Gottfried Wilhelm
Leibniz Universität Hannover
Faculty of Electrical Engineering and Computer Science
Institute of Practical Computer Science
Software Engineering Group

Identification and Analysis of Practices for Organizing Development Teams

Master Thesis

in Computer Science

by

Elefteria Merkohitaj

Examiner: Prof. Dr. Kurt Schneider
Second Examiner: Dr. Jil Klünder
Supervisor: Nils Prenner

Hannover, 26.04.2021

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Master Thesis selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 26.04.2021

_____

Elefteria Merkohitaj

Abstract

Recent research in software engineering teams suggests that different organizational aspects of teams, such as teamwork and collaboration, coordination and communication have been extensively studied. Still, relatively little research has been carried out on the generalizability of this knowledge. This thesis addresses this research gap by systematically reviewing the current literature, with the aim of identifying and analyzing the practices used by teams to organize themselves. A thematic analysis found team organizational structures corresponding to small, middle-sized and large teams. The arrangement of these structures suggested that especially middle-sized and large teams use agile principles at the team level, whereas traditional principles are used to facilitate the inter-team coordination by the management level. These structures were further supported by practices related to organizational aspects, such as communication, coordination, collaboration and decision-making. Finally, these results contribute in the formulation of a model, which should help teams evaluate their organizational needs and further, provides guidelines in the form of structures and practices.

Zusammenfassung

Aktuelle Forschungsarbeiten zu Software-Engineering-Teams weisen darauf hin, dass verschiedene organisatorische Aspekte von Teams, wie z. B. Teamarbeit und Kollaboration, Koordination und Kommunikation, ausgiebig untersucht wurden. Dennoch wird die Organizationsstruktur in ihrer Gesamtheit noch wenig erforscht. Diese Forschungslücke wird in dieser Arbeit durch eine systematische Literatursuche der aktuellen Forschungsliteratur adressiert, um die Praktiken, mit denen sich Teams selbst organisieren, zu evaluieren und zu analysieren. In einer thematischen Analyse wurden Team-Organisationsstrukturen identifiziert, die kleinen, mittelgroßen und großen Teams zugeordnet werden können. Die Zusammensetzung dieser Strukturen deutet darauf hin, dass vor allem mittelgroße und große Teams agile Prinzipien auf der Teamebene nutzen, während traditionelle Methoden verwendet werden, um die teamübergreifende Koordination durch die Managementebene zu erleichtern. Diese Strukturen wurden außerdem durch Praktiken unterstützt, die sich auf organisatorische Aspekte wie Kommunikation, Koordination, Zusammenarbeit und Entscheidungsfindung beziehen. Schließlich tragen diese Ergebnisse zur Formulierung eines Modells bei, das Teams helfen soll, ihre organisatorischen Bedürfnisse zu bewerten und darüber hinaus Richtlinien in Form von Strukturen und Praktiken bereitzustellen.

Contents

1 Introduction 1
1.1 Problem statement 2
1.2 Solution Approach 2

2 Related Work 5
2.1 Teams in Software Engineering 5
2.2 Organization of Development Teams 7

3 Foundations 11
3.1 Terminology and Context 11
3.2 Guidelines for Development Teams 13
3.3 Systematic Literature Review 14

4 Methodology: A SLR 19
4.1 Planning Phase 20
4.1.1 Step 1: Research Questions 20
4.1.2 Step 2: Database Selection 21
4.1.3 Step 3: Search String 21
4.1.4 Step 4: Inclusion and Exclusion Criteria 22
4.2 Execution Phase 23
4.3 Data Collection 25
4.4 Data Analysis 25

5 Findings 29
5.1 Data Analysis 29
5.2 Research Questions 30
5.2.1 RQ1: Structures for team organization 31
5.2.2 RQ2: Practices for team organization 41

6 Discussion 53
6.1 Organizational team structures and practices 53
6.2 Implications for practice 56
6.3 Generalizability, Limitations and Threats to Validity 56

7 Model 59
7.1 Model Presentation 59
7.2 Proposed Team Structure Templates 63
7.2.1 Team structures A1 to A4 63
7.2.2 Team structures T4 to T1 66

8 Summary and Future Work 73

A Review Protocol 75
A.1 Search String 75
A.2 Research Questions 75
A.3 Inclusion Criteria 75
A.4 Exclusion Criteria 76
A.5 Database Selection 76
A.6 Search Process 76

B Data Extraction Form 77

C Selected Primary Studies 79

Chapter 1

Introduction

Team organization is the act of coordinating or managing the activities of a group of people by establishing an orderly, functional, or coherent structure [67, 68]. Team organizations are a topical area of research, especially in the social sciences, such as sociology [98, 47] and management science [36]. Lunenburg [47] describes two forms of organizations: mechanistic and organic organizations. On the one hand, mechanistic structures are characterised by a centered control of processes and resources, formalized procedures and practices, where the chain of command is clearly defined. On the other hand, organic structures are characterised by decentralisation, low specialisation and less control, allowing a flexibility for the team to be part of the decision making process.

Many teams could find their needs reflected in the aforementioned characteristics of these structures. For instance, it is widely accepted that small teams are easier to handle because of their size. Therefore, one might encounter smaller teams with a flat structure, with few or no levels between sta and executives. These teams are apt to operate independently and have a short chain of command, showing characteristics of an organic team structure. However, managing a team becomes more challenging the more the team size increases. In such cases, the solution is often found in control-centered and hierarchical structures. [39]

The same organisational structures are applied in development teams inside software development organization. Based on which software development model is used inside the organization, teams combine a traditional, agile or hybrid model with well established team organization structures. For instance, Chau et al. [14] a rms that more traditional approaches, like the Waterfall model, promote usage of role-based teams. This team structure shows similar characteristics to the hierarchical team organisation, where each of the teams contains members of the same role. Other researchers, Hoda et al. [35] and Moe et al. [53], identify teams in an agile environment as self-organizing teams. This allows the teams to have both individual and

team autonomy in a change-driven development.

However, current research has shown that more software development organizations are transitioning to agile [5]. Paasivaara [70] states that the agile transformation requires changes in the organizational structure of the software project teams, especially when applying agility at the large-scale. For instance, a small agile software project can scale up rapidly, involving more people or even more teams. As the project grows, the responsibilities divide into specialized people or teams. This can force the team structure to fall back into having defined roles as in the mechanistic structure defined above. At this point, teams still want to benefit from the independence and the flexibility to respond to change, as well as having some control to managing the rapid growth.

A summary of prior research by Brick et al. [9] suggests that large and complex development projects can benefit by the combination of flexible structure of agile teamwork and the structured, plan-driven coordination of traditional project management into hybrid approaches. However, a hybrid approach allows the teams to design their team structure and coordination based on their own needs. Therefore, there is no fixed structure with which a hybrid team structure or coordination can be identified.

1.1 Problem statement

The research by DeFranco and Laplante [22] on software development teams found that software development teams have been extensively studied as well. This research provides an important opportunity to advance the understanding of organization inside software development teams, regardless of the traditional, agile or hybrid approach they implement. An overview of reported experiences regarding team structures in these settings can be useful to both researchers and practitioners. There is no single method from which every software team can benefit. In spite of that, the gained knowledge can serve as a foundation, that guide teams to shape the organisational structure based on their needs. Furthermore, it can help researchers to analyse and further investigate how the structures can be improved in order to support a better teamwork and coordination.

1.2 Solution Approach

The current state of the research on organization of software development teams suggests that a systematic understanding of how software development teams organize themselves is still lacking. This thesis addresses the lack of this overview by conducting systematic literature research for identifying the practices of organizing development teams with which an agile, traditional or hybrid organization is implemented. This literature review focuses its

attention to the development teams organizational structures. Moreover, by following a review plan we deliver comprehensive and transparent research and answer our research questions clearly.

With this review, this thesis investigates the following research questions:

- RQ1: What development team organizational structures exist in software development?

  With this research question, this thesis investigates and gathers team organizational structures from the literature. The focus is to analyse how project development teams are internally organized and if these structures show are any recurring patterns.

- RQ2: What are the practices for the organization of development teams?

  This research question should give insight into the actual practices or activities that software development teams use to support their organization, such as coordination, communication, collaboration and decision-making practices.

Finally, the team structures, practices and methods are further analysed to build a model, which can guide software development organizations and development teams to find the combination best suited for their needs and their development environment. With this model, this thesis seeks to answer the following research question:

- RQ3: How can these structures and practices be combined with traditional, agile or hybrid models?

The research questions are elaborated in the remaining seven chapters of this thesis. Firstly, this thesis provides an overview of the current literature on software development teams in Chapter 2, defines the terminology and context of this research in Chapter 3. Secondly, Chapter 4 introduces the research methodology and Chapter 5 presents the findings of the systematic literature review. Next follows the discussion of the findings in Chapter 6. Finally, this thesis presents the model for organizing software development teams in Chapter 7 and conclude this thesis by providing a summary and directions for future work in Chapter 8.

Chapter 2

Related Work

In general, teams have been the focus of numerous studies. Also in the field of Software Engineering teams are a topical area of research [21, 22]. However, this literature consists of single case studies or experience reports, which document valuable information about software development teams. Hence, this experience cannot be applied to other settings. Relatively little research has been carried out on the generalizability of these findings.

A recent mapping study conducted by DeFranco and Laplante [22] suggest that the current literature on software engineering teams provides an important opportunity to advance the understanding of development teams in general. Whereas software engineering teams are extensively studied, DeFranco and Laplante's themes further suggest, that there is a current paucity of research focusing specifically on synthesizing the knowledge about teams' organization. The aim of this chapter is to introduce the context in which teams have been investigated and give a short review of the current state of knowledge regarding the organization of software development teams.

The chapter is composed of four sections. First, section 2.1 will provide a general overview of teams in software engineering. Next, a summary of related work on the aspects of organization in software development teams will then be presented in Section 2.2. The last Section 3.2 will attempt to present previous research, which has intended to provide guidelines for choosing best suited methods, processes and frameworks based on specific project and team needs.

2.1 Teams in Software Engineering

Broadly speaking, teams in software engineering are typically studied in the context of a software development project. Software development projects can be of different sizes, where the size is mostly defined by the complexity of the software product to be developed. Thus, software development projects are divided into small, middle and large projects. The same applies to the

teams which build the software product. The literature review conducted by Keshta and Morgan in [39] suggests that teams with less than 15 members are considered *small teams* and those with 15 to 25 members *medium teams*, whereas *large teams* may have more than 25 members.

In 2004, Sawyer [82] studies teams regarding the software development methods they use. He mentions *three archetypes* of software development teams; *sequential*, *group* and *network archetypes*, based on how they fit into the traditional, iterative or agile methods respectively. The first archetype, the *sequential* one, represents a hierarchical, role-based and formal team organization structure, where the team follows a linear, task-driven and well-planned software development practice, often observed in the settings of traditional waterfall development. Next, the *group archetype* shows characteristics of a social structure which is based on collaboration, norms and regulations and where the collective skills and weaknesses of the members of the group are taken into consideration for the task completion. This is a typical structure of the first forms of iterative software development methods, such as the spiral model. In the last one, namely in the *network archetype*, the team is more concentrated on the product than on the processes. The social structure is defined here by people's connections, which give shape to the less formal development processes. Such characteristics are observed in iterative development as well as in teams implementing agile development methods.

In a more recent paper, Sharp et al. [89] consider the physical and temporal distance of teams members and mention *co-located* and *distributed teams*. While in a co-located team all team members share and work on the same physical work environment, a distributed team can be spread geographically and temporally. Sharp et al. [89] recognize in this context 'three main flavours' of distribution: *distributed teams*, *dispersed teams* and *hybrid teams*. Whereas distributed teams have sub-teams in different locations, in dispersed teams each team member is located in a different place. Hybrid teams are a combination of both distributed and dispersed teams.

Further, the set of teams working together in the same project is called the *project team* [6, 24, 54]. In a distributed project environment, the teams near the home organization are called *near-shore teams*, the ones in the home organization are called *on-shore teams* and the farthest are called *o -shore teams* [41, 77, 97]. When all these teams answer to the same organization, then they can also be called *remote teams* [86, 89, 90]. Moreover, in order to reduce development cost, software companies engage external work-force for performing project work, such as, in the case of software development projects, for the development or testing activities. The outsourced work-force is a *contracted team* which, on the one hand, is attached to the core project team to achieve the project common goal and, on the other hand, answers to its own organization [100, 93]. In this case, the whole project

team can be called a *virtual team* [55, 56, 74].

2.2 Organization of Development Teams

One of the earliest works on software development teams is the discussion of team archetypes by Sawyer [82], which is already described in the section above. It is one of the first attempts to summarize team types among different software development environments. These archetypes did not only put the teams in the traditional and iterative software development context, but also described aspects of organization such as decision-making, communication and collaboration. At the very beginning of the agile methods, this almost two decades old argumentation has correctly recognized substantial aspects of team organization.

However, in the last five years there have been several attempts to synthesize the current state of knowledge on software development teams. Even though these reviews do not directly address the organization of development teams, many of them have investigated and summarized some of the aspects of organization. This thesis identified four additional publications which investigated the following aspects: decision-making [17], communication [21], team size and project domain [39], and team autonomy [1]. An important fifth review [22] aimed to fundamentally synthesize research performed in the area of software engineering teams and provided valuable insight into this part of the literature.

In 2016, Cunha et al. [17] synthesize empirical studies on 'decision-making phenomenon in the software project management from a naturalistic perspective'. Among others, their review suggests that some of the factors influencing the decision-making process are the agile development practices, stakeholders involvement and communication. While the self-managing teams in the agile development practices challenge the traditional, controlling decision authority, the stakeholders closely involved in the project tend to dominate it by aligning the decisions with their business objectives. Further, the communication decisions are discussed as the competencies of project managers to decide what, who, how and when to communicate.

In addition, DeFranco and Laplante [21] analyze the literature to give insights into the current state of the software development team communication research and to point out the gaps of research in this field. First, their review showed that the most active communication research areas in software development are global software development, project effectiveness and effective teamwork. Next, they speculate that the research gaps may be found in the least prevailing research areas, such as agile processes, shared understanding and generic software engineering processes. However, they do not exclude the possibility that there are still research gaps present in the most prominent research areas listed above and further

speculate that the gap may 'point toward the rigor of research' in those areas. At the end, they summarize the major and common findings about communication in software engineering teams. Among these findings, two are the most outstanding: the emphasis on tools and strategies to improve communication and project performance and the ideal team dynamics and composition to improve communication for e□ective teamwork.

In a comparison literature review of traditional and agile methods regarding team size and project domain, Keshta and Morgan [39] conclude that there is a relation between project size and team size: the bigger the project, the larger the team. Further, these two factors seem to drive also the methodology size in the same proportion: the bigger the project and team size, the more heavier the methodology. Their argumentation implies that a heavy-weighted methodology is represented by a good organization of work, planning, documentation and high quality control, as typical characteristics of the more traditional, plan-based methods such as waterfall, used in projects with large teams. Further, they categorize agile methods as light-weight and a□rm the widely accepted statement, that agile thrives in small teams with 15 to 25 members. Nevertheless, with the evidence of implementing agile in large projects found in the literature, the authors do not rash to generalize that agile methods cannot be used with large teams. This implies that the application of agility in large-scale projects is still di□cult.

In the most recent literature review, Acharya and Colomo-Palacios [1] provide insights to self-organization of agile teams from the literature. Topics they address include the benefits and challenges of autonomous agile teams, the way these teams organize themselves as well as possible facilitation strategies for enforcing autonomous agile teams. While they describe findings from the selected literature which relate to the topics above, they fail to disclose an aggregation of their findings and to directly answer the research questions. However, worth mentioning key information regarding the organization of autonomous agile teams is that team members and leaders face challenges while shifting to new agile roles, which require a change on their mindset. Other valuable insights include the emphasis to the team leader's role in supporting and empowering self-organization, as well as communication increase inside and across teams for facilitating learning and taking ownership of tools and methods.

In 2018, DeFranco and Laplante [22] conducted a thorough literature research in the area of software engineering teams. They a□rm that, in general, software engineering teams have been a prevailing area of study. However, they recognize the necessity of an overview of the knowledge from this area and provide a fundamental assessment of the research on software engineering teams. With a keyword content analysis, DeFranco and Laplante classify this research into the following seven categories, listed by descending prevalence: teamwork/collaboration, process/design,

coordination, global/OSS[1], tools, project improvement, communication and agile. Among the most researched team types, the researchers identified traditional, global, pair programming, agile and OSS teams, also presented in decreasing prevalence.

Broy and Kuhrmann [13] view teams 'as a means of structuring personnel' and as a consequence, state that teams need clear roles and competencies. For that, development teams are embedded in the project organization and work together to achieve the common goal. Moreover, they present two forms of development team structures in software projects: hierarchical and democratic team organization. The hierarchical organization is build in a top-down approach and has the form of a pyramid. While managers, such as project managers, occupy the top of the structure, the development teams are located at the bottom of the organizational structure. In contrast, the democratic team organization moves from the strict reporting lines of the top-down approach and applies instead collective group decisions and responsibility. In this structure, the levels of organizations are flattened and the management style is based on leadership and collaboration.

Traditional versus Agile Organization

The hierarchical and democratic team organizations by Broy and Kuhrmann [13] described above are similar with the findings of several publications [2, 60], which have compared traditional and agile software development methods. These authors describe the team organization in traditional approaches as pre-structured, where team members are specialists, who work plan-oriented and have specific roles. These roles are assigned 'based on the skill levels of each individual' [2]. Overall, the organization culture in traditional approaches, similar to the hierarchical organization, is based on command and control [2, 60]. In contrast to this top-down approach, projects implementing agile methods show similarities with the democratic team organization. Agile teams are characterized by self-organizing teams, where the project management is based on leadership and collaboration. These teams are described as cross-functional, where team members have different skill sets. In addition, agile teams are known for the frequent involvement and participation of the customer in the development process. [2, 60] Table 2.1 summarizes team organizational aspects for traditional and agile teams, based on the descriptions above.

[1] Open-source software

Aspect	Traditional Methods	Agile Methods
Project management and organization	Based on command and control	Based on leadership and collaboration
Team organization	Pre-structured teams	Self-organizing teams
Team members	Specialists working in silos	Agile, knowledgeable, cross-functional
Customer involvement	Low, passive	on-site, active/proactive

Table 2.1: Project and team management in traditional and agile settings [2, 60]

Chapter 3

Foundations

The aim of this chapter is to introduce the reader to background information on the terminology and methods used in this study. The terminology and the context of this thesis are elaborated in Section 3.1. Section 3.2 presents a short summary of prominent research on guidelines for software development teams. A brief description of the research methodology, a systematic literature review, is presented at the end of this chapter.

3.1 Terminology and Context

To better understand the research focus of this research, it is necessary to clarify the terminology used to report facts and findings on this thesis. Some terms, being used in different research fields, carry different meanings, depending on the context in which they are applied. A typical example would be the term *development*, which is not only used within the field of Software Engineering, but also as a general term in the production of new goods and services. Nevertheless, in the context of this thesis, this term is used in association with software development. Furthermore, it is important to elaborate the aspects of *organizing* that this thesis investigates. In its most general sense, organizing refers to the act of arranging something into a structured whole [67]. Throughout this thesis and relating to teams, organizing is the act of coordinating or managing the activities of a group of people by establishing an orderly, functional, or coherent structure. Previous research has studied several key aspects of organizing software development teams [53, 102, 88, 9], such as communication, coordination, collaboration and decision-making authority, which give form to the structure of a team. This thesis also considers these four aspects for the investigation of the organization in development teams. However, there is still some ambiguity with regard to these terms, especially for communication and coordination. Definitions of each of the four aspects of organization used in this thesis are listed in Table 3.1.

In order to achieve their goals, team members have to constantly communicate to and exchange thoughts and ideas with their peers, coordinate work and tasks, work together to solve issues or dependencies and make important decisions throughout their project. It is clear that these activities are dependent on one another. For instance, to coordinate work and tasks, team members have to somehow communicate with one another. During the process of decision making, the communication of thoughts and ideas is also crucial. From this perspective, communication seems to be the center of these activities, and consequently, can boost or inhibit coordination and collaboration.

Communication. In general, the Oxford English Dictionary [15] defines communication as 'the transmission or exchange of information, knowledge, or ideas, by means of speech, writing, mechanical or electronic media, etc.;[...]'. Sharp and Robinson similarly define in [88] that, in the context of software development, communication 'takes place when two or more people exchange information or knowledge through verbal or non-verbal means'. Both definitions are used in this thesis when referring to communication in software development teams. Having said that, this thesis defines communication as the transmission or exchange of information, knowledge or ideas by means of speech, writing or signs. More specifically, the aim of this research is to identify how software development team members communicate with one another during their daily work, which communication channels they use and how it affects or defines the overall team organization.

