View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by AIS Electronic Library (AISeL)

Practices for vertical and horizontal coordination in the Scaled Agile Framework

Tomas Gustavsson *tomas.gustavsson@kau.se*
Karlstad University
Karlstad, Sweden

Abstract

Scrum and eXtreme programming, the first agile development frameworks, were designed with very few advice on coordination for work in larger scale were several teams cooperate toward a common goal. This lead to both wrong assumptions regarding the usefulness of agile ways of working in larger organizations as well as much individual tailoring with coordination practices in organizations. Now, the Scaled Agile Framework is gaining much attention in software development. This study contains an analysis of inter-team coordination practices prescribed in the framework and how they have been implemented in three organizations. Previous research on coordination has been criticized for having a static view on coordination with not enough focus on how to manage emerging dependency issues. The result of this study shows that the Scaled Agile Framework have four practices that cover both planning and emerging issues and three practices solely aimed at managing these emerging dependency issues.

Keywords: Agile software development, Large-scale, Scaled Agile Framework, Inter-team coordination.

1. Introduction

The first agile development frameworks, eXtreme Programming (XP) and Scrum, have become the norm for software product development ways of working in the IT industry today [1]. Since a fundamental agile value is to form self-organizing, cross-functional teams, coordination issues between individuals in the team are solved by following agile principles and practices. But if the organization grows and several teams are organized, coordination is suddenly required between the teams, sometimes called inter-team coordination. These early frameworks were designed without much consideration to scaling problems. Because of this lack of support for managing coordination issues in larger settings in the first frameworks, many diverse solutions have been created to handle different aspects of coordination problems. Recently, however, several agile frameworks designed for large-scale settings have emerged of which the most commonly adopted is the Scaled Agile Framework (SAFe) according to an industry study performed by VersionOne Inc. [1].

Coordination problems are caused by interdependencies between various elements of a situation which constrain how particular tasks are performed [2]. Malone and Crowston [2] therefore define coordination as managing dependencies. Solutions to coordination have been an important research topic in organizational and management studies for a long time. March and Simon [3] divided the solutions into three areas, or levels: Coordination 1) by standardization, 2) by planning and 3) by feedback or “mutual adjustment” as it was refrained by Thompson [4].

Thompson [4] made a distinction between horizontal and vertical coordination. The horizontal coordination is between employees or teams on the lowest level of the organizational hierarchy and, according to the author, the key to this area of coordination is mutual adjustments. Vertical coordination is managers or management teams deciding from the top down. According to the Thompson [4], all coordination issues that cannot be solved by mutual adjustments should be dealt with by a management team.

Besides the above-mentioned authors, several researchers have presented additional coordination practices such as Van de Vehn et al. [5] and Mintzberg [6] but critique on this earlier work, e. g. Jarzabkowski et al. [7] and Okhuysen & Bechky [8], is the static view of coordination where detailed solutions to emerging coordination issues are mostly overlooked and rarely explained as detailed practices. These practices have been difficult for scholars to measure and therefore remained largely unexamined. This is unfortunate because unplanned contingencies and the responses to them are important for the efficiency of workflow and processes in organizations [8].

SAFe have received criticism for being too rigid and focused on too much planning in advance, i.e. too much focus on the vertical, pre-planning, coordination practices. The question is if SAFe offers solutions for both static as well as emerging coordination needs or if emerging coordination is still being overlooked in a modern software development framework? The research question for this article is: *Which practices, for static and emerging coordination issues, are used in the Scaled Agile Framework?*

Since an important agile value is to inspect and adopt practices being used, we cannot look at the framework alone, but rather how it is being used. Therefore, three case studies are investigated to pinpoint the coordination practices being used in these organizations who have implemented SAFe.

Okhuysen and Bechky [8] summarized the practices used for both static and emerging coordination needs into five different mechanism types: 1) Plans and rules, 2) Objects and representations, 3) Roles, 4) Routines and 5) Proximity (or Location). These five types will be used as the analytic lens for investigating the usage of coordination practices with SAFe.

