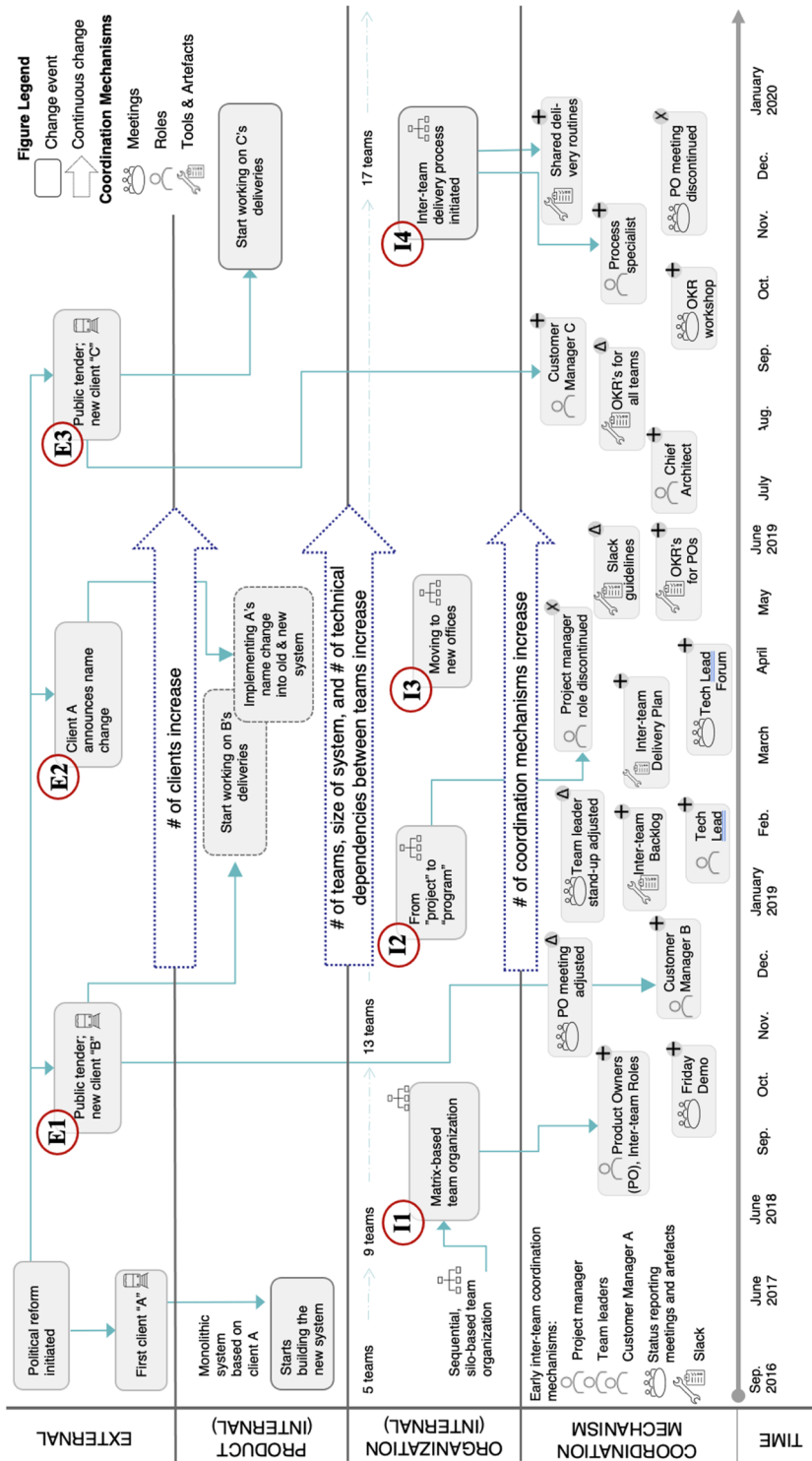
Fig. 4 Change events and changes in coordination mechanisms represented through a visual mapping strategy (Langley 1999). The red circles containing letters and numbers (e.g., E1, I2, etc.) correspond to the sub-sections in Sections 4.1 and 4.2. New coordination mechanisms are indicated with a plus sign (+), discontinued mechanisms with a cross (✗), while changes to existing mechanisms are indicated with a delta (Δ)

following the current size of the program. This also led to the introduction of more coordination mechanisms, as the existing ways of coordinating were no longer efficient. The product owner role was implemented for each of the nine delivery areas, and inter-team roles such as agile method specialist, customer manager, and development manager were formally implemented in the organization matrix. After we concluded our data collection, as Entur continued to scale, there was a need to consider yet another change in the team organization. Rather than being organized according to delivery areas, they would gradually organize according to product areas from 2020 onwards.

Internal Change Event 2 (I2): From 'Project' to 'Program' Initially, the software development at Entur was organized in a development project referred to as the Leap Project. This project was directly linked to the political reform and primarily focused on building the new software, which was done in parallel with running services on the old sales system inherited from Client A. However, because of the scope and magnitude of work associated with developing a new sales platform while running and maintaining the old system and because new clients were added, the project was expanded into an ongoing development program with no end date rather than running many different projects. However, while the Leap project officially ended by the end of 2018, traces of the project organization remained for some time as the program members oriented to new ways of coordinating, both in terms of coordination mechanisms and way of thinking. A product owner explained:*"We have simplified how and how much the teams report, but all the mechanisms are still there*" [I20, Product Owner]. For example, as shown in Fig. 4 and discussed in Section 4.3, the Project Manager role remained until April 2019. This example illustrates that coordination mechanisms are not only created in response to change events and that new patterns of coordinating can include the discontinuation of a mechanism.

Internal Change Event 3 (I3): Moving Offices In April 2019, a third internal change event occurred when Entur moved offices to a new building. When we began our data collection in August 2018, Entur was located on a single floor in a larger office complex. As they continued to grow, the office space became too small to support the program's need for inter-team coordination. This was reflected in our observations during 2018. Task boards hung wherever they fitted in, and there were few open spaces for informal meetings and socializing. Stand-up and prioritization meetings took place in corridors and were constantly interrupted by people passing. The meeting rooms were too small for any inter-team meeting and too small for many of the development teams (of which the largest counted 16 members). Moreover, due to the lack of space, several of the development teams had to sit off-site, which was an obstacle to efficient inter-team communication and coordination.

The office move was a big change event that took time and effort, but it was necessary for successful unscheduled coordination and communication across teams as the program continued to scale. *"Communication might get better now that all teams are in the same building. Because before, we couldn't walk over to each other, but now we can"* [I13, Team leader]. The new offices spanned two floors, connected by a large open staircase that could be used for informal seating and presentations. They had several large meeting rooms and two large open spaces that allowed for more efficient use of existing coordination mechanisms. For instance, task boards could be displayed in the open space, and inter-team meetings could now be held in well-suited areas (see Figs. 5 and 6 in Section 4.3). Despite this upgrade, the new offices were also at the risk of becoming too small as the program continued to scale. *"We just keep growing. We moved to get more space. Now, new desks are*

constantly being added, and we no longer have a dedicated stand-up room as we need the space for workstations" [I24, Tech lead].