Coordination. When talking about coordination, one can encounter different points of view. For example, Sharp and Robinson [88], also supported by a very similar definition of Malone and Crowston [48], refer to coordination as 'the process of managing dependencies among activities'. While Sharp and Robinson see the coordination as a process, Nguyen-Duc et al. [64] refer to the team coordination explicitly as a set of activities, and thus define it as 'activities required to maintain consistency within a work product or to manage dependencies within the workflow'. Still, both definitions similarly state that coordination somehow handles or manages dependencies, as a process or set of activities. It can be agreed that both meanings are important in the process of software development.

To follow a process means following predefined procedures or regulations. In software engineering that would be the case, when a team follows a specific development process or model to deliver a software product. For instance, practicing Scrum means following the activities such as sprint planning, daily stand-up meetings, sprint review and retrospective for each iteration. Consequently, the formal activities of these processes, models or frameworks can define a part of how a software development team coordinates its work. In contrast, defining coordination as activities makes the term less formal. In this case, the informal activities, such as peer-to-peer exchange of information and ideas, indirectly support the success of the

formal coordination.

Overall, communication seems inseparable from coordination.Considering the fact that communication in traditional development methods is defined as rather formal, whereas agile approaches are characterized by a more informal communication [2], this thesis recognizes two types of organizations: horizontal and vertical organization. Horizontal organization refers to the informal, peer-to-peer exchange of information, ideas and thoughts, whereas vertical organization is more formal and is defined by regulations in the team level or the different levels of an organization. In other words, the horizontal organization is seen here as the informal communication and coordination at the team level, while the vertical organization is represented by the formal communication and coordination occurring from bottom-up or top-down through the levels of the organization. Having said that, it can be speculated that in traditional settings, teams are characterized by a rather vertical organization, whereas agile teams practice a more horizontal organization.

Terminology	Definition
Organizing	is the act of coordinating or managing the activities of a group of people by establishing an orderly, functional, or coherent structure. [67, 68]
Communication	is the transmission or exchange of information, knowledge or ideas by means of speech, writing or signs. [15, 88]
Coordination	is the process of managing dependencies among activities. [88]
Collaboration	takes place when two or more people are working together on a task. [88]
Decision-making	is the act or process of making decisions. [20]

Table 3.1: Definition of the key terms

3.2 Guidelines for software development teams

In a study of finding balance between agile and plan-driven methods, Boehm and Turner [11] propose 'a tailorable, risk-based approach' for combining both methods in software development. The suggested approach is based on 5 steps, which make use of risk for guiding practitioners in choosing between agile and plan-driven methods or a combination of both for their software development project. A polar chart with five axes represented the factors which distinguish these two methods: size or the number of personnel, criticality of the system to be developed as the loss due to impact of defects,

dynamism as a measure of requirements change per month, skill levels of the personnel and the culture, as the percent of thriving on chaos versus order. These factors should aid the practitioners to reflect the characteristics of their projects, in order to better judge their risks while following the 5-step approach. Three out of 5 factors, that is personnel, size and culture, represent the human aspects of a project which shows that the human factor plays a significant role in the methodology decision.

Since the publication of this model in 2003, tailored methods have been used and discussed extensively in the literature. Reports from practitioners and researchers contain valuable insights about which combinations work best and how to mitigate any obstacles while combining agile and traditional methods. This thesis makes use of both, the findings from the literature review as well as the approach presented above, and formulates an approach, which should guide development teams to choose the development method and team structure that best fits their needs.

3.3 Systematic Literature Review

Kitchenham et al. [3] propose a systematic literature review as a useful method for reviewing evidence on a particular study field. A systematic literature review is 'a form of secondary study that uses a well-defined methodology to identify, analyse and interpret all available evidence related to a specific research question in a way that is unbiased and (to a degree) repeatable' [3, p. vi]. In the literature the studies containing the necessary evidence are identified during the execution of the review and are also called *primary studies*. The systematic literature review itself is a *secondary* study. While a primary study empirically researches and studies a specific research topic, a secondary study reviews all primary studies on a specific research topic with the aim of synthesising evidence related to this research topic.[3]

Moreover, systematic literature reviews are important studies. They are mostly conducted to summarize empirical evidence on a specific topic of interest. Other reasons for conducting a literature review are also to identify a research gap for proposing research topics for further investigation or to suggest totally new research topics. [3] By undertaking a systematic literature review, this thesis aims to identify and analyse practices for organizing software development teams. The gained knowledge should be used to build guidelines for practitioner teams, who have di□culties with their current organizational structure and need improvement or a total reform for their team.

The method of the systematic literature review suggested by Kitchenham et al. [3] includes a 3-phase process and several activities. The three phases are: planning the review, conducting the review and reporting the review. A list of the activities per phase are listed in the Table 3.2 below.

Planning the Review

Especially the activities started during the planning phase may involve iteration and they might require refinement during the execution phase. For example, the most common pre-review activities include defining the research questions and preparing a review plan which contains the most important review procedures and controls possible limitations such as researcher bias. The review plan also contains the research questions, inclusion and exclusion criteria for the literature selection, research databases and search and data extraction strategy, among other things. These activities can be piloted and may be refined several times before undertaking the proper review. [3]

Review Phase	Activities per Phase
Planning the Review	Identifying the need for a review Commissioning a review* Specifying the research question(s) Developing a review protocol Evaluating the review protocol*
Conducting the Review	Research identification Primary studies selection Study quality assessment Data extraction and monitoring Data synthesis
Reporting the Review	Specifying dissemination mechanisms Formatting the main report Evaluating the report*

Table 3.2: List of the sub-activities in each phase of the systematic literature review. The activities marked with * are not mandatory.

In addition to the research questions, the definition of study selection criteria is one of the most important pre-review activities as it provides the opportunity to reduce the likelihood of selection bias. These criteria are divided into inclusion and exclusion criteria and should be defined based on the research question(s). To increase the reliability of inclusion decisions, the study selection criteria can be refined during the search process if necessary. [3]

Moreover, the search strategy is represented by the definition of a set of terms, which can be used during the execution phase for the research identification. The set of terms are derived from the research questions of the review and should contain also synonyms, abbreviations and alternative spellings of the terminology used in the context of study. These terms can then be transformed in sophisticated search strings by using Boolean

operators AND and OR. Additionally, the search string may have to be adjusted to fit to the digital library's search syntax. There are several digital libraries relevant to software engineering: IEEE Explore, ACM digital library, Google Scholar, ScienceDirect, Citeseer, SpringerLink etc. [12].

Conducting the Review

Study Quality Assessment is a method to assess the quality of the primary studies and has mostly the form of quality checklists. These checklists contain questions for 'assessing the extent to which articles have addressed bias and validity' [3, p. 21]. Among others, this activity should facilitate the study selection by providing more detailed selection criteria and weighting the importance and quality of individual studies. By doing so, the quality assessment further guides the interpretation of findings and recommendations for future research. Kitchenham et al. [3] accumulate two lists of quality-check questions, one for each quantitative and qualitative studies and suggest using them or selecting subsets of them in the context of their own study and by considering the specific research questions.

Another activity which begins during the definition of the review plan and becomes more sophisticated during piloting, is the data extraction, which is recorded in the plan through data extraction form(s). By defining such forms, the SLR aims to accurately record the knowledge extracted from the primary studies and to reduce the possibility of bias during the extraction. These forms need also to be piloted to assess issues such as the completeness as well as clarity of the instructions or questions. Kitchemham et al. [3] further suggest that two or more researchers should independently perform the data extraction.

The final activity in the second phase of the review is data synthesis. Data synthesis is the process of analysing the collected data and summarising the results. How the data synthesis is to be performed should also be roughly specified in the review protocol. Moreover, the synthesis can be descriptive with a quantitative summary, obtained by using statistical techniques. Both the extracted information about the studies and quantitative data should be presented in tabular form, and structured in such a way that the similarities and differences between study outcomes are highlighted and can easily be compared. [3]

Reporting the Review

For the last phase of the review, Kitchenham et al.[3] suggest specifying dissemination mechanisms as well as formatting the systematic review report and evaluating it. Regarding disseminating the findings of the review, these can be reported in academic journals and conferences or by other means, such as in practitioner-oriented journals and magazines, short summary leaflets,

posters, web pages etc. Lastly, the most common forms to report the findings of a systematic literature review are technical reports, as part of a PhD thesis or in a journal of conference paper.[3]

Chapter 4

Methodology
A Systematic Literature Review

This thesis is set out to investigate and analyse the aspect of organization of teams in software projects. The information about the organization of development teams can be found described in the literature in the form of research reports. The aim is to use these reports to develop a model for organizing software development teams. This model can be used by new, inexperienced teams or matured ones, which seek better mechanisms for organizing themselves.

The evidence presented thus far in the literature is based on single experience reports of development teams working in a project or as part of a software company. These studies mostly investigate different aspects of software development processes and methods. Although the team itself may not have been the object of research, reliable knowledge about its organization can still be found in these reports.

Despite the considerable amount of such publications, the present understanding of software teams' organizational practices is limited. Furthermore, there is little or no systematic examination of development teams in general, or of their organizational practices in particular, that explores the generalizability of any findings from these reports. The model which this thesis seeks to build relies on successful practices of development teams, especially on those that other teams can benefit from. Thus it is necessary to identify and analyse studies on the organization of software development teams, with the aim to find recurring successful applications of tools, techniques or processes that enable a better communication, coordination and self-organization for a team.

To identify existing evidence about organizational practices of development teams, a Systematic Literature Review (henceforth referred as SLR) is

conducted as proposed by Kitchenham et al. [3]. The aim of the SLR is to gain an overview of these practices, which will be further analysed in order to answer the research questions.

The following Sections describe the planning and execution phases of the SLR for finding and extracting the information from relevant studies. Furthermore, the methods for the data analysis are described in Section 4.4.

4.1 Planning Phase

Prior to conducting the SLR, it is necessary to develop a structure about how the SLR is going to be conducted, what kind of literature should be included, where to find the literature, etc. To answer such questions Kitchenham at al. [3] suggest formulating a review protocol for developing a structured pre-review plan. Moreover, the review protocol establishes some basic procedures and activities to guide the reviewer in systematically extracting the needed knowledge from the literature. In particular, the review protocol procedures address concerns such as bias [3]. In this SLR, a potential source of bias is the influence the single reviewer may have upon the selection of relevant literature. However the risk can be controlled by following the review protocol.

The rest of this subsection presents the steps of the planning phase for developing the final review protocol. These steps are: definition of review research questions (Subsection 4.1.1), database selection (Subsection 4.1.2), search string (Subsection 4.1.3) as well as inclusion and exclusion criteria (Subsection 4.1.4). The review protocol can also be found in the Appendix A.

4.1.1 Step 1: Research Questions

As already mentioned at the beginning of this chapter, this SLR is exploratory in nature. The central goal is to identify relevant research with focus in software development teams and the way they organize themselves.

Prior to formulating the research questions, it is necessary here to clarify exactly what is meant by 'organization of development teams'. Throughout this thesis, the term *organization* will refer to the way a software development team arranges, distributes and administers its work, people and activities. Hence, a special interest lies in the team members' roles, coordination and communication practices, as well as shared tools or artifacts that have proved to be useful to teams.

The literature search should reveal the studies which report evidence of these aspects of development teams' organization. Furthermore, the structures identified in such studies will be later analyzed to determine general structures which represent recurring patterns. First, they should highlight the ultimate recurrent characteristics of teams' organization and second, make adjustable ones evident. These characteristics will in turn serve

as a starting point for building the model for the organization of software development teams. Consequently, this SLR formulates the first research question as follows:

SLR-RQ1: *What development team organisational structures exist in software development?*

Furthermore, the evidence extracted from current literature should provide insight into the actual practices established for managing the aspects of organization inside a team. Specifically, this SLR gathers practices about the following organizational aspects: communication, coordination, collaboration and decision-making. Therefore, the second research question this SLR addresses is the following:

SLR-RQ2: *What are the practices for the organization of development teams?*

4.1.2 Step 2: Database Selection

The literature in this review was drawn from five main sources: Google Scholar, IEEE, ACM, ScienceDirect, SpringerLink. The first four are also listed in Kitchenham et al. [3] as electronic sources of relevance to Software Engineering, whereas SpringerLink is suggested for journals from Empirical Software Engineering and Springer Conference Proceedings.

4.1.3 Step 3: Search String

An SLR provides a method for identifying and extracting from literature the relevant research to a particular research topic of interest. In this thesis, the SLR addresses two research questions regarding the organizational aspects of software development teams. This is achieved by using a clear search strategy, which includes a set of search terms or a search string, which is used to filter the relevant studies from the databases. Thus, it is important to build a search string that addresses the SLR's research questions in such a way, that it neither returns too many non-relevant studies nor excludes relevant ones [51].

To support building a strong and representative search string, a set of keywords was derived from the research questions. The first and most important keywords are:

keywords set = {development, team, structure, organization}.

The set was further extended by terms such as synonyms, so that they cover and are consistent with the terminology used in software development publications. For example, a synonym of the term *organization* often used in the research literature in the context of teams is the term *structure*. As a result, the synonym *structure* was added to the keywords set.

As previously mentioned in Subsection 4.1.1, coordination and communication practices are of particular interest for this SLR, as they

describe important aspects of organization and information flow inside the organizational structure of a team. Having said that, the key words *coordination* and *communication* were also added to the set.

keywords set = {development, team, structure, organization, coordination, communication}.

After that, the set was converted into a boolean expression by using the boolean operators AND and OR. The operator AND was used to connect words that, at any case, should be present in the literature, whereas OR links synonyms, which represent the same terms and concepts in the terminology of software development literature. By applying the boolean operators, the set of keywords is transformed into the following search string:

Search-string = development AND team AND (structure OR organization OR coordination OR communication)

Next, this expression was tested in all digital libraries mentioned in the previous section, from which the literature was going to be extracted. The evaluation showed that the keyword *development* is used in a more general context than just the software development. Considering the titles from the first and second pages of results confirmed that. Some of the results described project environments not from the field of software development. To prevent that, the expression was extended with the keyword *software*.

Search-string = software AND development AND team AND (structure OR organization OR coordination OR communication)

Moreover, the goal of this thesis is to analyse the team structures with regard to their usage within software development contexts which implement agile, traditional or hybrid software development methods. Consequently, the search string was adjusted accordingly to also target experience reports from these development environments. The final version of the search string is shown below.

Search-string = (agile OR 'traditional software development' OR plan-driven OR hybrid) AND (software OR development) AND team AND (structure OR organization OR coordination OR communication)

Finally, the research string was adjusted to fit each digital library's advanced search syntax.

4.1.4 Step 4: Inclusion and Exclusion Criteria

For controlling selection bias, Kitchenham et al. [3] suggest to determine study selection criteria. The criteria should guide the reviewer during the execution in determining which studies are going to be included in or excluded from the systematic review. The literature inclusion and exclusion criteria for this review are shown in Table 4.1 and Table 4.2. Inclusion criteria IC1 to IC3, as well as exclusion criteria EC1 to EC3, were formulated alongside the research questions and the search string described above. They were then evaluated on the database results from the search string test trials.

No.	Formulation
IC1	The paper or article describes team organisation structures.
IC2	The paper or article describes teams which implement principles of agile, traditional or hybrid approaches in their software development processes.
IC3	The paper or article reports a case study of an agile, traditional or hybrid software development approach, where the organisation of teams is discussed.

Table 4.1: Inclusion criteria

One of the goals of this thesis is to identify team practices from real life software development settings, such as actual software development projects or team descriptions inside a software development company. In such cases, the studied teams are composed of professional software developers and represent a mature software development environment. However, at the beginning of the execution phase, it was noticed that a considerable number of papers, which fulfilled the IC1 to IC3 and not the EC1 to EC3, reported experimental studies like pilot projects or simulated student projects in a controlled academic setting. In contrast to the professional environments of software development, such experimental studies have the disadvantage that they are simulated and controlled by the experimenter and can, therefore, be biased. Feldt et al. [27]'s comment on student-based experiments suggest that the difference 'in skill and motivation relative to the professionals' poses a threat to the validity of these experiments and further state that 'sampling from the same population that one aims to generalize to reduces threats to validity'.

Hence, a fourth criterion was added to the list of exclusion criteria and is presented in Table 4.2 as EC4. This change was advised with Kitchenham et al. [3] on study selection, which states that the selection criteria should be retained during the protocol definition, although their redefinition during the search process is not prohibited.

4.2 Execution Phase

The review execution started after the review protocol and especially after the steps 1 to 4 as described in subsections 4.1.1 to 4.1.2 above were consolidated. The search string was used in each of the databases to retrieve relevant literature.

Firstly, the literature selection was based only on the information given from the title and by applying the inclusion and exclusion criteria. The selection based on only the titles yielded in total 539 articles. Furthermore, 37 duplicate studies were in total identified and removed.

No.	Formulation
EC1	The paper or article reports a team organisation practice not from a software development setting.
EC2	The paper or article is written neither in German nor in English.
EC3	The paper or article is not subject to peer-review for conference proceedings or for publishing in a journal.
EC4	The paper or article reports a team organisation practice in an educational environment with student teams or a pilot study or project.

Table 4.2: Exclusion criteria

Secondly, the title, keywords and the abstract of the literature were read cautiously to perform a second selection, which resulted in 217 relevant titles. Moreover, this step provided more context for each study and further allowed the usage of all the inclusion and exclusion criteria, based on the information on the study contribution described in the abstract.

Finally, the remaining literature was selected based on the content of the full text. This step defined the set of the most relevant studies for the review, which resulted in 53 relevant studies. An overview of the selected literature on each of these steps is found in Table 4.3.

Database	Literature selected based on title	Duplicates	Literature selected based on Title, Keywords and Abstract	Literature selected based on Content
IEEE	247	0	92	18
Google Scholar	106	20	49	15
ACM	72	14	36	10
Science Direct	32	1	4	3
SpringerLink	82	7	38	7
Total	539	37	219	53

Table 4.3: Overview of extracted literature for each database in each step

4.3 Data Collection

The systematic search and selection of relevant literature, described in the previous sections of this chapter, yielded 53 relevant primary studies. Different studies, which can be selected as part of a SLR, do not and cannot report the findings in the same way. This was the case in the resulting studies selected for this SLR. The reason for this is the fact that relatively few studies have concentrated their research on the organization of development teams. Still, organizational aspects of development teams can be found in many different types of studies, which investigate other aspects of software development teams, such as team management, meetings, information flow, challenges in collaboration or coordination, etc. Even though their focus lies elsewhere, these studies manage to report enough information about the organizational aspects of the software project teams.

To obtain the knowledge from the selected primary studies, it is required to formulate a data extraction strategy. Kitchenham et al. [3] suggest for this objective to design data extraction forms, which should aid the reviewer in accurately recording the information obtained from the selected literature. Similar to the process of defining the search string, data extraction forms should be built in such a way that they facilitate the collection of the relevant information which is needed to address the review research questions. The data extraction form of this SLR is built to facilitate the gathering of the information about development teams' organizational aspects as presented and defined in the previous chapter in Section 3.1.

The aspects of teams' organization were included in a first version of the data extraction form. This first version was then further improved during the piloting of the search string and the selection criteria. The final version of the data extraction form can be found in the Appendix B in Table B.1. It is important to emphasize once again that, because the selected relevant literature represented a set of inhomogeneous study types, they do not report the information about teams' organization in the same way. Thus, some data items in the data extraction form may be relevant to some studies, but not to others. These items are marked with an asterisk (*), which should indicate that the information may not have been reported in some studies.

4.4 Data Analysis

There were 53 primary studies identified during the execution of this SLR. This literature presents a heterogeneous set of studies as it included single case, multiple case, mixed-method studies as well as experience reports. Figure 4.1 presents an overview of the methods of data collection throughout the primary studies. Thus, for the synthesis of inhomogeneous and mixed-method studies this thesis uses a qualitative approach as suggested by Wohlin

et al. [101, p. 50-51]. They further present several synthesis methods for summarizing qualitative studies with inhomogeneous empirical evidence.

