TDT4900 - Computer Science, Master's Thesis

Coordination Effectiveness in Large-scale Agile Software Development

Spring 2015

Author:
Espen Andreassen

Advisor:
Torgeir Dingsøyr

NTNU – Trondheim
Norwegian University of
Science and Technology

Department of
Computer and Information Science

Abstract

In later years agile development methodologies have seen a steady growth. Agile approaches were originally developed for small-scale contexts to cover the increasing need for flexibility and the urge to be first-to-market with technology in constant change. The benefits witnessed in this small-scale adoption has got large organisations to open their eyes. Therefore, it has not been surprising to see large-scale software development projects opt for the use of agile methodologies. However, the research regarding agile development in a large-scale context is still scarce.

Another aspect that has seen an increasing focus in the later years has been coordination effectiveness, which is identified as an important factor in software development and team performance.

These two aspects are combined and looked further into in this research study. The focus is on robust empirical studies performed on coordination in large-scale agile software development projects. Strode's theoretical model of coordination is also looked further into to identify its applicability in a large-scale context.

The main findings show similarities to coordination effectiveness in small-scale agile software development, but also some dissimilarities. Synchronisation, team co-location, an organisational openness culture, and appropriate infrastructure and supportive tools seem to have a positive effect on the team performance. On the other hand, number of sites and team size, as well as large time zone differences between teams, seem to have a negative effect on the level of effectiveness achieved through coordination in large-scale agile software development projects.

Keywords: Large-scale; Coordination; Coordination Effectiveness; Agile Software Development; Scrum

Preface

I am now entering my last year on a master degree in computer science where I specialise in software, or more specifically, software systems. I was introduced to agile development methodologies through different subjects at the "Norwegian University of Science and Technology", NTNU, and also got hands-on experience working with Scrum in a subject called "TDT4290 - Customer Driven Project". This subject in particular sparked my interest in agile development methodologies and the new ways of handling work and project organisation. After a summer internship with EY (formerly known as Ernst&Young) I got more intrigued with how communication and coordination was handled in real life business and IT projects. Therefore, my previous experiences led to a motivation in exploring the combination of agile development and coordination.

The work performed in this pre-study is carried out to give an insight in the field of coordination in large-scale agile software development projects. This insight will be of considerable importance for a planned master thesis prepared for my last semester at NTNU.

I would like to use this opportunity to thank Torgeir Dingsøyr for his support, assistance and knowledge throughout the research project as the advisor. I would also like to thank NTNU for giving me the opportunity to experiment with ideas within the boundaries of the research project, and letting me acquire interesting knowledge for the future.

Trondheim, June 1, 2015

Espen Andreassen

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

The introduction chapter takes a closer look at the motivation behind the study. It also looks at the concrete problem description and the background for this description, as well as the research question. Afterwards, a closer look at the scope and limitations of the research project is performed. Ending the chapter is a section giving a closer look at the report outline.

1.1 Motivation

1.2 Problem Description and Background

> ...

1.3 Scope and Limitations

1.4 Report Outline

Chapter 1: Introduction contains a brief and general introduction to the study at hand and the motivation behind it.

Chapter 2: Theory looks at important aspects of the research question, namely software development methodologies, coordination, large-scale, and performance in coordination.

Chapter 3: Method explains how the literature review was carried out throughout the research project.

Chapter 4: Results outlines the studies selected from the literature review, as well as their findings. It also links these studies to Strode's theoretical model of coordination.

Chapter 5: Discussion contains a summarised look at the findings from the results chapter, and connects these to the research question. Strode's theoretical model of coordination is also discussed further with regards to its applicability in a large-scale context.

Chapter 6: Conclusion carries out a summary of the most paramount points of the results and discussion chapters.

Chapter 7: Future Work outlines possible routes to take in the research field on inter-team coordination.

Chapter 2

Theory

Contents

In this chapter theory and literature relevant to the study is presented. It starts of with an introduction of both traditional and agile software development methodologies, with the main focus on Scrum. Afterwards a shift towards coordination is taken. Some well-known literature is looked at, for example Malone and Crowston's coordination theory. Further, to put the study into context, a definition of large-scale is given. To end the chapter a look at different aspects of effectiveness in team coordination is introduced.

2.1 Software Development Methodologies

The term software development methodologies has been around for quite some time now. These methodologies are frameworks for accomplishing a well-structured development process. In this section a brief introduction to the most prominent methodologies will be carried out. It will start with a quick look at the traditional software development, before ending with a presentation of the new and agile way of thinking. In the last section (on agile software development) the main focus will be on Scrum as this is the methodology found in most of the literature gathered from the literature review.

2.1.1 Traditional Software Development

Traditional software development methodologies have a distinct pattern. This pattern is sometimes called software development life cycle (SDLC) methodologies which is often found in system engineering. These "life cycles" are in contrast to the "iteration"-approach found in agile methodologies, such as Scrum. The most well-known of these traditional software development methodologies is Waterfall discussed further below.

Waterfall

The Waterfall methodology is one of the classic development models. It was first described in a paper by W. W. Royce in 1970 [1]. The model was not yet named in this paper, which it received later mostly due to its iconic structure (as shown in figure 2.1).

In the aforementioned paper, it is suggested that all software development models tend to go through two distinct phases: Analysis and Coding. The author argues that it is not possible to write a software project without having a somewhat deep understanding of the underlying problems that it needs to solve. Therefore an analysis phase will always be required in advance of writing the program itself. However, he also mentions that such a simple model is only suitable for programs that are completed in a matter of days. Larger software projects require an extended number of steps.

For larger projects, the following steps are suggested:

1. **System and Software Requirements:** The customer is involved with the specification of the scope and requirements of the system. The resulting documentation serves as a foundation to the next stages of development.
2. **Analysis and Program Design:** The requirements produced in the previous stage are used to create a system plan and various design documents.
3. **Coding and Testing:** The actual implementation of the project. This also involves continuously testing on various levels (for example unit and integration).
4. **Operation and Maintenance:** Once the project has been completed, it has to be maintained during its usage. In addition to improving the program in various

ways, this may also involve the inclusion of extra features if the customer so desires. These features can in themselves use the Waterfall model.

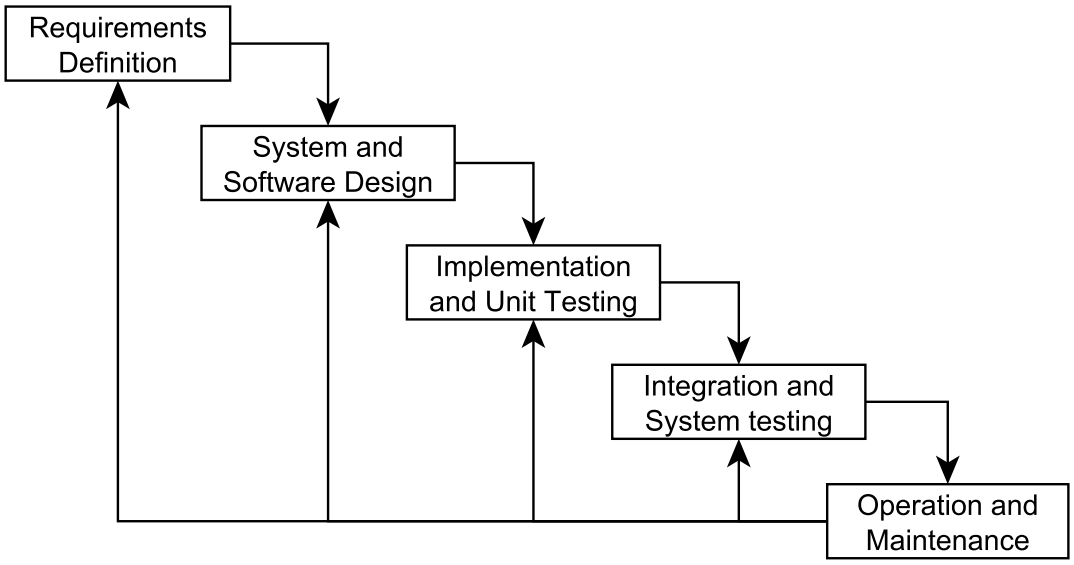

Figure 2.1: The Waterfall model.

The model initially suggested by W. W. Royce discusses a linear model in which each of the aforementioned stages are used as distinct steps in the development process. Each stage is required to be completed before the next is started. This may be a sound premise in theory, but as suggested in the paper it is likely to fail in practice. The argument used is that often during development, unforeseen problems in the design are encountered. The linear model does not allow for a return to a previous stage in development. Hence, it does not allow for changes in the design that could potentially resolve such problems.

Therefore, an alternative model is suggested that allows for the process to return to earlier stages if necessary. This may not be an ideal solution either, but it does allow for encountered problems to be addressed during development.

2.1.2 Agile Software Development

As can be seen from the ending of the Waterfall-section there were doubts about its applicability already at an early stage. With the advancement of business needs and customer involvement something had to change. This opened the door for the introduction of a new software development methodology, namely agile software development. This new way of thinking tries to deal with collaboration in a way that promotes adaptive planning, early delivery and continuous improvement, making the development phase faster and more flexible regarding changes [2].

Scrum

In this section an introduction to one of the most popular agile software development methodologies will be carried out, namely Scrum. This is based mainly on Abrahamsson, Salo, Ronkainen and Warsta's publication on agile methods [2]. In VersionOne's "7th Annual State of Agile Development Survey" Scrum or Scrum variants had a quoted 72% usage making it by far the most popular agile methodology in the survey [3].

Scrum is an iterative and incremental software development model (as shown in figure 2.2). It has come forth from the realisation that development methods that were common at the time of its introduction worked well in theory but did not in practice. These methods, Waterfall included, were designed to provide a structured and well-defined development process [4].

The agile software development processes, like Scrum, are part of a recent approach to software development. The idea with Scrum in particular is to divide the development into short periods called "sprints". This is done to focus effort for a limited time on short-term goals. Iterating over these goals allows the process to adapt the development plan based on progress but also to address any design problems that arise.

In short, the team concentrates on isolated parts, and through this prioritises on the most important tasks of the project first. The time span of a sprint is typically between one and four weeks long.

In order to implement the requirements step by step and in an orderly fashion, a repository is kept containing the features that have yet to be implemented. This repository is called the "product backlog". During development, the requirements could change over time. Therefore the product backlog is not static; it changes to the needs of the project with new topics being added, and obsolete ones being removed. The items from the backlog that a team works on during a sprint is called the "sprint backlog".

Meetings are also a key part of Scrum. There are several different types of meetings: sprint planning meeting, daily scrum meeting, backlog refinement, end of cycle and Scrum-of-Scrums. The sprint planning meeting is held at the beginning of each sprint cycle. Here the focus is on what work is to be done, and the sprint backlog for the coming sprint cycle is set. The daily scrum meeting, also called the daily stand-up, is a daily encounter (15 minutes) where each member of the project team answer these three questions:

1. What have you done since yesterday?
2. What are you planning to do today?
3. Are there any impediments in your way?

Further, there is the backlog refinement, also called "grooming". This is where tasks are created, large tasks are decomposed into smaller ones, tasks are prioritised, and the existing tasks are sized in the product backlog. Backlog refinement is often split into two meetings. In the first meeting the product owner and other stakeholders create and

refine stories in the backlog. In the second meeting the project team sizes the tasks in the backlog to make them ready for the next sprint. Planning poker is an example of how this can be carried out.