Internal Change Event 4 (I4): Inter-Team Delivery Routines In the large-scale program, teams often worked on the same deliveries or on inter-dependent deliveries. Accordingly, as Entur continued to scale, there was a growing need to align the delivery process across teams. In October 2019, measures were taken to establish an inter-team delivery process with common delivery routines. This was done to improve predictability in deliveries across teams to the clients. However, to keep with an agile way of thinking, these new processes were not implemented overnight but piloted and tested in one central team before scaling further. "*We try it out with a smaller group to see 'do we see the value of doing this?' We start off narrow, and if people think it gives value, then it is a good model for testing before we go full scale*" [I19, Program Architect]. We observed the piloting phase conducted in October-December 2019. As part of this phase, several new coordination mechanisms changed. As seen in the lower right corner of Fig. 4, new coordination mechanisms included a process specialist role and shared routines for using JIRA and Confluence.

4.3 Continuous Changes in Coordination Mechanisms

In addition to the changes in coordination mechanisms following the change events, there were *continuous changes* in coordination mechanisms that reflected the continuous growth and evolution of the program. Often, these were ongoing changes that went unnoticed. *"You don't put down in writing that 'this is how we do things here,' and then people know what it's like. It's more like... it flows a bit. And suddenly, things have changed a little. You just notice, like, 'oh yes, things have changed' [laughs]"* [I04, Product Owner]. Table 3 and the bottom lane in Fig. 4 illustrate these changes. In the following, we provide examples for each of the coordination mechanisms categories, *roles*, *meetings*, and *tools and artifacts*.

Coordination roles are coordination mechanisms performed by people coordinating with other people that contribute to managing knowledge dependencies within or across teams. Entur had several coordination roles, including team-level roles, such as product owners and team leaders, and inter-team roles, such as customer managers and architects. More coordination roles were added as the program scaled, and some roles changed in response to the program's growth. As mentioned in Section 4.1, the project manager role was discontinued after Entur changed from 'project' to 'program'. The person who filled this role, an external consultant, left in April 2019. However, it took some time for the developers to adjust to this change. "*The project has long been shut down, and our focus is now on product development. But we notice that the project way of thinking remains, and the idea of the project manager role also remains. After a stand-up last week, a team leader asked: 'Who's our project manager now? Who will follow up on us*?'" [I20, Product Owner].

As Entur grew in response to the scaling and development of the new software product, at the same time as the old system was kept in use, the number of technical dependencies increased. This led to an increased need to focus on the software architecture both within and across teams. As a result, the tech lead role was established in all development teams in January 2019. "*The role is about technical coordination and in a way be a person within the team that has the knowledge and insight about the team architecture that can discuss and be part of making technical decisions, within the team, and also outside the team*" [I21, Tech Lead]. Additionally, a chief architect role was added

in June 2019 "*responsible for coordinating the architects and be part of deciding the scope of the architecture function at Entur*" [I14, Program Architect].

The product owner role was a central role at Entur associated with many changes. As explained in Section 4.2, the role was established during the reorganization in 2017 to correspond with the nine delivery areas. Among the product owners' primary responsibilities was coordinating priorities towards the overall product deliveries and communicating the needs and prioritizations of each development team at an inter-team level. At first, there was a 1:1 correspondence between the delivery areas and development teams. "*What's interesting about the product owner role here is that it's influenced by the situation we're in. Now and in the future. When we implemented the role, the thought was that the product owners themselves would be part of shaping the role. To own their delivery area and be the CEO of their own product, so to speak*" [I06, Program Manager]. However, as the program scaled, this quickly changed such that some product owners became responsible for more than one team.

Importantly, at Entur, the product owners were considered as part of the development teams and not an inter-team role. Even the two product owners who had more than one team each (see Fig. 1) were primarily affiliated with the teams rather than with the product owner group. This primary affiliation with the teams represented a challenge for inter-team coordination and prioritization of deliveries across teams. "*They all have the same role. But they perform it very differently. That's the problem*" [I12, Program Manager]. Another manager explained: "*They have no sense of group affiliation. But it's a point to make coordination across the teams work. And if we say coordination across teams is one of our challenges, that includes the product owners. They don't seem to talk enough to each other*". [I03, Program Manager]. During our fieldwork, we witnessed several adjustments of the coordination mechanisms surrounding the product owners in order to manage inter-team dependencies more efficiently. They held quarterly retrospectives where inter-team coordination issues were addressed, and changes and adjustments to improve inter-team coordination was made. Most notably, the prioritization meeting (to be introduced below) but also changes in how they communicated on Slack, what to discuss in their weekly meetings and how they could improve inter-team coordination on an ad hoc basis. All along, there was a promise of change attached to the role. "*I do not believe the product owner role is the same now as next year or the year after. How many product owners do we need today, tomorrow, or in the long run? I think that number will vary* [I03, Program Manager].

Coordination meetings are coordination mechanisms where dependencies are managed by enabling people to discuss, share knowledge, and negotiate shared understandings. At Entur, both scheduled and ad hoc, unscheduled meetings were frequently used to manage dependencies within and across teams. During our data collection, we observed the ongoing adjustment and improvement of coordination meetings. The program members had the autonomy to adjust these mechanisms, which often happened during inter-team retrospectives. For example, the product owner prioritization meeting was adjusted in November 2018 based on input received in a retrospective meeting for the product owners. After this, the product owners kept experimenting with the meeting format, and in November 2019, they decided to discontinue the meeting. "*The last few weeks, the product owners have had stand-up meetings instead. I asked them if using the [prioritization] task board still made sense, and most said they did not want to use it anymore*" [I03, Program Manager]. Similarly, the team leader stand-up meeting was adjusted in early 2019 following a team leader retrospective. During the retrospective, held in February 2019, some team leaders

complained that the stand-ups had become time-consuming reporting meetings. "*In the retro, we decided to only focus on issues relevant across teams, which has saved us some time. So, the stand-up improved, for now at least*" [I13, Team Leader].

Coordination meetings were also added during the study to fit the program's needs. For instance, an inter-team Program Demo, where teams showcased parts of their development work every week, was established in September 2018. This demo contributed to coordination by enabling shared knowledge across teams. In March 2019, following the implementation of the tech lead role earlier in the year, the first "tech lead forum" was held. This was a bi-weekly meeting for the tech leads and the program architects aimed at sharing knowledge and coordinating technical dependencies across teams. After the forum was established, it took some time to adjust and find the right format. When the forum had been running for some months, a program manager explained: "*In the beginning, not everyone understood their role or wanted to speak up and share their opinion. We wanted to be careful with telling the tech leads what to do, want them to figure it out, and take responsibility themselves. They're starting to adjust, now we start to see discussions and the type of knowledge sharing that we wanted*" [I12, Program Manager]. The tech lead forum was modeled after the ideal of communities of practice and was planned to establish several such inter-team fora for other inter-team coordination areas, such as software quality and testing and DevOps. "*We wanted to start off with one such forum, not all at once, and see what we more we wanted over time*" [I06, Program Manager].

In addition to these scheduled meetings, unscheduled coordination meetings improved following the office move in April 2019, as there was more open space available and more meeting rooms that enabled spontaneous meetings (see Fig. 5). Additionally, the open staircase was used to display inter-team coordination mechanisms (see Fig. 6) and enabled easy access to members of other teams. This open staircase was also designed with seating and was used for informal lunches, company presentations, and hangouts.