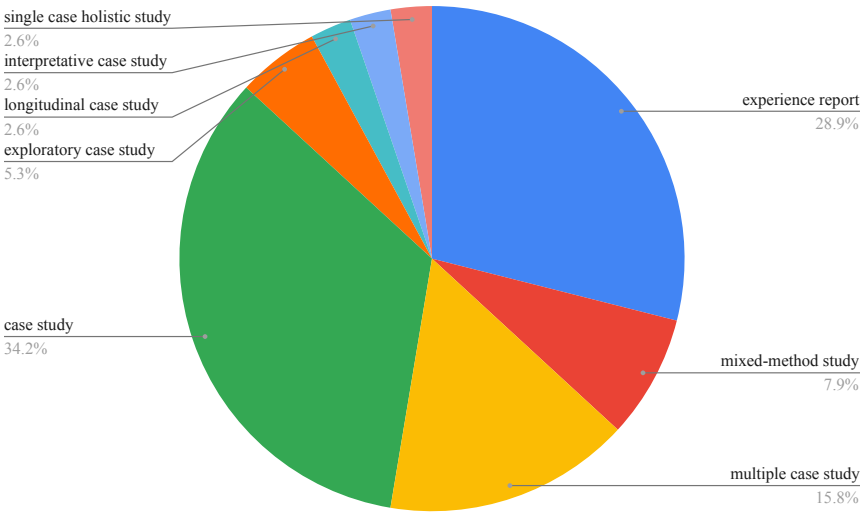

Figure 4.1: Method of data collection in primary studies

Due to the mixed nature of the selected primary studies regarding their type and the inconsistency of the data extracted from them, this thesis chooses to analyze the data by conducting a thematic analysis. To answer the research questions, this thesis looks for recurring patterns among the reported team organization structures and practices to identify general team structures. Thus, thematic analysis is used in this case, as it is the method that identifies, analyzes and reports such patterns. The thematic analysis organizes data in detail for further facilitating the interpretation and identification of patterns.

Moreover, the guidelines to experimentation in Software Engineering recommend that a sensitivity analysis should take place apart from the synthesis method [101, p. 51]. The sensitivity analysis should be able to assess whether the results are consistent across di□erent subsets of studies. Subsets of studies can be formed by primary studies of a specific type, high quality primary studies, or primary studies reporting complete and detailed evidence. The results of this analysis are considered in this thesis when reporting the limitations and threats to the validity of the findings of this SLR.

Finally, this thesis presents the findings on organizational practices and structures in a narrative synthesis. According to Cruzes and Dybå [16], the narrative synthesis adopts a narrative description to summarize findings from primary studies. This includes not only descriptions, but also comments and interpretation of the evidence in order to increase transparency and

trustworthiness. Based on that, during the data extraction phase of this SLR, key evidence from each study is tabulated side by side with the key aspects of teams' organization. Furthermore, whenever possible, the team structure is illustrated with a figure.

Chapter 5

Findings

This chapter presents the results of the SLR regarding development teams' organizational structures and practices. To better understand these results, this chapter first gives an overview of the extracted data, by providing general background information about the studies in section 5.1. Section 5.2 reports the results of the data analysis and answers the research questions of the SLR about teams structures and practices, in subsections 5.2.1 and 5.2.2 respectively.

5.1 Data Analysis

As already reported in the previous chapter (see Table 4.3), the selection of the literature based on study content resulted in 53 relevant publications. Because the team size is an important dimension when studying teams in general, this thesis also gathered information on team sizes for each case study found in the selected literature. The analysis of this data suggested that almost half of the teams from the literature were part of a large-scale environment. As shown in Figure 5.1, in about a quarter of the cases the team size was small, whereas 15.2% of the studied teams were middle-sized. In a notable number of the cases, the categorization to small, middle-sized and large team was already given by the researchers, without mentioning a concrete number of team members. Despite that, the collected data from the rest of the studies suggested that small teams' size ranged from 4 to 22 members, middle-sized teams had 15 to 50 members, whereas the size of large teams varied from 28 to 400 members. Table 5.1 provides an overview of the reported team sizes alongside the respective publication.

Regarding the software development method used in each case, it was widely reported that project teams implemented agile principles and development approaches. This trend towards more research on agile methods was expected, considering the fact that almost all selected studies were conducted after the Agile Manifesto in 2001, as shown in Figure 5.2. In spite

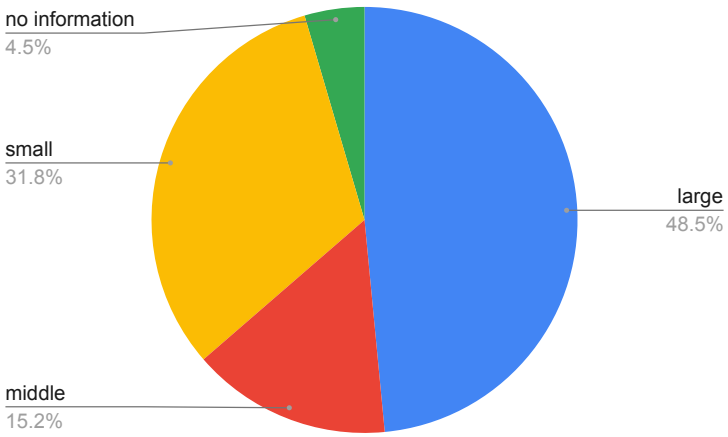

Figure 5.1: Project team sizes

of the fact that this thesis did not analyze in detail the reported development methods, it can be stated that, overall, the software development projects are still in an agile transition.

Small		Middle-sized		Large	
Size	Literature	Size	Literature	Size	Literature
4 - 5	[99]	15	[94]	22 - 48	[58]
6	[43, 79, 38]	16 - 26	[83]	42	[92]
8	[61]	27	[34]	60	[57]
8 - 9	[76]	39	[31]	100	[73]
8 - 11	[19]	50	[91]	120 - 176	[54]
4 - 15	[25]			144, 150, 200	[66, 4, 10]
15	[75, 94]			300	[62]
22	[46]			400	[71]
no information	[65, 18]	[29, 30, 95, 52, 78]		[42, 32, 7, 44, 50, 80, 33, 72]	

Table 5.1: Overview of the team size reported in the selected literature. The cases where a size range is given were multiple case studies.

5.2 Research Questions

This section presents the results and analysis of the data collected from the literature selected for the SLR. These are, in turn, used to answer the research questions presented at the beginning of this thesis and described in detail in Subsection 4.1.1. The research questions are addressed here separately in the following subsections. Subsection 5.2.1 presents the analysis

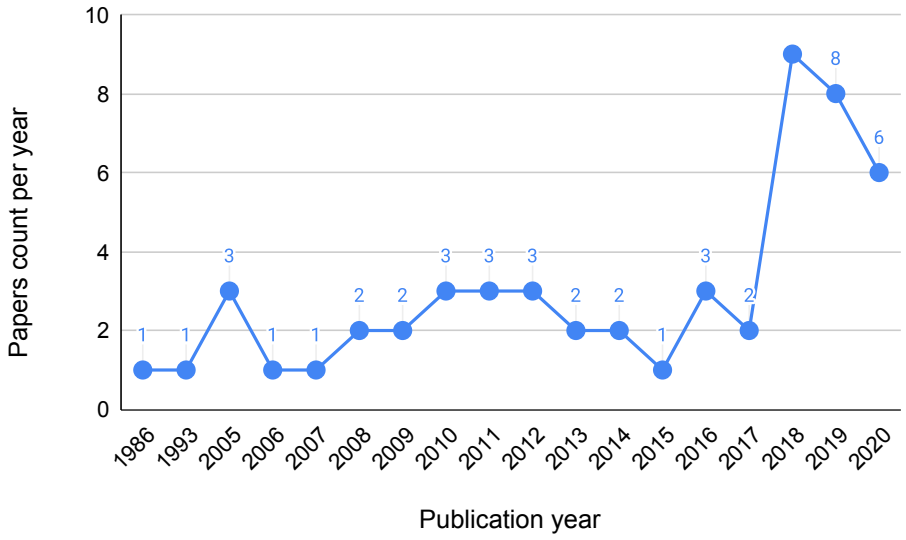

Figure 5.2: Chronological distribution of selected literature

and the synthesis of the findings about the structures of development teams. This is followed by the identification and analysis of the reported practices for the organization of development teams, introduced in Subsection 5.2.2.

5.2.1 RQ1: What development team organizational structures exist in software development?

In a significant amount of studies, the description of the development settings was detailed enough to allow an accurate extraction of the organizational structure for each team. A rather small number of cases, however, already included in their report a visualized structure of the teams under their investigation. All these structures underwent a thematic analysis in order to identify patterns which development teams recurrently use and benefit from. The results of the analysis are provided in this thesis as a narrative synthesis. To identify themes, the structures and the respective notes were compared in detail and finally organized in three groups; small, middle-sized and large teams, based on team size.

Small Teams

The analysis identified 21 cases of small project teams from 14 di□erent publications. In 8 of these cases, the researchers reported the use of agile methods [65, 26, 75, 99, 43, 61, 76, 19], whereas 3 cases implemented a traditional software development approach [26, 79, 99] and the other 10 represented a combination of development approaches [99, 25, 46, 18, 94,

26, 19, 38]. Figures 5.3, 5.4 and 5.6 below show typical development team organizational arrangements for traditional, agile and hybrid development approaches respectively.

Traditional team. By observing the traditional team structure in Figure 5.3, the first thing that can be noticed is its hierarchical structure. At the top of the structure these teams had a management level visible, whereas at the bottom, the teams were functional and worked in silos. Furthermore, the project and product manager as well as the team leaders were part of the management team. These roles were typically assigned to senior members of the teams or of the organization. Another important observation is the involvement of the customer and stakeholders. In traditional settings, they only communicated with the managers and were involved in the development process only at the beginning and the end of the project.

Agile team. The typical agile team was a scrum team. Almost all the teams which reported to use agile methods implemented as well the Scrum framework as part of their development process. However, some studies reported using also another methodology in addition to scrum such as eXtreme Programming [18, 19] or a tailored version of scrum [25]. Two other publications reported that the teams they investigated used only eXtreme Programming [99, 43], in a co-located and globally distributed team respectively. This thesis did not investigate in detail the actual software development methods, processes or frameworks used by the teams, because it would require an in-depth analysis, which is out of the scope of this thesis. Nevertheless, the collected data about each team suggested secondary findings which are presented at the end of this subsection (see Section 5.2.1).

Because the majority of publications reporting the implementation of agile principles used the Scrum framework, the team structure in this case was similar to the arrangement of a scrum team. This arrangement is illustrated in Figure 5.4. The agile scrum team consisted of a scrum master, a product owner and the developers. External developers or consultants were also hired occasionally as outsourced workforce to support the lack of resources in the core project team or for reducing project costs [25, 46]. Nevertheless, the teams were cross-functional with a broad range of technical skill sets. Furthermore, in some projects a project manager role was visible as well [19], and the overall team structure was similar to the arrangement presented in Figure 5.5.

This arrangement (see Figure 5.5) showed up from the cases which implemented eXtreme Programming for the development. The developers in these teams practiced pair programming when working on their tasks. Vidgen and Wang [99] reported for one of their cases, that developers, the project manager and the customer closely worked together to address the technical and business complexity of the product. This suggested that developers were involved in the management activities and held as well direct and frequent contact with the customer. They further practiced task self-

assignment and used pairing, especially for complex tasks and those requiring specific knowledge.

Hybrid team. A synthesized arrangement of these teams is illustrated in Figure 5.6. By comparing the team arrangements and descriptions for the thematic analysis, the cases categorized as hybrid teams not only reported the usage of hybrid development methods, but also a combination of the autocratic and decentralized management style. For instance, when using the scrum framework, it was often the case that the roles of scrum master and product owner were assigned to the most experienced team members or senior developers. In the cases which described a transition from traditional to agile methodology, these were project and product manager. Moreover, this way of role assignment is typical for autocratic management style, also seen in the traditional team arrangements [2].

In addition to that, it was noted that this allocation separated the team into management and development. While the management was rather traditional autocratic, the organization of development team was informal and resembled the cross-functional teams in composition. This is mostly because the development used agile development or a combination of agile principles with traditional methods, such as water-scrum-fall, Scrum-Xp-waterfall delivery, Scrum-XP-Unified proces, in [46, 94, 18] respectively. Di□erent from agile teams, hybrid teams assigned roles to each individual in the development team.

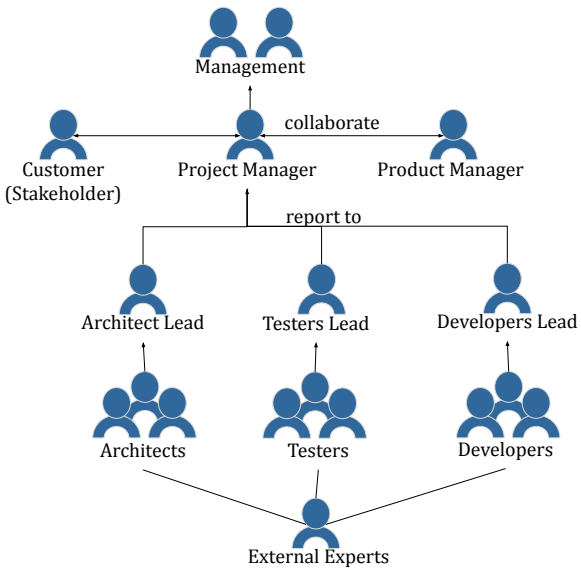

Figure 5.3: Common organizational structure for small development teams implementing software by following a traditional software development approach

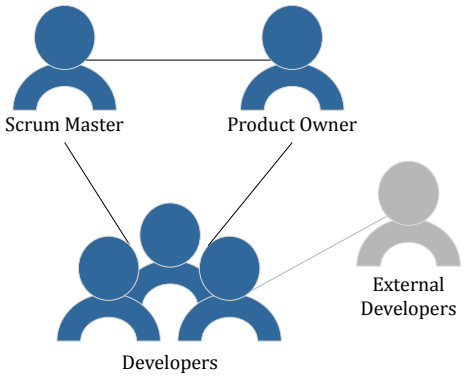

Figure 5.4: Common organizational structure for small development teams in an agile software development project

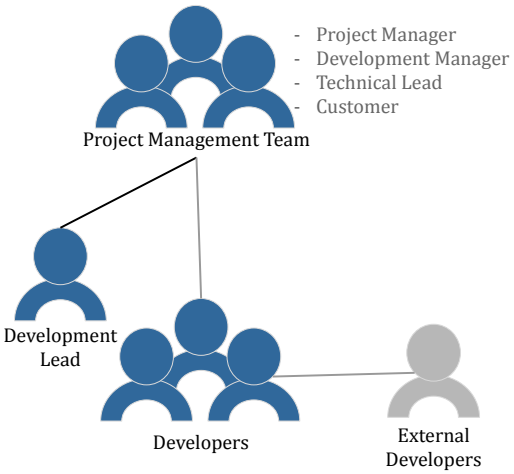

Figure 5.5: Organizational structure for small development teams implementing eXtreme Programming

Middle-sized and Large Teams

In total, 33 case studies reported large teams, whereas only 11 project cases had middle-sized teams. Figures 5.7 and 5.8 illustrate synthesized team arrangements for both middle-sized and large teams from the selected literature. Middle-sized and large teams are reported here together, because overall, they showed similar organizational characteristics. In both middle-sized and large team structures, there were clearly two organizational levels

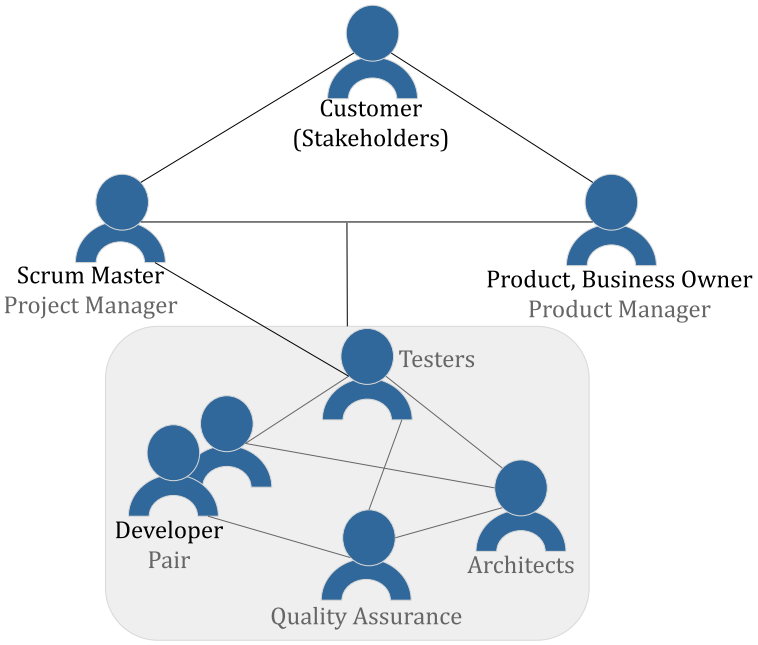

Figure 5.6: Typical organizational structure for small development teams implementing both traditional and agile principles

visible: the team level or the horizontal organizational level, and the management level, or the vertical organizational level. This can be also observed in the Figures 5.7 and 5.8, where the development teams are placed at the bottom of the structure, whereas the top level is occupied by management roles or teams.

Development teams were part of the horizontal organizational level. This was the lowest level of the project organization. The structure inside each team was similar to the one presented in Figure 5.6, where the inter-team communication and coordination was mostly controlled by team leaders or team coordinators [58, 32, 26, 75, 92]. Furthermore, the development teams were cross-functional and included people with di□erent areas of expertise or roles [71, 40, 65, 42, 57, 80]. Regardless of the fact that these teams appeared to be cross-functional and agile, when working for a long time in the same project, they tended to specialize in a single software component [62], feature [96, 69, 4, 73, 83] or module [30, 71, 79], and therefore, lost their cross-functionality.

In addition, project teams arranged in the center of all the development teams, another team, which mostly handled coordination aspects and had decision-making authority. This team was named ‘central team’ in the cases

presented by Bick et al. [7, 8], 'project management team' by Layman et al. [43] and 'core team' by Hossain et al. [25, 92]. Even larger teams and project cases had several management teams [80, 54], such as; a 'Central Planning team', a 'Product Owner team' and 'Architectural Governance team', and 'Bug Board', 'Architectural team' and 'Product Owners team' respectively. However, due to this team's managing role in the project, this thesis refers to it as 'project management team'. In some cases, members of this team were people with typical management roles, such as project manager or product manager [54, 32, 83, 30], as well as representatives of the customer and stakeholders, such as domain experts [7, 80, 30] or proxy customers [50, 71, 8, 69]. These people held the control over the most important decisions, such as plans, requirements, etc. A detailed overview of the most common practices in both, the team and management level, is presented in Section 5.2.2.

However, in agile or hybrid cases, this team was not directly visible and the management of the project was handled here by coordinating rounds, similar to Scrum of Scrums [95, 7, 8, 80, 71, 44, 32]. In these rounds, representatives and leaders from all teams met regularly in order to discuss issues and dependencies which development teams were facing. These rounds also handled different planning topics [7, 73], discussed requirements and prepared them for backlogs [7, 80], assigned features to the teams [66, 49], tracked the project progress and overview, etc. Other participants were also project and product managers, architects, quality assurance and customer representatives, such as proxy customer or domain experts. Figures 5.7 and 5.8 provide an overview of the most common organizational structures for middle-sized and large teams.