The last listed meeting occurs at the end of each cycle, and is therefore called end of cycle (meeting). This is actually two meetings: a sprint review meeting and a sprint retrospective. At the sprint review meeting the work that is completed and yet to be finished is reviewed. The completed work is also presented for the stakeholders, often called "the demo". At the sprint retrospective all members reflect on the past sprint. Two main questions are answered:

1. What went well during the sprint?
2. What could be improved in the next sprint?

The Scrum team usually consists of five to nine members. It is important to note that Scrum teams do not use traditional roles such as programmer, tester, designer or architect. Instead the main goal for the Scrum team is to collectively complete the tasks within the sprint.

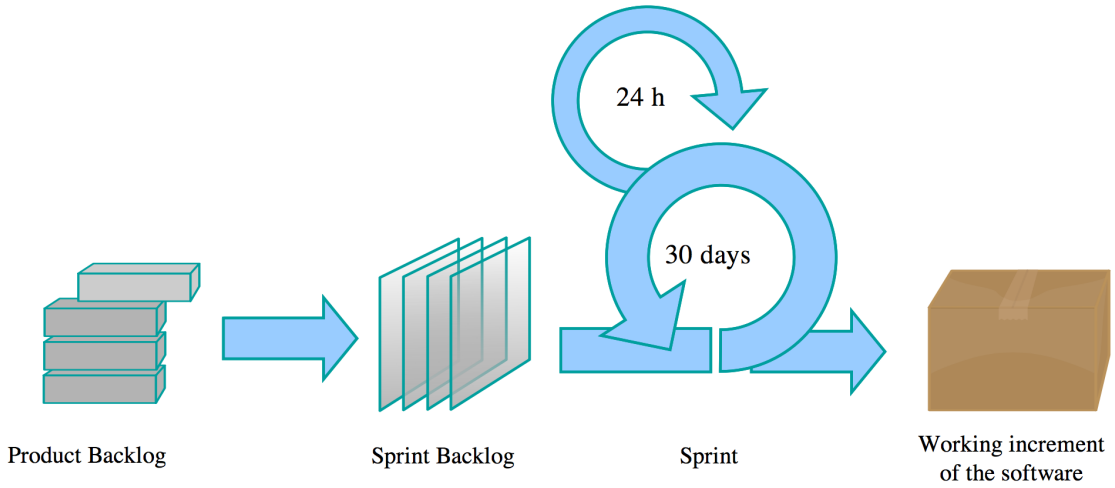

Figure 2.2: The Scrum cycle.

To end the section, as well as making a natural shift towards the next topic (Coordination), a look at Scrum-of-Scrums is carried out. It is a natural shift because Scrum-of-Scrums are used as the coordination mechanism across teams in the Scrum methodology. It works as the daily scrums (though usually implemented on a weekly basis because of time constraints and the complexity to find common times for all teams), but with one member assigned from each Scrum team to report completions, next steps and impediments for their respective teams. It is important that these impediments focus on the challenges that may impact coordination across teams and might limit other teams' work. The Scrum-of-Scrums will have their own backlog aiming to improve the cross-team coordination [5]. Below the suggested questions for the SoS meetings are listed [6]:

1. What did your team do since the previous meeting that is relevant to some other team?
2. What will your team do by the next meeting that is relevant to other teams?
3. What obstacles does your team have that affect other teams or require help from them?
4. Are you about to put something in another team's way?

Takeuchi et al. identified three strategies for distributed Scrum teams. The first type is isolated Scrum teams where teams operate as silos and no collaboration across teams is performed violating the agile principles. The second type is Scrum-of-Scrums which means overlapping Scrum teams. Here teams coordination, communicate and collaborate across teams through SoS meetings with participants from each team involved. Lastly, totally integrated Scrum teams are suggested. In this type teams are fully distributed and each team has members located at several sites. This approach creates similar characteristics as co-location. Type B is what is most common when several Scrum teams work together. The different types are visualised in figure 2.3 [7].

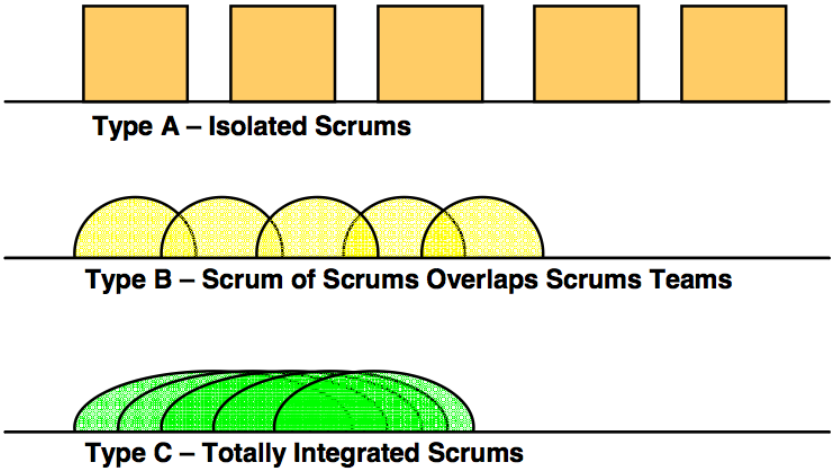

Figure 2.3: Different strategies for distributed Scrum teams.

2.2 Coordination

This section takes a look at different publications on coordination. It starts of with Malone and Crowston's well-known coordination theory. After this has been described a closer look at Strode's theoretical model of coordination is outlined. Ending the chapter is a brief look at the complexity factor introduced with a large-scale context in coordination.

2.2.1 Malone and Crowston's Coordination Theory

One of the most well-known papers on coordination theory was published by Malone and Crowston in 1990 and further redefined in 1994 (the focus will be on this paper) [8]. Their study spans different fields and can therefore be seen as an interdisciplinary coordination study. They list an extensive amount of different definitions of coordination, and through these proposed definitions and their own work come up with a rather simple definition:

> Coordination is managing dependencies between activities.

These dependencies can occur when some task has to be postponed or extended because of its connection to another task, resource or unit. Their theory is based on a combination of coordination from several different disciplines such as computer science, organization theory, operations research, economics, linguistics, and psychology. They state that coordination consists of one or more coordination mechanisms, and that each of these address one or more dependencies.

While Strode et al. acknowledges their coordination theory as very useful for identifying these so-called dependencies, categorising them, and identifying coordination mechanisms in a situation, they conclude that it is only a theory for analysis and not intended to be used for prediction. Despite this being true, and the coordination theory not being suitable for predicting outcomes such as coordination effectiveness, their theory adds important information for better understanding of how activities or artefacts support coordination in organisational settings [9].

2.2.2 Strode's Theoretical Model of Coordination

Strode et al. performed a multi-case study on three different co-located agile projects in 2012 [9]. From these projects the findings led to a theoretical model of coordination that will be outlined in this section. It is important to note that these projects were not large-scale, but the model will nonetheless be used to compare if there are similarities from the model proposed by Strode et al., and the findings from the literature review on large-scale agile project coordination. This will be performed in chapter 4, 5 and 6.

From these case studies three main components for the theoretical model were extracted: Synchronisation, Structure and Boundary Spanning. These components combine to what is called the "Coordination Strategy". Coordination strategy is in this context a group of coordination mechanisms that manage dependencies in a situation. The theoretical model of coordination can be seen in figure 2.4. Below the three main components will be explained in more detail:

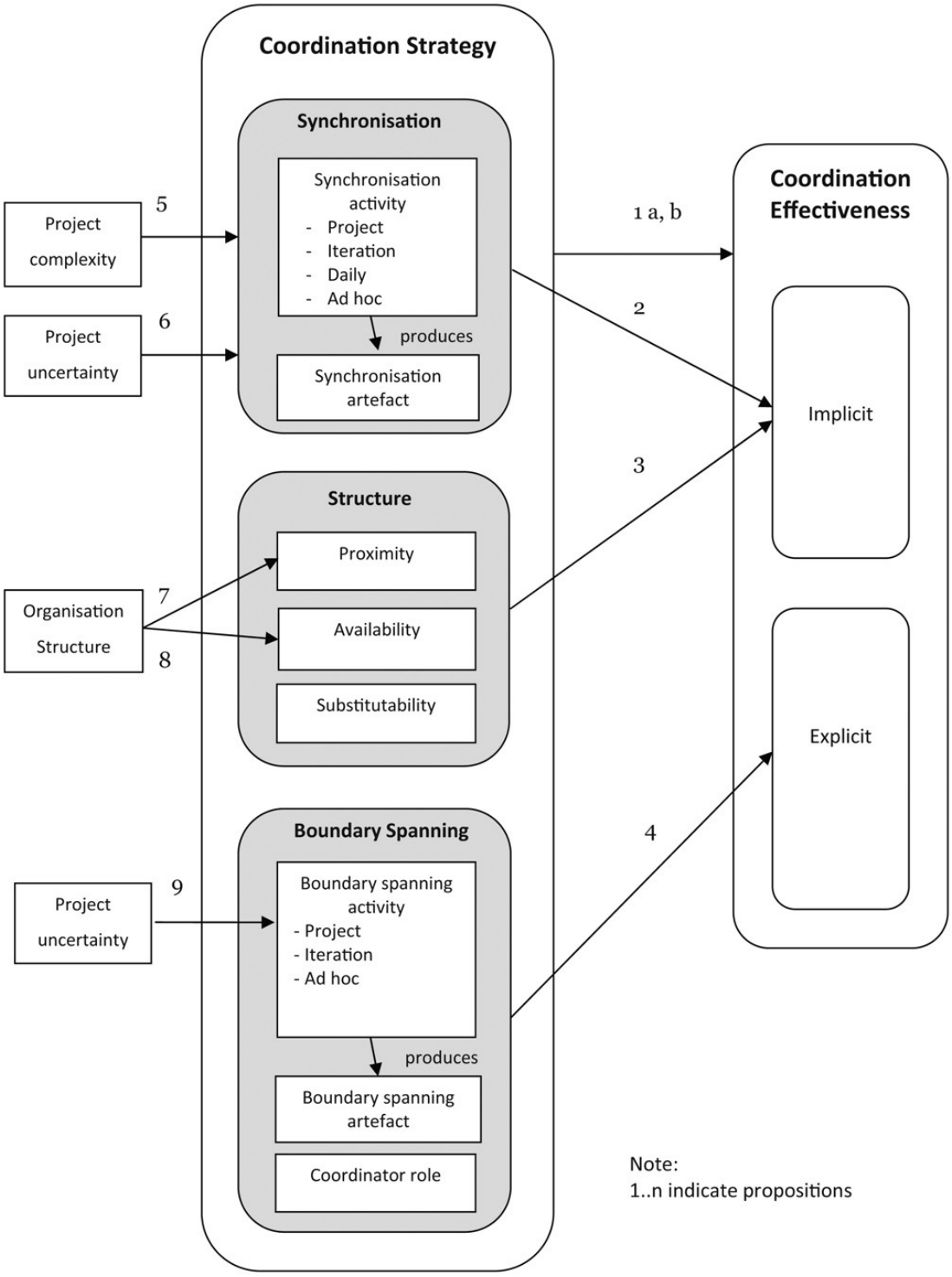

Figure 2.4: A theory of coordination in agile software development projects.

