Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Levi Sørum

Sharing of team knowledge in virtual software development teams

A case study

Master's thesis in Computer science, TDT4900
Supervisor: Torgeir Dingsøyr
Co-supervisor: Nils Brede Moe
March 2021

NTNU
Norwegian University of
Science and Technology

Levi Sørum

Sharing of team knowledge in virtual software development teams

A case study

Master's thesis in Computer science, TDT4900
Supervisor: Torgeir Dingsøyr
Co-supervisor: Nils Brede Moe
March 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Preface

Through the last years of my studies I've been fortunate to have a wonderful part-time job as a software developer at KnowIt. It is due to my employers that I was introduced to my supervisor Nils Brede Moe at SINTEF, who not only guided me through the process of deciding what to write about, but provided me with insightful knowledge throughout the project. It was also through my part-time work that I experienced how the COVID-19 pandemic caused a unique work situation, which gave me the interest and motivation to learn more. Prior to and during my master's project I was given the opportunity to work remotely, which was a new experience that introduced me to the challenges of not being co-located with one's colleagues. I quickly knew that I wanted to dedicate my thesis to research the new work circumstances for software developers caused by the lockdown.

I would like to thank my colleagues at KnowIt Objectnet who helped bring this project to life, and for being very understanding and flexible when I had to postpone the start-date of my full-time employment when the project took longer than expected. I would like to thank Nils Brede Moe for being a great guide, and for always providing me with a direction when I got lost in the project. I would like to thank my academic supervisor Torgeir Dingsøyr at NTNU; first for taking me as a master's student during the fall semester when many professors would only accept master's students during the spring; second for providing great guidance of how to do academic research; and lastly for proof-reading the thesis paper. Lastly, I would like to thank the participants of the study who were so kind as to show me how they work and share their insights with me.

On a more personal note, I would like to thank my family for always believing in me and for supporting me financially during the last few months so I could work on my thesis full-time, as well as to my room-mates whom I could always turn to for motivation when something was difficult. This has been a special time for me, and despite the challenges I can say that I am happy with the outcome.

I hope you enjoy your reading.

Levi Sørum

Trondheim, March 12, 2021

Abstract

Due to the COVID-19 pandemic lockdown, software development teams are forced to collaborate in highly virtual environments. With software development being knowledge intensive work, the field of knowledge sharing in virtual software development teams becomes increasingly relevant, but with a lacking amount of empirical research on this particular topic research possibilities are limited. This study describes the virtuality, knowledge sharing behavior and degree of shared team knowledge of a single virtual software development team.

The case team was observed at three occasions, and events demonstrating knowledge sharing were recorded. Next, semi-structured interviews were conducted with 5 of the team's members asking about their work situation and processes, as well as concrete questions which responses were later compared to measure an overlap of knowledge. Degree of virtuality and knowledge sharing activities were identified through qualitative analysis of the collected data, and the overlap measure of team-knowledge were found through semi-quantitative analysis by grouping similar statements and calculating the average number of respondents per statement. The results showed a moderate-to-high degree of virtuality, a moderate degree of shared team knowledge, and an array of knowledge sharing activities including Agile practices, use of knowledge management systems and certain meetings that had a higher focus on knowledge sharing.

A feedback meeting with the team's Scrum master supplemented our findings, and we learned that their degree of shared knowledge and knowledge sharing behavior was likely affected by the non-interdependence of team members. The research was limited by a small sample size and the inability to measure all types of shared team knowledge due to a limitation of scope. We recommend future researchers to conduct additional empirical studies of a similar nature, enabling the comparison and further study of knowledge sharing in virtual software development teams.

Contents

1 Introduction

Due to the outbreak of the COVID-19 pandemic in 2020, governments have introduced guidelines that impose social distancing and reduced travel on large populations. This has had significant implications for the software engineering industry, whose workers are commonly expected to work remotely from home, making software development teams highly distributed. As a consequence, software development teams are forced to communicate, collaborate and coordinate by using digital communication tools, thus becoming what is called a virtual team [7]. In short, virtual teams are teams whose communication, collaboration and/or coordination are enabled by communication technology because some members do not work in either the same place or at the same time, and therefore cannot collaborate face-to-face all the time [54].

The concept of virtual software engineering teams did not originate with the COVID-19 pandemic; globally distributed software projects have become quite common in the last decades, motivated by companies' desires for cost reduction, shorter time to market, access to a skilled labor pool, and increased innovation [56]. By removing the geographic boundaries of a project, virtual teams have become vital to maintaining our increasingly globalized social and economic infrastructure [47]. In other words, increased globalisation has long been a driving factor for increased virtuality of work teams. But it's not only global teams that are virtual. In fact, modern-day teams are often considered to be more-or-less virtual even if they are co-located, because their work rely on some form of technology [31]. Software development teams have been at least somewhat virtual for a long time due to the usage of technologies like version-control software and digital backlogs. What's new with the COVID-19 pandemic is the sudden requirement of social distancing and reduced travel, forcing team members to work remotely, drastically increasing their distribution and thus their degree of virtualness [35]. In other words, the pandemic has introduced a requirement for a high degree of team virtuality that has not been present before. Although increased team virtuality addresses the requirement of social distancing and reduced travel, it comes with its own set of challenges. Team virtualness has been associated with a lower communication frequency and a lower degree of knowledge sharing within teams [18], both of which are vital factors for the effectiveness of teams within the software industry [7].

Software engineering is a diverse and constantly evolving field, where knowledge is a central asset for a projects' success. Rus et al. (2002) [53] identified a set of motivations for why knowledge management is important in the field of software engineering. Much of the knowledge within an organization is acquired by its individuals through experience, and often involves a learning process of trial and error. Thus, the software engineering discipline is experimental in nature, and as a result, much of the acquired knowledge is held in the mind of the individual who learned it. An organization ideally want to re-use previous experiences for future projects, however if these experiences and practices are not captured, then future development teams will not benefit from them, and the team must acquire the knowledge all over again. The authors also found that while knowledge may often already exist in an organization, a lack of knowledge arises when it is not transferred to those who don't have it, and as a result, previous mistakes are repeated when they could have been avoided. And so the purpose of knowledge management is to organize acquisition of new knowledge, identify expertise, as well as capture, package, and share knowledge that exists in an organization. Knowledge sharing is an activity in which knowledge is moved from one donating party to another receiving party [58]. An exchange of knowledge takes place though either the donation or collection of knowledge, and can be performed on an individual or an organizational level [41]. Sharing knowledge is an essential group process for team effectiveness, it is needed for good decision-making and the building of the team's knowledge base, and it is a critical success factor for cross-functional teamwork [58]. A study by Staples et al. (2008) [58] examined the potential effects of different aspects of virtuality on knowledge sharing. They found that the degree of dispersion can affect the sharing of knowledge because co-location promotes both formal and informal contact and communication. Teams with more informal communication have greater cross-functional cooperation, which is more difficult in a distributed context. In other words, the study found that a high degree of dispersion negatively affects the amount of knowledge that is shared within a team, and that reduced knowledge sharing negatively affects team performance.

A type of knowledge that has been found to be especially important for virtual software development teams is *team knowledge.* According to Moe et al. (2016) [46] team members need to share the same understanding of many aspects of the team's work if virtual software development teams is to be successful. In other words, the members must share the same mental model of how the team works. Shared mental models allow team members to predict what their teammates are going to do and what they are going to need in order to do it , and so shared team knowledge provide mutual expectations that allow teams to implicitly coordinate without the need to constantly communicate [12]. In other words, team knowledge allows team members to make valid assumptions about the behavior of other team members, and thereby enable effective collaboration and cooperation despite being distributed [46].

According to Šmite et al. (2010) [57], a majority of the empirical research that has been previously performed on distributed software engineering projects have studied intra-organizational collaboration between two sites, and so there appear to be a lack of empirical research on highly-distributed software engineering projects where every participant is at their own remote site. In other words, there is a lack of studies on teams that are highly virtual. Furthermore, a majority of their identified studies has had a focus on the management aspects of virtual software engineering projects, and so there is little of research on the practical aspects of virtual software engineering, and this number is especially low in the field of knowledge management.

To summarize, there is an emergent requirement for a high degree of virtualness within software development teams, and the effectiveness of virtual software development teams rely on the successful sharing of team knowledge. Unfortunately, however, the studies on this particular topic is scarce. While it is desired to perform studies on how to improve the effectiveness of virtual software development teams, this is difficult without a sufficient empirical knowledge base. Recent events has not only made highly distributed virtual teams a relevant context to explore, but has also presented us with an excellent opportunity to do so. The main contribution of this master's project is to perform a descriptive case-study of a highly distributed, cross-functional virtual team within the software engineering industry, in order to learn which team-knowledge sharing activities they perform in a virtual setting. We will also evaluate the team's overlap of team knowledge, and use this as a measure of how well the team proceeds to share team-knowledge among each other. Thus we formulate two research questions:

- **R1:** Which team-knowledge sharing activities are used by the team?
- **R2:** What is the overlap measure of the team's shared team knowledge?

In other words, we are going to study a single case of a virtual software engineering team, in order to describe three things about that particular case. First, we wish to know which activities are used by the team that result in the sharing of team-knowledge (**R1**). A virtual team is not a binary or discrete phenomenon. There are as many flavors of virtual teams as there are virtual teams, and each team operate a little differently. If future researchers are to, for example, perform a systematic review in order to learn which practices lead to an improved sharing of team-knowledge within a virtual team, it is necessary that the reviewed studies include which practices are performed, as well as the result, which leads us to our next topic of study. The second topic that we wish to describe about our case is, simply put, how well the team achieves the sharing of team knowledge (**R2**). The term *team knowledge* will be described in more detail in the next chapter, but for now it will suffice to think of team knowledge as the sum of each individual team members' knowledge that is relevant and helps the team perform their tasks effectively. With the assumption that successful sharing of team knowledge leads to an increased "overlap" of shared team knowledge among its members, the second objective of this study is to measure the degree of overlap of the team's shared team knowledge.

The rest of this paper is organized as follows: Section 2 includes the background theory used to perform and discuss our analysis. Section 3 describes the method used to collect and analyse data. Section 4 shows the result of the analysis performed in the method section, and Section 5 includes a discussion of our findings. Section 6 will conclude our work and suggest directions for further research.

2 Background

Within this chapter we will review and present the findings of relevant literature in order to provide a theoretical background upon which this study is based. Section 1 of this chapter will review virtual teams, in terms of what makes a team virtual. As we will see, there are several ways to define a virtual team, and so we aim to identify which specific characteristics makes a team virtual. Moreover, we will explore the potential strengths and weaknesses of virtual teams. In Section 2 we will look deeper on the definitions of knowledge, as well as which characteristics knowledge may inhabit. Within Section 3 we extend on this knowledge in order to define shared mental models and, eventually, team knowledge, which is a type of knowledge highly relevant to virtual software development teams. We will also present a framework for shared team knowledge which will be very central for the remainder of this paper. In Section 4 we will review the basics of knowledge management by describing its relevance to the motivation of this study, categorizing different schools of knowledge management strategies, and presenting a research framework for knowledge management. In Section 5 we will build upon this knowledge in order to review knowledge sharing. We will first present a definition of knowledge sharing as well as factors that are known to influence knowledge sharing behavior. Then, finally, we will explore how Agile methodologies affect knowledge sharing within software development teams.

2.1 Virtual Teams

A *distributed team* consists of team members who are spread in different locations and work remotely on different parts and independent tasks of the project without any face-to-face interactions [32]. *Virtual teams* refer to teams that are geographically distributed and who rely on technology to communicate and collaborate [47], and the difference between distributed teams and virtual teams is that virtual teams work jointly on the same tasks [32]. According to Gheni et al. (2016) [25], the difference between traditional projects and virtual projects are the number of organizations and locations involved in the implementation of the project. In traditional projects a large majority of the team members are working for the same organization and in a single location, whereas virtual projects may be composed of team members in different organizations, dispersed geographically. Furthermore, the authors describe international projects as projects involving team members working in many locations across country borders, and global projects combine the characteristics of both virtual projects and international projects. *Global teams* are distributed teams working on international projects across country borders [47], and may be formed when an external company is subcontracted for providing software development services for a client company (outsourcing), or when a company creates its own software development centers located in different countries to handle the internal demand (offshoring) [32]. *Global virtual teams* (GVTs) combine the characteristics of global and virtual teams. They are groups of people who are working together from different places in the world with different languages and cultures, and who depend on information and communication technology to communicate with each other [25].

2.1.1 Definition of team virtuality

Hosseini et al. (2015) [31] conducted a literature review in order to construct a conceptual model for conceptualising team virtuality. They found that there are several different approaches to defining virtual teams. The first is the traditional approach, also called the dichotomy approach, through which team virtuality is a discrete category to which teams are either included or excluded based on a set of criteria. In other words, teams are either virtual or they are conventional (i.e. face-to-face). The main criteria for team virtuality is geographic dispersion of members, asynchronicity (i.e. temporal dispersion), reliance on information communication technologies, boundary spanning (e.g. cultural and nationality differences, or working for different organizations), temporality of teams (limited duration of team), and the fact that members are mutually accountable for a common purpose.

The dichotomy approach has several drawbacks, however. In the real world, a categorical approach fails to capture the whole picture, as a team may inhabit some of these criteria, or inhabit them to different degrees. In those cases, it is difficult to pinpoint a cut-off point of what constitutes a virtual team, and what doesn't. Another approach to defining team virtuality is one that the authors have named the virtuality approach. Researchers that follow this approach have regarded the dichotomy approach as an oversimplification of reality, as it is impossible to find modern-day teams that are purely virtual or traditional in contemporary organisations, because even co-located teams often rely on technology. The virtuality approach instead considers all teams along a continuum ranging from merely face-to-face to entirely virtual. In other words, team virtuality is a team characteristic of various levels, as opposed to a discrete category.

But even within the virtuality approach there are no agreed-upon conceptualisation for virtuality. Some approaches use discrete levels to classify degree of virtuality, while other approaches regard virtuality as a continuous measure. In the latter approach, researchers have introduced constructs or dimensions as the causes of virtuality increase in teams. These factors, which the authors refer to as constructs of virtuality, are factors which shift a team arrangement towards one extreme of the virtuality continuum, which are purely virtual teams. The other extreme of this continuum are the conventional co-located team working arrangements.

From several different sources of literature, Johnson et al. (2009) [35] summarized a list of dimensions that define a team's degree of virtualness, which are listed in Table 1. These factors were electronic dependence, geographic dispersion, level of technology support, percentage of time apart while working on a task, degree of physical distance, use of virtual communication tools, amount of informational value provided by those tools, synchronicity of communication, use of computer-mediated communication, temporality, and diversity. These keywords helps us paint a mental picture of what makes a team virtual, although several of these keywords are related and may be combined. Most of their reviewed literature, however, seemed to agree that team virtuality must include the use of computer-mediated communication. Furthermore, while there appeared to be a widespread agreement that virtual teams are geographically dispersed, some argue that geographic dispersion is not a prerequisite for virtual teams. In fact, Kirkman and Mathieu et al. (2005) [38] argued that the definitions of team virtualness that include a requirement for geographic dispersion makes the implicit assumption that when teams are co-located, then they are not likely to interface through virtual means. The authors contend that while teams with geographic and other forms of member dispersion are more likely to adopt more virtual means of communication and collaboration, a team with co-located members may also choose to employ virtual means of coordinating their actions, and that co-location does not prevent a team from being highly virtual. Gumm et al. (2006) [29] also claims that when evaluating a team's physical or geographical distribution, it is necessary to take the team's perceived distance into account, as opposed to only considering the team's actual physical distances. Perceived distance also applies to temporality. In a team with an established asynchronous communication culture temporal distance may seem lower than for a team that relies on highly synchronous communication mechanisms.