Coordination Tools and Artifacts Coordination tools are coordination mechanisms that manage dependencies by supporting the development process, for example, a chat tool, while coordination artifacts are considered by-products of the development process, for example, documentation. At Entur, coordination tools and artifacts were used widely, both at the team and inter-team levels. Over time, more inter-team tools and artifacts were added to align inter-team coordination. In the past, the teams had their own backlogs and delivery plans, which the product owners reported on during their prioritization meeting. However, as the program scaled, additional mechanisms were needed. In January 2019, an inter-team backlog was added, and in March, an inter-team delivery plan was put together in response to client growth, as described in Section 4.1. In May/June 2019, Entur started experimenting with OKRs, a goal management framework that Entur used as a coordination tool. They first tested OKRs with the product owners, and as that gave promising results, it was decided to expand the use of OKRs to involve the team leaders, the architects, and the management group. The goal was that all Entur were to use OKRs by 2020. "*The goal is to gain an overview and to give insights to the organization. And to be able to say, 'this is where we're at,' right. And use this insight to evaluate if something works or not and act* [I21, Program Architect].

In addition to the introduction of new coordination tools and artifacts, existing coordination tools were adjusted as needed. The digital communication tool Slack had been used since the outset in 2016. Slack is built up of channels, which users can create and name within certain boundaries set by the software. Many of the channel names were quite

Fig. 5 The new offices brought new coordination arenas like this multi-purpose room

similar. For example, there could be a channel called "ClientA_deliveries" and another called Client_deliveries," and so forth. By early 2019 the number of channels and various, often similar, names for inter-team and inter-organizational channels became confusing and misleading for Entur employees. This also represented a risk of information being shared with the wrong clients. Accordingly, the need to align communication on Slack resulted in new guidelines for creating and naming Slack channels. During April and May 2019, the new Slack guidelines were introduced. The development teams were encouraged to contribute with input before and during the implementation phase. After the new guidelines were introduced, there was a period of improving the new Slack practices. This example illustrates how the need for more alignment in the coordination process initiated new Slack guidelines, resulting in changes in how the coordination mechanism Slack was used and how the change led to a need to adjust further and improve the use of the coordination mechanism.

As a whole, continuous change and improvement were a part of the program's core culture and practiced at all organizational levels, from the 'change workshops' where managerial-level employees discussed structural and organizational changes to the team-level 'coffee and architecture'-meetings that some tech leads held to get their team members' input to the tech lead forum. The latter is an example of employee-driven changes resulting

Fig. 6 The open office space was used to display coordination mechanisms, such as this delivery plan

from knowledge sharing of "best practices" across teams. For example, a tech lead told us: "*'Coffee and architecture' is a 15-min meeting I initiated with the team. I picked it up from one of the other tech leads during a tech lead forum. I use it to gain input and involve the team*" [I21, Tech Lead].

The examples presented in this section demonstrate that at Entur, responding to change was part of everyday work. However, there was also a risk that introducing many initiatives at once could be counter-productive, as employees could perceive that there were too many changes: *"There's always a lot going on. And that's a factor: How much do we adjust at the same time? It might actually become stressful to have to get familiar with new things all the time. People might lose interest.*" [I20, Product Owner]. Further, despite the always ongoing changes in coordination mechanisms and Entur's ability to continuously improve, inter-team coordination remained a challenge in the large-scale program. "*The greatest coordination challenge is synchronization across [teams] in the overall deliveries. If there's one thing that haunts us, this is it* [I03, Program manager].

5 Discussion

Large-scale agile development projects and programs are often long-term and filled with changes, which has consequences for coordination. Change is often understood either in the form of events or patterns of events (Jarzabkowski et al. 2012) or as a continuous process or flow of activities (Langley et al. 2013). In this study, we took an explorative approach to both views of change. Using the theoretical lens of Jarzabkowski et al. (2012) explaining how coordination mechanisms are created in practice, we sought to better understand change and coordination in large-scale agile by investigating the research question: *What types of organizational changes influence coordination mechanisms in large-scale agile, and how do these mechanisms change over time?*

Through thematic analysis, we identified three themes covering the organizational changes that influence coordination mechanisms, namely external and internal *change events*, and *continuous changes* (Table 2). Further, we illustrated how coordination roles, meetings, and tools and artifacts changed over time (Table 3 and Fig. 3). The themes were derived partly based on our conceptual understanding of how coordination mechanisms are formed from disruptive events (Jarzabkowski et al. 2012) and our understanding of the importance of continuous improvement in software engineering (Fitzgerald and Stol 2017). However, through our close and detailed engagement with the data material during the analysis, the findings were strongly linked to the empirical material (Braun and Clarke 2012), and it became clear that our theoretical lens did not cover all aspects of coordination observed in the data. As an outcome, we present a model for understanding change in coordination mechanisms over time in large-scale agile, illustrated in Fig. 7. This model extends Jarzabkowski et al. (2012)'s theoretical framework and forms the basis for a model of change coordination mechanisms in large-scale agile. We now discuss the implications of our findings.

5.1 External and Internal Drivers of Change in Coordination Mechanisms

First, our findings show that changes in the external and internal environment lead to changes in the coordination mechanisms used to manage dependencies. Both internal and external change events can be compared with the disruptive events that cause
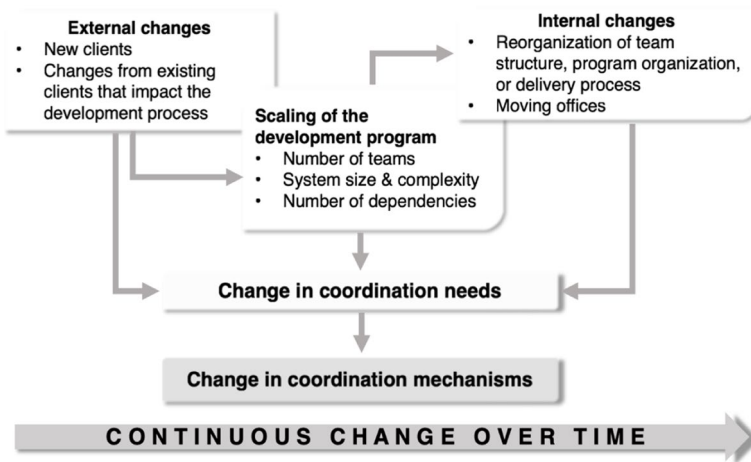
Fig. 7 A model of change in coordination mechanisms over time

coordination mechanisms to break down. However, Jarzabkowski et al. (2012) focused on one large disruptive event (i.e., the organizational restructuring). We find that events do not need to be of such a magnitude or disruptive level to lead to a change in coordination mechanisms. Moreover, contrary to Jarzabkowski et al. (2012)'s model, we find that the coordination mechanisms themselves do not necessarily have to 'break down' to change. Often, small adjustments or what we term continuous change were sufficient to cause a substantial change in how dependencies were managed. As illustrated by the example with the product owners' prioritization meeting described in Section 4.3, input received during retrospectives can lead to the adjustment of coordination mechanisms. This focus on continuous adjustment and improvement was a key strength at Entur and a core feature of agile that is not captured by the model of Jarzabkowski et al. (2012).

With respect to the external change events, these were initiated and controlled beyond the organization's boundaries. The upper left corner of Fig. 7 shows how external changes drive change in two ways. First, external change events may contribute to the scaling of the development program (upper middle box of Fig. 7), thus indirectly contributing to changes in coordination needs as the dependencies increase in number and complexity. The external changes caused by the public transportation reform and the addition of clients due to the public tenders can be seen as major drivers of change. These external change events contributed to the continued scaling and growth of the program and, as such, an increase in different types of dependencies (Strode 2016). From scaling follows a change in coordination needs and, subsequently, a change in coordination mechanisms. For example, technical dependencies increased with the size and complexity of the software, but also with the number of teams developing features. Entur established the tech lead role and the tech lead forum in response to the need to coordinate the continuously growing technical dependencies across teams. Additionally, external change events could directly lead to changes in coordination needs without impacting program growth, as with the name change of Client A (see the arrow that leads directly from the upper-left corner to the box in the middle of Fig. 7).