2. Five types of coordination mechanisms

Within the five types of coordination mechanisms described in the article by Okhuysen and Bechker [8], different practices for every single type are presented. In this section, the different practices within each coordination mechanism type are explained and discussed. An overview of practices is displayed in Fig. 1.

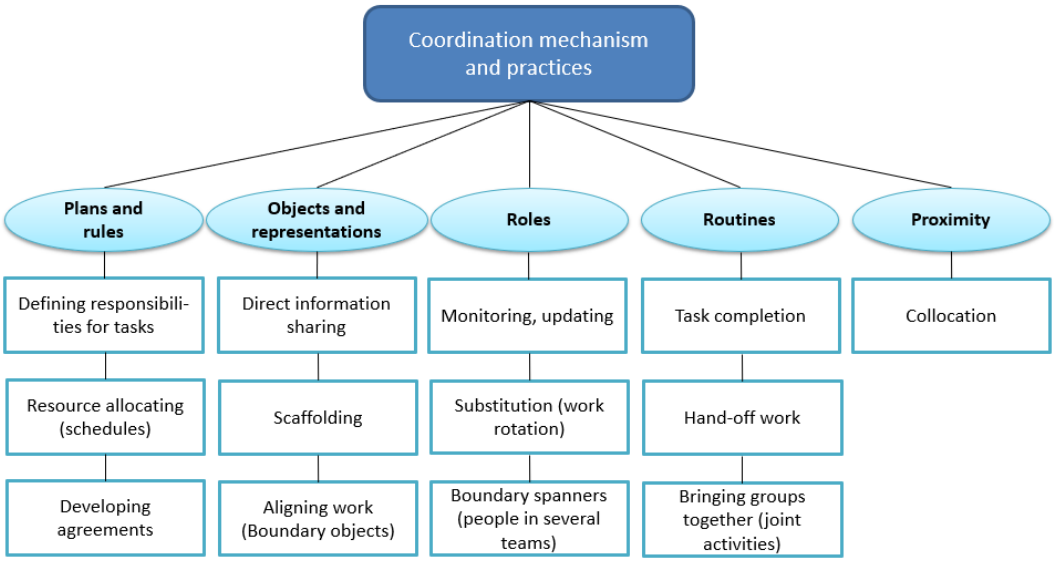

Fig. 1. Practices for coordination categorized in five mechanism types.

2.1. Plans and rules

Conceptualized as purposeful elements of formal organizations, plans and rules are essential to organizing and coordination [3]. Three different practices are put forth; the first is called

"Defining responsibility for tasks" which implies explaining the actions that different parties have to take to fulfill a specific common goal.

The second practice is called "Resource allocation" which practically often involves the use of schedules, placing people and activities on a temporal map. The third practice is called "Developing agreement" where the final product is a working agreement of how the mission or project will unfold. It serves as an orienting device for the people involved.

2.2. Objects and representations

Objects and representations coordinate by providing information and offer a common referent around which people interact, align their work, and create shared meaning. Four different practices are described in this type of which the first three will be used for analytical purposes. Reasons are explained below. First, "Direct information sharing" involves the use of data spreadsheets, large-size computer screens or physical billboards to share information that is necessary for the work to progress across different teams. Second, "Scaffolding" is to visualize the structure and progress of activities as a reminder of what work still needs to be done, who needs information in order to complete their work and who is supposed to deliver that information. Third, "Acknowledging and aligning work" is typically accomplished by using boundary objects [9]. A boundary object is "a sort of arrangement that allows different groups to work together without consensus" ([9], p 602).

The fourth practice is not a practice in itself but rather a desirable outcome of the other practices. Okhuysen and Bechker [8] calls the practice "Creating a common perspective". There are two examples provided in the article where the first is patient protocols at hospitals. This is rather an example of the first practice, "Direct information sharing" since the protocol can be viewed as a data spreadsheet of information. The next example is from computer chip design where maps are used to show interdependencies. This example is also used as the example for the second practice, "Scaffolding". So, since it is not possible to identify the difference between the suggested fourth practice from the previous three, only the first three practices will be used for analysis.