Dimensions of team virtualness
Electronic dependence
Geographic dispersion
Level of technology support
Percentage of time spent apart while working on a task
Degree of physical distance
Use of virtual communication tools
Amount of informational value provided by virtual communication tools
Synchronicity of communication
Use of computer-mediated communication
Temporality
Diversity

Table 1: Dimensions of team virtualness, as summarized by Johnson et al. (2009) [35]

2.1.2 Potential strengths and weaknesses

The geographical distribution of team members may be both a great strength as well as a potential weakness for virtual teams. Since virtual teams are not confined by geographical boundaries, the team may benefit from employing expertise from a much larger pool of candidates, enabling the assembly of teams that maximise functional expertise [21]. This increased pool of expertise can provide organizations with a competitive advantage [17]. An absence of geographic boundaries promotes knowledge sharing across organizational units and sites, and thus knowledge may be more broadly distributed within an organization [21]. Working in virtual teams also provides some benefits for the team member. Since team members are not required to commute into an office or work site, the time and cost spent on travel and relocation are significantly reduced [21]. The flexibility of working from home facilitates work-life balance and enhances employee satisfaction [17].

Many of the challenges faced by virtual teams are related to communication, collaboration, and coordination. The geographical and organizational dispersion of virtual teams may hinder communication and make coordination dependent on the team's shared knowledge [24]. Effective knowledge sharing is more difficult in virtual teams than in co-located teams , and for this reason virtual teams suffer a performance penalty [7]. When team members work remotely, there is no way to signal a person's availability for "spur-of-the-moment" informal communication, and as a result there are fewer instances of informal communication taking place than there would be in a co-located work environment [47]. In other words, virtual teams face a lower communication frequency than co-located teams, and may cause challenges when using agile approaches that require frequent communication.

The reduced communication frequency faced by virtual teams can lead to isolation and high levels of social distance between team members, making it more difficult to build trust and a shared sense of responsibility [21]. While members co-located teams rely on physically observable behaviors in order to assess trustworthiness of other team members, Alsharo et al. (2017) [3] claims that virtual teams rely upon different behaviors, that are unique to virtual settings, to compensate for the lack of physically observed behaviors. According to the authors, trust between members of a virtual team is less affect-based and more cognition-based, meaning that trust rely less on personal factors such as caring and emotion, but more based on one's perceptions of evidence of another person's trustworthiness, as well as their capabilities and competencies to perform the task at hand. Institution-based trust, as the authors calls it, allows team members to trust each others behavior based on the norms and rules of the organization. The authors discovered that institution-based trust affects virtual team effectiveness more so than personality-based trust.

de Guinea, Webster and Staples et al. (2012) [18] performed a meta-analysis of the consequences of virtualness on team functioning, and they found that short-term and long-term teams respond differently to higher levels of virtualness. Generally, they found that teams that were more virtual exhibit higher task conflict, lower communication frequency, less knowledge sharing, lower performance and lower satisfaction. However, they discovered that the time length of the project studied was a significant moderator on the effect virtualness had on performance. Long team teams, that is, teams that worked together for more than a single day, saw no negative effect on team performance and satisfaction as virtualness increased. Team conflict was even reduced with higher virtualness. The negative effects on communication frequency and knowledge sharing faced by both short- and long-term teams with increased virtualness, was significantly weaker in long-term teams. Their analysis also suggests that while the communication delays caused by asynchronous communication was a significant penalty to team effectiveness in short-term teams, it appeared to be less significant in long-term teams who had more time available to finish their tasks.

2.2 Knowledge

According to Rus et al. (2002) [53] there are four levels of knowledge: data, information, knowledge and experience. Data is discrete, objective facts about events, and may be either qualitative or quantitative. Data is raw material used to compose information, and say nothing about it's own importance or relevance. Information is data that has been organized in a way that makes it useful. Knowledge is the understanding of information, and can be thought of as "information about information". Experience is applied knowledge. While knowledge itself cannot be stored, it is possible to store information about knowledge. New knowledge can be created through experiences, observations and drawing rational conclusions.

Alavi and Leidner et al. (2001) [1] defines knowledge as information that has been possessed in the mind of individual. In other words, knowledge becomes information once it is processed in the mind of individuals, and knowledge becomes information once it is articulated and presented in the form of text, graphics, words, or other symbolic forms. There are a few important implications of this view of knowledge. In order for individuals to arrive at the same understanding of data or information, they must share a certain knowledge base. Systems designed to support knowledge organizations will be geared toward enabling users to assign meaning to information and to capture some of their knowledge in information and/or data. In order for knowledge held by an individual or a group to be useful for others, it must be expressed in such a manner as to be interpretable by others. Hoards of information are of little value, and only the information which is actively processed in the mind of an individual through a process of reflection and learning can be useful.

According to Wildman et al. (2012) [64], knowledge can be obtained either through top-down or bottom-up processing. Top-down processing is accomplished by first having access to explicit declarative knowledge, which then develops into procedural knowledge through practice. Bottom-up processing is accomplished when individuals or teams are subjected to a situation without prior explicit declarative knowledge or instruction. By engaging in the given task or situation, they implicitly understand how to engage in the task procedure and thereby obtain declarative knowledge from exposure.

2.2.1 Knowledge characteristics

There are several dimensions that can be used to describe the nature of knowledge. Whether the knowledge is held by an individual, a group, or an organization, it may be documented or undocumented depending on whether or not the knowledge has been captured and externalized [53]. It may be characterized as either implicit or explicit depending on the nature of the knowledge type, including whether it is declarative or procedural knowledge, and the manner in which the knowledge is obtained [64]. Explicit knowledge is more easily communicated, and thus easier to document and reuse. It usually contains processes, templates, and data captured in media [53]. Tacit knowledge refers to knowledge that is more abstract in nature, resides in the human brain, is embedded in individual experience and action, and is therefore not easily conveyed [49]. It is gained through experience, is highly personal to the individual, and is more easily influenced by the beliefs, perspectives, and values held by the individual [53].

Wildman et al. (2012) [64] categorizes knowledge as either static or dynamic, depending on the temporal nature of its content and structure. Static knowledge captures metal representations of information that remains stable over time, while dynamic knowledge captures mental representations of information that is rapidly changing or evolving. The authors claim it is possible that static team knowledge is better learned through top-down processing due to its stable and long-term nature, whereas dynamic team knowledge is better learned through bottom-up processing due to time and situational constraints.

According to Rus et al. (2002) [53], there are three different levels of knowledge abstraction, describing the extent to whether its application is specific or general. At the most specific level is point data, which describes (quantitatively or qualitatively) information about a single project or event. Examples of point data are metrics collected for a specific project, or lessons learned from a specific event. From a set of point data collected from multiple projects, models are created that contain more information applicable to new projects. From these models, best practices and standards may be built.

Furthermore, Rus et al. mention two more knowledge characteristics: knowledge awareness and knowledge scope. Knowledge awareness refer to the knowledge about existing knowledge, as well as knowledge about the lack of required knowledge. Knowledge scope refer to the domain in which certain knowledge is applicable.

2.3 Team knowledge

A mental model can be described as a person's own idea of the world around them, or a subjective representation of external reality [65]. More specifically, mental models can be defined as organized knowledge structures that allow individuals to interact with their environment, draw inferences, make predictions, understand phenomena, decide which actions to take, and experience events vicariously [44]. According to Rouse and Morris et al. (1989) [52], humans can have their own individual mental model of a system with which they interact. This mental model allows them to predict and explain the system's behavior, as well as recognize and remember relationships among its components. These mental models allows them to construct expectations of what is likely to occur next. The authors themselves proposed a more concise working definition, as follows:

> *"Mental models are mechanisms whereby humans generate descriptions of system purpose and form, explanations of system functioning and observed system states, and prediction of future system states".*
>
> *- Rouse and Morris et al. (1989) [52], p. 360*

To summarize, a mental model is an organized knowledge construct that allows a person to describe, explain, and predict events related to a specific system. Cannon-Bowers et al. (1993) [12] suggested that if the members of a team have *shared mental models*, then that would allow the team members to predict what their teammates are going to do and what they are going to need in order to do it. In other words, teams that have shared or common mental models are able to adapt quickly to changing task demands, making them more effective. Klimoski and Mohammed et al. (1994) [39] argues that the definition of *shared mental models* largely depend on what it means to "share" a mental model. Sharing can mean both "having in common", "dividing up", or "overlapping". Thus, shared mental models may refer to having common, distributed, or overlapping representations among several individuals. If the system in question is a team, and the individuals sharing mental models are the members of that team, then one might instead refer to their shared mental models as *team mental models*. A more formal definition of team mental models are team member's shared, organized understanding and mental representation of knowledge about key elements of the team's relevant environment [39]. In other words, team mental models are shared mental models for which the domain has been restricted to that of the team[13]. According to Cannon-Bowers et al. (1993) [12], team mental models should not be identical among team members, but rather, compatible, so that they provide mutual expectations that allow teams to coordinate and make predictions about the behavior and needs of their teammates. Cooke et al. (2000) [13] refer to team mental models as the collective task- and team-relevant knowledge that team members bring to a situation. It is acquired through training and experience, and is long lasting in nature. *Team situation models*, on the other hand, refer to the team's collective understanding of the specific situation [50], is developed in-situ while the team is actually engaged with the task, and is highly dynamic. The team situation model guides the team in assessing additional cues and patterns in the situation, determining strategies available to the team, assessing how the team is proceeding, predicting what teammates will do and need, and selecting appropriate actions to take.

Team knowledge has been defined as the collection of task- and team-related knowledge held by teammates and their collective understanding of the situation [13]. In other words, they refer to team knowledge as a set of knowledge structures including both the team mental models and the team situation models. It has also been described as an emergent structure in which knowledge that is critical to team functioning is organized, represented, and distributed within a team [64]. Due to the ambiguity of the word "shared" (e.g. in common, distributed, or overlapping), Cooke et al. (2000) [13] describe *shared team knowledge* as either knowledge that is similar within a team (i.e. homogeneous), or knowledge that is distributed among team members (i.e. heterogeneous). In heterogeneous teams, that is, teams in which different team members are assigned different roles, the possession of heterogeneous team knowledge is required, meaning that the different team members have role-specific yet compatible knowledge. While a certain degree of overlap of team knowledge is needed for effective coordination and shared expectations among team members [12], a situation in which every team member has identical knowledge is not only highly unlikely, but will also be highly dysfunctional [39]. Therefore, for the purposes of this paper, we will use the term *shared team knowledge* to describe homogeneous team knowledge, or in other words, team knowledge that is similar within a team.

2.3.1 Framework for Shared Team Knowledge

Wildman et al. (2012) [64] proposed a framework for categorizing team knowledge. The authors define team cognition as the conceptual sum of the knowledge of the individual team members, as well as the emergent knowledge structure that results from the interplay of the individual cognitions of each team member. Team knowledge refers to the structure of team-level mental representations such as mental models, transactive memory, situation awareness, strategic consensus, or other mental representations concerned with the organization, representation, and distribution of knowledge among team members. These team-level mental representations may be categorized into four basic types of knowledge content: task related, team-related, process-related, and goal related. Faegri et al. (2016) [24] proposed a framework of shared team knowledge that classify and describe classes of knowledge with particular importance for virtual teams. The main classifications of their framework is adopted from Wildman et al. (2012) [64], but includes more detailed attributes relevant to virtual software teams. In this section, we will provide an overview of this framework.

Task-related

Wildman et al. (2012) [64] describes task-related team knowledge as a mental representation focused on the task-work that a team is performing, such as task-mental models and knowledge about how a task should be accomplished as well as the criteria for completing the task successfully. The authors found that the sharedness of task mental models and task knowledge within a team was often

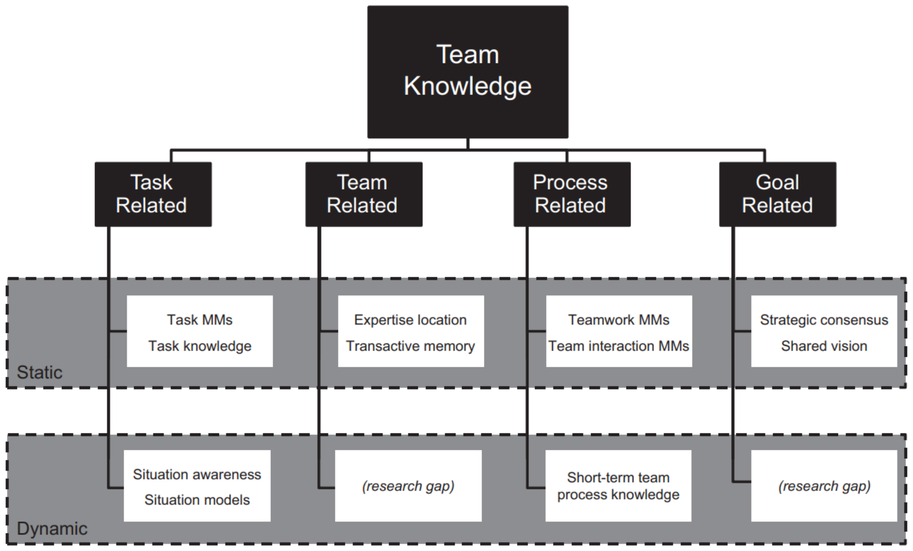


Figure 1: Wildman et al. (2012) [64] p. 91 - fig. 1: Organizing framework of team knowledge

measured using a score overlap between team members on a set of items assessing their individual knowledge about the key pieces of information in a particular task. The majority of their reviewed literature categorized task mental models and task knowledge as static in nature, meaning that it is relatively unchanging over time.

Team situation models are the mental representations held by teams about the dynamic moment-to-moment aspects of the task, team, process, and goals. However, according to Wildman et al. there was only a small amount of work that focused on dynamic mental-representations in teams, and the majority of said work focused on task-related situational information. The authors specifically mention two examples of dynamic task-related situational information: one was the measurement of what cues in the environment the team members were attending to, and the other was interruptive questionnaires where team members report on the current status of a task-related situation.

Faegri et al. (2016) [24] found two primary types of task-related team knowledge. The first type is task strategies, which is a shared understnding about how a task is supposed to be accomplished by the team so that a sufficient level of performance can be achieved. This type also includes knowledge of how task work is allocated to members of the team, and plans of how team subgroups should be used for working on parts of a task. The second type of task-related team knowledge is knowledge about the content of a task. This includes a shared understanding about the content about how the parts of the tasks interact, and a shared understanding about how a task is connected to its environment.