Internal change events, such as reorganizing the team structure, or moving to a new office, were also drivers of change in coordination mechanisms. Internal change events

can be understood as ways of adapting to the growth in the client base and the size of the development program. For example, as illustrated by the arrows running from the upper middle to the upper-right corner of Fig. 7, the increased number of teams increased the need for inter-team coordination due to more knowledge dependencies across teams. Adding new coordination meetings, such as the Friday Demo, and adjusting existing meetings, such as the team-leader stand-up meeting, were ways of managing inter-team knowledge dependencies. Also, introducing new Slack guidelines was an important way to manage knowledge dependencies when the lack of channel naming conventions caused confusion and the potential risk of information getting out of hand. These findings relate to a study of coordination in global software engineering (Stray and Moe 2020), where the findings showed that the lack of formalized coordination procedures on Slack constituted a challenge to effective dependency management. Additionally, the increase in resource and process dependencies at Entur, caused by more teams needing to coordinate deliveries, was managed by introducing new coordination tools and artifacts, such as inter-team delivery plans and new inter-team coordination roles, such as the process specialist.

We found that changes in coordination mechanisms not only happened reactively because of a disruption or a breakdown in coordination (Cataldo and Herbsleb 2012; Jarzabkowski et al. 2012) but also as a result of Entur's wish to continuously improve their development process (Fitzgerald and Stol 2017). This is in line with Edison et al.'s (2022, p. 14) review, who points out that when organizations seek to improve, "constant change is inevitable" across team structures, processes, tools, and tools metrics. Further, we found that changes were initiated both top-down and bottom-up. In the 'change workshops,' the managers and key inter-team roles discussed organizational and structural issues such as team organization and the establishment of communities of practice (e.g., the tech lead forum), gaining input on "best practice" from research and practice outside the organization. These workshops were examples of top-down drivers of change. From the bottom-up, retrospectives and the tech lead forum were arenas where team representatives could initiate change by discussing and identifying changed coordination needs and how to adapt to them. This is in line with findings from two other large-scale programs in the telecom industry where both top-down and bottom-up approaches to decision-making were used (Moe et al. 2021).

Understanding the dynamics of change in large-scale agile is not easy, as change is not clear-cut. It is difficult to pinpoint, for example, when a "decision" turns into "implementation" and exactly when something changes (Van de Ven and Poole 2005). These issues also pertain to the changes observed in our case study. Although we chose to arrange the themes in terms of change events, there were many instances where it was difficult to see the clear boundaries of the changes. In our results, this is perhaps most clearly illustrated by the change from 'project' to 'program' described in Section 4.1. and Section 4.3. Even though the change was an event that occurred on a specific date, the traces of the project organization remained for some time. In line with the process theoretical perspective (Pettigrew 1990; Langley et al. 2013; Ralph 2018), our findings show that scaling is a constant process that unfolds over time, shaping both organizations and the mechanisms used to manage dependencies. Our study underlines the importance for practitioners in large-scale agile to be mindful of both these aspects and to avoid deciding on a fixed coordination strategy upfront but recognizing the ongoing need to sense and respond to the situation and continuously improve coordination practices in growing organizations.

5.2 Continuous Growth Requires Continuous Change and Improvement

Second, our findings show that over time, the continuous scaling of the program leads to "more of everything." This is illustrated by the arrow running across the bottom of Fig. 7. This continuous scaling was largely fueled by external events that led to the increase in clients. As the number of clients grows, so does the number of teams needed to develop the system. The more teams, the more dependencies. As such, at Entur, it seemed that change events and the program's growth were closely associated and that both led to a continuous change in coordination needs. In this sense, our findings underscore the importance of dependency awareness in the face of change (Bick et al. 2018), as the ability to sense and respond to changing coordination needs requires understanding how dependencies change.

Our analyses have shown how coordination mechanisms were added, modified, and removed over time in response to changing external and internal environments. The strongest tendency was that the number of coordination mechanisms increased over time. Figure 4 shows that 13 new mechanisms were added, and four mechanisms were adjusted, in relation to the change events described, whereas only two were removed. This finding is interesting in comparison to a recent study where Dingsøyr et al. (2022) identified 27 mechanisms while the case program used a mix of traditional and agile project management techniques. After transitioning to autonomous cross-functional teams, 14 coordination mechanisms were used. In another study, Moe et al. (2018) found that their case started out with many scheduled meetings but that over time, unscheduled meetings were used more. This was explained by the maturing that happened over time. We, on the other hand, found that although some mechanisms were removed, overall, more coordination mechanisms were added over time. We explain this by the continuous growth of the program. Initially, Entur only had one railway client and five teams with low coordination needs, as the number of dependencies between teams was perceived as relatively low. Over time, however, the number of dependencies increased as Entur continued to scale, which required the introduction of more coordination mechanisms.

Our findings can further be related to Fuchs and Hess (2018)'s model, where large-scale agile transformation is understood as episodic phases where each phase is characterized by a radical change followed by a period of incremental changes, and to the lean concepts of *kaikaku* and *kaizen* (radical and incremental change, respectively) (Fitzgerald and Stol 2017). Jarzabkowski et al. (2012) describe phases of destabilization following a change event, during which mechanisms are abandoned, re-formed, or changed before they stabilize. In Entur, however, due to the continuous growth and the unpredictable external environment (i.e., new clients following public tenders and the political backdrop of the transportation reform), such a stabilization period never really seemed to occur. Instead, they needed to rely on sensing and responding to the situation at hand and adapt their use of coordination mechanisms to manage the relevant dependencies at any given time, which often meant adding new mechanisms in response to the continued growth. If the situation stabilizes when the software goes into a maintenance phase and no more clients are added, the need to continue to scale should vanish, a situation in which they might be able to reduce the number of coordination mechanisms in use.

5.3 Responding to Change by Using the Right Mechanisms at the Right Time

Third, our findings illustrate how having the right coordination mechanisms in place builds resilience to change. As seen in the example of the renaming of Client A (E2), although

the event was a significant external change that impacted the coordination needs associated with implementing the name change in the system, Entur was able to handle the event quite smoothly using the mechanisms that were already in place. Information about the event was efficiently distributed to the teams via inter-team coordination mechanisms (e.g., Slack and meetings), and existing inter-team coordination mechanisms, such as task boards and delivery plans, were adjusted as needed to meet the change. This example supports the notion that some coordination mechanisms may be more effective than others in managing certain dependencies (Strode et al. 2012; Stray et al. 2022a).