2.3. Roles

Roles represent expectations associated with social positions and can facilitate continuity of behavior over time. Understanding the relationship between roles, the role structure, helps people acquire a general sense of who does what in the work process [8]. Three practices are described and the first one is called "Monitoring and updating" which is one of the traditional functions of a formal hierarchy.

The second practice is called "Substitution" meaning that when individuals understand the tasks linked to each other's roles, they can substitute for one another in task execution. A practical example is when individuals rotate between different teams or departments. The name of the third practice is "Creating a common perspective" which is unfortunate since the same name is used as one of the practices in the "objects and representation" mechanism type. Okhuysen and Bechker [8] describes this practice with the example of "boundary spanning", which means a role appointed to transfer information between different competencies, translating information to create a common perspective to help teams coordinate. Therefore, the term "Boundary spanning" will be used in the analysis.

2.4. Routines

Routines have historically been an important part of conceptions of organizations and coordination, just like plans and rules [3]. As with "objects and representations, four different practices are described in this type of which only the first three will be used for analytical purposes. The first practice is called "Task completion" which is where a sequence of activities is established and routines enact a way for interdependent parties to observe

progress on the task. An example could be the routine of a university course with known activities before, during and after the course.

"Hand-off work" is the second practice, meaning a routine to move activities between people or teams when one group has completed their work. The third practice is called "Bringing groups together" which means that interdependent parties jointly work on an activity. For the third time, the authors [8] use the same name of their proposed fourth practice within the routines type: "Creating a common perspective". As with the mechanism type "Objects and representations", it does not seem to be a practice but rather an effect of the other practices described within the type.

2.5. Proximity (location)

The final mechanism type that supports coordination within organizations is the physical proximity to one another. The authors [8] describe four suggested practices who seem to be effects of practices, rather than practices themselves. These are "Visibility: [for easier] monitoring and updating", "Familiarity: [to] develop trust", "Familiarity: [for] anticipating and responding" and "Familiarity: [for] increased stored knowledge". Words used within brackets are added for a better understanding of the expressed effects. Instead, the suggested naming of one practice: "Collocate", will be used since that is the actual action described within the proximity coordination mechanism type.

3. Agile frameworks

In this section, the earlier frameworks for agile ways of working will be described as well as a short description of what SAFe entails.

3.1. Scrum and eXtreme Programming

The values and principles stated in the Agile Manifesto [10] mainly describe teamwork for a single team. This has sometimes led to the false assumptions that agile ways of working are suitable only for smaller projects and organizations (e. g. [11]). Authors of the first popular agile frameworks, XP [12], and Scrum [13], only touched the challenges of scaling the frameworks but did not get into any detail. The originator of XP was even skeptical of the idea of scaling, saying "you probably couldn't run an XP project with a hundred programmers" ([12], p 157), and only put forth the technical view of solving scaling problems by proposing a single integration machine to deal with several code streams.

To manage emerging interdependence problems between several teams, Schwaber [14] proposed the coordination mechanism Scrum of Scrums (SoS) in the Scrum framework. The process was originally described in a case study report by Sutherland [15] who introduced the Scrum framework at the company IDX Systems in 1996 where hundreds of developers worked on dozens of products. SoS is described as a weekly meeting between all team leaders, called Scrum Masters (SM), in a product line to discuss and solve dependency issues between teams. SoS, adopted in this form, has been widely criticized (e. g. Paasivaara et al. [16]). By allowing only SMs to the SoS, there is a risk for talking about solutions rather than solving them. Practitioners have since stressed that the SoS should be a place for solving coordination issues, not a meeting for managers, and put forth the need for allowing any team member with a coordination issue to participate in the SoS [16].

Another criticism of the SoS has been that a weekly meeting does not solve problems between teams fast enough. Since the Scrum teams meet on a daily basis to sort out problems, called the "Daily Scrum", having to wait for a whole week to sort out problems between teams is much too slow. Practitioners have therefore argued for more frequent SoS meetings; 2-3 times a week [17] or even daily [14].