Team-related

Wildman et al. (2012) [64] describes team-related knowledge as the mental representations concerning one's teammates or the team as a whole. More specifically, team-related knowledge is knowledge of the qualities and characteristics of the team members and the team itself, and not of team processes. Expertise location is an example of static, team-level mental representation concerning the location and structure of expertise within the team, such as which expertise is held by which individual, as well as the individual's degree of held expertise within a given topic.

Faegri et al. (2016) [24] defined three primary types of team-related knowledge: team membership knowledge, team member model, and expertise location. Team membership knowledge refers to knowledge about who is on the team and defined boundaries of who shares responsibility of the team's work. The team member model is a type of shared mental models containing information about the characteristics and qualities of the individual team members, including their knowledge, skills, attitudes, preferences, strength, weaknesses and tendencies. A shared team member model is important in distributed teams where interactions are infrequent, as it enables one to predict the behaviours of their fellow team members and act accordingly, increasing the automation of the process [44]. Expertise location refers to the extent to which team members know who knows what on the team, and the authors claim this type of knowledge is particularly important for virtual teams, and is positively associated with team performance. Techniques such as Planning Poker may improve expertise location within a team [24].

Process-related

Process-related team knowledge refers to mental representations regarding teamwork and the interpersonal processes involved in team interactions such as team interaction mental models and other mental representations focused on processes such as communication, leadership, and coordination. The difference between process-related knowledge and task-related knowledge, is that task-related knowledge is particular for each task, while process related knowledge will be generic and valid for a range of tasks [24].

Faegri et al. (2016) [24] classified two primary types of process-related team knowledge: team interaction mental models, and team norms. Team interaction mental models include knowledge of interaction and interpersonal processes in a team, such as communication, leadership, and coordination. The authors claim that team interaction mental models enable virtual teams to be adaptable as it creates expectations and drives team member behavior. Examples of the use of team interaction mental models in virtual teams are when the team have a rhythmic, temporal pattern of interaction, or use video calls. Team norms refer to codes of conduct that are accepted by the team members. They are formed and adopted as patterns of actions are found to be useful or effective.

Goal-related

Wildman et al. (2012) [64] describes goal-related team knowledge as mental representations concerning the goals and objectives for the team, as well as the mental representations concerning the achievement of these goals. This mental representation is not referencing the requirements of the task, the characteristics of the team, or the team interaction processes, but rather it is focused on the knowledge or understanding of an overarching goal or mission relevant to the team. An example of goal-related team knowledge is strategic consensus, a construct that captures the extent to which the members of the team share an understanding of the strategic vision that the team aims to achieve.

Faegri et al. (2016) [24] classified two primary types of goal-related team knowledge: Overarching team goals, and strategic consensus. The former are the mental representations of an overall goal or mission for the team, team expectations, or performance objectives, as well as the mental representations concerning the achievement of these goals. Strategic consensus refers to an agreement about strategic goals for the organization. The authors claim that goal-related team knowledge is very important for virtual agile teams. For self-organization to work, the team members must have a profound interest and commitment to the overall objectives of the team. Social bonds are weaker in virtual teams, making social contracts among the members more fragile. And for this reason, establishing and maintaining shared goals between the team members are both challenging and important.

2.4 Knowledge Management

During the 1980s emerged the belief that knowledge had an increasingly important role as an organizational resource, and that the success of an organization requires systematic knowledge management [63]. Through the 1990s managements not only realized that knowledge was a critical resource, but also that their organizations generally poorly managed it [22]. Knowledge had become a central asset in not only knowledge-focused industries, but also manufacturing, financial services and government organizations [15], and it became a more accepted belief that the success of a business relied on competitive knowledge assets and their effective utilization [63]. In other words, creating, providing, sharing, using, and protecting knowledge was believed to improve organizational performance [22]. The field of knowledge management rose to fulfill these needs, with a twofold objective: first, to make the enterprise act as intelligently as possible to secure its viability and overall success; and second, to otherwise realize the best value of its knowledge assets [63]. Davenport et al. (1998) [16] defined knowledge management as "a method that simplifies the process of sharing, distributing, creating, capturing and understanding a company's knowledge".

Rus et al (2002) [53] claim that organizations have knowledge management needs regarding domain knowledge for individual projects, knowledge of technology, knowledge about an organization's local policies, and the knowledge of who knows what within an organization. Software development requires not only knowledge about its own domain, but also knowledge of the domain of the problem which the developed software is supposed to address. Domain-specific knowledge takes a long time to to acquire, and is often complex and application-specific. Domain knowledge that no one in the organization possesses must be acquired first-hand through training or by hiring knowledgeable employees. However, once this knowledge is attained, successful knowledge management can make such knowledge available to other members of the organization, so that the learning process need not be repeated. The field of software engineering is heavily reliant on technical knowledge in order to achieve the desired outcome for projects. Due to the dynamic and constantly changing nature of technology, however, this can pose challenges. A fast pace of change and stream of new technologies can make it difficult for an organization to keep ahead in the competition, making software engineering a difficult domain to master. Knowledge management may help speed the learning curve for new technologies by facilitating sharing of previously acquired knowledge within the organization. Knowledge concerning existing software bases and local programming conventions within an organization is often maintained informally in the minds of experienced developers, and shared through informal conversations, such as during coffee breaks. While this type of knowledge sharing is generally encouraged, it is not sufficient for making the knowledge available throughout an organization, because the knowledge may not reach everyone who needs it. Additional formal means of communication is required in order to make the knowledge more available on a larger scale. The majority of an organization's assets reside as tacit knowledge within the minds of its employees. Tacit knowledge is very mobile, and if a person leaves the organization, then their knowledge goes with them, leaving a gap in the organization's knowledge. Knowledge management helps identify who knows what, so that structures can be made to retain some of the tacit knowledge held by employees. Furthermore, mapping knowledge gaps is useful for identifying what knowledge is required from a successor. As it is becoming increasingly common for teams to work across borders of location and time, knowledge management becomes necessary in order to help them communicate, collaborate and coordinate. Through knowledge management, an organization may facilitate transfer and mutual sharing of knowledge, as well as the storage of work artifacts and their status in a format that is available for all team members.

2.4.1 Knowledge management strategies

Earl et al. (2001) [22] proposed a taxonomy involving three categories of knowledge management schools: technocratic, economic and behavioural. The technocratic schools revolve around information or management technologies, of which the systems school is the most established and formal approach. Its key idea is to capture individual or group-held knowledge in knowledge bases which other people can access. The system school includes knowledge management strategies that use technology to manage a knowledge base, and examples of such technologies within the software industry are JIRA, Wiki and Github [4], and it is believed that the systems school is unfeasible without IT. Engineering design and maintenance are classic applications, and the managed knowledge tends to be domain-specific and aims to support and improve knowledge-intensive work tasks and decision making. The authors suggested that the systems school of knowledge management rely on two critical success factors: first, it would be required with incentives or rewards for knowledge creation and contribution to the knowledge base; second, if the contributed knowledge is to become "official knowledge", then the knowledge content must be validated. A second technocratic school of knowledge management is the cartographic school, which revolves around the mapping of organizational knowledge by recording and disclosing who in the organization knows what by building knowledge directories. Its main idea is that people holding expertise should be accessible for consultation and knowledge exchange. Since tacit knowledge is not easily expressed or articulated, the objective is to identify who might be a source of said knowledge, making the tacit knowledge accessible through conversation instead of a knowledge base. The cartographic school of knowledge management relies on a culture of mutual support and knowledge sharing as well as communication networks in order to be successful. In this school, the contribution of IT is to connect people via intranets to help them locate knowledge sources and providers using directories. The third and last technocratic school of knowledge management is the process school, which is driven by the ideas that performance of business processes can be enhanced by providing personnel with the relevant knowledge, and that management processes are inherently more knowledge-intensive than business processes. This school revolves around decision-relevant, contextual and best-practice knowledge. In this school, both knowledge and information are provided by systems and intranets to operatives, staff or executives. Its philosophy is enhancing the firm's core capabilities with knowledge flows.

The commercial school is the only school that was labeled as "economic", as it was concerned with both protecting and exploiting a firm's knowledge or intellectual assets to produce revenue streams. Its philosophy is pure commercialization of intellectual or knowledge property, with a concern not for what it is, but how to do it efficiently and effectively. It relies on the development of a specialist team or function to aggressively manage property so it is not too easily forgotten. It also requires the development or acquisition of techniques and procedures to manage intellectual assets as routinized processes, in order to avoid sub-optimization.

The last three schools are the behavioral schools, including the organizational, spatial and strategic schools. The organizational schools describe the use of organizational structures to share or pool knowledge, often described as "knowledge communities", which are a group of people with a common interest, problem or experience. The essential feature of these communities is that they exchange and share knowledge interactively in unstructured ways as an interdependent network. The communities are also typically supported by knowledge bases provided over networks, and so they combine both codification and personalization knowledge management strategies. The philosophy of this school is increasing connectivity between knowledge workers, and rely on two critical success factors: first, is a tradition of sociability and networking; second, a moderator is required for the communities, whose function is to know who knows what as well as what is known. The spatial school revolves around the use of spatial design to facilitate knowledge management, and consider the fact that people in organizations are social being who prefer socialization over documentation. The main objective is to encourage socialization as a means of knowledge exchange, with the belief that tacit knowledge is best exchanged through discussion. The strategic school sees knowledge management as a dimensions of competitive strategy, and knowledge is considered the key resource. This school is concerned with raising consciousness about the value creation possibilities available from recognizing knowledge as a resource,

2.4.2 Research framework for knowledge management

In Lee and Choi et al. (2003) [40], the authors propose a research framework in which they emphasize three major factors for managing knowledge: enablers, processes, and organizational performance. These are connected in the following way: knowledge management enablers provide infrastructure that facilitates knowledge management processes. These processes lead to intermediate outcomes which affects the organizational performance. This research framework is shown is Figure 2.

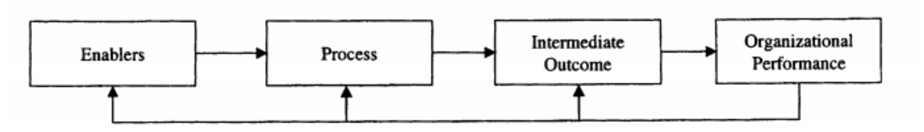


Figure 2: Lee and Choi et al. (2003), p. 182, fig. 1: An integrative research framework for studying knowledge management.

Knowledge management *enablers* are organizational mechanisms meant to stimulate knowledge creation, protect knowledge, and facilitate the sharing of knowledge in an organization. They are influencing factors that provide the infrastructure necessary for the organization to increase the efficiency of knowledge processes. According to the authors, the most important enabler for successful knowledge management is an appropriate organizational culture that encourages people to create and share knowledge within the organization. Another important factor is an organizational structure that put knowledge to use through centralization and formalization. Furthermore, an organization should acquire people with desirable skills in order to acquire their knowledge and competence. Technology may be used to connect people and facilitate the sharing of knowledge and creation of new knowledge, as well as infrastructure to store knowledge.

Knowledge *processes*, or knowledge management activities, are structured coordination for managing knowledge effectively, such as creation, sharing, storage, and usage. The authors emphasize knowledge creation as an important knowledge management process. Knowledge creation is a continuous process where individuals and groups share tacit and explicit knowledge. The authors adopt the SECI model to explore knowledge creation and knowledge transfer, and this model is made up of four activity modes; socialization, externalization, combination, and internalization. Socialization converts tacit knowledge into new tacit knowledge through social interactions between members. Externalization codifies tacit knowledge into explicit concepts. Combination converts explicit knowledge into more systematic sets by combining key pieces. Internalization embodies explicit knowledge into tacit knowledge.

Organizational performance is the degree to which companies achieved it's business objectives. These measures can be categorized into four groups: financial measures, intellectual capital, tangible and intangible benefits, and balanced scoreboard. The balanced scoreboard retains financial performance and supplements it with measures on the drivers of future potential. Directly linking knowledge management processes with organizational performance is not possible as different organizations measure performance differently, It is possible, however, to link knowledge management processes with *intermediate outcomes*, and these outcomes may then be linked to the organizational performance. According to the authors, an important intermediate outcome is organizational creativity, which is the transformation of knowledge into business value, and the seed of all innovation.

2.4.3 Knowledge management systems

Alavi et al. (2001) [1] define knowledge management systems (KMS) as a class of information systems applied to managing organizational knowledge. More specifically, they describe them as IT-based systems developed to support and enhance the organizational processes of knowledge creation, storage / retrieval, transfer and application. According to the authors, knowledge management systems can support knowledge management in several ways, one of which is knowledge sharing and collaboration in virtual teams.

Wu and Wang et al. (2006) [66] describe two common characteristics of knowledge management systems, namely knowledge repositories and knowledge maps. Knowledge repositories are databases of useful documents with the system that provides functions for capturing, organizing, storing, searching and retrieving the knowledge and information. Knowledge maps are searchable indexes or catalogues of expertise that help team members find individuals with particular knowledge. According to the authors, a major benefit of KMS comes from knowledge creation and sharing on the basis of "pull" by users, as opposed to a "push" of information on them.

Alavi and Tiwana et al. (2002) [2] identified a set of challenges faced by virtual teams, to which they propose ways a knowledge management system may help alleviate these challenges. First, knowledge management systems may support the development and maintenance of a virtual team's transactive memory. Secondly, knowledge management systems may support mutual understanding between the virtual team's members, which suffers due to the barriers to effective communication. Furthermore, dispersion of virtual team members cause a failure to share and remember contextual knowledge, leading to misunderstandings or misinterpretations of a remote team member's behavior, further leading to conflict and difficulty in coordination of team efforts. Supposedly, knowledge management systems may support the sharing and retaining of contextual knowledge in virtual environments. Lastly, knowledge management systems may support symmetry in shared knowledge, so that each member has access to the same information.

2.5 Knowledge sharing

The success of knowledge management initiatives depends on the sharing of knowledge between employees. Knowledge sharing is a major focus area of knowledge management because it provides a link between the level of the individual knowledge workers, where knowledge resides, and the level of the organization, where knowledge attains its economic or competitive value [30]. Wang and Noe et al. (2010) [61] describes knowledge as a critical organizational resource that provides a sustainable competitive advantage in a competitive and dynamic economy, and claims that organizations must consider how to transfer expertise and knowledge from experts to novices in order to gain that competitive advantage. In other words, successful knowledge management depends on effective exploits of an organization's existing knowledge-based resources. Knowledge sharing between employees and within and across teams allows organizations to exploit and capitalize on knowledge-based resources [16]. Knowledge sharing and combination is positively related to, among other things, reductions in production costs, team performance and faster completion of new product development projects [61]. According to Hendriks et al. (1999) [30], knowledge sharing is not only an important pillar in knowledge management efforts, but also a significant barrier to effective knowledge management. Factors such as inadequate organizational structures, sharing-unfriendly organizational cultures and denominational segregation are all factors identified to impede the sharing of knowledge in organizations. Moreover, the authors claim that employee motivation is of critical concern, as well as the willingness and ability to use ICT-systems that support knowledge sharing.