Other important insights regarding the organization structure of development teams

Overall, the analysis of the data reported from the selected primary studies suggested the following conclusions about team organizational structures:

- *Software development methods used by development teams influence the way these teams organize themselves.*

  For example, in a project team using traditional development methods such as waterfall [30], team members work in silos with clearly defined roles and are not aware of the dependencies between one another and with other teams, because of the lack of frequent and direct communication. The management of dependencies between work entities is centralized at management and leader roles, which manage the team based on a command and control style. On the other hand, when a team applies agile principles, teams tend to be cross-functional with no defined roles and have more freedom regarding decisions on

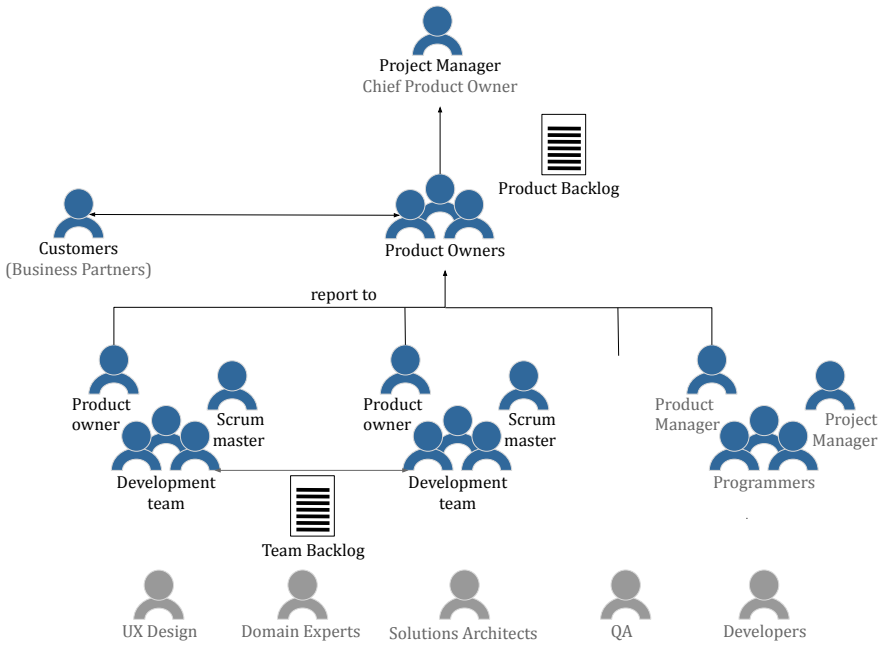

Figure 5.7: Summarized structure of middle-sized software development teams

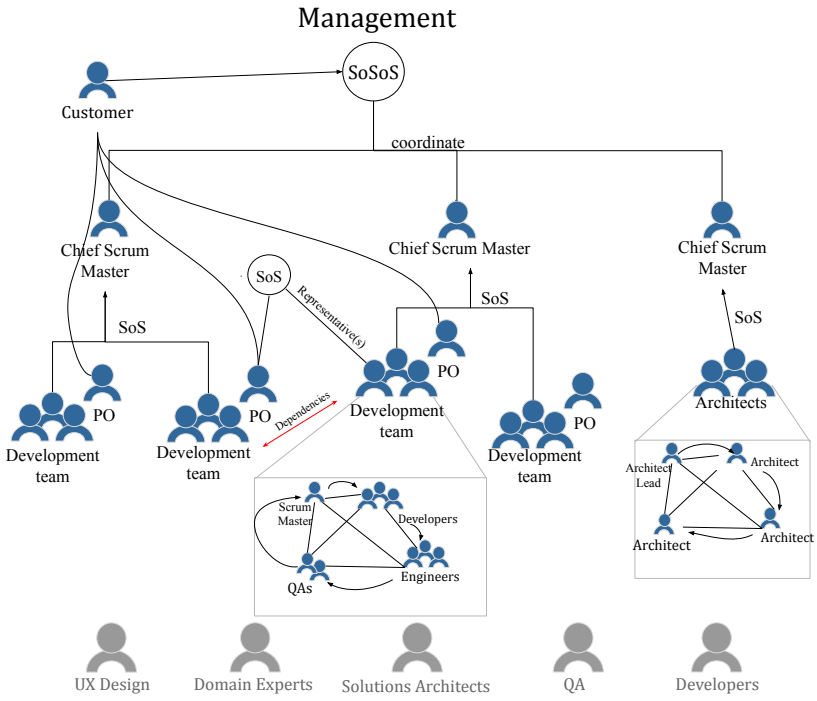

Figure 5.8: Summarized structure of large software development teams

how to work, coordinate tasks and monitor dependencies for identifying issues on time. Because agile principles put an emphasis to the individuals, the management of such teams is based on leadership and collaborations, where decisions are made by consensus.

While in traditional teams there is a clear di□erence between development and management level, agile teams mostly do not show any managerial roles, as management is a common responsibility of the team. Furthermore, specific software development processes or frameworks such as Scrum, eXtreme Programming etc. prescribe as well practices to facilitate the team management. Examples of such practices in Scrum are coordination meetings such as daily stand-ups or sprint planning and in XP, pair programming for collaboration. For multiple team systems such practices define and have an influence especially in the organization at the team level.

- *Project complexity, size and structure a ects the overall team structure.*

  In a small project, there are little to no dependencies between teams and the team size is proportional to that of the project. The latter was also found by Keshta and Morgan [39]. Especially in large and complex systems, the product is mostly divided in features or modules, which

have often dependencies from one another [28, 4, 62, 42, 96, 7, 8, 54, 71, 73]. This division and their dependencies are also carried by the teams when they are assigned to work on those features and modules.

- *Team and project size define the practices for decision-making authority and control.*

  As already stated above, the data from several publications [7, 4, 32, 95, 73] suggested that especially complex and large projects suer from issues related to dependencies in dierent parts of the product. To coordinate and manage these issues is a challenge, that is why the most complex and large projects and teams from the selected literature centralized the decision-making authority and control to the project management [7].

  Bick et al [7] reported on five large-scale projects that suered from the same issues. All their cases implemented tailored Scrum at the team levels and tailored Scrum of Scrums with decision-making authority and control at the project management levels to facilitate coordination of dependencies and work on multi team systems with up to 13 teams. In their traditional case, centralizing the control at the management did not solve such dependency issues; firstly, the management did not have the required technical understanding to identify such problems and secondly, the teams were not involved in the coordinating process in order to help with their technical expertise. However, their other cases where each team had the opportunity to collaborate with peers from other teams were perceived as well-coordinated, because the members had the freedom to exchange information and were more involved in the overall organization.

  Paasivaara and Lassenius [71], on the other hand, also reported a similar team structure. Their project team worked in product areas and applied Scrum of Scrums practices for coordination for each area. The product development responsibility, and with that some control over the project, was held by a product management team. In order to not confine the overall project overview just at the area or feature levels, the project started using communities of practices (CoPs), which were open for the whole team to participate and contribute to. Such communities seemed to increase collaboration and helped in quickly identifying and resolving dependencies and issues. In this case, there was much more freedom left to the teams regarding control over the project status, coordination and communication.

  Even small teams suered from dependencies between product features developed by dierent teams. Although the issues here were easier to handle, especially for co-located teams, small distributed teams were the most aected [18, 46]. Similar to the large projects,

a management role controlled the project and team progress and facilitated coordination inside the team.

- *Teams constantly changed aspects of organization for their team with better practices, mostly because of the challenges they face.*

  This was not only observed in well-established teams who sought an agile transition (e.g. [10, 59]), but also in new project teams, in which their current organizational arrangement did not satisfy the team's needs [71]. Also, the team organizational changes were necessary in the case of project and team growth [96]. This finding is also supported by previous research [23, 92].

- *Managerial roles in traditional teams are typically assigned to the most experienced or senior members of the team. The same is applied by agile and hybrid teams. Furthermore, new agile roles are not fully implemented as described in the corresponding guidelines. They are often merged with old traditional managerial roles.*

  Consider, for example, the roles of Scrum Master and Product Owner in an agile scrum team. In Scrum.org [85] and Scrum Guides [84], the Scrum Master is defined as a coach and a leader that serves the team and the Product Owner. Having said that, this role does not correspond to the product manager role, who leads and manages the team based on command and control principles. However, it was noted that teams implementing the Scrum Framework, often assigned the Scrum Master role to the most experienced team member (e.g. [4, 75]) or to the person, previously holding a manager role (e.g. [91, 32, 26]). This suggested that especially management roles from traditional settings are translated into the agile coach and leader roles, almost without changes in the occupation.

- *Given the findings described above, it can also be concluded that the new agile positions are still occupied by the same people. [69, 57, 76, 71]*

- *External stakeholders, especially customers, were not always involved in the development process of agile cases.*

  While in traditional methods it is not necessary for customers and external stakeholders to be part of the development process, agile settings require from them constant feedback and direct communication, in order for the teams to quickly react to changes and problems regarding requirements. In the project cases developing an in-house product [96, 58, 34, 25, 76, 7, 8], where the requirements came directly from within the project organization, that is, the organization itself was the customer, development teams received frequent and direct feedback

on the product's requirements. In these cases, the management of requirements was the responsibility of one or more dedicated Product Owners, which also comprised or were part the project management team [7].

In cases of external customers, these were partly or not at all involved in the development. Instead, they provided product specifications, plans or regulatory documents [83, 94, 25]. Others delegated the dictation to employees from their own organization or to representatives such as proxy-customers [69, 50, 43, 71] or domain experts [8, 30, 91, 66].

5.2.2 RQ2: What are the practices for the organization of development teams?

To answer this question, data from the selected studies underwent a thematic analysis. The analysis identified patterns in the organizational practices of development teams, which are introduced and described in detail in the following subsections. This thesis found and grouped the following practices: communication, coordination, collaboration and decision-making practices.

Communication

This thesis identified several communication practices used for different purposes, depending on what is being communicated. For example, team members used direct communication to exchange thoughts and ideas about their tasks [28, 95]. Further, communication with the purpose of coordination of work happened mostly in meetings, virtually through emails or chats or in the written form through documents. Table 5.2 presents an overview of communication practices grouped by the type of communication: spoken, written or visual.

One or more of the practices listed in Table 5.2 found usage in the following scenarios: one-on-one encounters or conversations, during meetings or gatherings, during collaboration sessions between team members and for the communication through documents.

One-on-one communication. This is the most common form of spoken and written communication, which takes place in the everyday work between two team members. Direct and informal communication forms are preferred in this case. Furthermore, the one-on-one communication is a practice represented by ad-hoc conversations [66, 34, 59], face-to-face or in the written form, for the following reasons: for knowledge sharing [58, 32, 26, 57, 71], coordination of work [71, 10, 28, 8], clarification of dependencies and issues [94, 43, 71, 91], receiving and giving feedback as well as for socializing [26, 75]. Depending on the team setting, there is reported to be more face-to-face and direct communication in co-located teams [28, 50, 43], whereas in distributed cases, team members depend on virtual

Communication	Practices
spoken	face-to-face
	ad-hoc conversations
	online conference (call or video call)
	meetings
	socializing, on-site visits
	feedback
	open office landscape, co-location
written	emails
	instant messages, chats
	wiki
	spreadsheets, word documents, quality interface
	requirements and product specification
	plans, release schedule
	guidelines, regulations, standards
	epics, features
	user stories, tasks
	acceptance criteria, test cases
	behaviour-driven scenarios
visual	project and team progress dashboards
	physical and electronic whiteboards
	backlogs (per team, sprint, project)
	desktop sharing

Table 5.2: Overview of communication practices

communication [18, 34, 25, 46]. The virtual communication happens in the written form through email, instant messages, chats etc. and as spoken communication through phone or video calls. In both cases, communication between the involved parties is further supported by other tools such as documents containing figures, diagrams, plans, etc. or desktop-sharing in the case of virtual conversations.

Communication during meetings or gatherings. Typical communication forms during meetings are spoken and visual communication. This is the form of spoken communication in group, where more than two people are involved in the conversation. In this case, depending on the type of meeting, the participants use communication to: coordinate work in general (who does what), to identify dependencies, issues, impediments, etc., to discuss solutions [95, 87, 65, 66], to share own thoughts, ideas, suggestions for improvements [32], to ask for or to give help, to give feedback [26] as well as to share knowledge and expertise [71, 34]. During meetings, it is common to sit together in one Table or be in the same room [58, 19, 61]. Thus, in these cases the communication happens orally. In the case that participants are not co-located, the communication is facilitated by online conferences through phone or video calls [83, 44, 40, 65, 18, 75, 50, 43]. Furthermore, visual communication practices such as desktop sharing, physical or electronic whiteboards [4, 18, 34, 43, 80], progress dashboards [54, 75, 43, 96, 28] are also commonly used by teams to facilitate the coordination process and discussions during meetings [99, 18, 34].

Communication during collaboration. This is as well a form of spoken, written and visual communication in group, used indirectly in collaboration sessions between two or more people who work on the same task. In this case, the same communication practices as in one-on-on communication and meetings can be used, which should facilitate the activity of working together. A typical example is the collaboration during pair programming [8, 40, 62, 58, 19, 34], where two developers work together on a common assignment. Communication in the context of collaboration is as well used for knowledge sharing, giving and getting feedback or for decision making, e.g. making decisions by consensus [79]. The communication practices described above should further support actual collaboration activities. These are described in the following Subsection 5.2.2 and summarized in Table 5.4.

Communication through documents. The practice of communicating through documents is a formal form of communication and it is the most common form of written communication. In this case, code, artefacts and all the other documents generated during the project duration such as guidelines, regulations, standards, specification are used as a means of communication and documentation of project status and knowledge. [66, 57, 80, 71, 94, 4, 43, 79, 49] Among the most used artefacts are those representing customers' requirements, such as backlogs, epics, user stories, tasks [4, 54, 34, 29, 44], but also team progress boards for tracking team and project

status, designs, acceptance tests, etc. Furthermore, most used documents include guidelines, regulations, standards, specifications, templates, notes, plans etc.

Overall, the analysis of all communication practices suggested that written and spoken communication are the most common form of communication among teams. Because communication is an essential part of work, its usage is found to accompany and support other team organizational aspects as well. This will be further elaborated in the following subsections about coordination, collaboration and decision-making.

Coordination

Software development teams use different coordination practices to coordinate their work and identify issues and dependencies. An in-depth analysis of the identified coordination practices suggested a categorization in coordination practices during meetings, through boundary spanners as well as through boundary objects. These practices are described below in details. Table 5.3 shows an overview of concrete practices from each category.

Meetings. Meetings are the most common practice for team coordination, in both horizontal and vertical level. Meetings are a way each team member can get an overview of the project status, of team progress and for coordinating the work to be done with other team members. Further, meetings give each participant the chance to get to know their peers' responsibilities and knowledge, which in turn, contributes to the identification of dependencies, issues, impediments, questions, which can then in the same place be addressed and clarified. This raises further the need for discussion, where team members can give their feedback, perspectives, ideas and make suggestions for solutions, improvements, plans and strategies.

Boundary spanners. A boundary spanner is a person who acts as a facilitator of communication and of work coordination inside the team or between teams, in order to protect their team from interrupts. In small development settings, especially in those implementing agile practices, this role is almost invisible [18, 26, 99]. Because agile practices require more direct communication, teams directly contact one another [58, 26, 19, 43, 95, 28] or bring the issue up in the next meeting for clarification [50]. In less traditional settings, this role is occupied mostly by management roles such as the project or product manager [49, 73, 8].

In the cases where teams are not so close to one another, the boundary spanner is represented by one or more team representatives [95, 94, 69, 4, 44], who can be assigned by the teams or is a rotating role [71]. This role often participates in the meetings with teams in the same level of the organization. These meetings are mostly used to identify and address dependencies and issues, for planning and for sharing of knowledge and expertise. It is common practice that, in meetings with managerial nature, the only participants are

Coordination	Practices
meetings	planning (project, sprint, iteration, release, monthly) daily stand-ups, daily meeting Scrum of Scrums project increment, requirements, user stories workshop sprint review, retrospective refinement sessions user stories reviews communities of practice semi-structured meetings emergency team meetings status and hand-over meetings demos task discussions direct coordination
boundary spanner	manager roles (project, product, technical, programme) domain/regulatory expert product owner (CPOs, domain PO, business owner) scrum master team representatives, rotated or not proxy customer team leader
boundary object	project dashboards backlogs (team, enterprise, product) kanban board quality interface documents (requirements, product specification, regulations) epics, features user stories acceptance criteria, test cases tasks wiki repository online project management program

Table 5.3: Overview of coordination practices

those from each team's management roles, such as product manager, scrum master, product owner etc.

The communication with the customer happens as well often through a boundary spanner [66, 18, 69, 70, 50], who may be a dedicated person like a proxy customer or domain expert from within customers' organization, or a person actually attached to the team, like the product owner.

Boundary objects. Boundary objects are part of the written communication. They are used for gathering requirements, tracking project progress, documenting knowledge and expertise, coordinating work through regulations, guidelines; or they can be interfaces or tools, such as bug tracking systems, wikis or CVS[1] repositories to handle the organization and communication during coordination activities. The most common boundary objects used for product requirements are items such as epics, user stories, tasks, acceptance criteria etc. and backlogs at different levels, such as product or enterprise backlog, sprint, release, dedicated team backlog etc. The documents that can be mentioned here include regulations, guidelines, standards, plans and other files such as meetings recordings, group chats or wiki pages.

It is evident that communication practices reappear and are part of the coordination practices. For example, meetings are the events where a large amount of group communication happens and with that also the organization of work. Further, meetings serve as a common practice for the coordination at both organizational levels, at team level as well as at the management level. Nevertheless, the participation of all the team members is high for meetings [81, 31, 94] at the lower levels of the organization, that is, at the team level. Participants of meetings at the management level are mostly only management or leader roles, such as team leaders or team managers, while other team representatives are rarely asked to join [57, 4]. In the management level, the team representatives serve as boundary spanners, which handle the discussions in the name of their team. Thus, the lines of communication are extended.

Moreover, almost all the cases included in this SLR report about the usage of written forms of requirements or indirect communication, such as backlog, epics, user stories, etc. It can be clearly stated that these boundary objects serve as a formal way of communication and coordination with stakeholders at the management level. However, the same boundary objects are used also at the team level.

Collaboration

Both communication and coordination support collaboration practices. The success of communication or coordination practices is reported to affect

[1] Concurrent Versions Systems

positively the collaboration. For instance, Noordeloos et al. [65] reported that frequent communication between team members, shorter lines of communication with the customer, as well as their direct involvement in planning, prioritization and daily stand-up meetings promoted overall a better collaboration.

The thematic analysis of the collaboration practices extracted form the literature resulted four themes: intra-team, inter-team, collaboration between development teams and management teams and collaboration with external stakeholders. The most common practices per each theme are listed in Table 5.4.

Collaboration	Practices
intra-team	working in pairs communities of practice learning, knowledge sharing planning (work) backlogs (local, sprint) ad-hoc conversations, socializing support, assistance and help feedback, issues discussion of issues, improvements, progress, ideas, dependencies co-location, shared o□ce
inter-team	coordination meetings (SoS, status and hand-over, demos, reviews, dailies, communities of practice, planning) retrospectives team representatives learning, knowledge sharing task swap or responsibility rotation shared collaboration tools shared team members
team and management	requirements (clarification, prioritization, planning, backlogs (product, sprint, team, iteration) project status meetings feedback, issues, improvements, suggestions, ideas, dependencies

Table 5.4: Overview of collaboration practices

Intra-team collaboration is the collaboration happening between team members of the same team. The most reported practices in this case were: co-located or remote pair programming, sitting together or co-location, collective discussions at workplace or during meetings, workshops

for knowledge-sharing, as well as ad-hoc encounters and socializing [8, 19, 99, 66].

Inter-team collaboration is practiced by members of different teams. The inter-team collaboration was found in: common coordination meetings, such as Scrum of Scrums [92, 91] and project management meetings [95], inter-team technical meetings and communities of practice (CoP) [71, 96]. In the coordination meetings collaboration was practiced for identifying dependencies and issues or during the discussion of solutions and decisions. However, only in a few studies, team members had the chance to directly participate in these meetings. Instead, boundary spanners such as team representatives or leaders with managerial role took part in such meetings and therefore, controlled the inter-team collaboration. Regarding the inter-team technical meetings, these were mostly organized by technical leaders and handled a specific technical topic. Communities of practice were rarely used. They were self-organized by the development teams at the lower levels of the organization and also handled a specific topic. Furthermore, communities of practice were reported to be held irregularly or on an as-needed basis, whereas technical inter-team meetings took place regularly. Especially in large-scale settings, technical inter-team meetings had a crucial role in coordinating work between dependent teams. All in all, collaboration during coordination meetings was the most reported form of inter-team collaboration, followed by inter-team technical meetings and communities of practice.

Collaboration between development teams and management teams. In the project cases where the control of the project is centered at the management team [49, 94, 8], there was little to no direct collaboration between the team members and the management team [26, 19]. The collaboration happened through the boundary spanner or the boundary objects, discussed above in the coordination practices. There were, however, cases where these coordination rounds were more open and transparent, giving each team the chance to participate in the discussion. Common practices worth mentioning were: firstly, the whole team could participate [28, 71] and secondly, teams elected representatives who participated in rotation or not [65, 44].

Collaboration with external stakeholders (especially customers or their representatives). Amongst most common practices of collaboration with external stakeholders were boundary spanners, such as proxy customers [50, 43], domain experts from customers' organization, external domain experts, product owners, project or product managers, etc. [8, 30, 91, 66] Boundary objects describing raw product and business requirements were another commonly used practice in these cases. Examples of boundary objects were documents containing high level requirements, such as requirements specifications and whole descriptions of features or modules.

Decision-making and control

For the thematic analysis, all the identified practices regarding decision-making and control were considered and compared to find themes. The analysis made evident three groups for these practices: firstly, practices where the decisions and control were held by the teams, secondly, practices that indicated control and decision-making authority at the management level. Finally, the third group were practices that showed some control and decision-making authority lying at the stakeholders and customers, mostly about requirements. In Table 5.5 these groups are referred to as decision-making and control at the team level, at the management level (including customers and stakeholders) respectively. Further, the Table displays a summary of the most common practices found for each group.