Sutherland [15] does not give any details on how the original SoS meetings were conducted but Schwaber [14] suggests that a SoS should follow the same rules as the Daily Scrum, by answering the three questions; 1) What have we done since the last meeting? 2)

What are we doing until the next meeting? 3) Are there impediments to get there? He also suggests using a fixed meeting time, time-boxed to last a maximum of 15 minutes. Cohn [17], however, suggests allocating 30 to 60 minutes for SoS meetings to let people be able to discuss and solve any big problems that might arise at the meeting, while the right people are present.

The SoS is an agile way of dealing with inter-team, emerging, horizontal coordination issues in a systematic process based manner. Using the notation from Okhuyser and Bechker [8], the SoS is within the coordination mechanism type "Routine" to work as a practice for "Hand-off work".

For the coordination mechanism type "Proximity", Schwaber [14] also stressed the importance to collocate and put forth the Agile Manifesto [10] stating the importance of face-to-face communication. In the Scrum framework, no other coordination practices are suggested.

3.2. The Scaled Agile Framework (SAFe)

An agile software development setting where the organization has scaled up is a multi-team system (MTS). An MTS is defined as a collective of two or more inter-dependent teams organized in order to achieve a shared superordinate goal [18]. In organizations embracing agile, an MTS is normally referred to as a large-scale agile development setting.

One important value of agile ways of working stated in the Agile Manifesto [10], is to create self-organizing teams, allowing much autonomy to the single team to decide how to be able to reach the desired goals on their own. Since coordination entails decisions to make the most benefit out of a number of cooperating teams, this implies a risk of less autonomy to every single team. This important value, to allow autonomy while still managing coordination, has strongly affected the choice of prescribed and suggested coordination mechanisms in the agile frameworks [19].

4. Method

In this section, the research design, case organizations and data collection methods are presented.

4.1. Research design

Within the five types of coordination mechanisms described in the article by Okhuysen and Bechker [8], different practices for every single type are presented. In this paper, these five mechanism types and their practices will be used to analyze which ones that are used both in the description of SAFe but also in practice, based on three different case studies. Regarding the description of SAFe, data from the website [20] will be used describing version 4.5 of the framework. Also, the practices prescribed in SAFe will be analyzed regarding if they are aimed at solving coordination issues by planning, the static approach, or to manage emerging dependency issues.

4.2. Case organizations and data collection

Observations and interviews have been conducted in three different organizations to gather data for this paper. The true names of the organizations have been anonymized but will be referred to as Auto, Gov, and Bank. These three organizations have used agile ways of working for four to six years, with self-organized autonomous teams working side by side. But all three of them realized the need to adopt practices for improved coordination and started implementing SAFe during the beginning of 2017. Auto was first, starting in January while Gov and Bank started in March and April, respectively.

Auto develops software as well as hardware and the observed department was organized in twenty cross-functional teams, divided into three different value streams or Agile Release Trains (ART) to use SAFe terminology [20].

The Gov-project is a SAFe implementation that started as a pilot project in a large Swedish Government Agency where large-scale agile processes were implemented with the aim of finding best practices to be used for the whole organization. A one-year review of the pilot projects was conducted to provide guidelines for new roll-outs of further large-scale agile projects in the organization. Gov consists of seven teams working in one ART.

Bank is a department in one of the major business banks in Sweden consisting of seven teams that had worked together for less than a year and decided to implement large-scale agile processes one month prior to my first visit.

Every organization has been visited with an interval of nine to twelve weeks, two to four days at a time during the two-day product increment (PI) planning workshop and some additional days for pre-planning or retrospective days, called Inspect & Adapt using SAFe terminology. During these visits, a total of 18 interviews have also been conducted. The number of visits and the total number of hours of observation is presented in Table 1.

Table 1. The number of visits and hours spent on observations.

Organization	Number of on-site visits	Hours of observation
Auto	5	160
Gov	5	89
Bank	6	70
Total:	16	319

5. Results

In this section, results are presented divided into five subsections based on the five coordination mechanism types presented by Okhuysen and Bechker [8]. Every subsection starts with a brief explanation of the prescribed SAFe practices relevant for the mechanism and ends with observations in the three case organizations, presenting the similarities and differences in how the framework has been tailored.