2.5.1 Definition of knowledge sharing

According to Hendriks et al. (1999) [30], knowledge cannot be shared like a commodity that can be passed around freely, but rather it is tied to a knowing subject, and requires an act of reconstruction in order to be shared. The party that possesses the knowledge must first externalize it. An act of externalization may take many forms, including performing actions based on this knowledge, explaining it in a lecture or codifying it in an intelligent knowledge system. Second, the party seeking to acquire knowledge must internalize the externalized knowledge. Internalization may also occur in various forms, such as learning by doing, reading books, or trying to understand the codified knowledge in a knowledge base. Internalization of externalized knowledge may be distorted by barriers such as distance in space, time, culture and language, as well as social distance and differences in mental or conceptual frames. This process is shown in Figure 3.

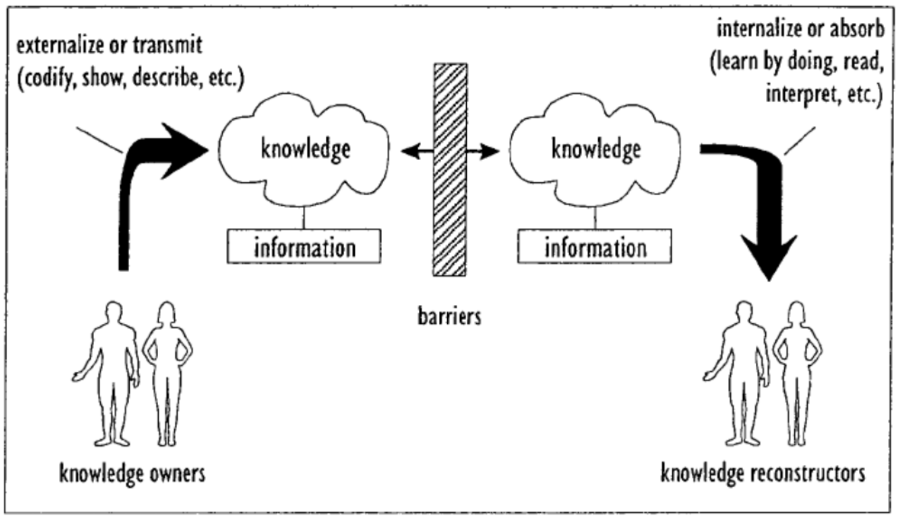


Figure 3: Hendriks et al. (1999) [30], p. 93, Figure 2 A simplified model of knowledge sharing

In Chapter 2.4.2 we saw that *knowledge transfer* involved socialization, externalization, combination and internalization [40]. The definition by Hendriks et al. (1999) [30], however, only includes externalization and internalization in their definition of knowledge sharing. While in other literature, the definition of knowledge sharing also includes socialization [4]. From the literature we have read the terms knowledge sharing and knowledge transfer have been used interchangeably, so what is the difference between the two? Wei et al. (2011) [62] summarizes knowledge transfer as dyadic exchanges of organizational knowledge between the source unit and a recipient unit that involves direct communication processes between the two parties. At the organizational level these units are separate organizations or departments, at the team-level the units are separate teams, whereas on the individual level, on which there will be most focus on in this paper, the units are individual people. Cummings et al. (2004) [14] defined *knowledge sharing* as "the provision or receipt of task information, know-how, and feedback regarding a product or procedure". According to them, knowledge sharing includes verbal communication about a task and exchange of tangible artifacts, as well as implicit coordination of expertise and information about who knows what in the group. It may occur via written correspondence or face-to-face communications through networking with other experts, or documenting, organizing, and capturing knowledge for others [61]. Wang and Noe et al. (2010) [61] describes the difference between "knowledge transfer" and "knowledge sharing" as follows:

> *"Knowledge transfer involves both the sharing of knowledge by the knowledge source and the acquisition and application of knowledge by the recipient. "Knowledge transfer" has typically been used to describe the movement of knowledge between different units, divisions, or organizations rather than individual."*
>
> *- Wang and Noe et al. (2010) [61], p. 117*

To summarize, it would appear that knowledge transfer is a broader term than knowledge sharing, and that knowledge transfer involves the movement of knowledge between higher-level units, in addition to how the knowledge is applied by the receiving unit. Knowledge sharing, on the other hand, concerns itself with a smaller scope, involving individual people instead of groups or departments. While knowledge transfer involves four conversion modes (socialization, externalization, combination, internalization) [40], knowledge sharing only involves three (socialization, externalization, internalization). In other words, knowledge sharing does not entail combination as a mode of conversion, which is the conversion of explicit knowledge into other forms of explicit knowledge. It does, however, entail moving of knowledge between two individuals through socialization, externalization and internalization.

2.5.2 Factors influencing knowledge sharing

The factors influencing the extent to which knowledge sharing takes place may be labeled as cognitive and motivational limitations towards knowledge sharing, where the former are related to an individual's ability to share knowledge, and the latter is related to an individual's willingness to share knowledge [60]. In the literature review by Nguyen et al. (2020) [49] the author proposes the four-dimensional ISTO model, which is short for the individual-social-technological-organisational model. In this model, online knowledge sharing behavior in organizations is categorized into four groups: individual, social, technological and organizational. We will make use of this model as we proceed to explore factors which may influence the sharing of knowledge in a team or an organization.

Individual factors refer to the factors that are personal to the individual, and its motivations derive from an individual's own perception of achieving a reward, benefit, or some form of enjoyment [49]. Lin et al. (2007) [41] identify two broad classes of individual motivation, namely intrinsic and extrinsic motivation, both of which influence individual intentions and behaviors regarding knowledge sharing. Extrinsic motivation to share knowledge refers to the cost-benefit analysis where rewards for performing knowledge sharing are compared to the efforts involved in the exchange [41]. Donating knowledge can take both time and mental effort, thus and employee will be more motivated to donate their knowledge if they expect benefits such as receiving organizational rewards or reciprocation from their colleagues. Reputation and peer recognition are extrinsic motivations known to encourage knowledge sharing [49]. Employees often share useful knowledge to gain the respect of their peers, or to establish a position as an expert in an organization. Intrinsic motivation, on the other hand, refers to engaging in an activity for it's own sake, out of interest, or for the pleasure or satisfaction derived from the experience [41]. People may experience enjoyment from helping others, or gain satisfaction from enhancing their knowledge self-efficacy (that is, the confidence in one's own ability to provide knowledge valuable to others) [41, 49]. Openness to experience has been positively related to individuals' self-report of knowledge exchange, suggesting that people who are open to new experiences tend to have a high level of curiosity, resulting in a pique interest to seek others' ideas and insights [9]. Several studies have shown that individuals who are more confident in their ability to share useful knowledge are mode likely to express intentions to share knowledge and report higher levels of engagement in knowledge sharing [61]. Lin et al. (2007) [41] concluded in their study that extrinsic rewards secure only temporary compliance, and that reciprocal benefits, knowledge self-efficacy, and enjoyment in helping others were significantly associated with employee knowledge sharing attitudes and intentions, while expected organizational rewards did not significantly influence employee attitudes and behavior intentions regarding knowledge sharing. This result, however, could be impacted by the fact that more than 67 percent of their sample respondents were executives.

The *social factors* refer to an individuals relationships with other people in the organization, feedback from their colleagues, and reciprocity [49]. Social interaction ties were shown to improve the depth, breadth, and efficiency of knowledge sharing between employees. Trust was also shown to increase an individual's willingness to share their knowledge. Furthermore, individuals who shares their knowledge with a colleague will often expect the receiver to share the same amount of knowledge back. Social norms also influence an individual's behavior; an employee who perceive a greater social pressure to share knowledge will have a greater willingness to do so [49, 34]. In virtual communities both he number of direct ties and personal relationships an individual has with other members have been shown to be positively related to the quantity and the perceived helpfulness of knowledge shared [61]. Diversity, which was found to be a dimension of team virtualness in Chapter 2.1, has also found to be a factor influencing knowledge sharing in organizations, teams and work-groups. Cultural diversity have also been found to influence knowledge sharing behavior in global virtual teams [37]. Team members who consider themselves a minority based on gender, marital status, or education have been found to be less likely to share knowledge with team members [61]. Diversity entails not only cultural diversity, but also other types of diversity. Different types of team diversity, such as different background within education, experience and technical knowledge, unfamiliarity of team members, physical distance between team members, different languages and time difference have all been perceived as barriers to effective knowledge sharing for agile software development teams [27]. Note that physical distance between team members and time difference (temporality) were both found to be dimensions of team virtualness in Chapter 2.1.

In a virtual team, face-to-face communication may not be possible, and so the team is dependent on ICT to facilitate knowledge sharing. For this reason, *technological factors* may carry more weight than they would in co-located teams. According to Van den Hooff et al. (2003) [60], the role of ICT in knowledge sharing is to facilitate easier exchanges (independent of time and place) between team members, and to promote connectivity and communality. Connectivity refers to an individual's ability to directly contact other members of the team, positively influencing both the ability and willingness of members to share knowledge. Communality refers to the collective storing and sharing of information so that it is accessible by all team members. With an increased shared intellectual capital follows a greater willingness to contribute to it, and with the increase of knowledge about other member's capabilities and interests follows an increased ability to share knowledge. Previous research has found that employees' comfort level and ability to use computers likely influence the usage of collaborative electronic media for information sharing [33]. According to Nguyen et al. (2020) [49], system quality, perceived ease of use, and the usefulness of information technology affects how likely employees are to make use of them. Employees who understand how to use all the functions in a virtual platform, and who perceive an improved performance upon using them, are likely to interact more and thus share more knowledge. Van den Hooff et al. (2003) [60] also found that task interdependence, computer effort, an individual's attitude towards computer-based information, and information culture in an organization are factors that influence the use of ICT to achieve knowledge sharing.

The *organizational factors* refer to the determinants within an organisational context, such as rewarding incentives, management support, commitment, shared goals, and leadership. Employees who strongly commit to organisations tend to contribute their knowledge to build a relationship with their colleagues and to contribute to the organisation [49]. Greater commitment to an organization are linked to an in increase prosocial behaviors and attitudes, and may engender beliefs that the organisation has rights to the information and knowledge one has created or acquired [34]. The beliefs of organisational ownership of knowledge and information could be related to, or reinforced by, organization culture, which refers to the shared values and attitudes of the members of an organization [34]. Management have a responsibility to create the organisational culture in which knowledge sharing among employees is encouraged [49]. Common goals improve trust among individuals and reduce the fear of self-interest by other members of the organisation, making employees more willing to share knowledge [49]. Supervisors who recognize employees' contribution and empower them is also a key determinant for knowledge sharing [49].

2.5.3 Knowledge sharing in agile software development

According to Ghobadi et al. (2015) [26], knowledge sharing is an important area of concern, especially within the field of software development. They describe software development as a collaborative and knowledge-intensive process that requires the blending and interweaving of diverse knowledge dispersed across domains of specialization. Developers, user interface designers, user representatives, and project managers engage in iterative development cycles that require intensive knowledge sharing in terms of rapid reflections to exploit diverse expertise and explore existing and potential opportunities in software development. Moreover, effective knowledge sharing is necessary to allow team members to discuss critical aspects of coordinating work across distributed spaces. Bjørnson and Dingsøyr et al. (2008) [8] found that out of the three categories of knowledge management schools proposed by Earl et al. (2001) [22] (see Chapter 2.4.1), the technocratic and behavioural schools were used in software engineering. The former refers to the knowledge management strategies that focus on explicating knowledge and its flows, and the latter refers to schools focusing on collaboration and communication as knowledge management strategies [5]. Ebert et al. (2008) [23] proposed a classification of knowledge in software engineering describing three types of knowledge. The first type is *project knowledge*, which is defined as "the knowledge about resources, functional and attributes requirements, work products, budget timing, milestones, deliverables, increments, quality targets and performance parameters". The second type is *product knowledge*, which is defined as "the knowledge about product features and how they relate to other products, standards, protocols and the like". The third type is *process knowledge*, which is defined as "the knowledge about business processes, workflows, responsibilities, supporting technologies and interfaces between processes".

According to Chau and Maurer et al. (2004) [11], long chains of communication between the customer and the coder will result in a high amount of information lost to the person who actually codes the software. For example, the customer may express what they want from their software, but if this information needs to go through an analyst, architect and designer before it reaches the coder, then there are many points vulnerable for communication error. Agile methodologies such as Scrum and Extreme Programming (XP) rely on direct face-to-face communication between customers and developers for knowledge sharing, reducing the information loss. They rely on individual, team and customer communication and interactions, emphasizing the management of tacit knowledge over explicit knowledge [5]. Andriyani et al. (2017) [4] found that Agile practices were associated with the three aforementioned types of software engineering knowledge: *project knowledge*, including timelines, team progress and plans; *product knowledge*, including requirements and designs; and *process knowledge*, including coding techniques and synchronised teamwork. Moreover, agile teams use three specific knowledge management strategies: discussions (e.g. sharing requirements), artefacts (e.g. user stories), and visualisations (e.g. burn down charts). Agile methodologies recommend the use of cross-functional (i.e. heterogeneous) teams as opposed to role-based (i.e. homogeneous) teams, because the former facilitate better collaboration and knowledge sharing [11].

Scrum is an Agile methodology in which the product is developed in a sequence of self-contained mini-projects called iteration, and the product functionality grows incrementally by adding new features during each iteration [6]. The software development project has three roles: *The product owner* is responsible for deciding functionality of the product and prioritization of its implementation; *the scrum master* is responsible for establishing scrum practices and rules, representing management to the project, shielding the team and removing obstacles; and the *team member* is someone belonging to the team that carries out the actual development activities, and the team is often cross-functional [6]. According to Adriyani et al. (2017) [5], Scrum practices such as sprint and release planning, daily Scrum meetings and sprint retrospective all support the sharing of knowledge, especially process and project knowledge. Through discussion the team achieves socialization, which is the process of sharing tacit knowledge (e.g. sharing mental models and technical skills). Externalization occurs when agile teams gain the shape of metaphors, concepts and models in written form, such as documentations, diagrams or artifacts. Through internalization process, the externalized knowledge is processed to gain understanding about "know-how". Finally, knowledge is combined when the team compile knowledge from different sources in order to transform it into action.

According to Chau and Maurer et al. (2004) [11], daily Scrum meetings, also referred to as stand-up meetings, are short daily meetings in which each team member report their work progress since the last meeting, state their goals for the day, and voice problems related to their tasks or suggestions to their colleagues' tasks. These meetings facilitate communication among the entire team, provide visibility of one's work to the rest of the team, raises awareness of who has knowledge about specific parts of the system, and encourage communications among team members who may not talk to each other regularly.