Synchronisation

Synchronisation in this context consists of synchronisation activities and synchronisation artefacts produced and used during these activities. Synchronisation activities are activities performed by all team members simultaneously. They contribute to a common understanding of the task, process, and or expertise of other team members. Synchronisation artefacts on the other hand are artefacts that are generated during synchronisation activities. These artefacts may be visible for the entire team or largely invisible but available. The artefacts can take a physical or virtual form, and are temporary or permanent.

Structure

Structure in this model is the arrangement of, and relations between, the parts of something complex. It consists of three categories: proximity, availability and substitutability. Proximity is the physical closeness of other (individual) team members. Availability means that other team members are accessible for requests or information. Lastly, substitutability has to do with the team members ability to perform others' work to maintain time schedules.

Boundary Spanning

The last component of the coordination strategy is boundary spanning. Boundary spanning has to do with the interaction with other organisations or other business units that are not involved in the project. It consists of three aspects: boundary spanning activities, boundary spanning artefacts and a coordinator role. Boundary spanning activities are activities performed to achieve help from some unit or organisation not involved in the project. The boundary spanning artefacts are artefacts produced to enable this external coordination. These artefacts have the same characteristics as synchronisation artefacts. Lastly, the coordinator role is a role taken by someone within the project team. His or her role is to support interaction to outside personnel to extract resources or information needed in the project at hand.

Coordination Effectiveness

There is another important part of the theoretical model of coordination, namely the coordination effectiveness concept. This concept will be further explained in section 2.4 that takes a look at coordination effectiveness.

Propositions

There are in total ten propositions (Proposition 1 has two parts) linking the coordination concepts in Strode's theoretical coordination model showed in figure 2.4. These are outlined below:

Proposition 1a: A coordination strategy that includes synchronisation and structure coordination mechanisms improves project coordination effectiveness when the customer is included in the project team. Synchronisation activities and associated artefacts are required at all frequencies – project, iteration, daily, and ad hoc.

Proposition 1b: A coordination strategy that includes synchronisation, structure, and boundary spanning coordination mechanisms improves project coordination effectiveness when the customer is an external party to the project. Synchronisation activities and associated artefacts are required at all frequencies – project, iteration, daily, and ad hoc. Boundary spanning activities and associated artefacts are required at all frequencies – project, iteration, and ad hoc.

Proposition 2: Synchronisation activities at all frequencies – project, iteration, daily, and ad hoc, along with their associated synchronisation artefacts, increase implicit coordination effectiveness.

Proposition 3: Structural coordination mechanisms i.e. close proximity, high availability, and high substitutability, increase implicit coordination effectiveness.

Proposition 4: High levels of boundary spanning coordination mechanisms, i.e. boundary spanning activities at all frequencies – project, iteration, and ad hoc, their associated boundary spanning artefacts, and a coordinator role, increases explicit coordination effectiveness.

Proposition 5: Under conditions of high project complexity, increasing the frequency of iteration and ad hoc synchronisation activities will maintain coordination effectiveness. The production of related synchronisation artefacts must be adjusted accordingly.

Proposition 6: Under conditions of high project uncertainty, to maintain synchronisation activity frequency and production of associated artefacts, changing the priority of stories will maintain coordination effectiveness.

Proposition 7: A mono-project organisation structure enables close proximity relative to multi- or matrix structures.

Proposition 8: A mono-project organisation structure improves availability relative to multi- or matrix style structures.

> **Proposition 9:** Under conditions of high project uncertainty, when the customer is not part of the team, increased boundary spanning coordination mechanisms will maintain coordination effectiveness. The production of related boundary spanning artefacts must be adjusted accordingly.

2.2.3 Coordination in Large-scale

This section takes a closer look at general studies performed on large-scale coordination and is not specifically focusing on software development. The section is added to highlight the introduction of complexity that a large-scale context brings with it.

Van der Ven et al. released an article in 1976 where they tried to identify determinants of coordination modes within organisations. They state that an increase in size will produce a trade-off between the increasing complexity and cost of coordination at the administrative level. From the research two different coordination forms are described, namely vertical and horizontal. The vertical communication includes coordination through curators, while the horizontal communication occurs by way of one-to-one communication. Their findings show that when team size increases the coordination moves towards a more vertical and impersonal style [10]. This is backed up by John Child in a publication from 1973. Here he states that with a growing complexity level there is likely that administrative problems will occur regarding coordination and control [11].

2.3 Large-scale

Having looked at coordination in large-scale in section 2.2.3, what is actually this so-called "large-scale"? This was a topic brought up at a workshop regarding research challenges in large-scale agile software development where opinions regarding how large-scale should be defined varied a lot. Some suggestions were to define it through project duration, project cost, number of people involved, number of remote sites and/or number of teams [12]. This issue was further analysed by Dingsøyr, Fægri and Itkonen trying to work out a taxonomy of scale for agile software development. Their results are summarised in table 2.1 where the taxonomy of scale is based on the amount of teams involved in the development project [13].

Level	Number of teams	Coordination approaches
Small-scale	1	Coordinating the team can be done using agile practices such as daily meetings, common planning, review and retrospective meetings.
Large-scale	2-9	Coordination of teams can be achieved in a new forum such as a Scrum of Scrums forum.
Very large-scale	10+	Several forums are needed for coordination, such as multiple Scrum of Scrums.

Table 2.1: A taxonomy of scale of agile software development projects.

Others have also discussed problems regarding large-scale. For example Schnitter and Mackert discuss the scaling of Scrum at SAP AG and concludes that in their case the maximum involved development employees that may be organised with regards to agile project management is 130 (This number sums up developers in 7 teams (max. 70 people), the product team (max. 16), development infrastructure responsible (about 10), quality assurance and testers (about 25), general management (about 10)) [14].

Another example is taken from Nord et al. defining large-scale by scope of the system, team size, and project duration. They say that the size of the development team must be more than 18 people and distributed into a few teams [15].

So the definition of a "large-scale agile project" used in this research will be:

> An agile project must consist of a minimum amount of two teams coordinating across the teams to be categorised as large-scale.

2.4 Efficiency, Effectiveness, Productivity and Performance in Coordination

There has been released a good amount of papers regarding effectiveness, productivity and efficiency in project literature. Unfortunately research in this area that focuses on large-scale is scarce. Therefore, the work highlighted in this section will mainly be extracted from small-scale studies. To start the section of a closer look at the aforementioned study by Strode et al. will be performed, before a summary of some different field studies on the matter will be carried out.

2.4.1 Strode's Coordination Effectiveness

Part of the theoretical model of coordination by Strode et al. seen in figure 2.4 is the so-called "coordination effectiveness". This concept was developed by Strode et al. in 2011 having used the same three agile projects discussed earlier, as well as a non-agile software development project as a foundation [16]. Coordination effectiveness is defined as the outcome of a particular coordination strategy. Coordination effectiveness is split into two components: an implicit and an explicit part.

The implicit part is concerned with coordination that occurs without explicit speech or message passing, this happens within work groups. It has five components: "Know why", "Know what is going on and when", "Know what to do and when", "Know who is doing what", and "Know who knows what". These aspects are pretty self-explanatory.

The explicit component on the other hand is concerned with the physical aspects of the project. It states that the objects involved in the project have to be in the correct place, at the correct time and in a state of readiness for use. A summary of the combination of explicit and implicit coordination effectiveness is provided in figure 2.5.

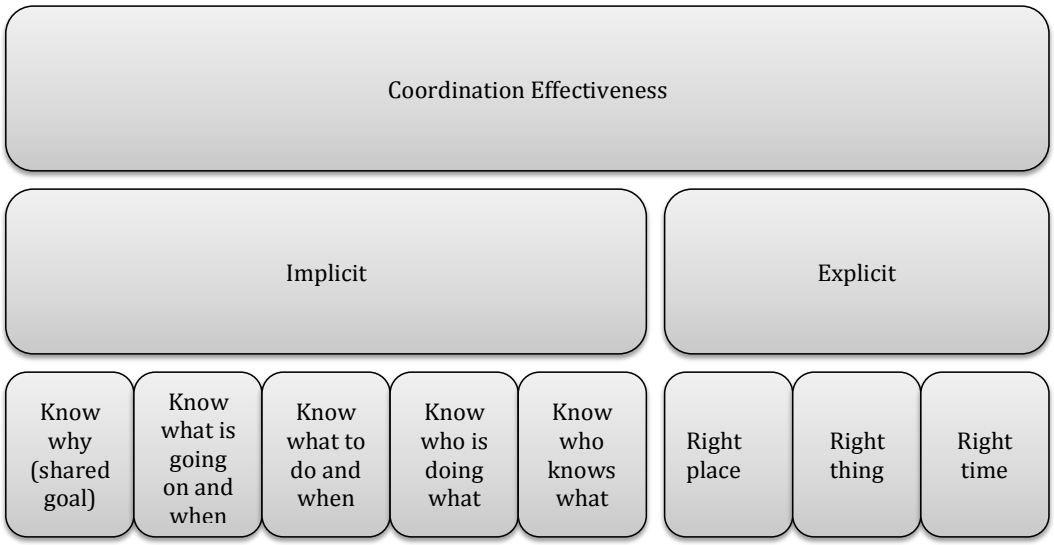

Figure 2.5: Components of coordination effectiveness from Strode et al. (2011).

To end this subsection a definition of coordination effectiveness from Strode et al. is provided:

> Coordination effectiveness is a state of coordination wherein the entire agile software development team has a comprehensive understanding of the project goal, the project priorities, what is going on and when, what they as individuals need to do and when, who is doing what, and how each individuals work fits in with other team members work. In addition, every object (thing or resource) needed to meet a project goal is in the correct place or location at the correct time and in a state of readiness for use from the perspective of each individual involved in the project [16].

2.4.2 Some Studies on the Field

Below four studies that try to identify important factors of coordination's impact on team performance are described.

Team Effectiveness 1997-2007: A Review of Recent Advancements and a Glimpse Into the Future

Mathieu et al. takes a look at literature published on team effectiveness in a ten year period. They look at several different aspects regarding the nature of teamwork [17]. It is important to note that the main focus of this article is on small-scale teams, and that the publications used are not gathered directly from the software and agile field. However, the article gives perspectives that are noteworthy. The main focal point here will be on Mathieu's chapter on organisational contexts, and the section on multi-team systems coordination in particular.

One aspect that was identified in several studies having a positive impact on performance was an "openness climate". What was concluded at the macro organisational level was that a support for a openness climate at the broader level of the organisation had a positive impact on team level processes.