At the team level. Decision-making authority and control at the team level can be understood as the freedom of each development team to decide about different organizational aspects regarding daily work. The most common practices here were: choosing own coordination activities [59], such as meetings, choosing own tools for collaboration, maintaining local backlog [44], estimation and self-assignment of tasks according to own interests, learning, task swap and collaboration with peers for knowledge sharing, etc. [10, 31, 83] Less common was the practice of directly involving the team in important decision rounds. In such cases, team members or their representatives were invited to share their perspective, give feedback and propose solutions [57, 29, 28].

At the management level. The most common decision-making and control practices were found to be held by the management. The management was composed here by roles, such as boundary spanners [96, 28, 40], as well as a dedicated project management team, which controlled budget and resources[81, 26, 49]. The dedicated management team was mostly visible in the cases of an in-house developed product [8] or in very large projects, as well as in the cases where the customer was not directly or only sporadically involved in the development [30]. In the first and latter cases, this team controlled also the project scope, which is normally managed by the customer and stakeholders.

However, in some cases, even though the management had the responsibility to keep track of dependencies between teams, it failed to do so, because of the lack of competence to identify such issues [8]. As a result, development teams were isolated from one another, worked on tasks assigned by the management and were not able to resolve issues on their own. It can be concluded that, considering these practices as well as the structures of middle-sized and large project teams (see Fig. 5.7 and 5.8), the authority of the management clearly stands out.

At customer and stakeholders. As already stated above, important decisions about requirements were held by the customer. Whether the

Decision-making and Control	Practices	Literature
at team level	own activities	
	team backlog	
	estimation, prioritization, time	
	task swap, assignment	
	participation in decision rounds, representatives	
	planning	
	learning and knowledge sharing	
	suggestions, solutions, improvements	
	communication with other teams and customer	
	autonomous	
at management level	product, team backlog	
	prioritization, planning, work assignment	
	resources, people	
	communication of decisions, information	
	coordination and resolving of dependencies, issues	
	collaboration and communication with customer, gathering and formulation of requirements	
	guidelines, regulation, standards	
	project scope, status, overview, progress, quality	
	centralized to different manager roles	
at customer	requirements dictation, selection, prioritization	
	quality check	
	project scope	
	monitoring project progress	
	delegated to proxy-customers, domain experts, customers' teams	

Table 5.5: Overview of practices for decision-making and control

customer was directly involved in the development process [99, 25, 81] or not, the control over requirements dictation [99], formulation, prioritization was held by them or their representatives, such as proxy customers [50, 43, 71] or domain experts from customers' organization.

Chapter 6

Discussion

One of the main goals of this thesis was the identification of practices for organizing development teams. The in-depth analysis of the findings from 53 primary studies highlighted team organizational structures for small, middle-sized and large development teams. Furthermore, this thesis found that these structures were supported by practices for organizational aspects such as communication, coordination, collaboration and decision-making authority.

This chapter provides a discussion and interpretation of the findings presented in the previous chapter. Section 6.1 discusses findings regarding team organizational structures and practices. Next, these findings are interpreted and compared with results from the current literature on software development teams. Finally, the discussion of limitations, threats to validity as well as generalizability of results concludes this chapter.

6.1 Organizational team structures and practices

The first research question in this study was to identify what *organizational structures* are used by software development teams. This thesis identified three groups of development team structures: small, middle-sized and large team structures. For small teams, there were evident three types of organizational structures: traditional, agile and hybrid team structures (see also Figures 5.3, 5.4 and 5.5, and 5.6 respectively). Cases reporting middle-sized and large teams were rather hybrid team structures, regardless of the methodology (traditional, agile or a combination of both) used in the software development process. This can be explained by the fact that, in smaller team settings, the software development methods are easier to implement, because a small number of people is being managed. Thus, the differences between traditional, agile and hybrid team structures are more evident in these cases.

However, by comparing the structure of a small traditional setting (see Figure 5.3) with the structures in middle-sized (Figure 5.7) and large teams

(Figure 5.8), it can be noticed that they share a hierarchical organizational structure based on a command and control management. This form of organization and management is found to be characteristic of traditional project management by previous studies [39, 2, 60], which compared traditional and agile perspectives on software development. Keshta and Morgan [39] found that, the bigger the team, the more control is required from the project management to govern the team and the project. They further concluded that larger teams often tend to apply traditional methods, which further explains the hierarchical organizational structure of middle-sized and large teams.

While the small teams summarized in Figure 5.3 reported using a traditional waterfall development approach, middle-sized and large teams applied mostly agile approaches such as Scrum of Scrums, LeSS or SaFE, which were tailored with some traditional practices. The results of the thesis at hand imply, however, that, even though the overall team structure appears to be hierarchical at first sight, in reality it is not. Providing the organizational practices identified and presented in the previous chapter (see Section 5.2.2), it can be assumed that development teams implement traditional practices to support their organization and project management. Actually, these teams' organizational structure used both the traditional hierarchical and the more agile approach of managing teams based on leadership and collaboration. Especially decision-making and control as well as coordination practices make this more evident.

Decision-making and control practices at the management and team level are presented in Table 5.5. By comparing these practices at the management level it can be noticed that the management controls resources, people, work assignments, coordination and communication and is centralized to managers. This is the same control as observed in the so-called 'top-down organizations', where the management has decision-making authority. Despite the limitations given by the control the management holds, teams still can have some autonomy.

Considering the practices at the team level, it is clear that teams can control their own activities, participate in planning as well as in decision rounds directly or through representatives, control the tasks they want to work on and have an open communication with stakeholders outside their own team. These practices have a more self-organizing nature, which is well known to be a characteristic of agile teams. Taken together, this suggests that development teams apply more traditional organizational practices at the management level, while at the same time, they organize themselves in a more agile way. This finding is an important step further in the better understanding of hybrid software development approaches, especially for identifying and evaluating the benefits and challenges coming from the combination of traditional and agile principles.

Another reason for larger teams showing hybrid characteristics might

be the fact that the majority of the cases included in the analysis came from recent research. Even though DeFranco and Laplante [22] related the recent publications on software development teams with agile team research, the data analysis of the thesis at hand suggested that teams tailored agile methods to their needs. The modifications are made especially in the team organizational practices. Moreover, it is important to notice that a significant amount of studies reviewed in this thesis, reported an agile transition from traditional settings. Based on the collected data, this further suggested that, during the transition, some of the old traditional practices were still used by the team. With regard to the team organization, the old traditional command and control is still applied at the management level, whereas development teams have received more freedom by implementing agile development practices. Thus, the combination of the two methods, agile and traditional, was also reflected in the overall project and team management, by showing characteristics of hybrid software development methods.

The fact that the agility was mostly visible at the team level may support the argument that large project teams have already found the way to apply agility at the team level. The findings from the thesis at hand suggest that the next step towards extending agility to the management level would be to invite more individuals from the team levels to management rounds and to apply decision-making by consensus.

The second research question posed at the beginning of this work was related to the identification of *organizational practices* used by development teams. This thesis found practices for each of the following aspects of team organization, that is, communication, coordination, collaboration and decision-making. These practices were applied at both project teams' levels, that is at the team and management level, and supported each teams' organizational structure.

The identified practices for one of these aspects often seem to be used as or support practices of another organizational aspect. Especially communication practices are applied in all of the other practices. For example, written communication practices such as documents, are also used for coordination as boundary objects. Documents are as well a practice that support collaboration between team members at the team level of the project organization, by helping them work on common ground. Furthermore, documents or other forms of written communication can be used by the customers or their representatives to document their decisions regarding requirements or planning.

Even though this thesis clearly defined each aspect (see Table 3.1), their intertwining does not consist in the terminology, but it lies on the concrete practices for communication, coordination, collaboration or decision-making. This argument is supported as well by Sharp and Robinson [88], who state that '[...] collaboration and co-ordination depend on communication, and

communication – in one form or another – is central to successful software development[...]'.

The summary of traditional and agile project management presented at the beginning of this thesis, suggested that traditional teams must be characterized by a hierarchical organizational structure, whereas the management levels in agile teams are flat. The analysis in the systematic literature review conducted in this thesis, a□rms once again these characteristics of traditional and agile teams. Moreover, the knowledge gained here suggests that, especially middle-sized and large teams, implement both these characteristics by building hybrid team structures. These structures show agile characteristics at the team level, where they implement more agile organizational practices, and traditional characteristics at the management level for controlling especially the decision-making activities.

6.2 Implications for practice

Teams structures and organizational practices found in this research can be used by teams that seek to improve one or more of the teams' organizational aspects studied here. Especially younger teams can mostly benefit from them as they build their team from ground up. This thesis presents further implications for practice in the form of guidelines in the next chapter (see Chapter 7).

6.3 Generalizability, Limitations and Threats to Validity

As with all similar studies, there are several limitations and threats to validity to take into account. This section reflects on and takes into consideration possible limitations and threats to validity of this study and its results. Furthermore, it presents the measures taken to mitigate these threats.

Typical limitations for systematic literature reviews are *selection bias* and *inaccurate data extraction*. These are also mentioned by Kitchenham et al. [3] in their guidelines for performing systematic literature reviews. To control the threat of *selection bias*, especially because of a single reviewer, this thesis first formulated a review protocol. This protocol defined a clear structure to follow for conducting the review. The research questions, inclusion and exclusion criteria as well as the search string helped in holding a similar selection procedure throughout all databases. Furthermore, Kitchenham et al. [3] suggest that some researchers may favor a specific study type, which also increases the possibility of *data collection bias*. Despite the fact that this review looked for studies with mature software development teams, the type of selected publications was heterogeneous, including single and multiple-case

studies, experience reports, mixed-method studies as well as interpretative and exploratory case studies (see also Figure 4.1).

In addition, it is not possible for different researchers to report their findings in the same way throughout their publications, this also depending on the topic under investigation. Because of this, there might have been some inaccuracy in the data reported in the primary studies, where interesting aspects for this study may have been described in different levels of detail. This may have been further intensified by the fact that this review did not identify studies directly focusing in the investigation of development teams' organization. All the selected primary studies had different focus than the organization of development teams. This variability to the topics under investigation in each included publication also poses a threat to reliability of the findings. To mitigate these limitations, this thesis developed a data extraction form prior to the phase of data collection, for guiding the reviewer in identifying and collecting the same kind of information throughout all the selected primary studies.

Regarding the *external validity*, it can be stated that, the findings of this thesis may be found useful from practitioners from several software development domains. The teams' characteristics studied in this thesis vary in different aspects, such as domain, size and context. The most frequent domains included healthcare, finance and insurance, consumer electronics, IT products and services, telecommunications, as well as enterprise software products. From these domains, it was possible to find and summarize several studies which presented data relevant to organizational aspects of development teams. This fact strengthens the external validity of this thesis' findings for these settings, which may be found useful by practitioners in these domains. Still, the collected evidence represents inconclusive evidence for other settings such as automotive industry, safety-critical systems other than healthcare or other software development settings, like Open Source Software Development. Therefore, it is uncertain whether the findings presented in the previous chapter are applicable in these other settings as well.

Chapter 7

Model

This chapter presents a model for organizing software development teams. The model is formed by guidelines, which are based on the insights on teams' organizational structures and practices presented in the previous chapter (see Chapter 5). The guidelines first propose team organization structures based on team's needs and then, provide relevant alternatives of practices, which are better suited in each case. Of course, there is no formula or direct method that can perfectly work for all teams, especially due to the fact that each team is different [21].

This chapter is organized as follows: subsection 7.1 presents the three-steps approach for choosing one of the seven template structures, suggested team organizational structures. The second section describes each template in detail and present suggestions on how to use them. Finally, the third section presents a list of optional practices to fill in the team structures.

7.1 Model Presentation

Similar to Boehm and Turner [11], it is necessary to first evaluate own software development setting as well as to identify what are the team's needs. The following guidelines can be used by both young and mature teams that need to build from ground up or improve their organizational arrangement.

Therefore, the proposed method consists of the following 3-phases:

1. Evaluation of the current organizational structure and practices, or of the new development setting

   In this step, the team should first create an overview of its current organizational practices and then identify the challenges or issues and the desired improvements in current practices. Mature as well as new project teams can evaluate their development setting by considering the five critical agility and plan-driven factors which are used by Boehm and Turner [11] in their approach for balancing agile and traditional

methods. For that, the practitioners should use the polar chart adapted from [11] (see Figure 7.1) to distinguish between agile or traditional, plan-driven methods.

It is suggested that, if the project is not characterized by a frequent change of requirements and if the loss due to defects is high, than teams should implement a plan-driven approach. In this case, the team can apply the team structure presented in Figure 5.3 and scale it to the team size appropriately. However, if the estimates per each factor are allocated very near the center of the polar chart, then the team can implement agile principles in their project. The team structures presented in Figures 5.4 and 5.5 are the most common arrangements used by small agile teams. In the other cases, it is necessary to further evaluate the aspects of organization for the specific development setting.

2. Evaluate the aspects of organization

   After the evaluation of the methodology, the typical characteristics of traditional and agile teams regarding organization should be consulted and compared with own needs. To accommodate this, the organizational characteristics of traditional or agile teams are summarized in Table 7.1, based on the findings from the SLR. The comparison with these characteristics should guide matured teams to evaluate whether their actual organization is more traditional, agile or has characteristics of both. The identification of its current model and the awareness of the problems should facilitate the decision whether to hold on to the already used methodology or to change it. For new teams, this step should help to decide in which organizational aspects they need to put emphasis to when arranging the team structure.

3. Level of control for communication, coordination and decision-making authority

   Communication, coordination and decision-making authority are the three aspects of organization that can be adjusted in order to best fit the team's needs. A way to adjust these factors is, for example, to allow teams or the management to control the practices for each of these aspects. For example, a project team might need to give more freedom to the development teams. This can be achieved by letting development teams control, for example, communication and coordination practices. In this case, the control over communication and coordination would be at the team, whereas management controls the decision-making authority. Therefore, if the control lies at development teams, then it can be stated that the team level of the project organization has control over one or more specific aspects. Similarly, if the control over these aspects lies at managers, then the management

level of the project organization controls and decides how to handle communication, coordination and decision-making. Having said that, this model assumes the following two levels of control: the *team level* and the *management level*.

After consulting the organizational characteristics of traditional and agile teams (Table 7.1), project teams should be able to decide whether to put the control for each of these aspects to the teams or to the management, that is, to the team or management level respectively. The model o□ers seven team structure templates. These are built based on the level of control for communication, coordination and decision-making authority. The team can determine its template by filling in T or M for control on team(s) or control on management respectively, in Table 7.2. The combination filled in here determines the team structure. For that, the corresponding team structure template can be extracted from Table 7.3 by using this combination. For example, if the team fills in T for communication, M for coordination and M for decision-making, then this model suggests template T4, illustrated in Figure 7.5).

The team structure templates A1 – A4 and T1 – T4 are further described and illustrated in the following section. In the description of each template, the practitioners can find further suggestions on how to apply them to their project team.

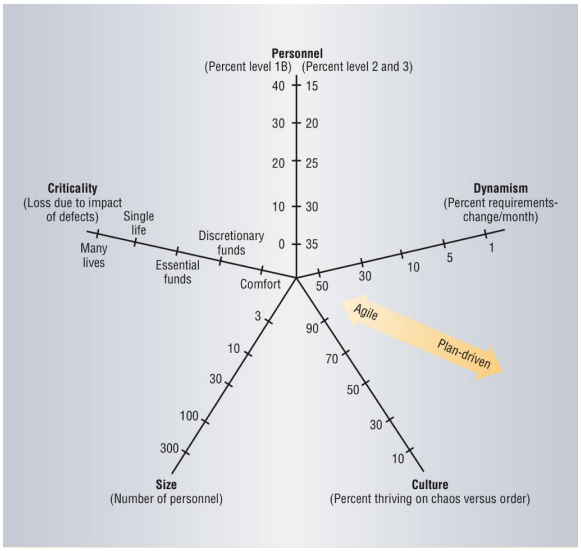

Figure 7.1: Polar chart with five axes representing the factors used to distinguish agile and traditional, plan-driven approaches by Boehm and Turner [11]

Organizational aspects	Traditional teams	Agile teams
roles	clearly defined	no roles evident
specialization	high, work in silos	low, are cross-functional
coordination	controlled by managers or leaders	controlled by the team
	through strict regulations and plans!	collaborative!
dependencies	not visual	visual and transparent
decision-making and control	at management level	at team level by consensus
communication intensity	scarce/sparse and formal	vast/extensive and informal
chain of command	rather long	rather short
span of control	wide	narrow
decision-making	centralized	decentralized

Table 7.1: Organizational characteristics of traditional and agile development teams

Organizational Aspects	More control on teams (T) or management (M)?
Communication	T or M?
Coordination	T or M?
Decision-making	T or M?

Table 7.2: The level of control for communication, coordination and decision-making authority

Organizational Aspects	More control on teams (T) or management (M)?							
Communication	T	T	T	T	M	M	M	M
Coordination	T	T	M	M	T	T	M	M
Decision-making	T	M	T	M	T	M	T	M
Template	A1	A2	A3	T4	A4	T3	T2	T1

Table 7.3: Template matrix: Templates A1-A4 and T1-T4 based on the level of control for communication, coordination and decision-making authority. Agile organizational practices dominate the templates A1 to A2, whereas templates T1 to T4 have more traditional organizational characteristics.

7.2 Proposed Team Structure Templates

The conclusions of the SLR presented in the previous chapter suggested that communication, coordination and decision-making are the aspects of organization that a□ect a team's organizational structure the most.

From Table 7.1 above it can be noted that in the organization of traditional and agile teams, the control lies at the management and at the team level respectively. Reconsidering Table 7.3, it can be noticed that the combinations on the left tend to give more control to the team, whereas those on the right give more control to the management. Thus, the overall organization characteristics of the team structures on the left have a tendency to be more agile and those on the right more traditional. In Table 7.3 the combinations showing more agile or traditional organizational characteristics are labeled A1-A4 or T1-T4 respectively.

For all of these templates the following practices apply:

1. Development teams at the team level of the project organization are cross-functional.

2. Customers and important stakeholders are regularly involved in the development, either directly or indirectly through proxy customers and domain experts.

This section describes the team organizational templates proposed in the approach from section 7.1. Most importantly, this section describes guidelines on how to use the templates in practice. Overall, how exactly and with which other concrete practices these structures can be further completed, is left in the hands of each individual team. However, some of the practices can be chosen from the lists of most used practices identified by the review (see also Tables 5.2, 5.3, 5.4 and 5.5) or can be adapted from the development framework already implemented by the project.

7.2.1 Team structures A1 to A4

As already stated above, the first three templates tend to show more agile characteristics, because of the amount of control they give to the teams for each of the three aspects of organization. For at least two out of the three aspects in the triples A1 to A4, the control is placed to the team level. A detailed description of each structure is given below.

A1

In this structure the control over communication, coordination and decisions is held by the teams. As already stated at the beginning of this section, the model suggests that each individual development team should be cross-functional, organize its internal coordination on its own initiative and make

decisions on the development practices and tools it employs in its day-to-day work. It is recommended that the team members should at best be co-located.

On the inter-team level, as the communication is controlled by the teams, there should be possible to implement communication practices that hold open and transparent channels throughout all the teams. Each individual from one team should be able to directly contact other individuals from the other teams. Moreover, it should be possible for each team to directly contact the customer as well, in order to hold the feedback cycle, especially for requirements clarifications, as short as possible.

Regarding coordination, there should be an open space for each team to propose joint coordination rounds, e.g. in the form of regular or irregular meetings, forums and discussion rounds about a specific topic or for resolving dependencies and issues. Nevertheless, it may be necessary to implement a facilitator as well, who will moderate the discussions in such rounds and help the teams to stay focused. Because the customer and some stakeholders hold the control over decisions, they or their representatives should also be involved in these rounds, especially in those where important decisions are discussed and made. Customer and stakeholders can be represented by proxy customers, product owners or domain experts.

Depending on the degree the customer and stakeholders wish and have the capacity to participate in the development process, this structure suggests to employ near the teams one or more individuals with both technical and domain background. The domain experts can already be part of the teams or can be outsourced consultants. In any case, they should be able to support the implementation with their knowledge of the domain.

A2

In the second left triple A2, the control over communication and coordination is given to the teams, whereas decision-making control is held by the management. In this case, the model suggests the employment of a management or leader role, which in direct collaboration with the customer or stakeholders has the authority to make decisions. However, because the teams still hold the freedom to organize the coordination and communication, the model further suggests to hold open channels of contact with both the management and the customer. Both these roles should participate at least in the important coordinating meetings such as planning meetings and support the teams with the results of their decision rounds. The rest of the arrangement is implemented similarly to A1.