Agile planning [68] includes two agile processes, called release planning and iteration planning. With release planning, the software development project is divided into multiple releases in order to accelerate the software's time to market. During release planning, the customer presents the desired features to the programmers, and the programmers provide an estimation of their difficulty, which the customers then use to lay out a plan for the project [42]. These initial plannings create a rough release plan, which is accurate enough for decision making, but imprecise enough to require regular revision. Each release is composed by several iterations, each of which lasts around 2-4 weeks. Iteration planning [11, 68] is performed at the beginning of each iteration to regularly give the team direction. The customers presents desired features for the next iteration, and the programmers break them down into tasks and estimate their cost at a finer level [42]. These estimations are used to predict whether the developers will complete all the proposed tasks, and if not then the iteration tasks are renegotiated with the customers [11]. This way of working allows the developers to demonstrate parts of the software product frequently, making it easier to detect misunderstandings and errors of communication between customer and developer [68]. The amount of progress is very visible to the customer, and with the ability to see and decide which features should be worked on next, the team are able to deliver more of what is most needed [42].

Another Scrum practice that involves feedback and communication is the retrospective meetings, which is a feedback meeting held at the end of every iteration. During the retrospective, the team does a short reflection on what went well during the iteration and what should be improved in the next iteration [42]. The objectives of the meeting is to gather data, generate insight, and decide what to do next [20]. Data is gathered when the team shares their review and feedback, report on what happened during the previous sprint and briefly discuss it with their teammates. Insight is generated by participating in a further discussion about which issues should receive focus, and how these issues should be solved. The team then decides which actions should be performed in order to solve the discussed issues. Retrospectives facilitate the identification of any success factors and obstacles of the current process, and provide the opportunity for these issues to be raised, discussed, and dealt with continuously during the project [11].

Extreme Programming (XP) is an Agile methodology based on four core values: communication, simplicity, feedback and courage [48]. The objective of XP is to find the simplest way of working that could possibly work, in the belief that predictive overhead is too expensive to be useful. A single team handles all aspects of the development using simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation [42]. Pair programming is one of the core processes of XP and involves two developers working together using a single computer to design, code and test software [48]. According to Chau and Maurer et al. (2004) [11], pair programming is a great facilitator for eliciting and sharing tacit knowledge, as the two developers share a range of knowledge between one another, some explicit but mostly tacit. The knowledge shared includes task-related knowledge such as system knowledge, best-practices and technology knowledge; contextual knowledge such as past experiences; or social resources such as personal contacts or referrals. These are all types of knowledge that developers tend not to document, and is often only shared through informal and casual conversation. Furthermore, XP recommends pairs to be rotated to ensure the shared knowledge is accessible by the entire team.

3 Method

The purpose of this study was to describe a single case of a virtual software development team in terms of their activities promoting team-knowledge sharing, and how well they achieve the sharing of team knowledge. There is a lack of studies performed on modern virtual software-development teams, especially within the field of knowledge sharing. Focus was placed on team-knowledge as opposed to general knowledge due to the fact that team-knowledge is crucial for the coordination and collaboration of virtual teams [46]. For this purpose, two research questions were posed about the case, which answers will provide empiric material which will, hopefully, be one of many cases, promoting the construction of more generalized theory in the future. The research questions posed are the following:

- **R1:** Which team-knowledge sharing activities are used by the team?
- **R2:** What is the overlap measure of the team's shared team knowledge?

This chapter presents the description of the research process and is divided into several sections. Section 1 presents the conceptual framework defining the concepts that are relevant for answering the posed research questions. Section 2 presents research design, describing the major design decisions made for this study along with their arguments. Section 3 provides a description case and informant selection, and Section 4 describes the data collection procedures. Section 5 describes how collected data was analysed with regard to the conceptual framework. Section 6 discuss quality of research, while Section 7 discuss the researcher's own positionality with regards to the study. Finally, in Section 8 we describe ethical considerations that were made for using information about the case and the informants.

3.1 Conceptual framework

Before we could answer any of the research questions it was necessary to establish a conceptual framework that defines the boundaries of which types of data are relevant, and which aren't. The purpose of this chapter is to provide a set of concepts and their definitions from the background chapter (see Chapter 2) which will guide the direction of the remainder of this study. For each of the two research questions we will list a set of concepts that are relevant in order to answer those questions. Additionally, we will define a conceptual framework in order to evaluate the team's degree of virtualness.

3.1.1 Team virtualness

To evaluate the team's degree of virtualness we will define our conceptual framework based on what we found in Chapter 2.3.1: "Characteristics of a virtual team". Johnson et al. (2009) [35] summarized a total of 11 different dimensions of team virtuality, and we quickly saw that several of these dimensions have similar themes, and may be grouped into two main categories: dimensions related to the use of technology, and dimensions related to team distribution (see Table 2). It was also noticed that some of these dimensions appeared similar in meaning. For example, "use of virtual communication tools" and "use of computer-mediated communication" were both interpreted to mean the same thing, that is, the use of digital tools for communication with other team members. And so both of these dimensions were combined into a single dimension named "Use of virtual communication tools". Furthermore, geographic dispersion and degree of physical distance could be interpreted to mean the same thing. Instead of combining them into a single dimension, we chose to instead define "geographic dispersion" as the team's dispersion on a global basis (e.g. which city and country each member is located), and "degree of physical distance" as the time and effort it would require for a team member to physically visit another, or for the entire team to physically gather at one place. Since it was outside the scope of this study to measure each of the remaining 10 dimensions in detail, the conceptual framework that was used to measure team virtualness would be simplified to mainly focus on the two main categories (use of technology and team distribution), and then probe more in-depth on individual dimensions where it seemed relevant for the case.

Use of technology	**Team distribution**
Electronic dependence	Geographic dispersion
Level of technology support	Percentage of time spent apart while working on a task
Use of virtual communication tools	Degree of physical distance
Amount of informational value provided by virtual communication tools	Synchronicity of communication
	Temporality
	Diversity

Table 2: Dimensions of team virtuality, grouped into two main themes: use of technology and team distribution

3.1.2 Overlap of shared team knowledge

In order to answer the research question **R2**: "what is the overlap measure of the team's shared team knowledge?", two separate items needed to be learned. First, it was necessary to find out what *relevant* knowledge each individual team member possessed. Note that there is an emphasis on the word "relevant", because only knowledge that was relevant to our context has been considered. If several of the team members possessed overlapping knowledge about, for example, European history, then that would not be very helpful in a software engineering context. A conceptual framework was required to define the boundaries of what knowledge was relevant in order to collect the correct information. In section 2.3.1 we introduced a conceptual framework for team knowledge by Wildman et al. (2012) [64] as well as the classifications of team knowledge types by Faegri et al. (2016) [24], and so the framework presented in Chapter 2.3.1 serves as a conceptual framework to define which types of knowledge are relevant. In the framework by Wildman et. al, team knowledge was divided into four categories based on content: task-related, team-related, process-related, and goal-related. Faegri et al. further divides these categories into different primary types of knowledge. Knowledge about task strategies and task content are primary types of task-related knowledge. Team membership knowledge, team member model and expertise location are primary types of team-related knowledge. Team interaction mental models and knowledge about team norms are primary types of process-related team knowledge. Knowledge about overarching team goals and strategic consensus are primary types of goal-related knowledge. These knowledge categories and their primary types of knowledge are shown in Table 3. Knowledge that belonged to one of these primary types of knowledge was considered to be relevant for our context.

Task-related	**Team-related**	**Process-related**	**Goal-related**
Task strategies	Team membership	Team interaction mental models	Overarching team goals
Task content	Team member model	Team norms	Strategic consensus
	Expertise location		

Table 3: Categories of team knowledge and their primary types of knowledge.

The second item that needed to be learned in order to measure the team's degree of shared knowledge, was a manner in which overlap between held knowledge may be measured. In other words, it was necessary to quantify the "sharedness" of held team knowledge, in order to create a measure of the team's overall shared knowledge. Cooke et al. (2000) [13] claims that quantification of team knowledge makes it easier to compare individual results within a team to assess knowledge similarity, and that team knowledge may be quantified in terms of category frequencies from the coding of qualitative data (e.g. interview transcripts). According to the authors, team knowledge similarity, also referred to as consensus, compatibility, or overlap among team members, is the metric which is most focused on by researchers, as there is a general implication that a high measure of team knowledge similarity leads to more effective teams. It must be noted, however, that this implication is not intrinsically true, as heterogeneous teams, that is, teams whose members have separate roles that requires separate sets of knowledge, require a degree of dispersion of knowledge in order to function. In other words, heterogeneous teams whose team members hold identical knowledge will be highly dysfunctional, and so using team knowledge similarity as the sole metric of quantifying team knowledge will be incomplete and will not provide an accurate picture of the team's collectively held knowledge. This is why a clear definition of boundaries are required to define which knowledge similarities should be quantified. In the paper by Faegri et al. (2016) [24], from which we retrieved a conceptual framework for team knowledge, the term "team knowledge" describes knowledge that is shared among team members. The types of team knowledge proposed in their framework are types of shared team knowledge that is relevant for all members a software development team in order to help them make valid assumptions about activities done by the other team members. The types of team knowledge that is described by Faegri et al. and is included in the conceptual framework of this study does not entail role-specific knowledge, nor in-depth technical knowledge. Instead, the primary types of team knowledge can be thought of as collectively produced frames of reference, or shared models, enabling the formation of accurate expectations of the tasks, the team, and the expected and accepted behaviors [39]. Since the types of team knowledge being quantified are the types that a software development team would collectively possess, and because we are not measuring role-specific or individualistic knowledge types, we argue that using similarity as a metric of quantifiable team knowledge, as described by Cooke et al. (2000) [13], is not an incomplete measure for the purposes of this study.

3.1.3 Team-knowledge sharing activities

The conceptual framework for shared team knowledge applies not only to measure the overlap of shared team knowledge, but also to define which processes was considered as team-knowledge sharing processes. In order to answer **R1:** "Which team-knowledge sharing activities are used by the team?", we need to define criteria for what constitutes a team-knowledge sharing activity. To recap Chapter 2.5.1, knowledge sharing involves a giving and a receiving party, where the giving party must externalize their knowledge to a format that can be internalized by the receiver. Externalization and internalization may take several forms: the giving party may perform actions based on their owned knowledge, and the receiving party may observe these actions in order to understand them; the giving party may explain their owned knowledge through a lecture, and the receiving party may listen; the giving party may codify their knowledge in a knowledge system, and the receiving party may read it. Using this definition we can describe a team-knowledge sharing activity as any activity that demonstrates the externalization and internalization of team knowledge, or the movement of team knowledge through socialization. Furthermore, Chapter 2.5.3 described how Agile practices were associated with sharing of (team) knowledge, with an emphasis on the Scrum methodology. There is a parenthesis around the word "team" because the practices were involved with the sharing of several types of knowledge, involving team knowledge. The practices include working in iterations, stand-up meetings, sprint- and release-planning, sprint retrospective and pair programming. The Scrum methodology brings the product owner into the team, which is cross-functional, in order to reduce barriers of communication. To summarize, we wish to find whether the case-team in question displays any of the practices associated with (team) knowledge sharing, in addition to any observed instances of team-knowledge sharing between two or more team members.

Demonstrated instances of	**Agile practices**
Moving team-knowledge through socialization	Scrum roles (Scrum master, product owner, team members)
Moving team-knowledge through externalization and internalization	Cross-functional team
	Working in iterations
	Stand-up meetings
	Sprint and release planning
	Sprint retrospective
	Pair programming

Table 4: Conceptual framework for identifying team-knowledge sharing activities.

3.2 Research design

This study was conducted as a cross-sectional, descriptive case-study following the positivist research paradigm. The ontological position of this study is that there exist a reality that can be systematically and rationally investigated empirically, and that this reality is driven by general laws that apply to human behavior [55]. In other words, this study was conducted in the belief that there may exist a causal relationship between which knowledge sharing activities are performed by a team and the team's degree of shared team knowledge, and that this relationship may be affected by the team's degree of virtualness. And thus this study aims to contribute to uncovering the reality of this relationship by providing empiric material from a single case, which will hopefully in the future be one of many cases, upon which hypotheses and theories may be built by following the scientific method.

With the objective of answering the proposed research questions in the context of a single case, a middle-ground between extensive and intensive (tight and loose) research design was chosen. According to Miles et al. (2018) [45], a loose research design is befitting for qualitative researchers who consider the item of research to be too complex for explicit conceptual frames, and instead let their conceptual frameworks inductively emerge throughout the course of their study. A tight research design, on the other hand, are fitting for researchers who work deductively with a previously defined construct, and puts more weight on procedures and overhead. Both approaches have their strength as well as weaknesses. With a loose design the data becomes richer as the data collection becomes less selective. This, however, may result in a higher work load of analyzing the data, and the result of analysis may be less generalizable for other cases, as there is no apparent structure which can be replicated. A tighter research design is more economic in terms of workload, and is more probable to yield generalizable results and conclusions. Unfortunately, however, a tight research design places limits on the complexity and flexibility of collected data. Since the objective of this study was to answer a set of concrete research questions proposed in chapter 1, an extensive approach appeared to be appropriate. Yet, due to the fact that the number of similar studies performed was low, it would be difficult to completely anticipate the nature of the data that would be collected. If the research design was too tight, then there could be situations in which data would be taken out of context in order to be forced into a strict framework that may not actually reflect the real world. For this reason, a standardized protocol for data collection was maintained, although there was left room for flexibility and openness for a broad variety of data in case anything unexpected emerged. Unfortunately, a somewhat loosened research design was less economic in terms of the information load to be analyzed, but seeing as the scope of the study was not large to begin with, this seemed to be a reasonable trade-off.

For similar reasons as those stated in the previous paragraph, a hybrid of qualitative and quantitative research design was chosen. According to Guba and Lincoln et al. (1994) [28], a highly quantitative research approach strips away much of the acquired data's context and may fail to provide clarity of the generalizability of the results. A qualitative approach, on the other hand, provides contextual information along with the acquired data, providing greater internal validity as well as an insight into its external validity. Quantitative data is more comparable than qualitative data, however, and so a quantitative approach may yield greater external validity, while a qualitative approach may yield greater internal validity. With the mindset of choosing an approach that would best answer the proposed research questions, it would seem that both a highly quantitative as well as a highly qualitative approach would be inappropriate. If this study was to be one out of many similar studies building a foundation of empiric material, then external validity would seem important, and so a quantitative approach would be fitting. Yet, the nature of our research required at least a partially qualitative approach. For example, an evaluation of the team's degree of virtualness required the measurement of a variable which has previously been defined as a multi-dimensional and continuous measurement

[35, 18]. Our first research question **R1:** "Which knowledge sharing activities are used by the team?" could accept responses of a somewhat discrete nature, but it seemed probable that if a quantitative approach was used to collect these answers (e.g. though a questionnaire), then the informants would be required to be familiar with the field of knowledge management, or else the internal validity of the results would suffer. A qualitative approach could yield discrete answers to **R1** as well, but would provide a greater flexibility to the data collection (e.g. semi-structured interviews), allowing the researcher to ensure that the received responses are, in fact, responses to the questions that they want to ask. Such approaches would also yield a greater richness of the collected data, further improving the internal validity of the results. For our second research question **R2:** "How much overlap is there between the individual team member's team knowledge?", it is evident from its wording that the answer would have to be at least partially quantitative. Yet, from our conceptual framework, a wide variety of information is required in order to measure an individual's held team knowledge, and acquiring those amounts of information through a quantitative manner would most likely be very tedious for the informants. Instead, the most fitting approach to answer this question appeared to be a hybrid of one that is qualitative and one that is quantitative. A semi-structured protocol was used in order to acquire rich information from the informants in order to identify concepts within thematic boundaries. The resulting concepts would later be quantified and compared between the informants.