Our study also provides examples of how to adapt when existing coordination mechanisms do not work effectively (Strode et al. 2012; Strode 2016; Bick et al. 2018). Both the example of adjusting the product owners' prioritization meeting and the need to pay attention to keeping the team-leader stand-up meeting relevant across teams illustrate this. It is already well established in practice that inter-team meetings should focus on sharing relevant information across teams (for example, the Scrum-of-Scrums meeting in the LeSS framework (Larman and Vodde 2016)). Most, if not all, of the meeting participants in Entur's inter-team coordination meetings were knowledgeable and experienced software engineering practitioners who were aware of this. However, in practice, it is difficult to keep this level of discipline and focus and not bring in other information relevant to one's own or one's team's prioritizations. Similar findings have been reported at the team level, where developers have perceived the daily stand-up meeting as too long and not relevant enough (Stray et al. 2016). Using other mechanisms, such as inter-team retrospectives to adjust and adapt regularly, has been found efficient for re-adjusting coordination practices, both at the team level (Strode et al. 2012) and the inter-team level (Edison et al. 2022).

Sometimes the challenging mechanism cannot easily be replaced or modified. In such cases, another way of adapting can be to improve the surrounding coordination mechanisms instead. At Entur, this was most notable with the product owners, where the large and varied group had such different perceptions on how and when to coordinate that it represented an ongoing challenge to the group. This can be explained partly by the team affiliation and partly by the diversity in the group in terms of work background and personalities (Berntzen et al. 2019), but also that the role was given great autonomy in managing their product area. Some studies recommend that inter-team roles, such as product owners, form teams to strengthen inter-team coordination (Bass 2015; Paasivaara et al. 2018). Other studies point to the tension between a strong team focus and a strong inter-team focus in large-scale agile (Gustavsson et al. 2022), where the need for team autonomy must be balanced with the need for inter-team alignment (Dikert et al. 2016; Bick et al. 2018). At Entur, when product owner coordination was challenged, the short-term solution was to adjust the product owner meetings and use of communication tools, and in the long-term, plan to change or even remove the role as a whole.

5.4 Implications for Theory and Research and Practice

The findings of this study contribute to calls for more research on how coordination mechanisms emerge, change, and terminate (Jarzabkowski et al. 2012; Moe et al. 2018), as well as expanding research on dependency management at the inter-team and large-scale levels (Strode 2016; Berntzen et al. 2021). This study also raises several arenas for future research, including further development of an emerging theoretical framework for coordination mechanisms in large-scale agile.

Software engineering researchers have been encouraged to adopt a more engaged relationship with theories (Sjøberg et al. 2008; Stol and Fitzgerald 2015). In this study, we adopted a process-theoretical lens, seeking to generate knowledge about *how* changes in coordination mechanisms unfold in a large-scale agile setting (Langley 1999; Ralph 2018). We analyzed our data building on the theoretical framework proposed by Jarzabkowski et al. (2012) that explains how coordination mechanisms are created in practice in response to a disruptive event. We extended this work by including a broader set of change events as our findings show that changes in coordination mechanisms occur not only in response to so-called disruptive events but also in response to internal and external change events, large and small. Moreover, our findings show that not only are coordination mechanisms *created* in response to such changes, but they may also be adjusted or removed altogether.

The model presented in Fig. 7 forms the basis for explaining changes in coordination mechanisms in large-scale agile. However, the model needs further theoretical development and empirical investigation (Stol and Fitzgerald 2015), which is an arena for future research. Additionally, future research could build on our findings and do a more thorough mapping of which types of dependencies and coordination mechanisms can be related to which types of change events using existing frameworks (e.g., Strode 2016; Berntzen et al. 2022). Future research can also use insights from this study and our previous work (Berntzen et al. 2022) to study how coordination mechanisms' social, technical, organizational, and physical characteristics change over time in response to changing dependencies. We encourage future work that can contribute to strengthen our findings, for instance by conducting follow-up case studies, or by collecting and analyzing survey data.

5.5 Implications for Practice

Our findings also generate several practical implications that are particularly relevant to large-scale agile programs characterized by high levels of complexity:

- While preparing for all external change events is impossible, having the right coordination mechanisms in place builds resilience to change over time.
- What constitutes an optimal combination of coordination mechanisms will vary over time, as coordination needs are not static.
- When scaling, we recommend using collaboration tools, such as Slack (preferably with communication guidelines), for swift and timely coordination, as face-to-face coordination is not always efficient or even possible.
- Having an overview of the current mix of coordination mechanisms enables companies to sense and respond in a timely and effective manner when coordination needs change.
- Having an explicit and clear focus on continuous improvement of coordination practices (for example, through retrospectives and change-focused workshops) facilitates a flexible way of changing coordination mechanisms in response to change events.

Above all, managers of large-scale agile programs that wish to improve coordination, or manage dependencies effectively in the face of change, should adopt an active and engaged relationship to coordination mechanisms. Agile development welcomes change, recognizing and embracing that it is impossible to avoid change, be it external or internal. At the same time, accurately predicting future coordination needs is nearly impossible. However, it is possible to increase dependency awareness (Bick et al. 2018) through an active and ongoing focus on which coordination mechanisms best address the coordination needs

at any given time. This can be achieved by using existing dependency and coordination mechanisms taxonomies (e.g., Strode 2016; Berntzen et al. 2022) to analyze and gain an overview of the coordination situation at hand. Further, awareness of the past, present, and possibly, future changes can be raised by reflecting on how the organization responds to change and how changes have influenced coordination in the past. Table 4 provides practical guidance for conducting such an activity. The questions are based on the analysis for this study and are meant to serve as inspiration for practitioners who wish to deep-dive into understanding change in coordination in their specific organizations.

5.6 Evaluation of Limitations

This study is a qualitative, interpretive case study. We now review limitations of this study by evaluating its credibility, confirmability, and transferability (Guba 1981). These quality criteria are applicable for assessing the trustworthiness of research and are often used within software engineering (e.g., Russo 2021; Hussain et al. 2022).

Credibility Because this study is interpretive, conducted by humans, and involves human participants, the question of credibility can be raised (Guba 1981; Walsham 2002). We used a range of procedures to make our findings as trustworthy and credible as possible. The ethnographic approach ensured a rich and varied data material (Sharp et al. 2016). The first author conducted the main part of the data collection, which was inevitably subject to her own interpretations and understandings of the case. This is explicitly recognized in interpretive studies; however, it is essential to take measures to safeguard the credibility and trustworthiness of the reporting. To this end, we used an observation protocol and interview guides to sort and systematize our data during collection. We later carefully analyzed the data following established analytical methods (i.e., thematic analysis). Moreover, the three other researchers contributed to the triangulation of the interpretations and the reported findings through ongoing discussions during the fieldwork (authors two and three) and throughout the analytical process (all authors). In longitudinal field studies like ours, the researcher and the participants will, over time, get acquainted with one another, which will influence the research process (Walsham 2002; Crang and Cook 2007). For example, it is impossible to ensure that respondents answer interview questions in an unbiased manner. Here, relying on several data sources and many data points was essential to get as nuanced impressions as possible. Data triangulation was ensured by collecting several data sources. The ongoing and iterative discussions among the authors further contribute to the credibility of our findings. Finally, regular member checks with Entur representatives provide additional trustworthiness (Crang and Cook 2007).