5.1. Plans and rules in SAFe

The SAFe framework prescribes the use of a release plan called Product Increment (PI) spanning from eight to twelve weeks, consisting of four timeboxed iterations where the teams demonstrate results at the end of every iteration to get feedback and thereby understanding the need for further coordination. At the end of a release, a two-day PI planning workshop is conducted to create a new release plan for the following PI. The PI planning is, among practitioners, what SAFe is most known for and is much debated whether it is a step backward towards the traditional waterfall model approach or if it is a step forward toward better predictability in agile projects. The PI planning is, therefore, an example of the static view, intended for the vertical planning of coordination efforts. Critics claim that eight to twelve weeks is too long as a time horizon to make detailed plans while supporters claim to perceive a better understanding of the big picture [16]. The workshop has a standardized detailed schedule, see Fig. 2, created in order to involve all team members as well as stakeholders in creating and committing to a detailed plan.

Fig. 2. The schedule for the two-day PI planning workshop.

In SAFe, the three practices for plans and rules, defining responsibility for tasks, resource allocation and developing agreements [8] are all covered by using the PI plan.

The three studied organizations only conducted minor tailoring in the way PI planning was performed. Auto and Bank condensed the workshop into one and a half day, instead of two days, but with the same agenda items as in Fig. 2. Gov used the exact proposed agenda presented in Fig. 2.

5.2. Objects and representations in SAFe

A boundary object is “a sort of arrangement that allow different groups to work together without consensus” ([9], p 602). SAFe use the objects called backlogs for visualization of upcoming work. The portfolio backlog, program backlog, and team backlog are three levels in a hierarchy for defining and detailing requirements. The portfolio and program backlogs are, according to the definition, boundary objects. Items in the list are described in different detail depending on how prioritized they are. The items are often expressed as business objectives (sometimes called epics) or more or less vaguely described wanted features of the product. Most of them are described by business people using their own terminology in such detail that it is possible to for team members in the development teams to implement the desired functionality in the IT system. These two backlogs cover the practice "acknowledging and aligning work" (Fig. 5) and are used to plan coordination in advance, vertically.

As items are being prioritized, developers and business people refine, split and add more details to the items. Therefore, items at the top of the program backlog are written and understood by both parties and can be divided into small work packages (sometimes called stories) and inserted into team backlogs, which are one list per team of committed work for the upcoming iteration of two to four weeks. The team backlog is thereby *not* a boundary object since it is a clearly defined and prioritized list of work fully understood by the development team.

Since there might be dependencies between team backlogs, the SAFe framework prescribes a visual representation called a program board. A program board shows the different teams involved in the upcoming release on the Y-axis and the team backlogs along the X-axis. Dependencies between features and stories are shown by using red strings as shown in Fig. 4. To handle emerging coordination issues, the program board should be updated when new dependency issues arise and could, therefore, claim to be aimed at both planning and solving emerging dependency issues.

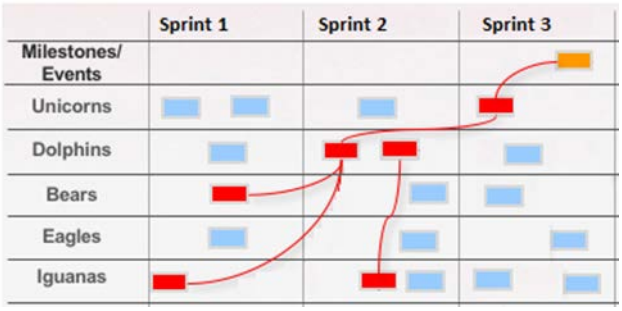

Fig. 4. Example of program board.

The coordination practice scaffolding [8] is therefore covered by the use of the program board.

There is no specific practice in SAFe that addresses the coordination practice called direct information sharing. Although the described backlogs contain information about requirements and proposed solutions on how to convert these features into software functions, they do not specifically contain information intended for coordination.