The case-study method appeared to be the most suitable method of acquiring knowledge about a group, which in our case was a single software development team. The strengths of the case-study method is that it allows for the examination of a phenomenon in depth using various kinds of evidence obtained from interviews with those involved, direct observation of events and analysis of documents and artifacts [67]. As the goal is not to construct or test a theory or hypothesis, but rather, describe the circumstances and dynamics of a single software development team, the method used was narrowed down to the descriptive case-study method. If the goal of a study is simply to describe in-depth a case within it's own context without the goal of constructing or testing a theory or hypothesis that applies to a broader population, then one might question whether the study is at all generalizable. Denzin et al. (2011) [19] posed four possible answers to this, one of which is the following:

> *The first is to say that the question is irrelevant because establishing typicality is not the intent of the researcher. The case is simply the case. The case portrayal might, however, have some utility beyond itself. For example, a researcher might argue that if one had several descriptive studies of the same phenomenon in hand, one might examine whether there is a trend, or if a descriptive study is well done, it might be used for comparison with other descriptive studies of the same or similar phenomenon.*
>
> *- Denzin et al. (2011) [19] - The SAGE Handbook of Qualitative Research, 5th ed, p.609*

As such it was concluded that the descriptive case-study method would serve as a fitting means to the purpose of this study. The goal is not for this study to construct or test a theory or hypothesis in and of itself, but rather to provide one building block out of many, which will contribute to an increased pool of empirical research in the field of knowledge sharing in virtual teams which may provide the grounds upon which to construct and test theories and hypotheses in the future.

In order to answer the second research question **R2:** ”How much overlap is there between the individual team member's team knowledge?”, a conceptual framework as described in chapter 3.1 was used. This framework required inquiry of the team knowledge of individual team members, which would later be compared to measure an overlap score. However, much of the team knowledge held by a team is highly dynamic in nature, and may change from one day to another [13]. For example, which tasks the team is currently working on will change with time as the team completes one task and begins another, and so the task-related team knowledge held by an individual may change from one week to another. For this reason it was required to sample the team members' held team knowledge from within a short window of time in order to avoid threats to internal validity, and so a cross-sectional time perspective was chosen.

3.3 Informants

For this study we needed to gather data from members of a single software development team. In order to choose which team to study there were 2 requirements. First, the team's main work had to be within the field of software development. Second, the team needed to work virtually, meaning that the majority of their work needed to be conducted through the aid of computer-mediated communication. The research institution for which this study was conducted (SINTEF) cooperated with several consultant companies within the software industry, and queried one of these companies whether they currently had any teams fulfilling said requirements.

Through this consultant company the researcher established contact with a cross-functional software development team working for a Norwegian organization. This team consisted of a total of 11 members, 7 of which were employees of the organization, along with 5 consultants from our contacted consultant company. The team included 6 developers and 5 non-developers, including 2 functionally responsible, 1 functional architect, 1 UX-designer and 1 product owner. The team's members had been required to work from home since the onset of the COVID-19 pandemic, and communicated and collaborated exclusively through digital means. By agreement with one of the members of the team, each team member was extended an invitation to participate in the study.

From a total of 11 team members who were invited, 5 team members accepted the invitation to participate in the study, including 3 developers and 2-non developers. Ideally it would be preferred that the entire team accepted, which would improve the internal validity of the results. However, as there are a lack of similar studies and thus no previously established recipe for collecting data of this nature, a flexibility of and openness to the data collection was required. In other words, large amounts of data would need to be collected from each informant, and so for the completion this study to remain feasible given the time limit of a master's project, 5 informants seemed to be an adequate number of informants, although perhaps on the lower-end of what would be preferred.

3.4 Data collection

The main method of data collection were in-depth, semi-structured interviews with each informant. However, prior to the initiation of data collection, the researcher were invited by the team to attend and observe three digital team meetings; a stand-up meeting, a retrospective meeting, and a sprint planning meeting. The three meetings took place through a video call on Microsoft Teams, to which each team member attended from their own remote location. The researcher entered the digital meeting room and turned off their audio and video feed before all the attendants arrived, then silently observed as the meeting carried on as usual. During the observed meetings the researcher wrote a very rough transcript of the events that took place as well as their own thoughts about them, while also noting any observed sharing of team-knowledge. After the meeting was concluded, the researcher wrote a summary. These observations served multiple purposes. First, they provided context knowledge that proved useful when writing an interview script, as these insights gave the researcher an idea of how questions may be phrased in order for them to make sense to the informants. Furthermore, it provided valuable insight that would help supplement the description of the context. Lastly, it provided an opportunity for to see the team in action, and see first-hand demonstrations of team-knowledge sharing, instead of just hearing about them from the interviewed informants. This brings us to the next source of data.

Each of the participating informants (5 in total) were interviewed, one at a time, within a 3-day time-span in the middle of the week: 1 interview on the first day, 3 interviews on the second day, and 1 interview on the last day. The interviews all took place over video-calls on Microsoft Teams which both the interviewer and the interviewees attended from their own home. The informants were told prior to starting that they were free to talk as much or little as they would like about the subjects that would be discussed, and that it was perfectly fine to not answer a question if they didn't want to. The informants were asked questions according to Interview Guide (see Appendix A), and the interview was recorded using an audio-recorder. These recordings would later be transcribed by the researcher, and the transcriptions served as basis for analysis.

After the transcribed interviews were analysed and their results were accumulated, a subset of the results were presented to the Scrum-master of the team during a feedback meeting. The purpose of this meeting was to provide the Scrum-master with insight about their own team, but also to ask them whether the results made sense to them, in order to detect any anomalies that would threaten internal validity. The meeting took place on Microsoft Teams, and was attended by the researcher, their supervisor, and the case team's Scrum master.

The researcher held a power-point presentation of the presented results, and the three attendants discussed the results during the presentation. At the end of the meeting, the Scrum-master was asked questions in order to fill any gaps in the researcher's previously possessed context knowledge. By agreement with all the attendants, the meeting was recorded using Microsoft Team's own recording feature. The recording was not transcribed or analysed, but re-played by the researcher while taking notes about items that should be added to the context chapter.

3.5 Data analysis

A set of 5 interview transcripts along with summaries of 3 observed meetings required analysis in order to structure our findings. A qualitative analysis method was used in which sentences and paragraphs were assigned to codes according to what kinds of information they could provide us. A pre-defined code hierarchy was first established according to the conceptual framework described in Chapter 3.1. For each unit of data (i.e. interview transcript or observed meeting summary) its containing sentences were assigned codes to the pre-defined code structure in an iterative manner, level by level, until the bottom of the code hierarchy was reached. Then, each of the bottom-level codes were analysed in a more open-ended manner; some codes were analysed qualitatively by identifying patterns and concepts that occurred, while others were analysed semi-quantitatively in order to quantify how many interviewed informants expressed in agreement with a specific statement. To code the data, a qualitative data analysis software named NVivo was used, which allows for the coding of text-based data. The 5 interview transcripts and 3 meeting observation summaries were uploaded into NVivo and read through once before the analysis process started.

3.5.1 Pre-defined code structure

Before the analysis process could begin, it was necessary to establish a pre-defined code hierarchy to provide structure for the analysis. At the top level a code was created for each data type to be analysed, that is, one for interviews and one for observations. The "Observations" code was given three children, one for each observed meeting, named "Retrospective", "Sprint-planning" and "Stand-up". See Figure 4 for reference.

Within the "Interviews" code, a pre-defined code hierarchy was set up based on the conceptual framework described in Chapter 3.1.

At first a code was created corresponding to each of the research questions, named **R1**: "knowledge sharing activities" and **R2**: "Team knowledge", in addition to a third code named "Team virtualness". These top-level codes were given child codes according to the conceptual framework described in Chapter 3.1. The top level code named "Team virtualness" was given two level-2 child codes, named "Use of technology" and "Team distribution", respectively. The level-1 code named "**R1:** Knowledge sharing activities" was given three level-

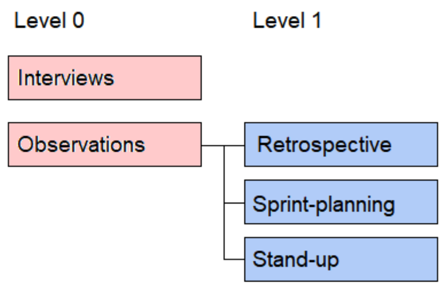


Figure 4: The top levels of the code hierarchy. At level 0 a code was made for each type of data to be analysed. At level 1, a code was created for each observed meeting under the "Observations" code.

2 child codes, named "Socialization", "Externalization + internalization", and "Agile practices", respectively. The level-1 code named "**R2:** Team knowledge" was given four level-2 child codes: 1) "Goal-related team knowledge" which is given two level-3 child codes, named "Overarching team goals" and "Strategic consensus"; 2) "Process-related team knowledge" with two level-3 child codes named "Team interaction mental models" and "Team norms"; 3) "Task-related team knowledge" with two level-3 child codes named "Task content and "Task strategies"; and lastly 4) "Team-related team knowledge" with three level-3 child codes named "Expertise location", "Team member model", and "Team membership". The pre-defined code hierarchy is shown in Figure 5.

Within the "Observations" code, a pre-defined code hierarchy was set up within each of its three child codes, named "Retrospective", "Sprint-planning" and "Stand-up" respectively. The pre-defined codes used were identical between the three level-1 observation codes, and so the figures will only show the code hierarchy with respect to the level-1 code named "Retrospective". Within each observation code, two level-2 codes were created, named "Protocol" and "Knowledge sharing activities" respectively. Within the "Knowledge sharing activities" code, a code hierarchy was set up according to the conceptual framework for knowledge sharing in Chapter 3.1. Level-3 child codes were created named "Socialization", "Externalization", and "Externalization and internalization" accordingly. This pre-defined code hierarchy is shown in Figure 6.

3.5.2 Pre-coding the data

The interview transcripts were coded in an iterative manner, first by coding sentences into the pre-defined code hierarchy, one level at a time, and then later in a more open-ended manner. In the first iteration, the interview transcripts were read through while each sentence of the informants' responses were considered whether it carried information about the team's degree of virtualness, or information that would help answer any of the research questions, in which case they were assigned level-1 codes respectively. For example, if a response mentioned

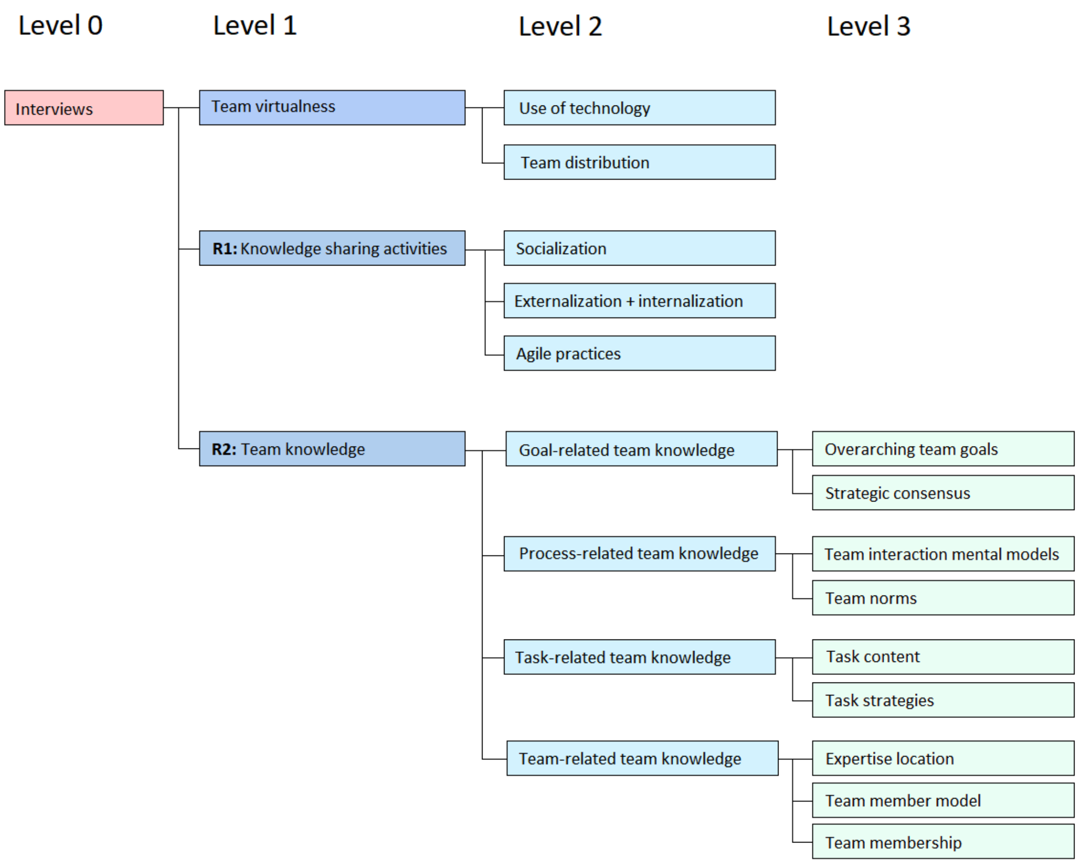


Figure 5: Pre-defined code hierarchy used for analysis of interviews. These codes were then divided into sub-codes at level 2 and 3 according to the conceptual framework used.

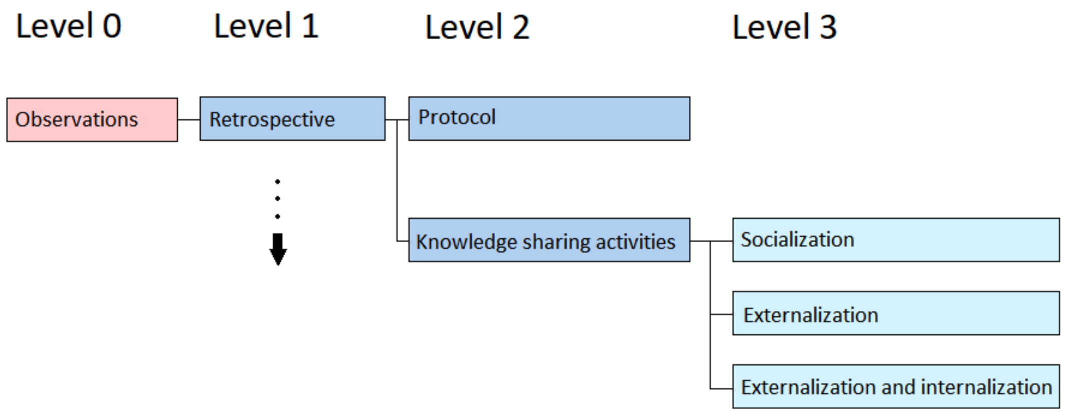


Figure 6: Pre-defined code hierarchy used for analysis of observed meetings, shown with respect to the "Retrospective" code. A code was created for information about the meeting protocol, and another code was created for displays of sharing team knowledge.

anything that could be related to the team's degree of virtualness, then the relevant sentence was coded into the "Team virtualness" code, along with any surrounding sentences if they provided context for the relevant sentence. In the second iteration, the accumulated sentences assigned to each of the level-1 codes were read through again, while simultaneously re-coding the sentences into any of the code's level-2 child codes. For example, if a sentence previously coded into "Team virtualness" mentioned anything relevant to the use of technology, then that sentence would be re-coded into the level-2 code "Use of technology", along with any of its surrounding sentences if they provided context for the relevant sentence. This process was repeated in the third iteration, where the sentences assigned to level-2 codes were read through and re-coded into its relevant level-3 child codes, if any. The remaining bottom-level codes would later be analysed in a more open-ended manner, in which each of the informants' statements would be assigned their own codes, and similar statement codes would be combined to get an overview over how many informants expressed any given statement.