Quite a few studies were identified on multi-team systems coordination as well. Here, the findings showed a positive correlation between inter-team coordination and intra-team coordination. Hyatt et al. indicated that teams perform more effectively as self-contained units when they have robust information networks, as well as communication and cooperation channels, both within and between teams [18]. This again highlights the importance of studies focusing on coordination in large-scale.

Interpretative Case Studies on Agile Team Productivity and Management

Melo et al. performed a multi-case study on three large Brazilian IT companies that were using agile methods in their projects [19]. The objective of the research was to provide a better understanding of which factors that had an impact on agile team productivity. To document teamwork effectiveness they used the well-known theoretical model "Input-Process-Outcome" (IPO). Their input factors were "Individual and Group characteristics", "Stage of team development", "Nature of task", "Organizational context" and "Supervisory behaviors". One process-category was identified: "Group processes". Lastly they identified two outcome-groups, namely "Agile team productivity" and "Attitudinal and Behavioral". All of these are summarised constituting the conceptual framework for their agile team productivity in figure 2.6.

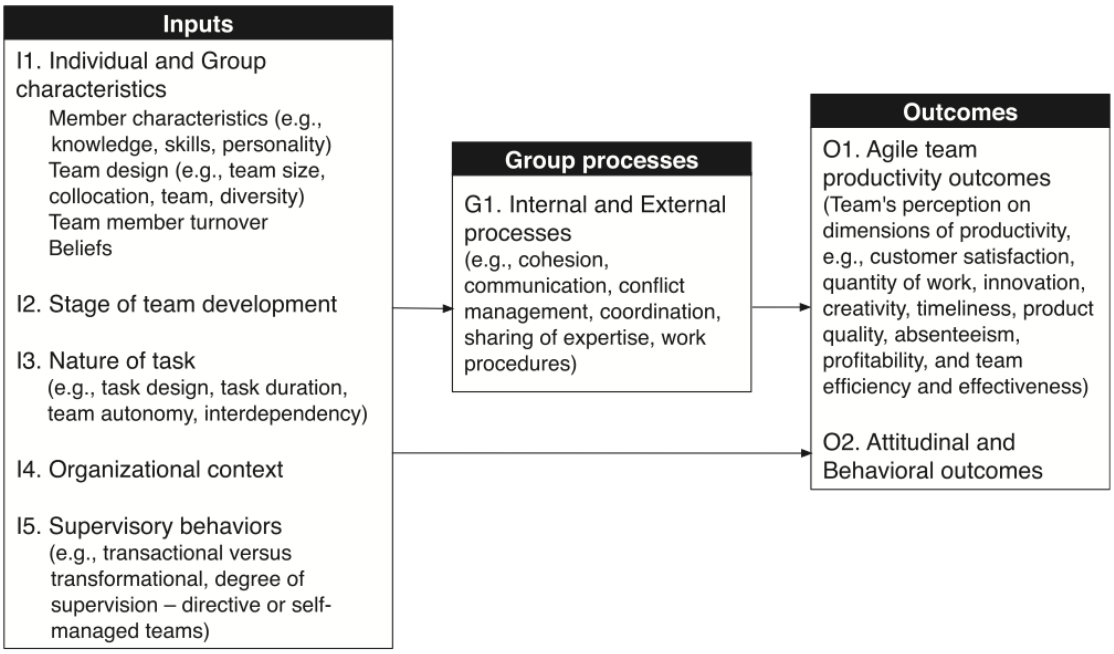

Figure 2.6: Agile team productivity conceptual framework.

After collecting the data from their multi-case study they mapped the results in a thematic map on agile productivity factors. These findings showed three main groups of team management and their impact on productivity. For this study it is the "Inter-team coordination" and "Team design choices" that are interesting because of their impact on coordination to a larger degree, meaning "Team member turnover" is left out.

In "Team design choices" four roots of impact were identified: "Team size", "Team members skills", "Team collocation" and "Team members allocation". Out of these team collocation and team size seem to effect coordination effectiveness the most. Their findings showed that smaller teams led to better communication and alignment, while collocation had a positive influence on team productivity as it helped overcome invisible barriers between teams in a hierarchical company.

For "Inter-team coordination" two roots were identified: "Lack of commitment among

teams" and "Inappropriate coordination rules among teams". One of the main reasons for negative impact was identified to be external dependencies because projects often were left waiting for results of entities outside the project team. So a problem in inter-team coordination was misalignment, hence, synchronisation is an important factor.

Dispersion, Coordination and Performance in Global Software Teams: A Systematic Review

Anh et al. performed a systematic literature review (SLR) to collect relevant studies on dispersion, coordination and performance in global software development (GSD), and highlighted the findings of impact factors in a thematic mapping [20]. It is important to note that the findings are not from agile software development, but they are still interesting because of the global aspect in the literature used. The results are briefly summarised in table 2.2:

Type	**Impact on team performance**
Presence of geographical dispersion	Negative (work takes longer time, less effective communication and coordination)
Number of sites/Team size	Negative (complicates coordination and hampers communication)
Large time zone differences between teams	Negative (creates coordination problems because of the complexity introduced)

Table 2.2: Impact of geographical dispersion on performance.

Team Performance in Agile Development Teams: Findings from 18 Focus Groups

Dingsøyr and Lindsjørn carried out a focus group study looking at which factors the agile software practitioners in the research perceived as influential on effective teamwork [21]. This paper focuses on the team performance of individual teams, but is included because of its agile nature. To place the suggestions from the participants into categorise Dingsøyr et al. decided to use the "Big Five" model proposed by Salas et al. [22] leading to eight teamwork components: "Team leadership", "Mutual performance monitoring", "Backup behaviour", "Adaptability", "Team orientation", "Shared mental models", "Mutual trust" and "Closed-loop communication". A summary of the distribution of all suggestions over these components is outlined in table 2.3.

Teamwork component	Foster	Hinder	Total
Team leadership	90	139	229
Mutual performance monitoring	49	22	71
Backup behaviour	44	57	101
Adaptability	46	50	96
Team orientation	91	65	156
Shared mental models	104	59	163
Mutual trust	97	58	155
Closed-loop communication	122	90	212
Sum	643	540	1183

Table 2.3: Summary of the distribution of suggestions over teamwork components.

The teamwork component with the strongest connection to coordination is "closed-loop communication". Looking at table 2.3 a lot of emphasis was aimed towards the component from the practitioners (second highest total count). This again illustrates the importance of coordination. The sub-components identified of closed-loop communication are outlined in table 2.4.

Sub-component	Foster	Hinder
Co-location	Physical presence Co-location Physically placed together	People are distributed Distance Not co-located
Openness	Open communication Openness in the team Open dialogue	Secrecy Retaining information
Infrastructure	Process support tools Suitable office spaces Tools that work	Bad tools Bad office facilities
Visualising status and progress	Informative workspace Visualise things that go well Whiteboard/task-board	No whiteboards
Social atmosphere	Good atmosphere Fun Friendly tone	Scolding Antisocial environment Bad atmosphere

Table 2.4: Sub-components identified of closed-loop communication with their respective performance items.

As can be seen from table 2.4 a lot of attention was directed towards location of team members, infrastructure and supportive tools, and organisational culture. The presence of co-location, a good infrastructure and supportive tools, and an open and social climate seem to all have a positive effect on team effectiveness.

Summary

The findings from the different studies are summarised in table 2.5. Note that it could be argued that misalignment and synchronisation, as well as team collocation and presence of geographical dispersion, are contrasts of each other. They are however included in the summary table because they were identified as important aspects in the different studies.

Type	**Impact**
Organisational openness culture	Positive
Misalignment	Negative
Synchronisation	Positive
Team co-location	Positive
Presence of geographical dispersion	Negative
Number of sites/Team size	Negative
Large time zone differences between teams	Negative
Infrastructure/Supportive tools	Positive

Table 2.5: Summary of impacts identified in the studies.

2.5 Multiteam Systems

As mentioned earlier the work environments have become more challenging and complex in line with the growth of communication and information technology. This growth has led to the globalisation of organisational work. With the globalisation an increase in interconnectivity across organisational boundaries has become apparent. Because of this trend new questions and problems have surfaced. Unfortunately, these questions and problems have not been possible to adapt to traditional organisational forms. This has led to the introduction of new and different organisational forms, e.g., matrix and virtual organisations, and cross-functioning and ad hoc project teams. One of these new organisational forms focus on projects where collaboration exists across traditional team and organisational boundaries. This form does not resemble traditional organisations or large-scale teams, but can be seen as an aggregation that includes tightly coupled arrangement of teams, where the different teams may have noticeable different norms, expertise, missions, structures and operating procedures to the overall work. Mathieu, Marks, and Zaccaro [**?**] defined the organisations corresponding to the aforementioned form as multiteam systems (MTSs). Below their definition follows:

> Two or more teams that interface directly and interdependently in response to environmental contingencies towards the accomplishment of collective goals. MTS boundaries are defined by virtue of the fact that all teams within the system, while pursuing different proximal goals, share at least one common distal goal; and in doing so exhibit input, process and outcome inter-dependence with at least one other team in the system [**?**].

From this definition it is easy to see similarities with so-called "large-scale" projects and organisations. Both large-scale projects' and multiteam systems' taxonomies look at the amount of teams involved, where the minimum number is two, but are typically larger than this number by a considerable margin. In both categories the teams have to be somewhat interconnected, and the organisational boundaries may be crossed, meaning teams can reside in different organisations. Mathieu et al. have therefore categorised MTSs into "internal MTSs" where the whole system or project is situated within an organisation, and "cross-boundary MTSs" where teams are located in different organisations, hence organisational boundaries have to be crossed to achieve collaboration [**?**].

One of the most distinguishing factors of multiteam systems is their focus on goal hierarchies. As mentioned above interdependencies are not only witnessed within teams, but also across them. From the definition of MTSs the teams have different proximal goals, but all share at least one distal goal. Mathieu et al. define the feature of these goal hierarchies that are relatively common across different MTSs as:

1. MTS goal hierarchies have a minimum of two levels
2. Goals at higher levels entail greater interdependent actions among more component teams than goals at lower levels
3. The superordinate goal at the apex of the hierarchy rests on the accomplishment by component teams of all lower order goals
4. Higher order goals are likely to have a longer time horizon than lower order goals
5. Goals vary in their priority and valence

2.5.1 MTS Characteristics

Having looked at the features of multiteam systems the attention is shifted towards their attributes. These attributes are what separates different MTSs. The attributes are classified into three dimensions, compositional, linkage and development attributes, and will be presented in the following sections. The different dimensions are summarised in table 2.6.

Compositional Attributes

In the compositional dimension several demographic features of the MTS and characteristics of component teams are looked at. In total there are ten attributes, and these will be outlined in this section. Regarding the magnitude of the MTSs two attributes are used. Firstly the "number" of component teams located within the MTS, and secondly the total "size" of the MTS, meaning the amount of individual members involved in the multiteam system.