In the studied organizations, the use of backlogs and program board are according to the prescribed framework. However, the prescribed use of the program board to be updated when new information arises seems to work only in theory. Neither Auto nor Gov updated information between the PI planning sessions. During the first five months, Bank also only made updates during PI planning but increasing problems with emerging dependency issues caused Bank to start with a new practice: the mid-sprint review. Halfway into an iteration, all SMs in the Bank ART meet during half an hour in front of the Program board to add new strings for new dependencies, mark implemented features as done and decide on responsibilities for newly detected dependencies. This practice is not suggested in SAFe but was invented at Bank to make the program board useful even for emerging coordination issues.

5.3. Roles in SAFe

Since self-organizing is an important value in agile ways of working, according to the manifesto [10], and the basic principle is that the teams should work out coordination issues themselves, without adding additional roles besides the SM and the Product Owner (PO) who acts as a project sponsor and business analyst, prioritizing requirements for the team. However, if the organization fails to solve coordination issues fast enough on its own, additional roles might be useful [14]. Regarding the SM, SAFe puts forth that horizontal inter-team coordination, managing emerging dependency issues, is a responsibility for the SM but cannot be delegated entirely to the SM since the team members should share the responsibility as well.

SAFe prescribes one PO per team. In order to coordinate these POs, an additional role is prescribed named Release Train Engineer (RTE). The RTE has the overall responsibility for an upcoming release as well as coordinating through POs and SMs. The RTE could therefore also be called a “Chief PO” [21] or “Chief SM” [22] responsible for both of the practices “Monitoring and updating” and “Boundary spanning” [8]. By using the portfolio and program backlogs, the RTE and the POs are responsible for planning the coordination between teams but also to solve emerging dependency issues raised by the teams.

To cover the practice “Substitution”, SAFe prescribes the use of a routine called Communities of Practice which is further described in Section 5.4.

The roles of RTE, PO, and SMs are implemented in all of the three case organizations with the same responsibilities as described in the SAFe framework.

5.4. Routines in SAFe

SAFe recommends using two routines for coordination: Communities of Practice (CoP) and Scrum of scrums (SoS). Regarding Communities of Practice (CoP), the concept is described in the SAFe framework as organized groups of people who have a common interest in a specific technical or business domain who collaborate regularly to share information, improve their skills, and actively work to advance the general knowledge of the domain [22]. Several examples of organizing CoPs are also described such as role-based communities (CoPs for PO:s, architects or testers) or topic-based communities (CoPs for java development or automated testing). Although CoPs manage the practice "substitution", described by Okhuysen and Bechker [8] as a practice belonging to the type "Roles" as put forth in Section 5.3, it is clearly an important routine and also described as a routine in SAFe. The CoPs are supposed to both plan for coordination, such as prescribing architectural changes in advance, but also manage emerging dependency issues raised within the specific domain of the CoP.

Regarding the routine SoS, it is suggested to be performed two to three times per week and divided into two parts: 1) SoS for everybody attending and 2) Meet-After/Problem solving for representatives that need to sort out dependency issues not solved during the first part of the meeting. As described in Section 3.1, the SoS is a practice specifically designed for management of emerging dependency issues.

SAFe prescribes that the SM should be the team representative in the SoS. SAFe suggest adaptions of the meeting agenda to focus on what issues that are of importance to the other teams regarding what has been done, what will be done and obstacles to reach the current goal of the iteration. The SoS is covering the practice "Hand-off work" [8].

Besides SoS and CoP, SAFe put much effort into emphasizing the importance of routines for cadence and synchronization. Therefore, the routines for development is prescribed to always consist of two weeks of work for all iterations, ending in integration and demonstration to synchronize results between teams. These demonstrations are called "system demo" and covers the practice called "task completion".

Also, above team level, release management is coordinated through weekly meetings with senior management but no details are described for this kind of meeting. Besides that, no other routines are prescribed or suggested for coordination in SAFe.

The framework does not prescribe or suggest practices for "bringing groups together" regarding the actual work. All though the SAFe way of teams planning together in the PI planning workshop, there are no routines for actually working together during iterations.