Similarly to the interviews, the observation summaries were read through while simultaneously assigning codes to sentences and paragraphs. For each observation summary, sentences were assigned codes from the pre-defined core hierarchy descending from the observation's own code. For example, when coding the summary of the observed retrospective meeting, sentences and paragraphs were assigned either of the pre-defined level-2 codes descending from the level-1 "Retrospective" code (note that identical code hierarchies descended from each of the three level-1 codes). This process was carried out in an iterative manner. During the first iteration, the summaries of each observed meeting was read through while simultaneously assigning sentences to either of its descending level-2 codes as shown in Figure 6. Every sentence or paragraph that contained information about the meeting itself, or could provide a description or outline of the protocol of the meeting, was assigned the level-2 code named "Protocol". Every sentence or paragraph that displayed instances of team-knowledge sharing was assigned the level-2 code named "Knowledge sharing activities". During the second iteration, the accumulated sentences and paragraphs in the level-2 code named "Socialization", "Externalization", or "Externalization and internalization", depending of the nature of which the knowledge was shared.

3.5.3 Observed meeting protocol

From the written summaries of the observed meetings, any sentences that either contained spoken statements by the attendants that described the meeting protocol, or notes that described what happened during the meeting, were assigned the level-2 code named "Protocol" in their respective level-1 code (there is one for each observed meeting). The result of this coding was a chronological list of strings describing events that happened throughout the meeting, as well as statements that were made by the attendants about the meeting protocol. This list was used as the basis for writing a narrative about what is done during these meetings, and which practices are used. These narratives were used in the next chapter when describing the team's workflow when presenting the case context. Moreover, these narratives supplemented the results from the analysis of interviews with regards to the virtualness of the team as well as the team-knowledge sharing activities used by the team.

3.5.4 Team virtualness

During the first iteration of the interview transcript coding, any sentence that provided insight into the team's degree of virtualness was assigned the level-1 code of "Team virtualness". More specifically, any sentence that included information related to any of the dimensions of team virtualness listed in Table 2 (see Chapter 3.1) was assigned to the level-1 code named "Team virtualness". Moreover, if any of its surrounding sentences provided relevant context for the meaning of the sentence in question, then those sentences were included as well. On the second iteration, all of the sentences previously coded as the level-1 code "Team virtualness" were re-coded into either of its level-2 child codes "Use of technology" or "Team distribution", depending on which group (of Table 2) the sentence's relevant dimension of virtualness belonged to. Again, any surrounding sentences that provided context to the sentence in question were also included. The "Team distribution" code was assigned to any sentences that were related to any of the dimensions of virtualness falling under the "Team distribution" group of Table 2, including team's geographic dispersion, percentage of time spent apart while working on a task, degree of physical distance, synchronicity of communication, temporality and diversity. The "Use of technology code was assigned to any sentences that were related to any of the dimensions of virutalness falling under the "Use of technology" group of Table 2, including the team's electronic dependence, level of technology support, use of virtual communication tools, amount of informational value provided by those tools, and the use of computer mediated communication.

During the third iteration the contents of "Team distribution" and "Use of technology" were analysed in an open-ended manner, meaning that no predefined codes were used. Within the "Team distribution" code we noticed that all of the sentences described one of the six virtualness dimensions related to team distribution, and so level-3 child codes were created for each of these dimensions, as shown in Figure 7. We also noticed that within the contents of "Use of technology", the sentences described the use of technology related to either documentation, meetings, or other forms of communication. Thus three level-3 child codes were created, named "Documentation", "Meetings" and "Other interactions", respectively. Even within these level-3 child codes new patterns were detected, and so level-4 child codes were created. The "Documentation" code was extended to include three level-4 child codes named "Jira", "Teams" and "Wiki". The "Meetings" code was extended to include three level-4 codes named "Miro", "Teams" and "Technology satisfaction". The "Other interactions" code was extended to include two level-4 child codes named "Teams and "Communication threshold". The resulting code structure is shown in Figure 7. During the fourth iteration, the contents of the level-3 codes descending from "Use of technology" were re-coded into either of its level-4 codes depending on which technology they referred to. Moreover, within the "Meetings" code, sentences containing an expression of the satisfaction of the usage of technology for meetings were assigned the "Technology satisfaction" code. Within the "Other interactions" code, sentences describing the threshold for reaching out to other team members by using technology was assigned the "Communication threshold" code. When the pre-coding was complete and new child codes were created for emerging patterns, we were left with a set of open-ended codes that would later be analysed, which are listed in Table 5.

Within each of the open-ended codes listed in Table 5, the content was read through while simultaneously assigning an individual code to each statement. For example, within the "Geographic disperison" code, individual codes were made for each statement that were related to the team's geographic dispersion. This process was repeated for every open-ended code, in order to extract informant statements related to the topic of their parent code. Similar statements were later combined into more general ones, resulting in the statements presented in Chapter 4.4.1.

3.5.5 Team-knowledge sharing activities

Both the interview data and meeting observation data was analysed with the goal of identifying the team's knowledge sharing activities. In Chapter 3.1 a conceptual framework was established in which a set of concepts were deemed relevant to identify team-knowledge sharing activities. The first item we would search for was mentions or observed demonstrations of the movement of team-

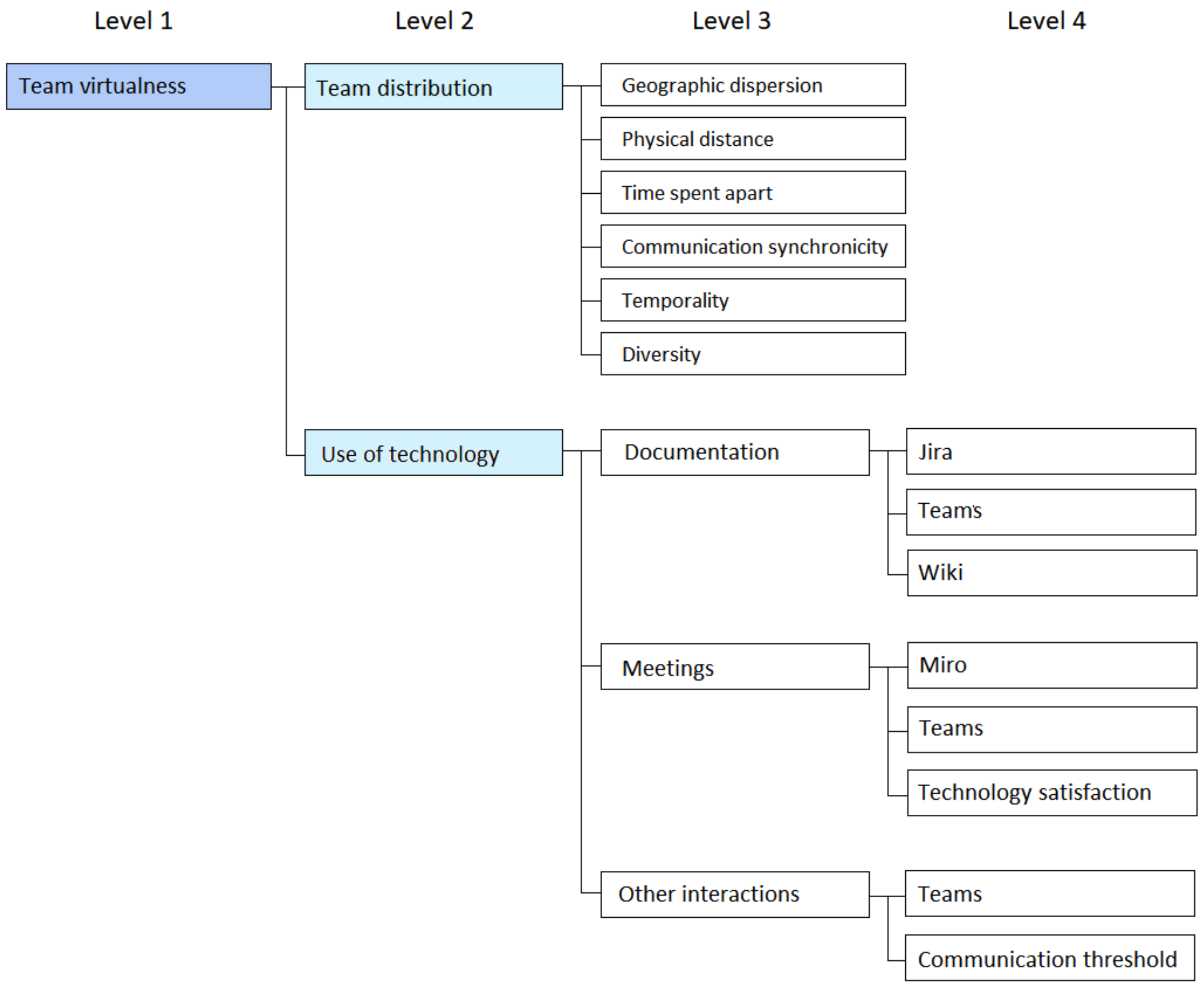


Figure 7: Emerged code structure from analysing the content of the pre-defined codes descending from ”Team virtualness”.

Open-ended codes
Geographic dispersion
Physical distance
Time spent apart
Communication synchronicity
Temporality
Diversity
Jira
Wiki
Miro
Teams (Documentation)
Teams (Meetings)
Teams (Other interactions)
Technology satisfaction
Communication threshold

Table 5: Open-ended codes descending from "Team virtualness"

knowledge, either through socialization, or through externalization and internalization. The second item we would search for was the mention or observed usage of the Scrum methodology or Agile practices such as Scrum roles, cross-functional teams, working in iterations, stand-up meetings, sprint and release planning, sprint retrospective or pair programming.

The interviews were coded in an iterative manner, and in the first iteration, any sentence that either mentioned the sharing of knowledge through socialization or externalization and internalization, or mentioned the use of agile practices, were assigned the level-1 code named "**R1**: Knowledge sharing activities." Any surrounding sentences that seemed relevant for context was included as well. During the second iteration, the contents of the level-1 code was re-coded into either of its level-2 codes. Any sentences that mentioned the sharing of team-knowledge through socialization was assigned the level-2 "Socialization" code. Any sentences mentioning the sharing of team-knowledge through externalization and internalization was assigned the level-2 code named "Externalization + internalization". Any sentences mentioning the use of Agile practices associated with knowledge sharing were assigned the level-2 code named "Agile practices". The level-2 codes "Agile practices" and "Socialization" were left open ended, but within the code "Externalization + internalization" a pattern of responses were detected, and so the "Externalization + internalization" code was given five level-3 child codes, named "Documentation", "Lecture", "Training", "Workshop" and "Other". Sentences that mentioned either of the first four activities were assigned their respective level-3 code, while the remaining sentences were assigned to the "Other" code. After three iterations of coding, we were left with the resulting code structure as shown in Figure 8.

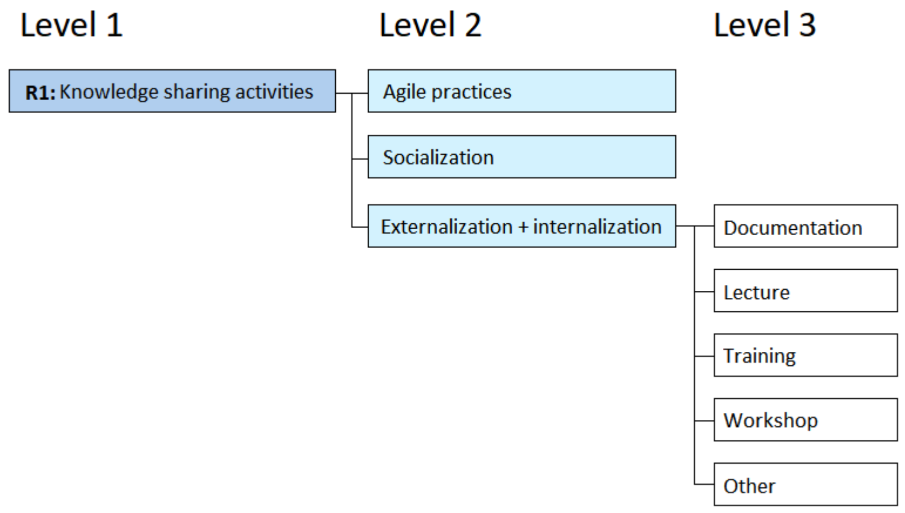


Figure 8: Emerged code structure from analysing the content of the pre-defined codes descending from "**R1:** Knowledge sharing activities".

After three iterations, we were left with a set of open-ended codes (i.e. codes with no child codes), which all had a set of sentences assigned to them. These codes were the level-2 codes named "Agile practices" and "Socialization", as well as the level-3 codes named "Documentation", "Lecture", "Training", "Workshop" and "Other". The open-ended codes are listed in Table 6. Each of these open-ended codes were then analysed, one at a time, by creating a child code for each of the mentioned knowledge sharing activity. These codes, along with the activities seen in the observations, would form the basis for the results with the objective of answering **R1**: "Which team-knowledge sharing activities are used by the team?".

The written summaries from the three observed meetings were also analysed iteratively with the objective of identifying knowledge sharing activities. First, the entire contents of every meeting were assigned to each their own respective level-1 code as described in Figure 6. The observed retrospective meeting was assigned the level-1 code named "Retrospective", the sprint planning meeting was assigned the code named "Sprint planning", and the stand-up meeting was assigned the code named "Stand-up". Each of these level-1 code had a child code structure as described in Figure 6. In the second iteration the contents of each of the three level-1 codes were re-coded into either of its level-2 child codes named "Protocol" or "Team knowledge". More specifically, any sentence that described an event of the meeting, or anything that contained information regarding the meeting protocol, was assigned the code named "Protocol". Any sentence that described an event in which team-knowledge was shared was

Open-ended codes: Interviews
Agile practices
Socialization
Documentation
Lecture
Training
Workshop
Other (Externalization + internalization)

Table 6: Open-ended codes descending from "**R1:** Knowledge sharing activities"

assigned the code "Team knowledge". For any of the coded sentences, surrounding sentences were included if they seemed relevant for context. During the third iteration the contents of the level-2 "Team knowledge" code was then re-coded into either of the level-3 codes named "Goal-related team knowledge", "Process-related team knowledge", "Task-related team knowledge" or "Team-related team knowledge", depending on which type of team-knowledge was described being shared. During the fourth iteration, the contents of these level-3 codes were again re-coded into their respective primary types of shared team knowledge as described by the conceptual framework in Chapter 3.1. No other codes were created in addition to the pre-defined codes, and the set of codes left open-ended for further analysis are shown in Table 7.