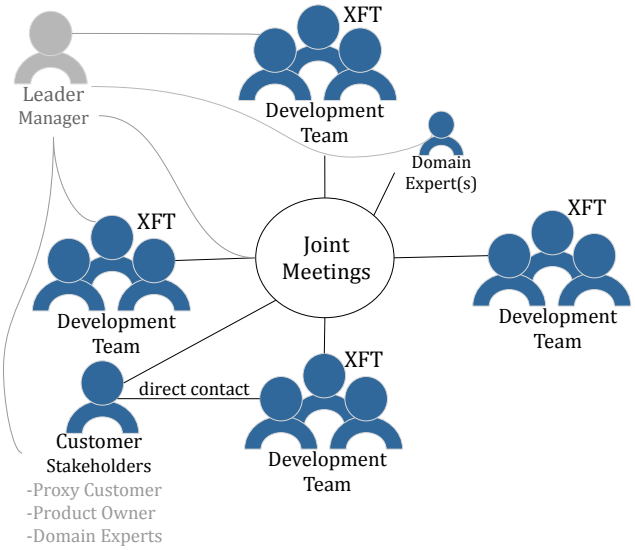

Figure 7.2: Template A1 – Development teams control communication, coordination and decision-making. Gray elements represent practices which can be adjusted by the practitioners.

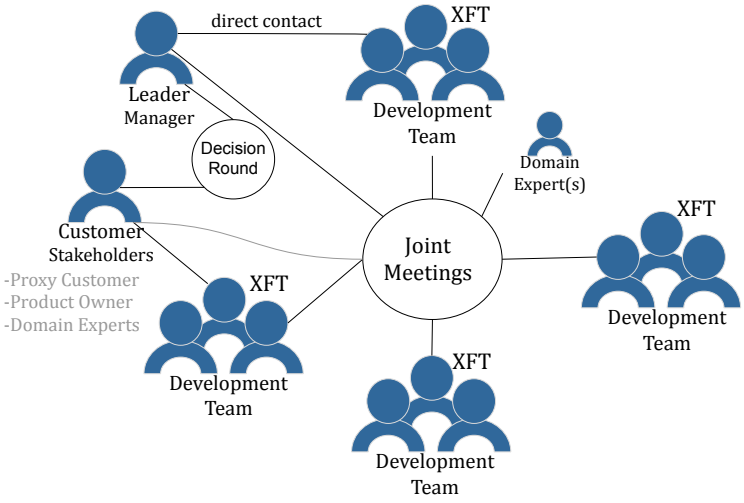

Figure 7.3: Template A2 – Development teams control communication and coordination, management controls decision-making. Gray elements represent practices which can be adjusted by the practitioners.

A3 and A4

The third and fourth triples A3 and A4 present combinations of organizational aspects that may at first be conflicting. They suggest to leave the control over communication and the control of dependencies to the management and at the same time, the team should be able to handle the decisions. At this point the question may arise: How should the team be able to make decisions when it is not aware of the dependencies and issues?

To solve this, both these combinations are merged into one single structure. The structure for A3 and A4, shown in Figure 7.4, allows both the management and the teams to have control over the decisions and coordination. This is arranged in the structure by giving the teams direct access to the decision rounds, where the team members or their representatives take active part in the joint discussions by sharing their perspective, thoughts and ideas to the whole group. The final decisions are taken by consensus and in agreement with the customer and stakeholders.

To further keep the feedback loops as short as possible, it is suggested that, if the customers cannot be directly involved in the development, it should be at least possible for team leaders or team representatives to directly contact them. Regardless of the customers' involvement, the project management should stay regularly in contact with both team leaders or the team members themselves and o□er its support whenever it is needed.

Similarly to the arrangement of A2, in A3 and A4 teams still can have self-coordinating rounds among themselves. Moreover, depending on the overall team size, teams can choose to send representatives to the important managerial rounds. The representatives can be elected temporarily or permanently, or can even be rotated among members of the same teams.

7.2.2 Team structures T4 to T1

By considering the four last triples from left to right and the structures illustrated in Figures 7.5 to 7.8, it becomes once again evident that with each step to the right, the control held by the management increases. It is important to recall that, overall, both traditional and agile organizational aspects should be implemented in these structures. Despite the fact that, with triple T4 the models' suggestions move in more traditional grounds, the organization inside each individual development team should lean toward agile characteristics. Thus, at this point it is recommended to continue implementing the same practices for each teams internal organization as in structures A1 to A4.

In the following subsections, the more traditional structures are presented in the descending order, from the least traditional T4 to the most traditional T1.

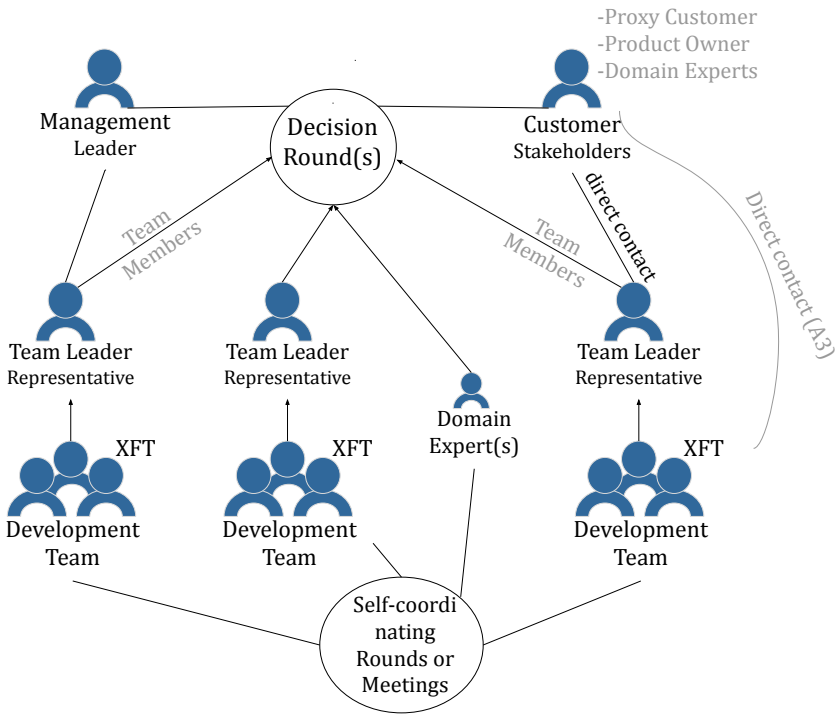

Figure 7.4: Template for A3 and A4 – Development teams control communication and decision-making (A3) and coordination and decision-making (A4), management controls coordination (A3) and communication (A4). Gray elements represent practices which can be adjusted by the practitioners.

T4

In this structure, the team holds only control of the communication, whereas management handles coordination and decisions. Because the control over communication lies with the teams, it is recommended to still allow teams to directly contact each other and also initiate cross-team working groups. These can be forums or communities of practice called by teams or team members in on-need basis. They can serve for cross-team collaboration, knowledge sharing or discussions of issues and dependencies. Furthermore, team members should be able to directly contact the project management as well as their team leader, and keep the feedback loop short. However, because the coordination is held at the management level, the communication with the customer and stakeholders as well as the formal coordination rounds at the management level are handled by boundary spanners, such as team leaders.

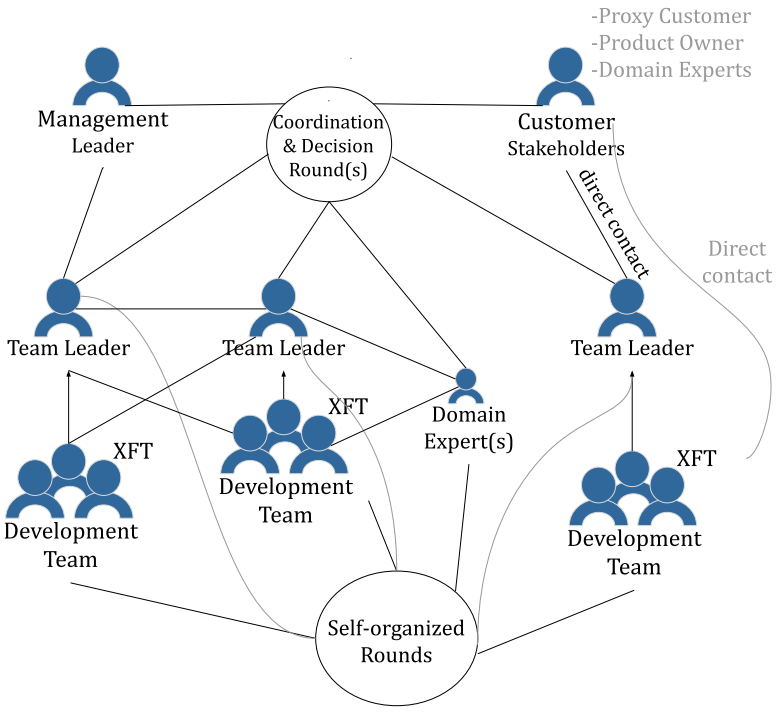

Figure 7.5: Template T4 – Management controls communication, development teams control coordination and decision-making. Gray elements represent practices which can be adjusted by the practitioners.

T3

By moving to this structure, the team has only control of coordination. In this case, it is suggested that team representatives organize regular coordinating rounds, where dependencies and issues can be discussed. However, the inter-team coordination should not be restricted only to these rounds. It is further suggested that team representatives, or the people holding the role of boundary spanners for each team, frequently contact one another. This practice should facilitate getting instant feedback in the cases where issues arise. Especially in complex systems, this kind of issue resolving may not be su□cient. For that, these guidelines suggest organizing inter-team gatherings on the specific topic, such as system architecture or quality assurance, where a small group of people from each team can participate. Because the management rules the communication and decision-making practices, practitioners may find it helpful to organize special decision rounds with the interested groups. These can be members from the management, customers and other important stakeholders. The decisions taken here can be then later communicated to the teams.

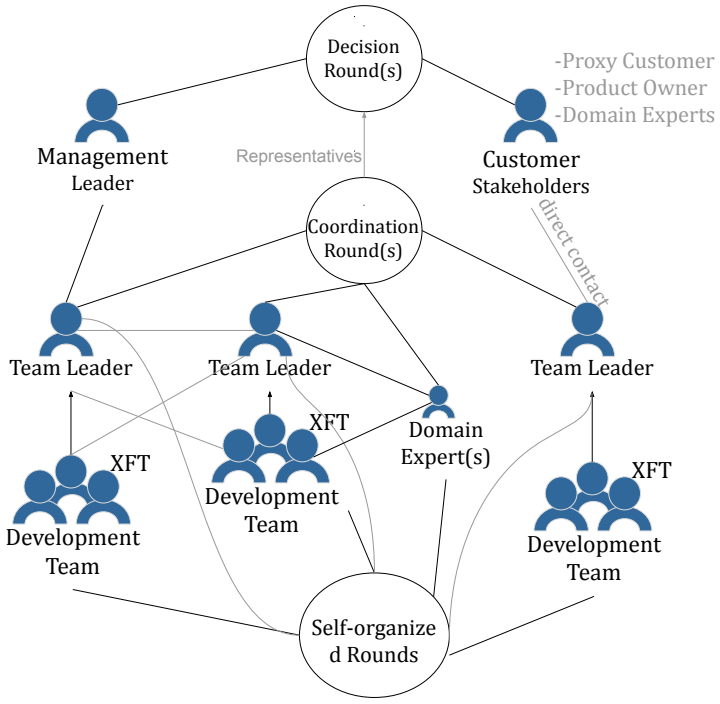

Figure 7.6: Template T3 – Management controls communication and decision-making, development teams control coordination. Gray elements represent practices which can be adjusted by the practitioners.

T2

With this triple, practitioners chose to put the control over communication and coordination at the management, while leaving the decision-making authority to the teams. These settings may be contradictory, however, they are still applicable in a team structure. For example, teams can apply their decision-making authority by giving feedback on problems or sharing their own perspective with the management in coordinating rounds.

Because the communication and coordination is controlled by the management, teams can only communicate with one another through boundary spanners or boundary objects. Boundary spanners should further facilitate the coordination by maintaining an overview over the project progress. This should help them to quickly identify and address issues and dependencies. Because teams do not directly communicate with one another, the role of the boundary spanners and boundary objects is crucial as they should replace the direct communication in the best way possible.

Furthermore, as the communication with the customer and stakeholders becomes more scarce, teams may often lack the necessary knowledge to directly address issues related to requirements. To avoid the long feedback loop while waiting for the customers response for questions, practitioners could find it helpful integrating the domain knowledge into the teams. At this point, domain expertise plays a key role in supporting teams and for that reason, it is suggested to place at least one domain expert close to them.

T1

With this combination, the management level of the project team structure holds the control over all the three aspects of organization. It is clearly evident also from the structure visualized in Figure 7.5 that the communication lines at the team and especially at the inter-team level are even more scarce. In this case, the project teams' practices will be dominated by those who put the control at the top of the organizational structure. Nevertheless, especially for complex projects, it is important to support inter-team coordinating rounds.

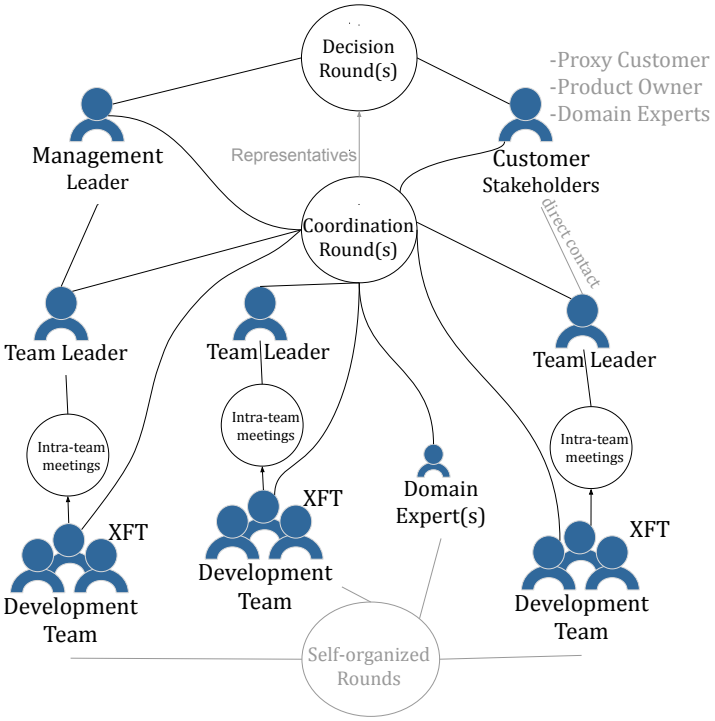

Figure 7.7: Template T2 – Management controls communication and coordination, development teams control decision-making. Gray elements represent practices which can be adjusted by the practitioners.

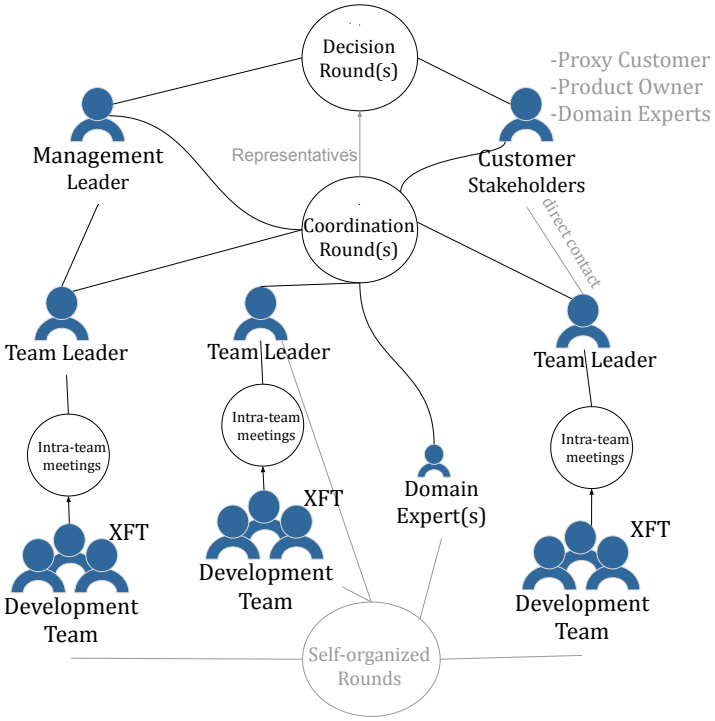

Figure 7.8: Template T1 – Management controls communication, coordination and decision-making. Gray elements represent practices which can be adjusted by the practitioners.

Chapter 8

Summary and Future Work

This thesis has investigated the organizational aspects of software development teams. The review has collected the evidence from the recent literature and has been able to analyse the organization of software development teams. With the help of a thematic analysis, it has been possible to identify organizational team structures and practices. The findings have indicated that there are three different organizational structures for small development teams, emerging from the software development method they apply, that is traditional, agile or hybrid. Further, the identified structures for middle-sized and large teams were almost identical.

In the larger team settings, this thesis identified a structural division between team and management levels. In the first, that is the team level, project teams implemented not only development agile principles, but also management and organizational agile principles to support the intra-team organization. In contrast, in the majority of the cases the management level was found to still implement traditional management techniques, where the coordination happened based on the principles of command and control.

Moreover, the case studies from the recent literature have used a various number of organizational practices, which further supported each team's structure. Overall, this thesis has found four types of organizational practices: communication, coordination, collaboration and decision-making practices. The in-depth thematic analysis of these four types of practices has further implied that, all in all, these practices support each other to achieve a better organization inside a project team. For example, an open and transparent communication of information has been found to strengthen the intra- and inter-team coordination and collaboration, and facilitated the application of consensus to support collective decision-making.

Taken together, the identification of development team organizational structures and practices provided an excellent opportunity to summarize them and finally, formulate team organizational guidelines for practitioners. Based on the analysis of the evidence and experiences reported in primary

studies, this thesis has developed several team organizational structures which should guide the practitioners in developing better team arrangements. Furthermore, the guidelines have provided organizational practices which can be chosen by teams to complete these structures based on their own needs. In summary, this thesis not only contributed to a better understanding of the organizational aspects of software development teams, but also provided guidelines for development teams and practitioners to achieve a better organization.

Directions for further research

The findings of this thesis should help the researchers and practitioners to better understand the aspects of organization in software development teams. The knowledge summarized in this work can find usage in various software development settings and domains. Nevertheless, the application and validity of the generalizations presented here in settings such as Open Source Software Development cannot be guaranteed, as project teams from these settings were not part of the selected literature. It can be argued that the further investigation of such settings is important to better understand very large, distributed teams, which are common in Open Source Software projects. Because large-scale projects face a lot of challenges regarding team organization and management, the solutions, or at least improvements to such problems may be found in this field.

Furthermore, the review of organizational aspects in development teams was based mostly on the evidence provided by several projects, which had previously delivered software products by implementing traditional, plan-driven methods, and which, with the birth of agile principles, completely transitioned to or partly adapted agility in their development process. According to analysis of this evidence in this thesis, these projects presented rather hybrid development approaches, as they implemented more of a combination of both traditional and agile approaches. Having said that, this thesis raises the question whether young development teams, with no experience in using traditional principles, will still engage them in their software development process or will they be able to fully implement the agile principles.

Finally, this thesis has proposed a model to guide teams into evaluating their organizational needs, which should guide them into choosing a team structure as well as organizational practices that best fit their needs. This is a new approach, which is developed based on the findings of the literature review conducted for this thesis. However, whether this approach finds applicability to different software development teams is not further investigated. Clearly, it is necessary to further evaluate this model with teams and, if needed, to provide changes and adjustments accordingly.

Appendix A

Review Protocol

A.1 Search String

[agile OR (traditional software development OR plan-driven) OR hybrid] AND (software OR development) AND team AND (structure OR organisation OR coordination OR communication)

A.2 Research Questions

RQ1: What development team organizational structures exist in software development?

RQ2: What are the practices for the organization of development teams?

A.3 Inclusion Criteria

IC1: The paper or article describes team organisation structures.

IC2: The paper or article describes teams which implement principles of agile, traditional or hybrid approaches in their software development processes.

IC3: The paper or article reports a case study of an agile, traditional or hybrid software development approach, where the organisation of teams is discussed.

A.4 Exclusion Criteria

EC1: The paper or article reports a team organisation practice not from a software development setting.

EC2: The paper or article is written neither in German nor in English.

EC3: The paper or article is not subject to peer-review for conference proceedings or for publishing in a journal.