Confirmability The presented results stem from rich process case data (Langley 1999), where the analysis is based on researcher interpretations. We have gone to lengths preventing that we oversimplified our interpretations of the instances and processes described in this study. By following the six phases of thematic analysis (Braun and Clarke 2006, 2012), we have ensured a rigorous analytical process. However, the interpretive research approach makes it difficult for others to repeat the process to confirm our findings, which is not the goal of such approaches (Walsham 2006). Despite this limitation, it is possible to continue this line of study of change, for example, by

Table 4 A practical approach to analyzing change in coordination mechanisms

Suggested participants are team representatives and inter-team coordination roles. The analysis can be run in one setting or in separate steps, depending on the time available and the complexity of the situation

Step	Goal	Questions
Step 1. Understanding coordination mechanisms	Gaining overview of which coordination mechanisms are currently in use. Ask questions to identify mechanisms	• Which meetings to we use to coordinate (between teams)? • Which roles deals primarily with coordination with others? • Which tools and artifacts enable coordination (between teams?) • Which dependencies are managed by these mechanisms? • Are the mechanisms perceived as effective?
Step 2. Understanding past changes	Becoming aware of past change events and continuous changes and how they have influenced coordination. Ask questions to explore and understand	• What changes have we dealt with in the past [insert relevant time period]? Focus on both specific events, as well as changes that have occurred more subtly over time (i.e., continuous changes) • How have these changes influenced how we coordinate? • How long have we used our current coordination mechanisms? When did they appear? Have they changed?
Step 3. Understanding present and future changes	Gaining awareness of ongoing and future changes to potentially be ahead of major changes in coordination needs	• Do we know about any upcoming internal or external change events that will influence our coordination needs? • What changes can we do to existing coordination mechanisms to meet these needs? Will any mechanisms need to be adjusted, removed, or replaced? • Is there a need for other mechanisms? • How will we test any new or adjusted mechanisms and what do we aim to learn?

using existing dependency (Strode 2016) and coordination mechanism (Berntzen et al. 2022) frameworks to analyze the coordination situation and use a visual mapping strategy as we did to map the developments over time. Future research should aim to reproduce and improve the method and analytical procedures in this study, not to directly replicate the findings but to test the confirmability of the research method.

Transferability A third limitation relates to the transferability of our findings. This research was conducted within a single organization, and we have focused much on the context-specificity of our case. Other large-scale agile organizations will have a different external and internal context and are, therefore, likely to have a different mix of coordination mechanisms. While we do not claim that our findings are directly transferable to other organizations, the key implications of our findings are likely transferable to other high-complexity large-scale agile settings. Although the emerging theoretical framework needs further development (Eisenhardt and Graebner 2007; Stol and Fitzgerald 2015), the practical insights from our study provide an empirically based approach to analyzing coordination and change that can aid practitioners in managing dependencies in large-scale agile over time. Even though our case organization did not use any of the large-scale agile frameworks, we believe the findings apply also in companies that have implemented SAFe or any of the other frameworks, because Entur can be considered a 'critical case' for coordination in large-scale agile software development. According to Flyvbjerg (2006, p. 230), the generalization characteristic of critical cases can be summed up as "If it is valid for this case, it is valid for all (or many) cases." In its negative form, the generalization would be, "If it is not valid for this case, then it is not valid for any (or only few) cases." As such, our findings are theoretically generalizable (Crang and Cook 2007) because other large-scale organizations using agile methods are likely to experience similar coordination challenges and use similar agile practices (Edison et al. 2022).

6 Conclusions

In large-scale software development, change is inevitable because of the complexity and long-term duration of such projects or programs (Edison et al. 2022). Agile practices and the use of coordination mechanisms help navigate the complex dependencies associated with software development at scale. Yet, understanding how change impacts coordination appears important to successful large-scale development (Dingsøyr et al. 2018b, 2022). In this study, we addressed the research question, "*What types of organizational changes influence coordination mechanisms in large-scale agile, and how do these mechanisms change over time?*" In previous research, change has been understood either as disruptive events or patterns of events that influence the formation, destabilization discontinuation of coordination mechanisms (Jarzabkowski et al. 2012) or as a continuous process or flow of activities (Langley et al. 2013; Fitzgerald and Stol 2017). In this study, both approaches informed our research question and our analysis.

To investigate changes in coordination mechanisms over time, we analyzed data from 1.5 years of fieldwork using thematic analysis (Braun and Clarke 2006, 2012). We built on the theoretical framework for creating coordination mechanisms in practice proposed by Jarzabkowski et al. (2012) but considered not only disruptive change events and how coordination mechanisms are created, but also how they are adjusted or removed

in response to changes in the internal and external organizational environment. Overall, our findings show that external and internal change events were related to changes in coordination needs and, subsequently, changes in coordination mechanisms. Based on our findings, we presented a model of change in coordination mechanisms over time, which we hope will make way for future research on change in coordination over time in large-scale agile. Further, we find that continuous growth requires a constant focus on improvement, which is also related to the continuous change and adjustment of coordination mechanisms. Our findings illustrate that having the right coordination mechanisms in place can contribute to building resilience to change. We suggest that large-scale agile practitioners should actively and continuously focus on coordination mechanisms. This makes it possible to continuously respond to changes in coordination needs, thereby efficiently managing dependencies.

Finally, our research shows that it is possible to change and adapt in response to challenges brought by scaling without relying on a set of mechanisms provided by commercial scaling frameworks. Rather, our research demonstrates that having an organizational mindset of continuous improvement is key to being truly agile in the face of changing external and internal environments in large-scale software development.

Appendix 1

Observation protocol. Template based on Crang and Cook (2007)

* Field notes to be taken in a handwritten notebook to be always brought around. Fieldnotes to be written up digitally at the end of each day of fieldwork, or as soon as possible after.

*If any photos taken, either add to this document, or give similar name as this file with a brief description of caption in the document.

Title, date

What is this record about? Meeting, observation of daily work, lunch etc.

Description of activity

– who, what, when, where, why, how? Stick to the facts & descriptions, no analysis-related here.

Direct quotes, snippets of conversation, any textual (SMS, email).

Recognize these are a glimpse of a point in time from a particular perspective.

Reflections

How did fieldwork go today?

Did I influence events in any way?

Did anything go wrong?

Can anything be done differently next time?

How do I feel about it?

Emerging questions/analysis

Any emerging analytical questions?
  Any potential lines of inquiry?
  Potential theories that might be useful?

Future action

Anything to follow up?
  Any particular people to talk to?
  Any new information to obtain?

Appendix 2

Interview guide

Background/about the role.

1. What is your role and how long have you had the role?
2. Please tell me about your educational background and previous work experiences
3. How would you describe your role. What are the most important tasks?
4. Has your role changed over time? How?

About the development program

5. Please tell me about the program.
6. Tell me about the use of agile here. Which methods and practices are used? Within teams, across teams?
7. Can you illustrate the team organization and tell me about how you see it? < Draw on a board or blank sheets. >

   a. What's going on here? Why did it happen?
   b. Where are the dependencies? (Now, in the past, in the future)
   c. How do the teams coordinate with each other?
   d. With stakeholders outside the team?

8. Has there been any changes to the program organization or team organization?
9. What challenges do you see now and in the future in the development program?

About coordination

10. In your role, who do you coordinate with on a regular basis?
11. Which coordination practices do you use here? Can you list specific coordination arenas and provide some examples?
12. How is coordination between the teams / developers?
13. What do you think are the most important challenges for coordination across teams? Why?

14. What do you think have been the biggest developments here in relation to coordination across teams?
15. Is there anything else you want to tell that I have not asked about, or do you have any questions?

Follow-up questions for recurring interviews

a. Since last time, has there been any changes to the program organization or team organization?
b. What challenges do you see now and in the future in the development program

Data Availability The data material is unavailable due to non-disclosure agreements.

Code Availability N/A.

Declarations

Ethics approval Study reported to the Norwegian Centre for Research Data.