Another compositional attribute that was earlier mentioned as a distinguishing factor is "boundary spanning". This attribute is concerned with where the different component teams originate from. If all component teams come from the same organisation it is an internal MTS, while if the component teams come from two or more organisations it is an external MTS. External MTSs are more complex and are more likely to run into task and social complexity than its counterpart. In this context, social complexity refers to diversity, scale, scope, and dynamism of stakeholders in the MTS's environment [**?**]. There are two more attributes concerned with boundary spanning which are at a higher detail level. Firstly the "organisational diversity" looks at the total amount of organisations represented in the MTS. With a higher number of organisations the likelihood of a higher level of social complexity rises. Secondly the "proportional membership" outlines the percentage of teams from different organisations. With an unbalanced proportional membership there is a risk that the influence level will be greater from the organisation(s) with the highest amount of teams.

The sixth compositional attribute is concerned with how similar the different component teams' core task and goals are. This attribute is called "functional diversity". With an increase in this so-called functional diversity problems may occur. Another important factor in MTSs is "geographic dispersion". There are three degrees of geographical dispersion, namely co-located, partially co-located, and fully dispersed. Some problems that have been witnessed in dispersed projects has been communication issues, coordination difficulties and trust building. Building on the geographic dispersion is an attribute called "cultural diversity". If teams are dispersed and the boundaries extend the national borderline this could lead to cultural clashes.

The ninth attribute in the compositional dimension is "motive structure". This attribute refers to the degree to which the different teams commit to the MTS, and how compatible and closely linked the team goals and the MTS goals are. A problem that can occur in this compositional attribute is that a team's proximal and/or distal goals are in conflict with the overall goals of the MTS leading to more complex interteam processes. With an increase in incompatibility in goals between the MTS and the compositional team(s) this can lead to team members being less committed to the overall goal hierarchy of the multiteam system. Motive structure may be associated with the last compositional attribute called "temporal orientation". Temporal orientation is concerned with the amount of resources dedicated to the MTS by each component team.

As can be seen the compositional attributes are important factors in interteam dynamics within MTSs. Focus on team composition is important to keep the level of effectiveness high, as well as prohibiting evolution of subgroups.

Linkage Attributes

Moving on the focus is shifted towards the so-called linkage dimension. Linkage mechanisms and attributes are concerned with how teams are arranged and connected within a multiteam system. The first attribute is concerned with the amount of coordination between different component teams that is needed and is called "interdependence". The degree of interdependence will differ from different MTSs, but with an increasing interdependence the amount of interteam processes necessary will increase to achieve high MTS effectiveness.

Two other linkage attributes that often correlate are "hierarchical arrangement" and "power distribution". Hierarchical arrangement focuses on how teams are organised within the MTS with regards to their responsibility level of goal attainment. The more proximal goals the component team is involved with, the higher in the hierarchical arrangement they are. As for power distribution the focus is on the relative influence that component teams have within a multiteam system. Often the teams placed higher in the hierarchical arrangment, meaning they have more proximal goals, also have a bigger influence and power. Other factors that could lead to higher power can be team size, the team's functional centrality to the core mission of the MTS, and/or the parent organisations having assigned the team with authority. Both attributes will likely influence communication, interaction and collaboration between the component teams in a multiteam system.

Moving on to the forth and fifth linkage attributes the focus is shifted towards the communication structures of MTSs. Firstly "communication networks" refer to the most common interaction and communication patterns between and within component teams. These networks can be fully decentralised (everyone interactive with everyone), fully centralised (everyone communicate to and through one single member of the MTS), and various patterns between these two boundary points. It is important to notice that the chosen communication network in a multiteam system will have a great impact on the task efficiency of the MTS. Lastly "communication modality" is concerned with which modes are used to communicate across component teams within a multiteam system. These can be, e.g., face-to-face interaction, electronic communication, or a mixture of the two. Often the degree of which modes are used is closely linked to the aforementioned compositional attribute called "geographic dispersion". With co-located teams preferring a higher amount of face-to-face communication.

Developmental Attributes

The last dimension of multiteam systems is the developmental. Developmental attributes are concerned with the developmental dynamics and patterns of MTSs. The first attribute looks at how the MTS was put to life, its "genesis". The origin of MTSs can either occur through appointment from parent organisations, or they may emerge from collective initiative of several teams. The type of genesis can have an impact on different aspects of a MTS, e.g., the distal goal. Another developmental attribute is "direction of development" and looks at the direction the MTS takes from its origin. For example the MTS could have emerged from a specific event, but then move towards a more formalised entity as

time passes. Another development path could be the MTS being formally planned in anticipation of a possible situation occurring, but when the event does occur acctually evolve in membership and linkages.

Two other developmental attributes are "tenure" and "stage". The tenure attribute is concerned with the anticipated duration of the MTS, while the stage attribute looks at which particular stage of development the MTS is in. Starting as a newly formed multiteam system it will evolve through different phases to finally becoming a mature MTS. The stage of MTS development will often give a hint to the efficiency of the MTS's interteam processes.

The last two developmental attributes together combine to the group term "transformation of system composition", meaning if there are changes to the composition as the MTSs develop and move through the different phases of development. The first of these focuses on "membership constancy" and refers to how constant or fluid the number of component teams are. Often in more complex and turbulent environments the amount of component teams may change over the course of the MTS's lifespan. Lastly "linkage constancy" is concerned with how the component teams are connected. The focus is on if these linkages between the component teams in a multiteam system is constant or if they change as the MTS progresses. Again the likelihood of fluidity in coordination structures between teams is higher when the MTS is located in more turbulent and dynamic environments.

In the above sections three dimensions of multiteam systems and their attributes have been presented. It is important to note how the different attributes can be factors in achieving effective MTSs and MTS processes. A simple model of MTS effectiveness is outlined in figure 2.7, where the attributes of the compositional, linkage and developmental dimensions can be seen as predecessors of different intrateam and interteam processes. The effects of these attributes on the total effectiveness of the multiteam system would be arbitrated by these intra- and interteam processes.

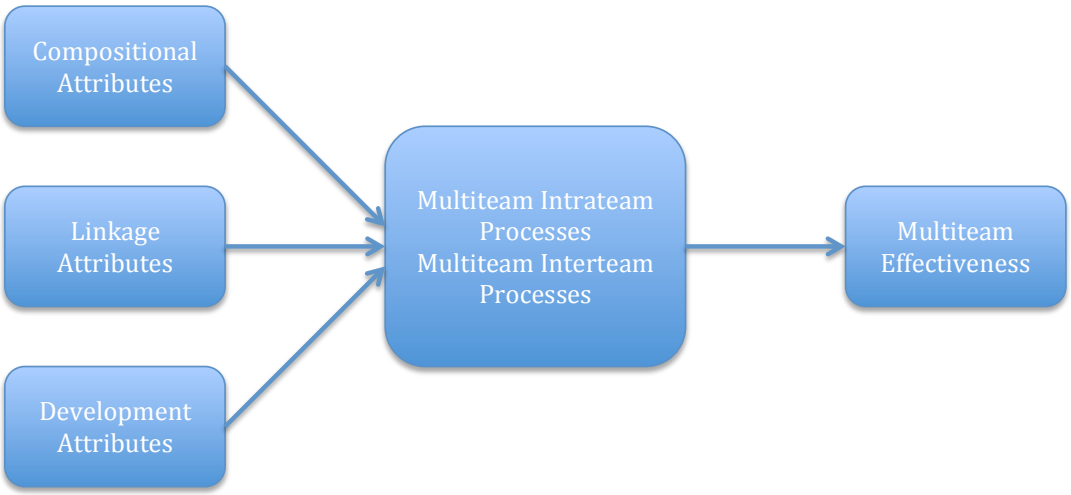

Figure 2.7: A model of multiteam system effectiveness.

Dimension	Attribute	Explanation
Compositional	Number	Number of component teams within the MTS
	Size	Total number of individual members across teams
	Boundary status	Component teams come from single organization (internal) versus multiple organisations (external or cross-boundary)
	Organisational diversity	In a cross-boundary MTS, the number of different organisations represented among the component teams
	Proportional membership	In a cross-boundary MTS, the percentage of teams from different organisations
	Functional diversity	Degree of heterogeneity in the core purposes and missions of component teams
	Geographic dispersion	Co-located or dispersed component teams
	Cultural diversity	Degree to which component teams come from different nations or cultures
	Motive structure	Degree of commitment of each component team to the MTS; the compatibility of team goals and MTS goals
	Temporal orientation	Level of effort and temporal resources expected of each component team
Linkage	Interdependence	Degree of integrated coordination (e.g., input, process, outcome) among members of different component teams
	Hierarchical arrangement	Ordering of teams according to levels of responsibility
	Power distribution	The relative influence of teams within the MTS
	Communication structure: Network	The typical patterns of interteam communication
	Communication structure: Modality	The modes of communication (e.g., electronic, face-to-face, or mixed) that occur across component teams
Developmental	Genesis	The initial formation of an MTS as either appointed or emergent
	Direction of development	From emergent to formalised; an evolution from an early formal state
	Tenure	The anticipated duration of the MTS
	Stage	The stage of MTS development from newly formed to mature
	Transformation of system composition: Membership constancy	Fluidity versus constancy of component teams as members
	Transformation of system composition: Linkage constancy	Fluidity versus constancy of linkages among component teams

Table 2.6: Dimensions of multiteam system (MTS) characteristics.

Chapter 3

Method

Contents

The research study used different methods to gather the relevant publications that were selected. These are further outlined in this chapter starting with a detailed look at the literature review performed, as well as highlighting other parts of the gathering methodology, namely snowball sampling and general accumulation of papers.

3.1 Literature Review

For this study a literature review was chosen as the information gathering method. For the searching process and selection of articles in the literature review certain recommendations from systematic reviews were followed. The general procedure of such a review is outlined in L1 below. It is important to note that the searching had an open-mindedness regarding search words and the selection process.

L1 - The steps of a systematic review [23]:

1. Framing questions for a review.
2. Identifying relevant work.
3. Assessing the quality of studies.
4. Summarizing the evidence.
5. Interpreting the findings.

Some of the benefits and objectives of a literature review are summarised in L2 below.

L2 - Objectives of a literature review [24]:

- Show that the researcher is aware of existing work in the chosen topic area.
- Place the researcher's work in the context of what has already been published.
- Point to strengths, weaknesses, omissions or bias in the previous work.
- Identify key issues or crucial questions that are troubling the research community.
- Point to gaps that have not previously been identified or addressed by researchers.
- Identify theories that the researcher will test or explore by gathering data from the field.
- Suggest theories that might explain data the researcher has gathered from the field.
- Identify theories, genres, methods or algorithms that will be incorporated in the development of a computer application.
- Identify research methods or strategies that the researcher will use in the research.
- Enable subsequent researchers to understand the field and the researcher's work within that field.

3.1.1 General Outline

As explained in subsection 3.1.3 a set of articles and publications were provided by the supervisor to give an overview on the field and agile software development in general. This made it easier to classify which studies to look for and how to evaluate their relevance and rigour. The databases used in the literature review are summarised in table 3.1. When searching in these databases concepts and keywords were combined to match the research question as well as other interesting combinations. These concepts and keywords are outlined in table 3.2. It is important to note that the last concept was an additional search word used because a lot of research seemed to either have focused on a co-located or a distributed manner.