EC4: The paper reports a team organisation practice in an educational environment with student teams or a pilot study or project *(updated 06.12.2020)*.

A.5 Database Selection

The search string will be used on the following digital libraries:

- IEEExplore
- Google Scholar
- ACM Digital library
- ScienceDirect
- SpringerLink

A.6 Search Process

1. Run an automated search by using the search string in each database.
2. Apply inclusion and exclusion criteria based on the title only.
3. Exclude duplicates.
4. Reapply inclusion and exclusion criteria based on abstracts and keywords on each paper or article left.
5. Considering the whole paper or article content, reapply inclusion and exclusion criteria once again. The papers or articles which were not excluded during this step, are the papers to be used for the review.
6. Read and analyse each paper or article. Use the data extraction form B to systematically collect data from each publication.

Appendix B

Data Extraction Form

Table B.1: Content of the data extraction form, the data items marked with * are optional

Data item	Value
DOI	
Author(s)	
Title	
Publication year	
Method of data collection	
Project team, team inside a company?	
Branch	
Small, middle, large setting?	
Number of subjects/teams participating in the observation	
On-/O□-shore, distributed, remote/virtual team(s)?	
*Number of locations	
*Organization type: hierarchy, matrix, flat, etc.	
Software development method: agile, traditional, hybrid, in agile transition	
Processes and frameworks: Scrum, Scrum of Scrums, SAFe, Kanban, DevOps etc.	

Continued on next page

Table B.1 – *Continued from previous page*

Data item	Value
Main team: Size	
Main team: Roles	
Main team: Leader role	
Decision making, chain of control: centralised, decentralized...	
*Tasks and responsibilities	
On-/o□-site customer?	
Sub-team(s): size	
Sub-team(s): roles	
Sub-team(s): leader role	
Sub-team(s): tasks and responsibilities	
*Development practices	
Communication	
Coordination	
*Documents	
Artifacts	
*Tools	
Pros, positive aspects	
Cons, negative aspects	

Continued on next page

Appendix C

Selected Primary Studies

Table C.1: List of the literature included in the systematic literature review

No.	Title	Year	Domain	Method	Size
1	Transitioning to Agile—In a Large Organization [59]	2020	Amazon USA	Scrum, non-agile teams, Scrum of Scrums traditional waterfall	large
2	Achieving Agile Big Data Science: The Evolution of a Team's Agile Process Methodology [81]	2019	BigData science, Entertainment industry, US	Kanban	large
3	Blueprint Model: A new Approach to Scrum Agile Methodology [29]	2019	R and D, Mobile products, samsung	Scrum and Kanban	middle
4	Challenges in Adopting Continuous Delivery and DevOps in a Globally Distributed Product Team: A Case Study of a Healthcare Organization [31]	2019	Healthcare organization, Siemens Healthcare	DevOps, lean	middle

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
5	Behavior-Driven Development as an Approach to Improve Software Quality and Communication Across Remote Business Stakeholders, Developers and QA: two Case Studies [83]	2019	Large Publisher of Financial Information and Business News	Scrum with agile and Lean SD principles, Kanban, XP, Devops, TDD... Behavior-Driven Development (BDD)	middle
			Collaboration platform vendor.	Scrum based	middle
6	Coping Strategies for Temporal, Geographical and Sociocultural Distances in Agile GSD: A Case Study [94]	2019	Pharmaceutical setting	Scrum + XP, delivery in waterfall fashion	small
7	Investigating the Adoption and Application of Large-Scale Scrum at a German Automobile Manufacturer [96]	2019	Automobile manufacturer	extended LeSS (+ Kanban + DevOps), LeSS Huge	large
8	Inter-team Coordination in Large-Scale Agile Development: A Case Study of Three Enabling Mechanisms [28]	2018	Public department, o□ce automation system	Scrum, Scrum of Scrums, LeSS	large

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
9	Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings [8]	2018	Standardized Enterprise software	Scrum, SoS	large
10	Our Journey to Becoming Agile: Experiences with Agile Transformation in Samsung Electronics [40]	2016	Consumer electronics, Samsung	Scrum, (Pair-programming, TDD, CD)	large
				Scrum of Scrums, cross-functional	
11	Adapting Agile in a Globally Distributed Software Development [30]	2016	Mobile platforms - Siemens Technology and Services	waterfall	middle
				scrum, SoS	
12	From RUP to Scrum in Global Software Development: A Case Study [65]	2012	Finance	scrum	small
13	Experiences in Scaling the Product Owner Role in Large-Scale Globally Distributed Scrum [69]	2012		scrum	large
				scrum	large

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
14	Influence of Large-Scale Organization Structures on Leadership Behaviors [62]	2009	Healthcare industry -Health Services, Siemens Medical Solutions USA Inc.	Scrum	large
15	Dependency Management in a Large Agile Environment [4]	2008	R and D	Scrums, Scrum of Scrums, SoSoS, clustered SoSoS	large
16	Implementing Program Model with Agile Principles in a Large Software Development Organization [42]	2008	Software development tools	scrum	large
17	Implementing Scrum in a Distributed Software Development Organization [91]	2007	BMC Software, Enterprise management solutions	scrum	middle

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
18	Agile o□shore techniques - a case study [18]	2005	-	Scrum + XP(iterative development) and Unified Process (Phases: Inception, Elaboration, Construction + Transition)	small
19	Enabling Team Autonomy in a Large Public Organization [58]	2020	Public sector	kanban	large
20	Key Factors in Scaling up Agile Team in Matrix Organization [32]	2019	Healthcare industry, -legacy, mission-critical software system	scrum, Scrum of Scrums	large

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
21	Analysis of the changes in communication and social interactions during the transformation of a traditional team into an agile team [26]	2018	Software Enterprise, software industry		small
				scrum	small
22	To schedule or not to schedule? An investigation of meetings as an inter-team coordination mechanism in large scale agile software development [56]	2018	Public department, office automation system	planned based on PRINCE2, development Scrum	large
			Norway's largest IT-programme	planned based on PRINCE2, development Scrum	
23	Inter-team coordination mechanisms in large-scale agile [66]	2017	Pension, bank and insurance	scrumban	large
				scrumban	
				scrum	

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
24	Interpretative case studies on agile team productivity and management [19]	2013	Financial industry	adopt Scrum + XP practices	small
			e-Commerce and infrastructure services	adopts Scrum, XP and lean principles and practices	small
			Internet content and access provision, recommendation system	adopts mainly Scrum and some XP and lean principles and practices	small
25	Why Scrum Works: A Case Study from an Agile Distributed Project in Denmark and India [75]	2011	Financial services, bank	Scrum	small
26	Inter-Team Coordination in Large Agile Software Development Settings: Five Ways of Practicing Agile at Scale [7]	2016	Enterprise Software	Scrum	large
					large
					large
					large
					large

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
27	Information Flow within a Dispersed Agile Team: A Distributed Cognition Perspective [34]	2012	Enterprise software components (in-house)	XP@Scrum	middle
28	Distributed agile: project management in a global environment [44]	2010	my Yahoo!	isolated scrums, distributed SoS, Fully distributed Scrum (with XP practices)	large
29	Using Agile Practices to Build Trust in an Agile Team: A case study [50]	2010	Financial services orga	customized agile	large
30	Coevolving Systems and the Organization of Agile Software Development [99]	2009	Network security and management systems, small software house	XP	small
			IT products and services, application for internal use	traditional waterfall	small

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
31	Essential communication practices for Extreme Programming in a global software development team [43]	2006	Telecommunications software and hardware industry	XP, GSD	small
32	Using open spaces to resolve cross team issue [95]	2005	Collaborative project, resource and portfolio management solutions	Waterfall Scrum, SoS	middle
33	A Qualitative Study of the Determinants of Self-managing Team E□ectiveness in a Scrum Team [61]	2011	Software Development Company, Software for small appliances	scrum	small
34	Adopting Continuous Delivery and Deployment: Impacts on Team Structures, Collaboration and Responsibilities [87]	2017	Consulting and IT Services, Financial, e-Commerce, Telecommunication Services		no information

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
35	Networking in a large-scale distributed agile project [57]	2014	Telecommunication, Ericsson	Scrum	large
36	Towards an Understanding of Tailoring Scrum in Global Software Development: A Multi-case Study [25]	2011	Process Industry Machinery and Systems, Paper Mill Industry	Scrum with tailored scrum practices	small
			Power, Energy and Oil Refinery Industry		small
			Enterprise Collaboration Software		small
			Telecommunication industry		small
37	Transition from a Plan-Driven Process to Scrum – A Longitudinal Case Study on Software Quality [38]	2010	IT Services	plan-driven then agile scrum	small

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
38	Control based management to self organizing agile teams - A case study [76]	2018	Digital Factory Process Industries, Motion Control Product	Scrum	small
39	Transitioning from plan-driven to lean in a global software engineering organization: a practice-centric view [80]	2018	Healthcare industry, safety-critical systems	Scrum with lean principles	large
40	From Scrum to Agile: a Journey to Tackle the Challenges of Distributed Development in an Agile Team [45]	2018	-	-	-
41	How autonomy emerges as agile cross-functional teams mature [46]	2018	Public sector, Norwegian Labor and Welfare Services	Scrum, water-scrum-fall	small
42	A project planning and development process for small teams [79]	1993	Linguistics	Spiral life cycle	small

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
43	Team Software Development Techniques [49]	1986	-	waterfall	no information
44	Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack [92]	2020	Engineering domain	Scrum, Kanban, ScrumBan, (7% waterfall)	large large
45	Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects [73]	2015	Global communications software and services	Scrum	large
46	Communities of practice in a large distributed agile software development organization – Case Ericsson [71]	2014	Telecommunications infrastructure	distributed Scrum, ScrumBan with CoPs	large
47	Transitioning from a First Generation to Second Generation Large-Scale Agile Development Method: Towards Understanding Implications for Coordination [10]	2020	Public organization	Scrum	large

Continued on next page

Table C.1 – *Continued from previous page*

No.	Title	Year	Domain	Method	Size
48	Meshing agile and plan-driven development in safety-critical software: a case study [33]	2020	Safety-critical software	Scrum in plan-driven V-model	large
49	Agile Autonomous Teams in Complex Organizations [52]	2019	financial	agile	middle
50	Large-scale agile transformation at Ericsson: a case study [72]	2018	Telecommunications	Scrum	large
51	Inter-organizational Co-development with Scrum: Experiences and Lessons Learned from a Distributed Corporate Development Environment [78]	2013	IT Services	Scrum,SoS	middle
52	Integration of Software Engineering Techniques Through the Use of Architecture, Process, and People Management: An Experience Report [63]	2005	Telecommunications	XP	middle
53	Cross-Continent Development Using Scrum and XP [37]	2003	IT Services	XP, Scrum, SoS	large

List of Tables

2.1 Project and team management in traditional and agile settings [2, 60] 10

3.1 Definition of the key terms 13
3.2 List of the sub-activities in each phase of the systematic literature review. The activities marked with * are not mandatory. 15

4.1 Inclusion criteria 23
4.2 Exclusion criteria 24
4.3 Overview of extracted literature for each database in each step 24

5.1 Overview of the team size reported in the selected literature. The cases where a size range is given were multiple case studies. 30
5.2 Overview of communication practices 42
5.3 Overview of coordination practices 45
5.4 Overview of collaboration practices 47
5.5 Overview of practices for decision-making and control 50

7.1 Organizational characteristics of traditional and agile development teams 62
7.2 The level of control for communication, coordination and decision-making authority 62
7.3 Template matrix: Templates A1-A4 and T1-T4 based on the level of control for communication, coordination and decision-making authority. Agile organizational practices dominate the templates A1 to A2, whereas templates T1 to T4 have more traditional organizational characteristics. 62

B.1 Content of the data extraction form, the data items marked with * are optional 77

C.1 List of the literature included in the systematic literature review 80

Bibliography

[1] Bibek Acharya and Ricardo Colomo-Palacios. “A Systematic Literature Review on Autonomous Agile Teams”. In: *2019 19th International Conference on Computational Science and Its Applications (ICCSA)*. IEEE, 2019, pp. 146–151. isbn: 978-1-7281-2847-4. doi: 10.1109/ICCSA.2019.00014. url: https://doi.org/10.1109/ICCSA.2019.00014.

[2] Rehan Akbar and Sohail Safdar. “A short review of Global Software Development (GSD) and latest software development trends”. In: *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*. 2015, pp. 314–317. doi: 10.1109/I4CT.2015.7219588.

[3] Kitchenham BA and Stuart Charters. “Guidelines for performing Systematic Literature Reviews in Software Engineering”. In: 2 (Jan. 2007).

[4] Eric Babinet and Rajani Ramanathan. “Dependency Management in a Large Agile Environment”. In: *Agile 2008 Conference*. IEEE, 2008, pp. 401–406. isbn: 978-0-7695-3321-6. doi: 10.1109/Agile.2008.58.

[5] Leonor Barroca, Torgeir Dingsøyr, and Marius Mikalsen. “Agile Transformation: A Summary and Research Agenda from the First International Workshop”. In: Aug. 2019, pp. 3–9. isbn: 978-3-030-30125-5. doi: 10.1007/978-3-030-30126-2_1.

[6] R. C. Beckett. “An integrative approach to project management in a small team developing a complex product”. In: *2008 IEEE International Conference on Industrial Engineering and Engineering Management*. IEEE, 2008, pp. 1028–1032. isbn: 978-1-4244-2629-4. doi: 10.1109/IEEM.2008.4738026. url: https://doi.org/10.1109/IEEM.2008.4738026.

[7] Saskia Bick, Alexander Scheerer, and Kai Spohrer. “Inter-Team Coordination in Large Agile Software Development Settings: Five Ways of Practicing Agile at Scale”. In: *Proceedings of the Scientific Workshop Proceedings of XP2016*. New York, NY, USA: ACM, 2016, pp. 1–5. isbn: 9781450341349. doi: 10.1145/2962695.2962699.

[8] Saskia Bick et al. “Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings: this is the detailed description of one of the SAP cases by Bick”. In: *IEEE Transactions on Software Engineering, title=Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings* 44.10 (2018), pp. 932–950. issn: 1939-3520. doi: 10.1109/TSE.2017.2730870.

[9] S. Bick et al. “Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings”. In: *IEEE Transactions on Software Engineering* 44.10 (2018), pp. 932–950. doi: 10.1109/TSE.2017.2730870.

[10] Finn Olav Bjørnson and Torgeir Dingsøyr. “Transitioning from a First Generation to Second Generation Large-Scale Agile Development Method: Towards Understanding Implications for Coordination”. In: *Agile Processes in Software Engineering and Extreme Programming – Workshops*. Ed. by Maria Paasivaara and Philippe Kruchten. Vol. 396. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2020, pp. 84–91. isbn: 978-3-030-58857-1. doi: 10.1007/978-3-030-58858-8{_}9.

[11] B. Boehm and R. Turner. “Using risk to balance agile and plan-driven methods”. In: *Computer* 36.6 (2003), pp. 57–66. doi: 10.1109/MC.2003.1204376.

[12] Pearl Brereton et al. “Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain”. In: *J. Syst. Softw.* 80.4 (Apr. 2007), pp. 571–583. issn: 0164-1212. doi: 10.1016/j.jss.2006.07.009. url: https://doi.org/10.1016/j.jss.2006.07.009.

[13] Manfred Broy and Marco Kuhrmann. “Projektorganisation und Management im Software Engineering”. In: Springer-Verlag, 2013, pp. XVI, 416. isbn: 978-3-642-29289-7. doi: 10.1007/978-3-642-29290-3.

[14] T. Chau, F. Maurer, and G. Melnik. “Knowledge sharing: agile methods vs. Tayloristic methods”. In: *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* 2003, pp. 302–307. doi: 10.1109/ENABL.2003.1231427.

[15] *"communication, n."* In: *OED Online*. Oxford University Press, Mar. 2021. url: https://oed.com/view/Entry/37309?redirectedFrom=communication (visited on 03/08/2021).

[16] Daniela S. Cruzes and Tore Dybå. “Research synthesis in software engineering: A tertiary study”. In: *Information and Software Technology* 53.5 (2011). Special Section on Best Papers from XP2010, pp. 440–455. issn: 0950-5849. doi: https://doi.org/10.1016/j.infsof.2011.01.004. url: https://www.sciencedirect.com/science/article/pii/S095058491100005X.

[17] José Adson O.G. Cunha, Hermano P. Moura, and Francisco J.S. Vasconcellos. “Decision-making in Software Project Management: A Systematic Literature Review”. In: *Procedia computer science* 100 (2016), pp. 947–954. issn: 1877-0509. doi: 10.1016/j.procs.2016.09.255. url: https://doi.org/10.1016/j.procs.2016.09.255.

[18] A. Danait. “Agile o□shore techniques - a case study”. In: *Agile Development Conference (ADC’05)*. IEEE Comput. Soc, 2005, pp. 214–217. isbn: 0-7695-2487-7. doi: 10.1109/ADC.2005.9.

[19] Claudia de O. Melo et al. “Interpretative case studies on agile team productivity and management”. In: *Information and Software Technology* 55.2 (2013). Special Section: Component-Based Software Engineering (CBSE), 2011, pp. 412–427. issn: 0950-5849. doi: https://doi.org/10.1016/j.infsof.2012.09.004. url: https://www.sciencedirect.com/science/article/pii/S0950584912001875.

[20] *"decision-making, (n.d)"*. In: *Collins English Dictionary – Complete and Unabridged*. 2014. url: https://www.thefreedictionary.com/decision-making (visited on 04/24/2021).

[21] J. F. DeFranco and P. A. Laplante. “Review and Analysis of Software Development Team Communication Research”. In: *IEEE Transactions on Professional Communication* 60.2 (2017), pp. 165–182. doi: 10.1109/TPC.2017.2656626. url: https://doi.org/10.1109/TPC.2017.2656626.

[22] Joanna DeFranco and Phillip Laplante. “A software engineering team research mapping study”. In: *Team Performance Management* 24 (June 2018), pp. 203–248. doi: 10.1108/TPM-08-2017-0040. url: https://doi.org/10.1108/TPM-08-2017-0040.

[23] Torgeir Dingsøyr, Nils Brede Moe, and Eva Amdahl Seim. “Coordinating Knowledge Work in Multiteam Programs: Findings From a Large-Scale Agile Development Program”. In: *Project Management Journal* 49.6 (2018), pp. 64–77. doi: 10.1177/8756972818798980. eprint: https://doi.org/10.1177/8756972818798980. url: https://doi.org/10.1177/8756972818798980.

[24] Tore Dyba and Torgeir Dingsoyr. “Agile Project Management: From Self-Managing Teams to Large-Scale Development”. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. IEEE, 2015, pp. 945–946. isbn: 978-1-4799-1934-5. doi: 10.1109/ICSE.2015.299.

[25] Emam Hossain, Paul L Bannerman, and Ross Je□ery, eds. *Towards an Understanding of Tailoring Scrum in Global Software Development: A Multi-case Study: Proceedings of the 2011 International Conference on Software and Systems Process*. New York, NY: ACM, 2011. isbn: 9781450307307. doi: 10.1145/1987875.1987894.

[26] Ismael Edrein Espinosa-Curiel et al. “Analysis of the changes in communication and social interactions during the transformation of a traditional team into an agile team”. In: *Journal of Software: Evolution and Process* 30.9 (2018), e1946. issn: 20477473. doi: 10.1002/smr.1946.

[27] Robert Feldt et al. “Four commentaries on the use of students and professionals in empirical software engineering experiments”. In: (Jan. 2018).

[28] FO Bjørnson et al., eds. *Inter-team Coordination in Large-Scale Agile Development: A Case Study of Three Enabling Mechanisms*. 2018. doi: 10.1007/978-3-319-91602-6{_}15.

[29] Cristiano P. Godoy et al. “Blueprint Model: A new Approach to Scrum Agile Methodology”. In: *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2019, pp. 95–99. isbn: 978-1-5386-9196-0. doi: 10.1109/ICGSE.2019.00014.

[30] Rajeev Kumar Gupta and Prabhulinga Manik Reddy. “Adapting Agile in a Globally Distributed Software Development”. In: *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, 2016, pp. 5360–5367. isbn: 978-0-7695-5670-3. doi: 10.1109/HICSS.2016.663.

[31] Rajeev Kumar Gupta, Mekanathan Venkatachalapathy, and Ferose Khan Jeberla. “Challenges in Adopting Continuous Delivery and DevOps in a Globally Distributed Product Team: A Case Study of a Healthcare Organization”. In: *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2019, pp. 30–34. isbn: 978-1-5386-9196-0. doi: 10.1109/ICGSE.2019.00020.