Conflicts of interests/Competing interests The authors report no conflicts of interest.

References

Aldave A, Vara JM, Granada D, Marcos E (2019) Leveraging creativity in requirements elicitation within agile software development: a systematic literature review. J Syst Softw 110396. https://doi.org/10.1016/j.jss.2019.110396

Allison I, Merali Y (2007) Software process improvement as emergent change: A structurational analysis. Inf Softw Technol 49:668–681

Bass JM (2015) How product owner teams scale agile methods to large distributed enterprises. Empir Softw Eng 20:1525–1557

Batra D, Xia W, VanderMeer DE, Dutta K (2010) Balancing agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry. CAIS 27:21

Berntzen M, Moe NB, Stray V (2019) The Product Owner in Large-Scale Agile: An Empirical Study Through the Lens of Relational Coordination Theory. In: Kruchten P, Fraser S, Coallier F (eds) Agile

Processes in Software Engineering and Extreme Programming. Springer International Publishing, Cham, pp 121–136

Berntzen M, Stray V, Moe NB (2021) Coordination Strategies: Managing Inter-team Coordination Challenges in Large-Scale Agile. In: Gregory P, Lassenius C, Wang X, Kruchten P (eds) Agile Processes in Software Engineering and Extreme Programming. Springer International Publishing, Cham, pp 140–156

Berntzen M, Hoda R, Moe NB, Stray V (2022) A taxonomy of inter-team coordination mechanisms in large-scale agile. IEEE Trans Software Eng 49:699–718. https://doi.org/10.1109/TSE.2022.3160873

Bick S, Spohrer K, Hoda R et al (2018) Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. IEEE Trans Software Eng 44:932–950

Braun V, Clarke V (2006) Using thematic analysis in psychology. Qual Res Psychol 3:77–101

Braun V, Clarke V (2012) Thematic analysis. APA handbook of research methods in psychology, Vol 2: Research designs: Quantitative, qualitative, neuropsychological, and biological. American Psychological Association, Washington, DC, pp 57–71

Carroll N, Conboy K, Wang X (2023) From Transformation to Normalisation: An Exploratory Study of a Large-Scale Agile Transformation. J Inf Technol 02683962231164428. https://doi.org/10.1177/02683962231164428

Castañer X, Oliveira N (2020) Collaboration, coordination, and cooperation among organizations: Establishing the distinctive meanings of these terms through a systematic literature review. J Manag 46:965–1001

Cataldo M, Herbsleb JD (2012) Coordination breakdowns and their impact on development productivity and software failures. IEEE Trans Software Eng 39:343–360

Crang M, Cook I (2007) Doing ethnographies. Sage

Dikert K, Paasivaara M, Lassenius C (2016) Challenges and success factors for large-scale agile transformations: A systematic literature review. J Syst Softw 119:87–108

Dingsøyr T, Nerur S, Balijepally V, Moe NB (2012) A decade of agile methodologies: Towards explaining agile software development. J Syst Softw 85:1213–1221. https://doi.org/10.1016/j.jss.2012.02.033

Dingsøyr T, Fægri TE, Itkonen J (2014) What is large in large-scale? A taxonomy of scale for agile software development. International Conference on Product-Focused Software Process Improvement. Springer, Cham, pp 273–276

Dingsøyr T, Moe NB, Fægri TE, Seim EA (2018a) Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. Empir Softw Eng 1–31

Dingsøyr T, Moe NB, Seim EA (2018b) Coordinating Knowledge Work in Multi-Team Programs: Findings from a Large-Scale Agile Development Program. Proj Manag J 49:64–77

Dingsøyr T, Bjørnson FO, Schrof J, Sporsem T (2022) A longitudinal explanatory case study of coordination in a very large development programme: the impact of transitioning from a first- to a second-generation large-scale agile development method. Empir Softw Eng 28:1. https://doi.org/10.1007/s10664-022-10230-6

Edison H, Wang X, Conboy K (2022) Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review. IEEE Trans Software Eng 48:2709–2731. https://doi.org/10.1109/TSE.2021.3069039

Eisenhardt KM, Graebner ME (2007) Theory building from cases: Opportunities and challenges. Acad Manag J 50:25–32

Espinosa JA, Slaughter SA, Kraut RE, Herbsleb JD (2007) Team knowledge and coordination in geographically distributed software development. J Manag Inf Syst 24:135–169

Fitzgerald B, Stol K-J (2017) Continuous software engineering: A roadmap and agenda. J Syst Softw 123:176–189

Flyvbjerg B (2006) Five misunderstandings about case-study research. Qual Inq 12:219–245

Fowler M, Highsmith J (2001) The Agile Manifesto. http://agilemanifesto.org/. Accessed 23 Mar 2023

Fuchs C, Hess T (2018) Becoming agile in the digital transformation: The process of a large-scale agile transformation. In: Proceedings of the 39th International Conference on Information Systems (ICIS 2018)

Gregor S (2006) The Nature of Theory in Information Systems. MIS Q 30:611–642. https://doi.org/10.2307/25148742

Guba EG (1981) Criteria for assessing the trustworthiness of naturalistic inquiries. Ectj 29:75–91

Gustavsson T, Berntzen M, Stray V (2022) Changes to team autonomy in large-scale software development: a multiple case study of Scaled Agile Framework (SAFe) implementations. Int J Inf Syst Proj Manag 10:29–46

Gustavsson T (2019) Dynamics of Inter-Team Coordination Routines in Large-Scale Agile Software Development. In: Proceedings of the 27th European Conference on Information Systems (ECIS). Uppsala, pp 1–16

Hoda R, Salleh N, Grundy J (2018) The rise and evolution of agile software development. IEEE Softw 35:58–63

Hoda R, Noble J (2017) Becoming agile: a grounded theory of agile transitions in practice. IEEE Press, pp 141–151

Hussain W, Perera H, Whittle J et al (2022) Human Values in Software Engineering: Contrasting Case Studies of Practice. IEEE Trans Software Eng 48:1818–1833. https://doi.org/10.1109/TSE.2020.3038802

Jarzabkowski PA, Lê JK, Feldman MS (2012) Toward a Theory of Coordinating: Creating Coordinating Mechanisms in Practice. Organ Sci 23:907–927

Kalenda M, Hyna P, Rossi B (2018) Scaling agile in large organizations: Practices, challenges, and success factors. J Softw Evol Process 30:e1954

Kwan I, Schroter A, Damian D (2011) Does socio-technical congruence have an effect on software build success? a study of coordination in a software project. IEEE Trans Software Eng 37:307–324

Langley A (1999) Strategies for theorizing from process data. Acad Manag Rev 24:691–710

Langley A, Truax J (1994) A process study of new technology adoption in smaller manufacturing firms. J Manage Stud 31:619–652

Langley A, Smallman C, Tsoukas H, Van de Ven AH (2013) Process studies of change in organization and management: Unveiling temporality, activity, and flow. Acad Manag J 56:1–13

Larman C, Vodde B (2016) Large-scale scrum: More with LeSS. Addison-Wesley Professional

Lin B, Zagalsky A, Storey M-A, Serebrenik A (2016) Why Developers Are Slacking Off: Understanding How Software Teams Use Slack. In: Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion. ACM, New York, NY, USA, pp 333–336

Madampe K, Hoda R, Grundy J (2022) The Emotional Roller Coaster of Responding to Requirements Changes in Software Engineering. IEEE Trans Softw Eng 1–1. https://doi.org/10.1109/TSE.2022.3172925