The case organizations all use the SoS routine but in slightly different ways. Auto and Gov have a weekly SoS meeting for half an hour and Bank has a daily SoS meeting for 15 minutes. Auto and Bank have implemented topic-based CoPs but Gov has not implemented CoPs at all mainly because management at this point in time consider attending CoP-meetings as waste, time that could have been used for productive work.

5.5. Proximity in SAFe

Regarding proximity, SAFe prescribes that the best solution is to have all teams collocated thereby decreasing the risk of coordination problems, making it easier for people to meet and solve emerging dependency issues. All practices in SAFe are described as being performed in an environment where people can meet face-to-face. SAFe does not give any advice on what to do in distributed, or virtual, organizational settings.

The Bank is the only organization where all teams are collocated, everyone on the same floor in the same office building. The teams at Auto and Gov consists of teams working in different locations; Gov in different cities in Sweden and Auto in Sweden, France, Brazil, and the USA.

5.6. Pre-planned coordination and emerging issues in SAFe

Table 2 displays all the presented SAFe practices and whether the practice aims at planned, vertical, coordination or for management of emerging dependency issues.

Table 2. Coordination practices in SAFe and their intended focus on planning or managing emerging coordination issues.

Coordination mechanism types and their practices	Equivalent SAFe practice	Practice intended for planning or managing emerging issues
1. Plans and rules		
Defining responsibility for tasks	PI plan	Planning
Resource allocation	PI plan	Planning
Developing agreement	PI plan	Planning
2. Objects and representations		
Direct information sharing	N/A	N/A
Scaffolding	Program Board	Both
Acknowledging and aligning work	Backlogs	Planning
3. Roles		
Monitoring and updating	RTE & POs	Both
Substitution	CoP	Both
Boundary Spanners	RTE	Both
4. Routines		
Task completion	System demos	Emerging issues
Hand-off work	SoS	Emerging issues
Bringing groups together	N/A	N/A
5. Proximity		
Collocate	Collocate	Emerging

6. Discussion and further work

This paper is a descriptive analysis of prescribed and suggested coordination practices for inter-team coordination in SAFe. Practices divided into five types of coordination mechanisms described by Okhuysen and Bechker [8] was used as an analytical lens to investigate practices described in SAFe as well as the implementation of SAFe in three different organizations.

Regarding plans and rules, SAFe prescribes planned coordination by conducting a release plan, called the PI plan, roughly every quarter involving all team members as well as stakeholders.

For objects and representations, SAFe prescribe the use of portfolio backlogs and program backlogs as boundary object used by both business people and developers. For work during the actual iterations, SAFe also prescribes the use of a Program board to visualize dependencies between teams during the upcoming release which cover the practice called “scaffolding”, intended to be used both for planned and emerging dependency issues. In the studied organization, however, the program board was only used for planned coordination. SAFe does not suggest any practice for “Direct information sharing” [8].

Regarding roles, SAFe explains the SM to be responsible, however not entirely, for coordination of emerging dependency issues. SAFe also prescribes one PO per team and prescribes the vertical coordinating responsibility between POs to an additional role called

Release Train Engineer (RTE), covering the practice called boundary spanning. The RTE together with the POs are supposed to both plan and manage emerging dependency issues. In all the case organizations, the roles are implemented exactly as intended by SAFe.

SAFe propose the use of Communities of Practice (CoP) and Scrum of Scrums (SoS) as routines for horizontal coordination. The CoP should also plan coordination within the specific domain intended for the CoP. Only two out of the three studied organizations had implemented CoPs and they also performed SoS meetings in different manners.

SAFe also emphasize the importance of routines for cadence and synchronization but nothing more in detail. Also, SAFe does not suggest any practices for "bringing groups together" [8] on a daily basis.

Regarding proximity, SAFe does not give any advice on coordination in distributed settings, they only highlight the importance of collocation.