[32] Rajeev Kumar Gupta et al. “Key Factors in Scaling up Agile Team in Matrix Organization”. In: *Proceedings of the 12th Innovations on Software Engineering Conference (formerly known as India Software Engineering Conference) - ISEC'19*. Ed. by Ravindra Naik et al. New York, New York, USA: ACM Press, 2019, pp. 1–5. isbn: 9781450362153. doi: 10.1145/3299771.3299793.

[33] Lise Tordrup Heeager and Peter Axel Nielsen. “Meshing agile and plan-driven development in safety-critical software: a case study”. In: *Empirical Software Engineering* 25.2 (2020), pp. 1035–1062. issn: 1382-3256. doi: 10.1007/s10664-020-09804-z.

[34] Helen Sharp, Rosalba Giu□rida, and Grigori Melnik, ed. *Information Flow within a Dispersed Agile Team: A Distributed Cognition Perspective*. XP 2012: Agile Processes in Software Engineering and Extreme Programming. 2012. doi: 10.1007/978-3-642-30350-0{_}5.

[35] R. Hoda, J. Noble, and S. Marshall. “Self-Organizing Roles on Agile Software Development Teams”. In: *IEEE Transactions on Software Engineering* 39.3 (2013), pp. 422–444. doi: 10.1109/TSE.2012.30.

[36] Paula Jarzabkowski, Jane Lê, and Martha Feldman. “Toward a Theory of Coordinating: Creating Coordinating Mechanisms in Practice”. In: *Organization Science* 23 (Aug. 2012), pp. 907–927. doi: 10.2307/23252441.

[37] Bent Jensen and Alex Zilmer. “Cross-Continent Development Using Scrum and XP”. In: *LNCS* 2675 (2003), pp. 146–153.

[38] Jingyue Li, Nils B. Moe, and Tore Dyba, eds. *Transition from a Plan-Driven Process to Scrum – A Longitudinal Case Study on Software Quality*. 2010. doi: 10.1145/1852786.1852804.

[39] Nesma Keshta and Yasser Morgan. “Comparison between traditional plan-based and agile software processes according to team size & project domain (A systematic literature review)”. In: *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2017, pp. 567–575. isbn: 978-1-5386-3371-7. doi: 10.1109/IEMCON.2017.8117128. url: https://doi.org/10.1109/IEMCON.2017.8117128.

[40] Suhyun Kim et al. “Our Journey to Becoming Agile: Experiences with Agile Transformation in Samsung Electronics”. In: *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2016, pp. 377–380. isbn: 978-1-5090-5575-3. doi: 10.1109/APSEC.2016.064.

[41] Andreas Kornstädt and Joachim Sauer. “Mastering Dual-Shore Development – The Tools and Materials Approach Adapted to Agile O□shoring”. In: (2007). doi: 10.1007/978-3-540-75542-5_7. url: https://doi.org/10.1007/978-3-540-75542-5_7.

[42] Maarit Laanti. “Implementing Program Model with Agile Principles in a Large Software Development Organization”. In: *2008 32nd Annual IEEE International Computer Software and Applications Conference*. IEEE, 2008, pp. 1383–1391. isbn: 978-0-7695-3262-2. doi: 10.1109/COMPSAC.2008.116.

[43] Lucas Layman et al. “Essential communication practices for Extreme Programming in a global software development team”. In: *Information and Software Technology* 48.9 (2006), pp. 781–794. issn: 0950-5849. doi: 10.1016/j.infsof.2006.01.004.

[44] Seiyoung Lee and Hwan-Seung Yong. “Distributed agile: project management in a global environment”. In: *Empirical Software Engineering* 15.2 (2010), pp. 204–217. issn: 1382-3256. doi: 10.1007/s10664-009-9119-7.

[45] Pernille Lous et al. “From Scrum to Agile: a Journey to Tackle the Challenges of Distributed Development in an Agile Team”. In: *Proceedings of the 2018 International Conference on Software and System Process*. Ed. by Marco Kuhrmann, Rory V. O’Connor, and Dan Houston. New York, NY, USA: ACM, 2018, pp. 11–20. isbn: 9781450364591. doi: 10.1145/3202710.3203149.

[46] Kjell Lundene and Parastoo Mohagheghi. “How autonomy emerges as agile cross-functional teams mature”. In: *Proceedings of the 19th International Conference on Agile Software Development: Companion*. Ed. by Ademar Aguiar. New York, NY, USA: ACM, 2018, pp. 1–5. isbn: 9781450364225. doi: 10.1145/3234152.3234184.

[47] F. Lunenburg. “Mechanistic-Organic Organizations — An Axiomatic Theory : Authority Based on Bureaucracy or Professional Norms”. In: 2012.

[48] Thomas W. Malone and Kevin Crowston. “The Interdisciplinary Study of Coordination”. In: *ACM Comput. Surv.* 26.1 (Mar. 1994), pp. 87–119. issn: 0360-0300. doi: 10.1145/174666.174668. url: https://doi.org/10.1145/174666.174668.

[49] Mary McGuir, ed. *Team Software Development Techniques*. Association for Computing Machinery - New YorkNYUnited States, 1986. doi: 10.1145/317210.317234.

[50] Orla McHugh, Kieran Conboy, and Michael Lang. “Using Agile Practices to Build Trust in an Agile Team: A Case Study”. In: *Information Systems Development*. Ed. by Jaroslav Pokorny et al. New York, NY: Springer New York, 2011, pp. 503–516. isbn: 978-1-4419-9790-6.

[51] Germano Duarte Mergel, Milene Selbach Silveira, and Tiago Silva da Silva. “A Method to Support Search String Building in Systematic Literature Reviews through Visual Text Mining”. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC ’15. Salamanca, Spain: Association for Computing Machinery, 2015, pp. 1594–1601. isbn: 9781450331968. doi: 10.1145/2695664.2695902. url: https://doi.org/10.1145/2695664.2695902.

[52] Marius Mikalsen et al. “Agile Autonomous Teams in Complex Organizations”. In: *Agile Processes in Software Engineering and Extreme Programming – Workshops*. Ed. by Rashina Hoda. Vol. 364. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2019, pp. 55–63. isbn: 978-3-030-30125-5. doi: 10.1007/978-3-030-30126-2{_}7.

[53] N. B. Moe, T. Dingsøyr, and T. Dybå. “Understanding Self-Organizing Teams in Agile Software Development”. In: *19th Australian Conference on Software Engineering (aswec 2008)*. 2008, pp. 76–85. doi: 10.1109/ASWEC.2008.4483195.

[54] N. B. Moe, Torgeir Dingsøyr, and Knut Rolland. “To schedule or not to schedule? An investigation of meetings as an inter-team coordination mechanism in largescale agile software development”. In: *International Journal of Information Systems and Project Management* 6.3 (2018), pp. 45–59. doi: 10.12821/ijispm060303.

[55] Nils Brede Moe et al. “Coaching a Global Agile Virtual Team”. In: *2015 IEEE 10th International Conference on Global Software Engineering*. IEEE, 2015, pp. 33–37. isbn: 978-1-4799-8409-1. doi: 10.1109/ICGSE.2015.26. url: https://doi.org/10.1109/ICGSE.2015.26.

[56] Nils Brede Moe et al. “Enabling Knowledge Sharing in Agile Virtual Teams”. In: *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2016, pp. 29–33. isbn: 978-1-5090-2680-7. doi: 10.1109/ICGSE.2016.30. url: https://doi.org/10.1109/ICGSE.2016.30.

[57] Nils Brede Moe et al. “Networking in a large-scale distributed agile project”. In: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM ’14*. Ed. by Maurizio Morisio. New York, New York, USA:

ACM Press, 2014, pp. 1–8. isbn: 9781450327749. doi: 10.1145/2652524.2652584.

[58] Parastoo Mohagheghi, Casper Lassenius, and Ingrid Omang Bakken. "Enabling Team Autonomy in a Large Public Organization". In: *Agile Processes in Software Engineering and Extreme Programming – Workshops*. Ed. by Maria Paasivaara and Philippe Kruchten. Vol. 396. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2020, pp. 245–252. isbn: 978-3-030-58857-1. doi: 10.1007/978-3-030-58858-8{_}25.

[59] Karthik Mohanarangam. "Transitioning to Agile—In a Large Organization". In: *IT Professional* 22.2 (2020), pp. 67–72. issn: 1520-9202. doi: 10.1109/MITP.2019.2902345. url: https://doi.org/10.1109/MITP.2019.2902345.

[60] A B M Moniruzzaman and Syed Hossain. "Comparative Study on Agile software development methodologies". In: (July 2013).

[61] Monteiro, Cleviton V.F. and da Silva, Fabio Q.B. and dos Santos, Isabella R.M. and Farias, Felipe and Cardozo, Elisa S.F. and do A. Leitão, André R.G. and Neto, Dacio N.M. and Pernambuco Filho, Miguel J.A. "A Qualitative Study of the Determinants of Self-managing Team E□ectiveness in a Scrum Team". In: 2011. doi: 10.1145/1984642.1984646.

[62] Erik Moore. "Influence of Large-Scale Organization Structures on Leadership Behaviors". In: *2009 Agile Conference*. IEEE, 2009, pp. 309–313. isbn: 978-0-7695-3768-9. doi: 10.1109/AGILE.2009.14.

[63] Christopher Nelson and Jung Soo Kim. "Integration of Software Engineering Techniques Through the Use of Architecture, Process, and People Management: An Experience Report". In: *LNCS 3475* - (2005), pp. 1–10.

[64] Anh Nguyen-Duc, Daniela S. Cruzes, and Reidar Conradi. "The impact of global dispersion on coordination, team performance and software quality – A systematic literature review". In: *Information and Software Technology* 57 (2015), pp. 277–294. issn: 0950-5849. doi: https://doi.org/10.1016/j.infsof.2014.06.002. url: https://www.sciencedirect.com/science/article/pii/S0950584914001414.

[65] Ramon Noordeloos, Christina Manteli, and Hans van Vliet. "From RUP to Scrum in Global Software Development: A Case Study". In: *2012 IEEE Seventh International Conference on Global Software Engineering*. IEEE, 2012, pp. 31–40. isbn: 978-1-4673-2357-4. doi: 10.1109/ICGSE.2012.11.

[66] Helga Nyrud and Viktoria Stray. “Inter-team coordination mechanisms in large-scale agile”. In: *Proceedings of the XP2017 Scientific Workshops on - XP ’17*. Ed. by Roberto Tonelli. New York, New York, USA: ACM Press, 2017, pp. 1–6. isbn: 9781450352642. doi: 10.1145/3120459.3120476.

[67] *"organize, v."* In: *OED online*. Oxford University Press, Mar. 2021. url: https://oed.com/view/Entry/132456?redirectedFrom=organize (visited on 03/09/2021).

[68] *"organizing, (n.d)"*. In: *Collins English Dictionary – Complete and Unabridged*. 2014. url: https://www.thefreedictionary.com/organizing (visited on 04/24/2021).

[69] Maria Paasivaara, Ville T. Heikkila, and Casper Lassenius. “Experiences in Scaling the Product Owner Role in Large-Scale Globally Distributed Scrum”. In: *2012 IEEE Seventh International Conference on Global Software Engineering*. IEEE, 2012, pp. 174–178. isbn: 978-1-4673-2357-4. doi: 10.1109/ICGSE.2012.41.

[70] Maria Paasivaara and Casper Lassenius. “Communities of practice in a large distributed agile software development organization – Case Ericsson”. In: *Information and Software Technology* 56.12 (2014), pp. 1556–1577. issn: 0950-5849. doi: 10.1016/j.infsof.2014.06.008.

[71] Maria Paasivaara and Casper Lassenius. “Communities of practice in a large distributed agile software development organization – Case Ericsson”. In: *Information and Software Technology* 56.12 (2014), pp. 1556–1577. issn: 0950-5849. doi: 10.1016/j.infsof.2014.06.008.

[72] Maria Paasivaara et al. “Large-scale agile transformation at Ericsson: a case study”. In: *Empirical Software Engineering* 23.5 (2018), pp. 2550–2596. issn: 1382-3256. doi: 10.1007/s10664-017-9555-8.

[73] Georgios Papadopoulos. “Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects”. In: *Procedia - Social and Behavioral Sciences* 175 (2015), pp. 455–463. issn: 18770428. doi: 10.1016/j.sbspro.2015.01.1223.

[74] Ravi Paul, John R. Drake, and Huigang Liang. “Global Virtual Team Performance: The E□ect of Coordination E□ectiveness, Trust, and Team Cohesion”. In: *IEEE Transactions on Professional Communication* 59.3 (2016), pp. 186–202. issn: 0361-1434. doi: 10.1109/TPC.2016.2583319. url: https://doi.org/10.1109/TPC.2016.2583319.

[75] Lene Pries-Heje and Jan Pries-Heje, eds. *Why Scrum Works: A Case Study from an Agile Distributed Project in Denmark and India: ??? like a SoS example*. 2011. doi: 10.1109/AGILE.2011.34.

[76] B. V. Rajeev and Vinod Hejib. "Control based management to self organizing agile teams - A case study". In: *Proceedings of the 13th International Conference on Global Software Engineering*. Ed. by Maria Paasivaara, Darja Šmite, and Roberto Evaristo. New York, NY, USA: ACM, 2018. isbn: 9781450357173. doi: 10.1145/3196369.3196394.

[77] R. ChandanaK. Ranasinghe and Indika Perera. "E□ectiveness of scrum for o□shore software development in Sri Lanka". In: *2015 Moratuwa Engineering Research Conference (MERCon)*. IEEE, 2015, pp. 306–311. isbn: 978-1-4799-1740-2. doi: 10.1109/MERCon.2015.7112364. url: https://doi.org/10.1109/MERCon.2015.7112364.

[78] Raoul Vallon, Stefan Strobl, Mario Bernhart, Thomas Grechenig. "Inter-organizational Co-development with Scrum: Experiences and Lessons Learned from a Distributed Corporate Development Environment: LNBIP 149 -". In: *xp 2013* (2013).

[79] Marc Rettig and Gary Simons. "A project planning and development process for small teams". In: *Communications of the ACM* 36.10 (1993), pp. 45–55. issn: 0001-0782. doi: 10.1145/163430.163440.

[80] Roopa M. S., Ratnanabh Kumar, and V. S. Mani. "Transitioning from plan-driven to lean in a global software engineering organization: a practice-centric view". In: *Proceedings of the 13th International Conference on Global Software Engineering*. Ed. by Maria Paasivaara, Darja Šmite, and Roberto Evaristo. New York, NY, USA: ACM, 2018, pp. 1–5. isbn: 9781450357173. doi: 10.1145/3196369.3196395.

[81] Je□rey S. Saltz and Ivan Shamshurin. "Achieving Agile Big Data Science: The Evolution of a Team's Agile Process Methodology". In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 3477–3485. isbn: 978-1-7281-0858-2. doi: 10.1109/BigData47090.2019.9005493.

[82] Steve Sawyer. "Software development teams". In: *Communications of the ACM* 47.12 (2004), pp. 95–99. issn: 0001-0782. doi: 10.1145/1035134.1035140.

[83] Andre Scandaroli et al. "Behavior-Driven Development as an Approach to Improve Software Quality and Communication Across Remote Business Stakeholders, Developers and QA: two Case Studies". In: *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2019, pp. 105–110. isbn: 978-1-5386-9196-0. doi: 10.1109/ICGSE.2019.00030.

[84] *Scrum Guide*. url: https://scrumguides.org/scrum-guide.html#scrum-master (visited on 04/21/2021).

[85] *Scrum.org - What is a Scrum Master?* url: https://www.scrum.org/resources/what-is-a-scrum-master (visited on 04/21/2021).

[86] C. Sepulveda. "Agile development and remote teams: learning to love the phone". In: *Proceedings of the Agile Development Conference, 2003. ADC 2003*. IEEE, 2003, pp. 140–145. isbn: 0-7695-2013-8. doi: 10.1109/ADC.2003.1231464. url: https://doi.org/10.1109/ADC.2003.1231464.

[87] Mojtaba Shahin et al. "Adopting Continuous Delivery and Deployment: Impacts on Team Structures, Collaboration and Responsibilities". In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering - EASE'17*. Ed. by Emilia Mendes, Steve Counsell, and Kai Petersen. New York, New York, USA: ACM Press, 2017, pp. 384–393. isbn: 9781450348041. doi: 10.1145/3084226.3084263.

[88] Helen Sharp and Hugh Robinson. "Three 'C's of Agile Practice: Collaboration, Co-ordination and Communication". In: *Agile Software Development: Current Research and Future Directions*. Ed. by Torgeir Dingsøyr, Tore Dybå, and Nils Brede Moe. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 61–85. isbn: 978-3-642-12575-1. doi: 10.1007/978-3-642-12575-1_4. url: https://doi.org/10.1007/978-3-642-12575-1_4.

[89] Helen Sharp et al. *Remote working in an Agile team*. Tech. rep. Agile Research Network, 2016. url: http://agileresearchnetwork.org/wp-content/uploads/2016/09/Remoteworkingwhitepaper.pdf (visited on 03/12/2021).

[90] Darja Šmite, Nils Brede Moe, and Richard Torkar. "Pitfalls in Remote Team Coordination: Lessons Learned from a Case Study". In: *Product-Focused Software Process Improvement*. Ed. by Andreas Jedlitschka and Outi Salo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 345–359. isbn: 978-3-540-69566-0. url: https://doi.org/10.1007/978-3-540-69566-0_28.

[91] Hubert Smits and Guy Pshigoda. "Implementing Scrum in a Distributed Software Development Organization". In: *Agile 2007 (AGILE 2007)*. IEEE, 2007, pp. 371–375. isbn: 0-7695-2872-4. doi: 10.1109/AGILE.2007.34.

[92] Viktoria Stray and Nils Brede Moe. "Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack". In: *Journal of Systems and Software* 170 (2020), p. 110717. issn: 01641212. doi: 10.1016/j.jss.2020.110717.

[93] Je Sutherland et al. “Distributed Scrum: Agile Project Management with Outsourced Development Teams”. In: *40th Hawaii International Conference on System Sciences 2007* (2007). doi: 10.1109/HICSS.2007.180.

[94] David Marcell Szabo and Jan-Philipp Steghofer. “Coping Strategies for Temporal, Geographical and Sociocultural Distances in Agile GSD: A Case Study”. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 161–170. isbn: 978-1-7281-1760-7. doi: 10.1109/ICSE-SEIP.2019.00025.

[95] C. M. Tartaglia and P. Ramnath. “Using open spaces to resolve cross team issues”. In: *Agile Development Conference (ADC’05)*. IEEE Comput. Soc, 2005, pp. 173–179. isbn: 0-7695-2487-7. doi: 10.1109/ADC.2005.49.

[96] Omer Uludag et al. “Investigating the Adoption and Application of Large-Scale Scrum at a German Automobile Manufacturer”. In: *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2019, pp. 22–29. isbn: 978-1-5386-9196-0. doi: 10.1109/ICGSE.2019.00019.

[97] Uwe van Heesch. “Collaboration patterns for oshore software development”. In: *Proceedings of the 20th European Conference on Pattern Languages of Programs - EuroPLoP ’15*. Ed. by Veli-Pekka Eloranta and Claudius Link. New York, New York, USA: ACM Press, 2015, pp. 1–10. isbn: 9781450338479. doi: 10.1145/2855321.2855343. url: https://doi.org/10.1145/2855321.2855343.

[98] Andrew H. Van De Ven, Andre L. Delbecq, and Richard Koenig. “Determinants of Coordination Modes within Organizations”. In: *American Sociological Review* 41.2 (1976), pp. 322–338. issn: 00031224. url: http://www.jstor.org/stable/2094477.

[99] Richard Vidgen and Xiaofeng Wang. “Coevolving Systems and the Organization of Agile Software Development”. In: *Info. Sys. Research* 20.3 (Sept. 2009), pp. 355–376. issn: 1526-5536. doi: 10.1287/isre.1090.0237. url: https://doi.org/10.1287/isre.1090.0237.

[100] Jessica K. Winkler, Jens Dibbern, and Armin Heinzl. “The Impact of Cultural Dierences in Oshore Outsourcing: Case Study Results from German–Indian Application Development Projects”. In: *Information Systems Outsourcing* (2009). url: https://doi.org/10.1007/s10796-008-9068-5.

[101] Claes Wohlin et al. “Experimentation in Software Engineering”. In: Springer-Verlag Berlin Heidelberg, 2012, p. 236. isbn: 978-3-642-29044-2. doi: 10.1007/978-3-642-29044-2.

[102] Monica Yap. *Successful Distributed Agile Team Working Patterns*. 2010. url: https://api.semanticscholar.org/CorpusID:14506646 (visited on 04/26/2021).