Name	**Impact**
ISI Web of Science	apps.webofknowledge.com
ACM Digital Library	dl.acm.org
Science Direct (Elsevier)	sciencedirect.com
Google Scholar	scholar.google.com

Table 3.1: Databases used in the literature review.

Concept	**Keywords**
Coordination	Communication, collaboration
Agile	Scrum, XP, Crystal, Lean, Kanban, Extreme Programming, Xtreme Programming
Large-scale	Global, multiteam/multi-team (systems), distributed, international
Effectiveness	Efficiency, productivity, performance
Location (Additional search words)	Co-located, collocated, colocated, co located, distributed, dispersed, global, globally, international

Table 3.2: Search words used in the literature review.

The literature review provided an extensive amount of findings, unfortunately a lot of the publications were focusing on small-scale development. Therefore, a selection process had to be carried out. Here all abstracts of the collected literature were read and publications with the highest relevance were chosen. The articles that were still left after this selection process were then read thoroughly where some were discarded to give an appropriate amount of publications. The analysis outlined above focused mainly on finding articles focusing on large-scale agile inter-team coordination, meaning such articles were given a higher score when identified. Some other aspects that contributed to the score were mentioning of global projects, effectiveness and inter-team coordination in general. This process was important because of the time constraints specified on the study, and to obtain relevant and rigorous literature to insure a robust study.

3.1.2 Snowball Sampling

Snowball sampling is a term that reflects how new studies are selected through already chosen studies (based on their similarities) [25]. This was done in two ways in the research. In table 3.1 a list of databases used for the literature review are summarised. Some of these databases provided snowball sampling in the way of suggesting similar articles when a specific publication was selected from a search. This is the first way of snowballing used. The second way was through using reference lists in selected articles and publications. This extraction lead to a lot of well-written and recognised papers.

3.1.3 General Accumulation

Articles were also accumulated through a supervisor and fellow research students. At the start a handful of publications were received from the supervisor, and other papers were also acquired throughout the study. It is important to note that all the articles were inspected in the same manner as the publications found from the literature review to make sure their relevance and rigour were appropriate.

3.2 Research Method

Because there has been few studies on the field of inter-team coordination in large-scale agile software development an exploratory case study was chosen as the research method. An exploratory case study is an excellent fit to get a greater understanding of somewhat unexplored territories, as well as spawning possibilities for further research. Details about exploratory case studies is further outlined in the quotation below.

> An exploratory study is used to define the questions or hypotheses to be used in a subsequent study. It is used to help a researcher understand a research problem. It might be used, for example, where there is little in the literature about a topic, so a real-life instance is investigated, in order to identify the topics to be covered in a subsequent research project [].

To choose an appropriate case organisation and project certain criteria had to be present. These are further discussed in section 3.2.1. In the end one case was selected as the focus because of its availability, as well as the case successfully fulfilling the case criteria. The case was described as the most successful large-scale agile software development project in Norway so far. The project will be referred to as “Omega” throughout this thesis and was project to develop a new office automation system for the public department “Gamma”. The Omega-project ran from 2008 to 2012, and had at most 13 development teams

involved. The distribution of these teams were six Gamma-teams, as well as four Alfa-teams and three Beta-teams. “Alfa” and “Beta” were contracted consulting companies in the Omega-project. The project as a whole is outlined in more detail in chapter 4.

Data collection was granted by all three organisations. Before the data collection took place it was important to get a better overview of the project and case at hand. Therefore available data was looked at in detail, e.g., public presentations. Because of this early review the complexity level of the Omega-project was identified at the initiating stage of the case study with a total of 175 people being involved in the project and sub-projects. As a result of the data collection starting some time after the project had ended a few challenges had to be dealt with, e.g., personnel availability and possible holes in memories of participants.

3.2.1 Case Selection

Before the case study was conducted several criteria for a fitting case project were agreed upon. Seeing the research was suppose to focus on large-scale development/multiteam systems it was important to find a case where minimum two teams were present in the project, as well as collaborating across the teams. It was also important that the project performed in the case was an agile software development project. Another criteria was that the length of the project had to be suitable, meaning that the project had been ongoing for quite some time. The reasoning behind this was that the amount of data would be larger, and it would be easier to find patterns over a longer period of time.

There were also other criteria which were preferable, but not mandatory. One of these criteria was that it would be desirable if there were several roles within the project as a whole and the project teams. This had to do with the possibility of people with different roles within a project having various experiences from the course of the project leading to valuable data, or put in other words, having different points of view within the project. Another preferred criteria was having a large-scale project with several organisations involved. With several organisations involved there will be different cultures and protocols involved, and therefore a lot of interesting data could surface when comparing the approach of the different organisations.

3.2.2 Data Collection

For the data collection in the exploratory case study focus groups were selected. In these focus groups aspects that are known to be challenging in large-scale agile software development were brought up, as well as general discussion on the topic of large-scale software development. Focus groups are further outlined in L4, and were primarily selected because of their ability to accumulate extensive and valuable amounts of research data.

L3 - Focus group []:

> Focus groups are a form of group interview that capitalises on communication between research participants in order to generate data. Although group interviews are often used simply as a quick and convenient way to collect data from several people simultaneously, focus groups explicitly use group interaction as part of the method. This means that instead of the researcher asking each person to respond to a question in turn, people are encouraged to talk to one another: asking questions, exchanging anecdotes and commenting on each others 'experiences and points of view.' The method is particularly useful for exploring people's knowledge and experiences and can be used to examine not only what people think but how they think and why they think that way.

The interaction in focus groups can be used to achieve seven main goals:

- To highlight the respondents' attitudes, priorities, language, and framework of understanding;
- To encourage research participants to generate and explore their own questions and develop their own analysis of common experiences;
- To encourage a variety of communication from participants, tapping into a wide range and form of understanding;
- To help to identify group norms and cultural values;
- To provide insight into the operation of group social processes in the articulation of knowledge (for example, through the examination of what information is censured or muted within the group);
- To encourage open conversation about embarrassing subjects and to permit the expression of criticism;
- Generally to facilitate the expression of ideas and experiences that might be left underdeveloped in an interview and to illuminate the research participants' perspectives through the debate within the group.

In total three focus groups were conducted, one for each of the organisations involved. The topic that was looked at in the focus groups was "Inter-team coordination and knowledge sharing". The reasoning behind conducting focus groups for each of the organisations, and not performing them on a project level, was to make sure that an openness was achieved,

and that data concerning specific organisations were not lost. As mentioned in section 3.2.1 there might be differences in cultures and methodologies between organisations, and these might not have been present in the focus groups if they were held on a joint basis.

The organisations were asked to provide their most relevant personnel to attend each focus group. In total 8 participants were involved in the focus groups. The participants had several roles in the Omega-project: development managers, scrum masters, (sub-)project managers, developers, delivery managers, functional architects and technical architects. It is important to note that most of the focus group participants were employees in management positions in the project. Most of these participants started as developers before switching to management roles with the course of the project. Because of the availability of personnel and topic in the focus groups no pure developers were present. The distribution of participants in the different focus groups is summarised in table 3.3.

Theme	**Organisation**	**Number of participants**
Inter-team coordination and knowledge sharing	Alpha	2
	Beta	3
	Gamma	3

Table 3.3: Participants in focus groups.

Before the focus group sessions were conducted interview guides were developed, as well as a rough timeline of the project. The timeline was used to freshen the participants' memories about key events. In the focus groups the role of the researcher was to moderate the discussion and take notes. At the start of each focus group the participants were asked to explain their role(s) in the Omega-project. All of the focus group meetings were recorded digitally and transcribed at a later point in time, and whiteboard drawings were documented through pictures. In total the three focus groups resulted in 94 pages of transcribed material. Minutes of each focus group was also sent to each of the corresponding participants for needed information and review.

3.2.3 Data Analysis

After all the research data from the data collection phase was transcribed the data analysis could commence. An important aspect in qualitative data analysis is abstracting the research data to themes and patterns important to the research topic. Several steps were taken to find these themes and patterns. To start of the data analysis the 94 pages of transcribed material were read through twice to get a general overview of the data. After this was done a more thorough read-through was performed where the text was segmented into different themes, e.g., roles, methodologies and general descriptive information. Some of the themes that were identified were then further examined because of their relevance towards the research topic.

Chapter 4

Results

Contents

In this chapter relevant articles gathered on ...

4.1 Introduction and Clarification

Article Name	Author(s)	Year	Keywords
Inter-team Coordination in Large-scale Globally Distributed Scrum: Do Scrum-of-Scrums Really Work?	Maria Paasivaara, Casper Lassenius, Ville T. Heikkilä	2012	Agile Software Development; Distributed Scrum; Global Software Engineering; Inter-team Coordination
Communities of Practice in a Large Distributed Agile Software Development Organization – Case Ericsson	Maria Paasivaara, Casper Lassenius	2014	Communities of Practice; Large-scale Agile Software Development; Scaling Agile
Operational Release Planning in Large-scale Scrum with Multiple Stakeholders – A Longitudinal Case Study at F-Secure Corporation	Ville T. Heikkilä, Maria Paasivaara, Kristian Rautiainen, Casper Lassenius, Towo Toivola, Janne Järvinen	2015	Agile Software Development; Scrum; Large Projects; Release Planning; Software Project Management
Towards a Governance Framework for Chains of Scrum Teams	Jan Vlietland, Hans van Vliet	2015	Agile; Chain of Scrum Teams; Coordination; Priority; Alignment; Predictability

Table 4.1: Summary of articles used in this chapter.

Role	Description of role
Scrum master	
Functional architect	
Technical architect	
Tester	
Developer	

Table 4.2: Team roles present in Scrum teams.