March and Simon [3], Thompson [4], Van de Vehn [5] and Mintzberg [6] do not describe detailed practices to accomplish the important area of mutual adjustments, only that it is critical for horizontal coordination. But in this large-scale framework for agile ways of working, the focus is mainly the opposite. Much emphasis is instead put on detailed practices to manage emerging coordination issues, rather than on static, pre-planned vertical coordination. The focus on emerging coordination issues in agile ways of working is not surprising since, as Mintzberg [6] explains, mutual adjustments are most common in very simple and very complex organizations. Large-scale agile software development settings could be considered as very complex organizations.

Since this study shows that the up-scaled agile way of working according to the framework SAFe puts much focus on emerging coordination issues as well as pre-planned coordination, an obvious focus for future work is to investigate what impact this has. An area for future research is to study if this increase speed of delivery, to find out if productivity will increase. Another area would be to focus not only on output but the impact on teamwork itself – how is teamwork affected by the increased focus on emerging coordination?

References

1. 11th Annual State of Agile Report (2017), http://www.agile247.pl/wp-content/uploads/2017/04/versionone-11th-annual-state-of-agile-report.pdf, Accesses March 29, 2018
2. Malone, T. W., Crowston, K.: The interdisciplinary study of coordination. ACM Computing Surveys (CSUR). 26(1), 87-119 (1994)
3. March, J. G., Simon, H. A.: Organizations. John Wiley & Sons, New York (1958)
4. Thompson, J. D.: Organizations in action: Social science bases of administrative theory. Transaction Publishers, New Brunswick, New Jersey (1967)
5. Van de Ven, A. H., Delbecq, A. L., Koenig Jr, R.: Determinants of coordination modes within organizations. American sociological review, 322-338 (1976)
6. Mintzberg, H.: Structures in fives. Prentice-Hall, New-Jersey (1983)
7. Jarzabkowski, P. A., Lê, J. K., Feldman, M. S.: Toward a theory of coordinating: Creating coordinating mechanisms in practice. Organization Science. 23(4), pp. 907-927 (2012)
8. Okhuysen, G.A., Bechky, B.A.: Coordination in Organizations: An Integrative Perspective. The Academy of Management Annals. 3(1), pp. 463-502 (2009)
9. Leigh Star, S.: This is not a boundary object: Reflections on the origin of a concept. Science, Technology, & Human Values. 35(5), pp. 601-617 (2010)
10. Manifesto for Agile Software Development (2001), http://www.agilemanifesto.org. Accessed March 29, 2018
11. Rising, L., Janoff, N. S.: The Scrum software development process for small teams. IEEE Software. 17(4), pp. 26-32 (2000)
12. Beck, K.: Extreme programming explained. Addison-Wesley, Indianapolis (2000)

13. Schwaber, K., Beedle, M.: Agile software development with Scrum. Prentice Hall, New York (2001)
14. Schwaber, K.: Agile Project Management with Scrum. Microsoft Press, Redmond (2004)
15. Sutherland, J.: Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies. Cutter IT Journal. 14(2), pp. 5-11 (2001)
16. Paasivaara, M., Lassenius, C., Heikkilä, V. T.: Inter-team coordination in large-scale globally distributed scrum: Do Scrum-of-Scrums really work? In: Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement, pp. 235-238, ACM-IEE, (2012)
17. Cohn, M. (2007). Advice on conducting the scrum-of-scrums meeting.
18. Mathieu, J., Marks, M. A., Zaccaro, S. J.: Multi-team systems. International handbook of work and organizational psychology. 2, pp. 289-313 (2001)
19. Larman, C., Vodde, B.: Practices for Scaling Lean and Agile Development: Large, Multisite and Offshore Product Development with Large-Scale Scrum. Addison Wesley, Boston (2010)
20. Scaled Agile Framework 4.5 (2017), http://www.scaledagileframework.org Accessed March 26, 2018
21. Stojanov, I., Turetken, O., Trienekens, J. J. M.: A Maturity Model for Scaling Agile Development. In: Proceedings of 41st Euromicro Conference on Software Engineering and Advanced Applications, SEAA, IEEE (2015)
22. Leffingwell, D.: Scaling Software Agility: Best Practices for Large Enterprises. Addison Wesley, Boston MA (2007)