Malone TW, Crowston K (1994) The interdisciplinary study of coordination. ACM Comput Surv (CSUR) 26:87–119

March JG, Simon HA (1966) Organizations. John Wiley & Sons, New York

Mintzberg H (1989) Mintzberg on management: Inside our strange world of organizations. Simon and Schuster, New York

Moe NB, Dingsøyr T, Rolland K (2018) To schedule or not to schedule? An investigation of meetings as an inter-team coordination mechanism in large-scale agile software development. Int J Inf Syst Proj Manag 6:45–59

Moe NB, Šmite D, Paasivaara M, Lassenius C (2021) Finding the sweet spot for organizational control and team autonomy in large-scale agile software development. Empir Softw Eng 26:101. https://doi.org/10.1007/s10664-021-09967-3

Munir H, Wnuk K, Runeson P (2016) Open innovation in software engineering: a systematic mapping study. Empir Softw Eng 21:684–723. https://doi.org/10.1007/s10664-015-9380-x

Murray E, Treweek S, Pope C et al (2010) Normalisation process theory: a framework for developing, evaluating and implementing complex interventions. BMC Med 8:1–11

Niven PR, Lamorte B (2016) Objectives and key results: driving focus, alignment, and engagement with OKRs. John Wiley & Sons, Hoboken, New Jersey

Okhuysen GA, Bechky BA (2009) 10 coordination in organizations: An integrative perspective. Acad Manag Ann 3:463–502

Paasivaara M, Behm B, Lassenius C, Hallikainen M (2018) Large-scale agile transformation at Ericsson: a case study. Empir Softw Eng 1–47

Pettigrew AM (1990) Longitudinal field research on change: Theory and practice. Organ Sci 1:267–292

Ralph P (2018) Toward methodological guidelines for process theories and taxonomies in software engineering. IEEE Trans Software Eng 45:712–735

Runeson P, Höst M (2008) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14:131. https://doi.org/10.1007/s10664-008-9102-8

Russo D (2021) The Agile Success Model: A Mixed-methods Study of a Large-scale Agile Transformation. ACM Trans Softw Eng Methodol (TOSEM) 30:1–46

Sharp H, Dittrich Y, de Souza CRB (2016) The Role of Ethnographic Studies in Empirical Software Engineering. IEEE Trans Software Eng 42:786–804

Sjøberg DI, Dybå T, Anda BC, Hannay JE (2008) Building theories in software engineering. In: Shull F, Singer J, Sjøberg DIK (eds) Guide to advanced empirical software engineering. Springer, London, pp 312–336

Spiegler SV, Heinecke C, Wagner S (2021) An empirical study on changing leadership in agile teams. Empir Softw Eng 26:1–35

Stol K-J, Fitzgerald B (2015) Theory-oriented software engineering. Sci Comput Program 101:79–98

Stray V, Sjøberg DI, Dybå T (2016) The daily stand-up meeting: a grounded theory study. J Syst Softw 114:101–124. https://doi.org/10.1016/j.jss.2016.01.004

Stray V, Moe NB, Strode D, Mæhlum E (2022a) Coordination Value in Agile Software Development: A Multiple Case Study of Coordination Mechanisms Managing Dependencies. In: Proceedings of the 15th

International Conference on Cooperative and Human Aspects of Software Engineering. Association for Computing Machinery, New York, NY, USA, pp 11–20

Stray V, Moe NB, Vedal H, Berntzen M (2022b) Using Objectives and Key Results (OKRs) and Slack: A Case Study of Coordination in Large-Scale Distributed Agile. https://doi.org/10.36227/techrxiv.16892161.v1

Stray V, Moe NB (2020) Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack. J Syst Softw 170:110717. https://doi.org/10.1016/j.jss.2020.110717

Strode DE (2016) A dependency taxonomy for agile software development projects. Inf Syst Front 18:23–46

Strode DE, Huff SL, Hope B, Link S (2012) Coordination in co-located agile software development projects. J Syst Softw 85:1222–1238

Thompson JD (1967) Organizations in action: Social science bases of administrative theory. McGraw-Hill, New York

Uludağ Ö, Philipp P, Putta A, et al (2022) Revealing the state of the art of large-scale agile development research: A systematic mapping study. J Syst Softw 111473

Van de Ven AH, Delbecq AL, Koenig Jr R (1976) Determinants of coordination modes within organizations. American sociological review 322–338

Van de Ven AH, Poole MS (2005) Alternative approaches for studying organizational change. Organ Stud 26:1377–1404

Walsham G (2006) Doing interpretive research. Eur J Inf Syst 15:320–330. https://doi.org/10.1057/palgrave.ejis.3000589

Walsham G (2002) Interpretive Case Study in IS Research: Nature and Method. In: Myers MD, Avison D (eds) Qualitative Research in Information Systems. Sage Publications, London

Wohlin C, Aurum A (2015) Towards a decision-making structure for selecting a research design in empirical software engineering. Empir Softw Eng 20:1427–1455

Marthe Berntzen is a Ph.D. candidate in Software Engineering at the department of Informatics, University of Oslo. She holds an M.Sc. degree from BI Norwegian Business School and has four years of industry experience. Marthe's Ph.D. research centers around inter-team coordination in large-scale agile software development. Her research focus on agile methods and practices, teamwork, leadership, and coordination in large-scale and distributed settings. She presents her work in journals and conferences within software engineering, information systems and management.

Viktoria Stray is an Associate Professor at the University of Oslo's Department of Informatics. She also holds a senior research position at SINTEF. Her research interests include agile methods, coordination, global software engineering, software testing, and large-scale development. The focus of her research is to improve the productivity of developers and testers and increase the success of software projects. Stray has a Ph.D. in Software Engineering and has worked several years in the industry participating in some of the largest software development projects in Norway.

Nils Brede Moe is a chief scientist at SINTEF. He works with software process improvement, intellectual capital, autonomous teams, and agile and global software development. He has led several nationally funded software engineering research projects covering organizational, sociotechnical, and global/distributed aspects. Moe received a Dr.Philos. in Computer Science from the Norwegian University of Science and Technology and holds an adjunct position at the Blekinge Institute of Technology in Sweden.

Rashina Hoda is an Associate Professor of Software Engineering at Monash University, Melbourne. Rashina specializes in human-centered software engineering, including agile transformations, self-organizing teams, agile project management, and large-scale agile, and has introduced socio-technical grounded theory to software engineering. She serves as an Associate Editor of the IEEE Transactions on Software Engineering and as co-chair for ICSE-SEIS 2023, and previously, on the advisory board of IEEE Software and as PC co-Chair for CHASE2021. For more: www.rashina.com.

Authors and Affiliations

Marthe Berntzen[1] [iD] **· Viktoria Stray[1,2] · Nils Brede Moe[2] · Rashina Hoda[3]**

✉ Marthe Berntzen
   marthenb@ifi.uio.no

   Viktoria Stray
   stray@ifi.uio.no

   Nils Brede Moe
   Nils.B.Moe@sintef.no

   Rashina Hoda
   Rashina.hoda@monash.edu

[1]   Department of Informatics, The University of Oslo, Gaustadalléen, 23B, 0373 Oslo, Norway

[2]   SINTEF Digital, Strindveien 4, 7645 Trondheim, Norway

[3]   Faculty of Information Technology, Monash University, Clayton, Melbourne, VIC 3800, Australia