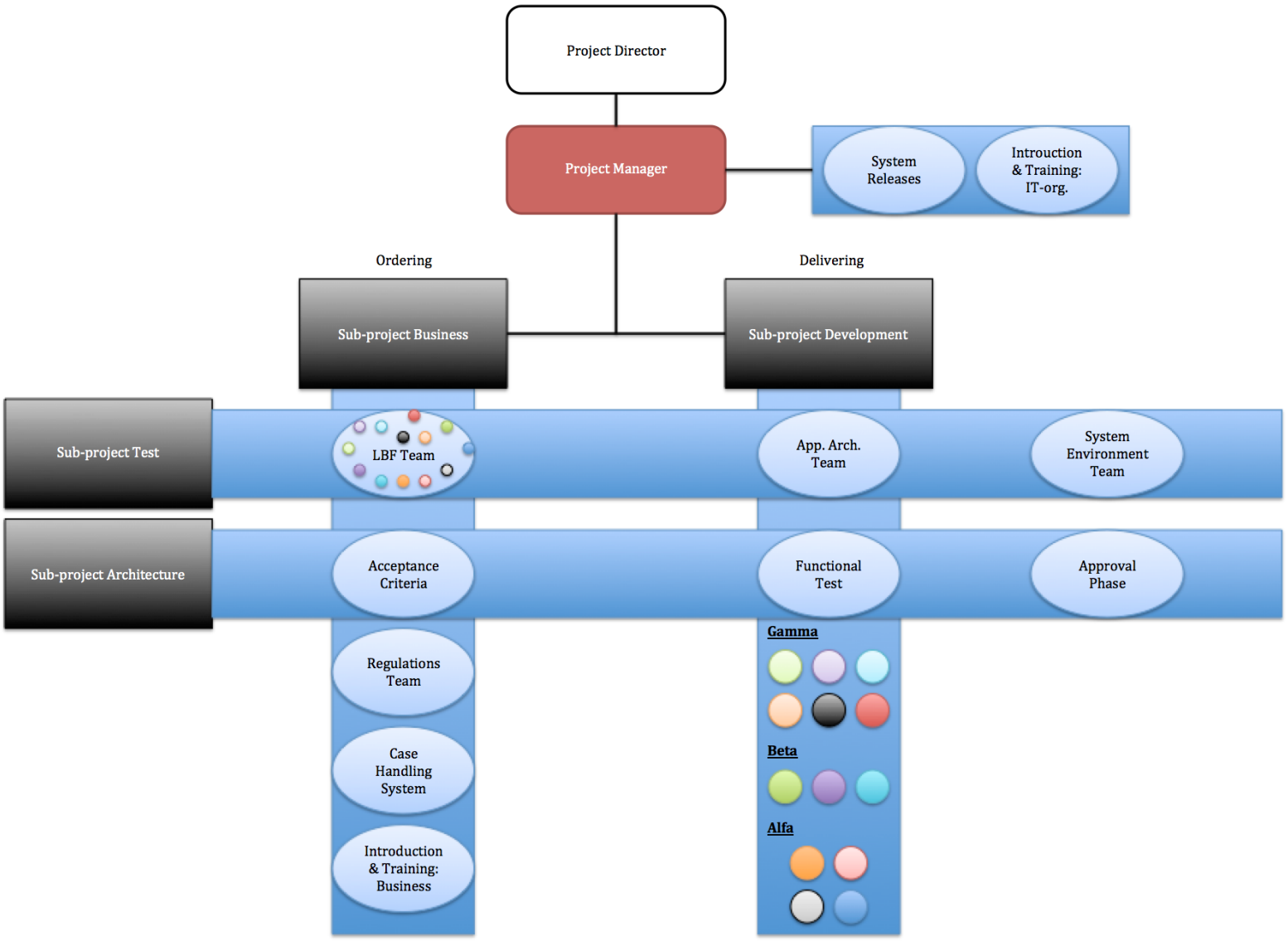

Figure 4.1: Omega-project's organisation.

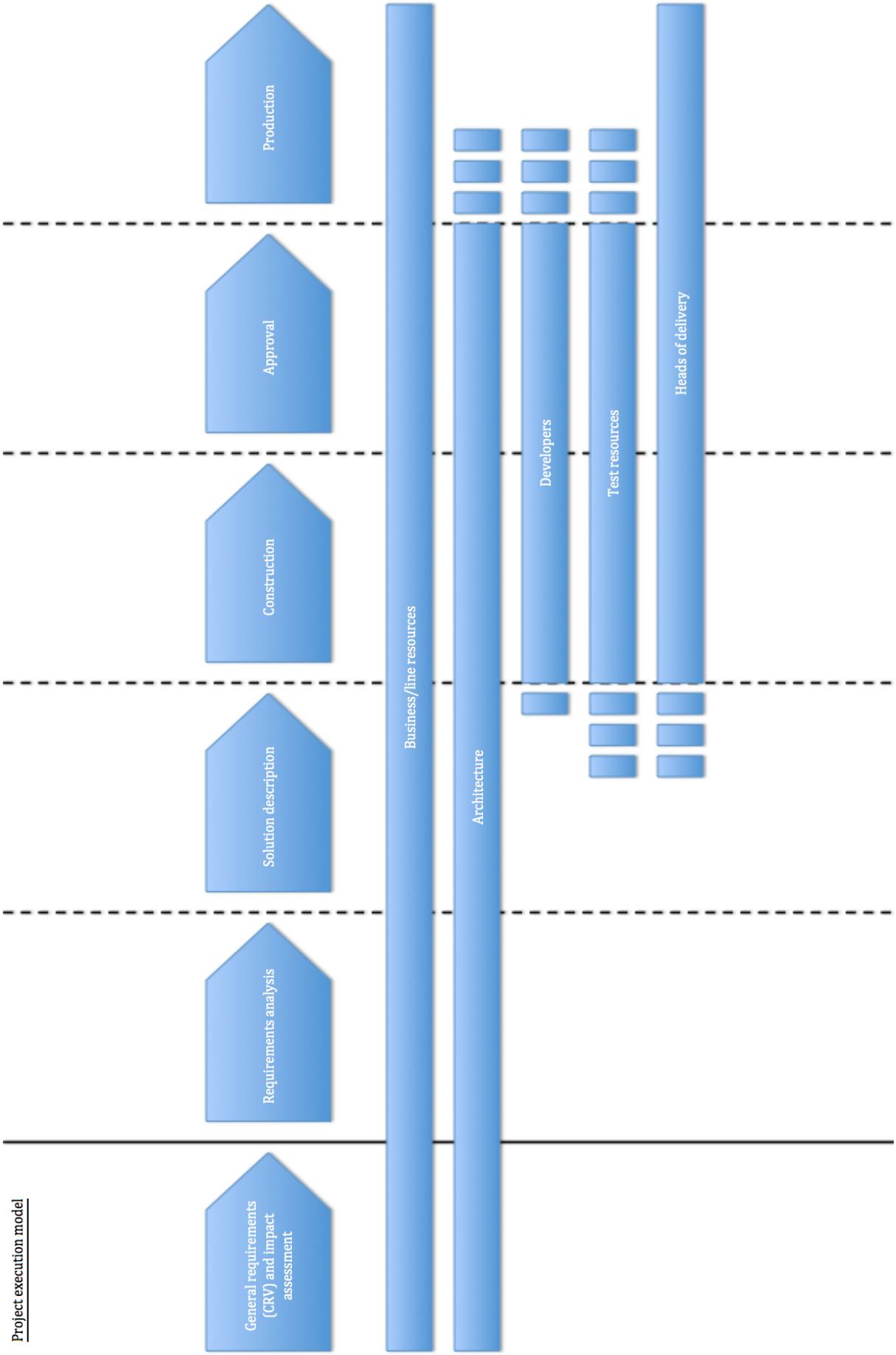

Figure 4.2: Project execution model.

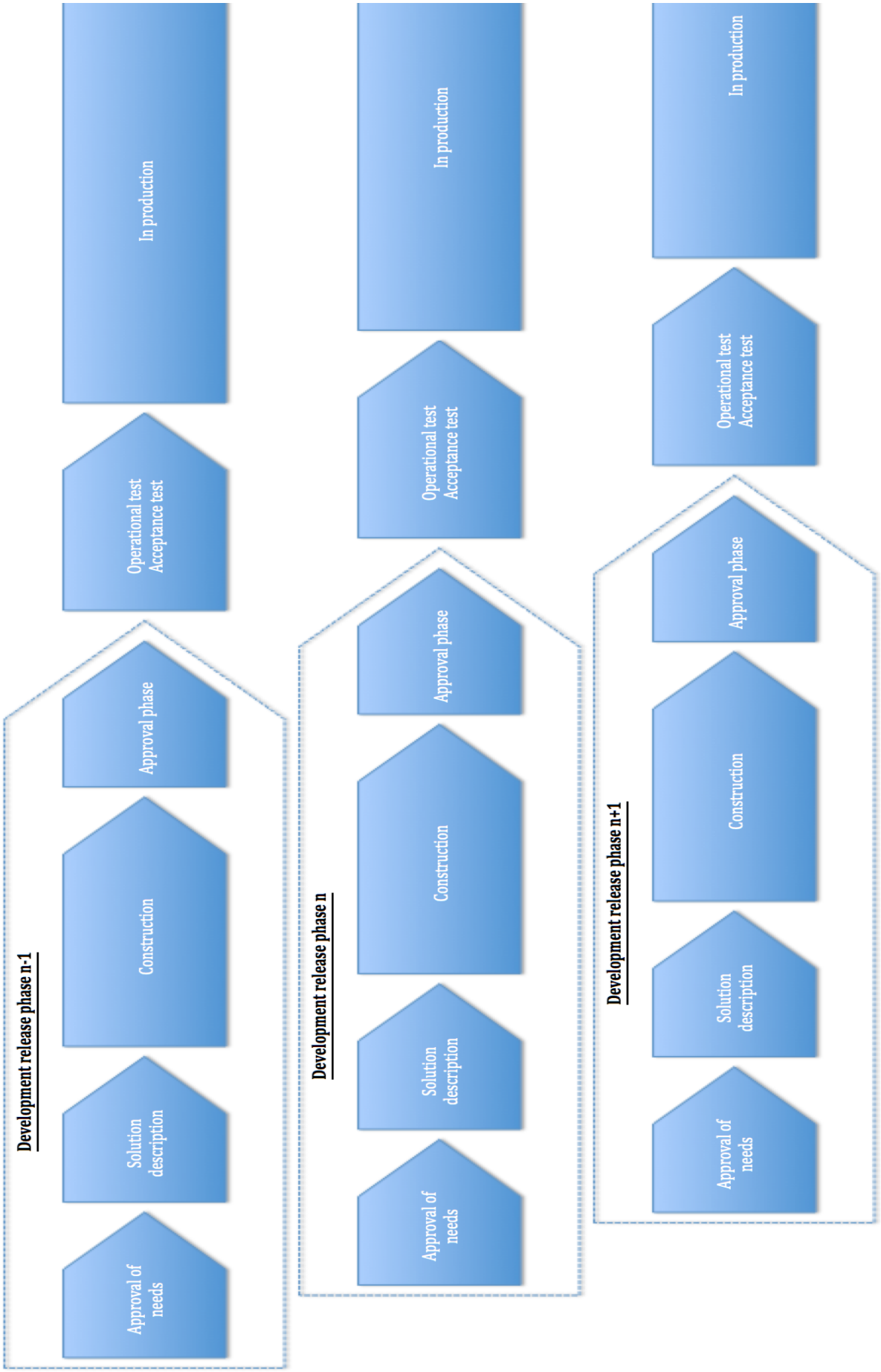

Figure 4.3: Initial development process.

Coordination mechanism	Description of mechanism
Metascrum	A meeting similar to Scrum of Scrums but with less details which was held twice per week. Attending the metascrum was the project leaders and all the sub-project leaders from test, architecture, business and development. A "technical metascrum" was tried, but was shortly shut down after initiation.
Planning day	The planning day was a form of kick-off for each sprint iteration where the project members met up with the project owner. The planning day was performed on three levels: project, organisation (Alpha, Beta and Gamma) and team. A rough sketch of the focus areas and work to be performed in the coming sprint was presented with a distribution towards each of the three organisations by the project owner. After this the organisations distributed the work on their respective teams, and lastly the teams got together separately and worked out a contract with estimated work to be performed which was delivered to the project owner team. Before the planning day commenced the developers also had a "developer forum" where development-oriented information and discussion was carried out. This was however held on an organisation basis, and not across the three organisations.
Demo	Demo presentations were held by all Scrum teams at the end of each sprint iteration where everyone could attend. Each team was allocated approximately 10 minutes. There were also larger demo presentations for the project owner when a new release was finished. Some teams in addition started performing smaller demo sessions within the iterations to get rapid feedback.
Pre-planning day	Before the "planning day" was carried out a pre-planning day was performed. Here typically different types of architects (especially functional architects) and the project owner (as well as some other members of the project owner's team) met to create a rough classification and allocation of work to the different Scrum teams for the coming sprint iteration. The allocated work was listed in a prioritised manner.
	Continued on the next page...

Table 4.3 – continued from previous page

Coordination mechanism	Description of mechanism
Dependency meeting	A meeting held between all Scrum masters from the Alpha, Beta and Gamma teams. This meeting was held on the "Planning day" where the focus was on discovering dependencies across Scrum teams. However, these meetings faded away early on because of the dependencies being discovered and handled elsewhere.
Solution description / "Master plan"	At the start of the Omega-project a larger solution description phase was performed involving a lot of architects (as can be seen from figure 4.2). This lead to a "master plan" for the project and was documented in an issue tracker program called Jira. The "master plan" was continuously altered throughout the course of the development phase as outlined in figure 4.3. In the solution description meetings important aspects were discussed such as coordination across organisations and management of activities. An example of what came out of these meetings was a dependency map of the whole Omega-project, which was in constant change. Part of the solution description meetings were also negotiation and estimation meetings which were important for the contract for each release.
Jira and Wiki/Confluence	Different programs and forums were used for documentation and tracking within the project. In Jira all user stories and epics were located, and different information about the project and current sprint iteration could be seen on different levels, such as project and team level. The dependency map for the whole Omega-project was also located in Jira. Confluence was the main program used as a wiki. Here solution descriptions, team routines, routines across teams, system documentation, check lists, retrospectives, architectural guidelines, functional test etc. were all located.
Open-space	An arena held on a voluntary and need basis, which was used for exchanging experiences. Only used during a few of the releases. Participants suggested the topics beforehand, leading to agendas for open-space sessions.

Continued on the next page...

Table 4.3 – continued from previous page

Coordination mechanism	Description of mechanism
Jabber	Jabber was introduced as an instant messaging service in the Omega-project after being identified as something needed in one of the Open-space sessions. Project members could ask both formal questions, e.g., technical questions, and informal questions or activities, e.g., wine lotteries.
Lunch seminars	Kind of similar to the "open-space" sessions. Typically two to three topics were held by project-personnel on relevant and interesting topics, often regarding themes correlated to the current situation of the project. As with the "open-space" session these seminars were also held on a certain period of the project before fading away.
Front-end meeting	The front-end developers worked with a complex framework called Flex. Because of this a lot of coordination had to be handled between teams working with this framework from all organisations. Therefore, front-end meetings where held were typically the most prominent Flex-developers were present.
Technical architecture forum	At the technical architecture forum all technical architects met up to discuss what was to be done in the coding base to prevent coordination issues. These meetings were slowly fading away because the need was covered in other arenas.
Architecture council	At these gatherings an architecture council listened to all team architects present their respective team's tasks for each sprint iteration.
Business meeting	The business part of the Omega-project was coordinated through meetings where the business architects from Alpha, Beta and Gamma met up with the business unit from the project owner. Here the sprint iteration queue, and the current status of the project and sprint was presented. This meeting was held around one time each week or every other week.

Continued on the next page...

Table 4.3 – continued from previous page

Coordination mechanism	Description of mechanism
Bug-board discussion	The quality assurance unit with its testers had frequent meetings around bug-boards, especially after new releases and around acceptance testing. In the period after a new release these meetings were often held on a daily basis. Here all the bugs were gone through and allocated to the responsible Scrum team in either Alpha, Beta or Gamma.

Table 4.3: Coordination mechanisms used across the whole Omega-project.

Coordination mechanism	Description of mechanism
Scrum of Scrums (SoS)	Scrum of Scrums were meetings held by all organisations (Alpha, Beta and Gamma) ranging from two to three times per week. In these meetings all Scrum masters from the corresponding organisation, as well as project management (project leader, test leader, head technical architect, head functional architect, business leader and development leader). The main goal of the SoSs was to identify and handle obstacles. There were also held a few SoS meetings across organisations to handle potential changes to the contracts.
Technical corner	The "technical corner" was a meeting Beta had in an early stage of the project. It was held on Fridays for about 1-1,5 hour. Here team architects presented important themes for the Beta-members. After a while it was shut down because of lack of interest and topics.
Experience forum	The experience forum was an arena established in the Alpha-organisation for exchanging experiences. Here Scrum masters and the development manager met to discuss topics such as retrospectives, the planning day, and how work was performed by the Alpha-organisation's Scrum teams. It could be seen as a coaching-session with exchange of ideas and thoughts.
Retrospective	Different levels at Alpha (team, solution description and project management), Global retrospektiv testet ut på Beta
Technical architecture meeting	

Continued on the next page...

Table 4.4 – continued from previous page

Coordination mechanism	Description of mechanism
Functional architecture meeting	
Supplier meeting	Alpha
Meeting about queue	Alpha

Table 4.4: Coordination mechanisms used across teams within the specific organisations (Alpha, Beta and Gamma) in the Omega-project.

Mechanism/Aspect	Description
Stand-up	
Board discussion	An important aspect for coordination, discussion and status updates in the project was the frequent use of whiteboards. The stand-up meetings were for instance held around these boards, and on these boards the workload for each sprint iteration was put up and updated as the sprint moved along. The backside of the boards were left open to carry out informal discussion when needed.
Co-location	
Informal communication	
Trust	
Joint coffee break	Alpha (generelt?)
Pair-programming	
Self-organising	
Rotation of team members	
Rotation of team placement	
Alfa/Beta-personnel placed in Gamma teams	
Project management in same location	"Walking around, talking around"
Continuous planning	
3-level hierarchy from product owner	

Table 4.5: Other coordination mechanisms and important aspects.

Chapter 5

Discussion

Contents

In this chapter a closer look...

5.1 Research Question

> “ ... ”

Chapter 6

Conclusion

Contents

6.1 Research Question

> “ ... ”

Chapter 7

Future Work

Contents

In this chapter possible research for the future is highlighted.

7.1 Suggestions for Future Research Focus

Bibliography

[1] Dr. Royce, Winston W., "Managing the Development of Large Software Systems," 1970.

[2] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods - Review and analysis," Tech. Rep. 478, VTT PUBLICATIONS, 2002.

[3] V. O. N. E. Com, "7th Annual State of Agile Development Survey." `http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf`, 2013. [Online; accessed 15-November-2014].

[4] Takeuchi, Hirotaka and Nonaka, Ikujiro, "New New Product Development Game," 1986.

[5] J. Sutherland, "Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies," vol. Vol. 14, No. 12, Dec. 2001.

[6] M. Cohn, "Advice on Conducting the Scrum of Scrums Meeting." `https://www.scrumalliance.org/community/articles/2007/may/advice-on-conducting-the-scrum-of-scrums-meeting`, 2007. [Online; accessed 11-December-2014].

[7] H. Takeuchi and I. Nonaka, *Hitotsubashi on Knowledge Management*. Wiley, 2004.

[8] T. W. Malone and K. Crowston, "The interdisciplinary study of coordination," *ACM Comput. Surv.*, vol. 26, pp. 87–119, Mar. 1994.

[9] D. E. Strode, S. L. Huff, B. Hope, and S. Link, "Coordination in co-located agile software development projects," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1222 – 1238, 2012. Special Issue: Agile Development.

[10] A. H. Van De Ven, A. L. Delbecq, and R. Koening Jr., "Determinants of Coordination Modes within Organizations," *American Sociological Review*, vol. 41, no. 2, pp. 322–338, 1976.

[11] J. Child

[12] T. Dingsøyr and N. B. Moe, "Research challenges in large-scale agile software development," *ACM SIGSOFT Software Engineering Notes*, vol. 38, p. 38, Aug. 2013.

[13] T. Dingsøyr, T. E. Fægri, and J. Itkonen, "What is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development." 2013.

[14] J. Schnitter and O. Mackert, "Large-scale agile software development at sap ag," in *Evaluation of Novel Approaches to Software Engineering* (L. Maciaszek and P. Loucopoulos, eds.), vol. 230 of *Communications in Computer and Information Science*, pp. 209–220, Springer Berlin Heidelberg, 2011.

[15] I. O. Robert L. Nord and P. Kruchten, "Agile in distress: Architecture to the rescue." 2014.

[16] D. E. Strode, B. G. Hope, S. L. Huff, and S. Link, "Coordination effectiveness in an agile software development context.," in *PACIS* (P. B. Seddon and S. Gregor, eds.), p. 183, Queensland University of Technology, 2011.

[17] J. Mathieu, M. T. Maynard, T. Rapp, and L. Gilson, "Team Effectiveness 1997-2007: A Review of Recent Advancements and a Glimpse Into the Future," *Journal of Management*, vol. 34, pp. 410–476, June 2008.

[18] D. E. Hyatt and T. M. Ruddy, "An examination of the relationship between work group characteristics and performance: Once more into the breech," *Personnel Psychology*, vol. 50, no. 3, pp. 553–585, 1997.

[19] C. De O. Melo, D. S. Cruzes, F. Kon, and R. Conradi, "Interpretative case studies on agile team productivity and management," *Inf. Softw. Technol.*, vol. 55, pp. 412–427, Feb. 2013.

[20] N.-D. Anh, D. S. Cruzes, and R. Conradi, "Dispersion, coordination and performance in global software teams: A systematic review," in *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '12, (New York, NY, USA), pp. 129–138, ACM, 2012.

[21] T. Dingsøyr and Y. Lindsjørn, "Team performance in agile development teams: Findings from 18 focus groups," in *Agile Processes in Software Engineering and Extreme Programming* (H. Baumeister and B. Weber, eds.), vol. 149 of *Lecture Notes in Business Information Processing*, pp. 46–60, Springer Berlin Heidelberg, 2013.

[22] E. Salas, D. E. Sims, and C. S. Burke, "Is there a " Big Five" in Teamwork?," *Small Group Research*, vol. 36, pp. 555–599, Oct. 2005.

[23] K. S. Khan, R. Kunz, J. Kleijnen, and G. Antes, "Five steps to conducting a systematic review.," *Journal of the Royal Society of Medicine*, vol. 96, pp. 118–121, Mar. 2003.

[24] B. J. Oates, *Researching Information Systems and Computing*. Sage Publications Ltd., 2006.

[25] L. A. Goodman, "Snowball sampling," *Ann. Math. Statist.*, vol. 32, pp. 148–170, 03 1961.