Open-ended codes: Observations
Protocol
Overarching team goals
Strategic consensus
Team interaction mental models
Team norms
Task content
Task strategies
Expertise location
Team member model
Team membership

Table 7: Open-ended codes descending from each of the level-1 codes named "Retrospective", "Sprint planning" and "Stand-up".

The open-ended codes descending from each observed meeting served multiple purposes. The "Protocol" code was used to get an outline of the chronological order of events that happened throughout the meeting, and was analysed to get a picture of which practices were used by the teams throughout the meeting, or which events took place during the meeting that lead to the sharing of team knowledge. The remaining codes in Table 7 were used to get a picture of which types of team knowledge were shared during each observed meeting.

3.5.6 Shared team knowledge

During the first iteration of analysis, any sentence from the interview transcripts that contained information about the informant's team knowledge (i.e. it was related to any of the pre-defined level-3 codes of Figure 5), was assigned the level-1 code "**R2:** Team knowledge". On the second iteration, any sentences that said anything about overarching team goals or strategic consensus was assigned the level-2 code of "Goal-related team knowledge". Sentences that said anything about team interaction mental models or team norms were assigned the level-2 code of "Process-related team knowledge". Sentences that said anything about task content or task strategies were assigned the level-2 code of "Task-related team knowledge". And finally, sentences that said anything about expertise location, team member model or team membership was assigned the level-2 code of "Team-related team knowledge". On the third iteration, all the sentences previously assigned to either of the level-2 codes "Goal-related team knowledge", "Process-related team knowledge", "Task-related team knowledge" or "Team-related team knowledge" were assigned to their respective pre-defined level-3 codes as shown in Figure 7.

During the fourth iteration, the resulting set of level-3 codes were then analysed in a more open-ended manner, meaning that child codes were created as patterns of concepts was recognized in the data. Within the "Team interaction mental models" code, sentences could be further divided into three groups as they were either related to regular team meetings, irregular team meetings, and other interactions, and so each of these groups were assigned to level-4 child codes with respective names. Within the "Task content" code, the sentences could be further divided into three groups and assigned level-4 codes accordingly. The first group was named "Tasks", and contained sentences related to which tasks the team was currently working on. The second group was named "Dependencies", and contained sentences related to dependencies between the team's tasks. The third group was named "Environment", and contained information about how the team's tasks affected the environment surrounding them. Within the "Task strategies" code, a set of sentences appeared to be related in that they contained information regarding the team's task allocation strategies, and so a subset of the statements that emerged within the "Task strategies" code was assigned it's level-4 code named "Task allocation strategies", while the rest was assigned a level-4 child code named "Other task strategies". The resulting codes are shown in Figure 9. The end result of this iterative coding of the contents in "**R2:** Shared team knowledge" was a set of leaf-nodes from which more concrete statements could be derived. The resulting leaf-codes after the fourth iteration are shown in Table 8.

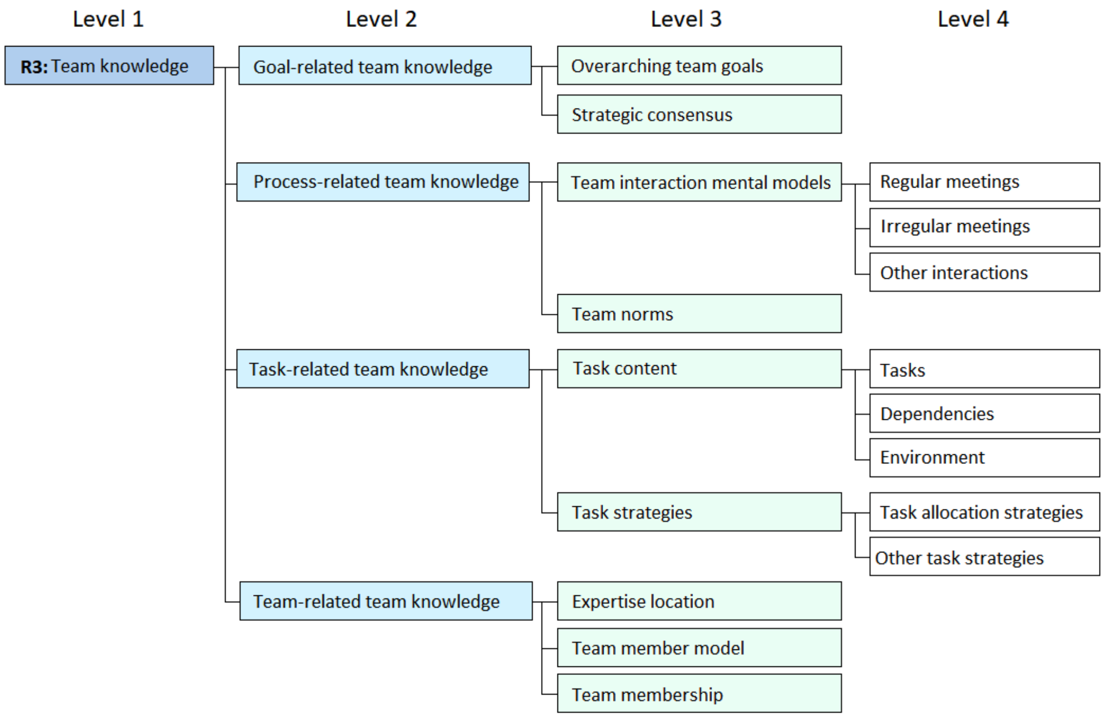


Figure 9: Emerged code structure from analysing the content of the pre-defined codes descending from "**R2:** Team knowledge" for four iterations.

Resulting codes from fourth iteration
Overarching team goals
Strategic consensus
Regular meetings
Irregular meetings
Other interactions
Tasks
Dependencies
Environment
Task allocation strategies
Other task strategies
Expertise location
Team member model
Team membership

Table 8: Leaf-codes resulting from the fourth iteration of the coding process of the contents of "**R2:** Team knowledge"

During the fifth iteration of coding, individual codes were created for each of the statements made by the informants within the resulting codes from the fourth iteration, and during the sixth iteration similar statements were combined to form more general ones. When the sixth and final iteration of coding was complete, what was left was a set of statements related to any of the codes listed in Table 8. Moreover, a number had been attached to each statement, representing how many informants had expressed a statement of similar intent. For each of the listed leaf-codes, its related statements were presented together using a bar graph showing how many informants expressed a given statement. Furthermore, if a set of informants expressed a given statement, and a subset of those informants expressed a related and more specific statement, then a bar graph of a stronger colour and smaller font was shown inside the original bar graph. This provided a visualization of how many informants expressed each statement, and to which degree they did specify in their statement.

3.5.7 Calculating overlap measure

To calculate a measure of overlap within a group of statements, the average number of respondents for each statements within the group was calculated. The overlap measure was assigned a label ranging from low to high according to it's average value, according to Table 9. Initially, overlap measures were calculated by including every single received response. A second calculation was then performed after filtering of responses that were unspecified or unrelated, as well as grouping together responses that likely referred to the same thing.

The filtering and combination of responses were dependent on the researcher's own interpretation of the results, and thus for transparency reasons both the initial and the second calculation were included in the results.

Numeric average	**Label**
4.0 to 5.0	High
3.5 to 4.0	Moderate-to-high
2.5 to 3.5	Moderate
2.0 to 2.5	Low-to-moderate
1.0 to 2.0	Low

Table 9: Labels assigned to the numeric average value of respondents of statements.

3.6 Quality of the research

The feasibility of this study was affected by the circumstances around which this thesis project was conducted. Normally, master's students at NTNU spend the first semester of the year to conduct a project in which they gain background knowledge within a field, which they later use to write their master's thesis in the next semester. The researcher's first-semester project was in relation to a field very unrelated to the fields of knowledge sharing and agile teams. For this reason, the researcher was required to learn their background knowledge while simultaneously performing the thesis project. In hindsight, a consequence of this is that some early decisions, such as which questions to include in the interviews, were not thoroughly informed, resulting in missed opportunities to get richer data.

The researcher's inexperience as an interviewer caused some ambiguity to spesificness of some of the interview questions, and as a result the answers to some questions had varying degree of detail. Moreover, the researcher did not probe the informants in order to compensate for their varying degree of detailed answers. These factors may affect the internal validity of results related to overlap measures of team knowledge.

The number of interviewed informants and observed meetings were also somewhat low, and in some cases the results may have been too weak to draw concrete conclusions. Ideally we would have interviewed the whole team, but only 5 informants agreed to interviews. If the project had a longer duration it would have been possible to find a different team in which a larger portion of the members were able to participate in interviews, but the researcher had already committed to this one team, partly due to time restrictions, and partly due to the researcher's own positionality. Furthermore, there were several types of meetings that we did not observe, such as estimation meetings and requirement phase meetings, simply because the team did not hold any such meeting for the duration of the data collection period.

3.7 Researcher positionality

The selection of case team was largely affected by the researcher's part-time employment for a consultant company. Before the beginning of the thesis project, the student contacted their employers and asked them whether they had any specific topics that they wished for him to write about in the master's thesis. His employers put him in contact with a researcher in a partnering research institution (i.e. SINTEF) who would serve as a guide and supervisor. Due to this partnership between SINTEF and the researcher's employers, the case team was found through the consultant company's network. As a result, several of the members of the case team were hired consultants employed at the same consultant company as the researcher. Although the researcher was not familiar with any of the team's consultants, they all were aware that they were colleagues, which may have affected how the researcher interacted with the informants who were hired consultants.

Furthermore, the researcher was a software developer, although a somewhat inexperienced one. However, being a developer, the research will most likely have a larger focus on the developer's perspectives and that of the non-developers, simply because that is what the researcher is familiar with. Interviews were performed on both developers and non-developers, but because the researcher, due to his trade, more naturally speak the same language as developers, the interview questions may be more suited for developers than for non-developers. Had the interview guide been written in collaboration with a non-developer, then the interview questions would likely have more quality assurance in terms of generalisability between the roles.

3.8 Ethical considerations

Seeing as this study would require the collection and analysis of personal and organizational data, a set of ethical considerations had to be made. Before conducting the data collection, an application was sent to the Data Protection Official at the Norwegian Centre for Research Data (NSD), including a description of the study along with the data collection and processing protocols. Shortly after, the NSD responded with a letter of approval (see Appendix B), stating that the implementation of this study was considered to be legal and ethical on the condition that it was implemented in the manner which had been stated on the application. It was also necessary to ensure that each participant was informed in their decision to participate, along with their rights. A letter of invitation was then electronically forwarded to all of the team's members (see Appendix C). This letter described the purposes and scope of the study, which questions it hoped to answer, and what participation would entail. It also included information about how a participant's data would be processed, and who the participants could contact if they wished to view, alter or delete their data. Moreover, the letter informed the team members that participation was voluntary, that individuals could choose to not participate in their study, choose not to answer individual interview questions, or to withdraw their participation from the study at any time, without any negative consequences to them or to their team. Lastly, the letter of invitation included a consent form through which participants could express their consent to participate in either the interviews, observations, or both, which were collected prior to the data collection procedures. Before each interview took place, a short summary of the participant's rights were repeated orally in order to ensure the participant was thoroughly informed, before asking the participant for permission to audio record the interview, to which all the informants agreed. The audio recordings of the interviews along with their transcriptions were labeled with the chronological order in which the interviews took place, and stored on a secure SINTEF server.

When presenting the context and results, the data was anonymized so that neither the informants nor their employing organization would be identifiable. When referring to specific informants, they were simply referred to as "they", and their roles would be generalized from specific roles into being either "developer" or "non-developer". The employing organization would simply be referred to as "the organization" or something of a similar nature. Furthermore, whenever the informants mentioned identifiable names of the services or projects that they were currently working on, their names were exchanged for something generic, such as "Project P1" and "Project P2" ect.

4 Case context

This chapter will present the case that was studied in order to provide context to the results of the next chapter. The organization under which the case team worked will be presented in the first sub-section. Next we will present the team including its responsibilities, members and structure. In the third section we present the team's workflow, and describe the process of their regular in-team meetings. Lastly, we present our results related to the team's degree of virtualness.

4.1 The organization

The studied case team was one out of several teams working under an organization of the Norwegian public sector whose employees worked remotely from home due to the COVID-19 pandemic. The organization was responsible for a domain which involved a total of 4 processes in addition to the management of a large amounts of users and their personal data, and the service of their customers. A total of three departments were responsible for their own sub-domain, and each department's sub-domain were further divided between a set of teams belonging to each department. "Department 1" and "Department 2" both had 4 teams each, and "Department 3" had only one team. Each team was composed of somewhere between 6 and 14 team members. In addition to these teams were a set of teams whose support was shared among the three departments, such as hardware, platform and dev-ops teams. For each department, a single person were designated the role of "team-lead", although the organization's definition of "team-lead" reminded more of a department or project manager, as the "team-lead" was not directly involved with the team's daily work, but rather responsible for managing all of the department's teams in a more overarching manner, such as keeping an overview of the teams' statuses,ensure they have access to their required resources, and act as an intermediary between the teams and higher management. As a consequence, the Scrum-masters of each team were inclined to adapt some of the more traditional "team-lead" responsibilities by overseeing their team's daily work.

4.2 The team

The case team which was studied belonged to "Department 1", which was the department that concerned itself with the service of end-users, management of user data and two of the organization's processes. The team itself were responsible for several items. First, they were responsible for the re-implementation of a work tool used by the organization's case-managers. The case-managers were employees of the organization who handled the service of the organization's customers and members, and followed up on their cases. Their work tools were previously built in Flash, which would soon be obsolete, and so the work-tools had to be re-implemented in React. The case team were responsible for re-implementing the tool's front-end, while another team were responsible for it's back-end. The second item the team were responsible for was a portal through which the organization's customers would be able to log in and report data about their members. The reported data were then to be parsed and stored in the organization's systems, and so a third responsibility of the team was the management of the reported data. At the time when the study was conducted, the team were also responsible for managing a change in customer and member rights due to the merging of several municipalities and counties in Norway.

The team itself was composed of 11 members, including six developers, two functionally responsible, one functional architect, one UX-designer and a product owner. Out of these 11 members, 2 developers and 2 non-developers were hired consultants from an external IT consultant company, while the remaining 7 team members were employees of the organization. Additionally, an architect were grouped with the team because they were newly employed and needed a set of faces whom they could be familiar and talk to daily. The team worked by following the Scrum methodology, and so one of the developers also had the role of Scrum-master, who, in addition to being responsible for enforcing Scrum practices and rules also entailed responsibilities that would traditionally be defined as team-lead responsibilities, such as guiding the team and making sure the team members are able to complete their tasks and ask for help when they require it, and generally manage the team's day-to-day activities. Moreover, a set of case-managers had a part-time position in which they supported the team by informing them about their needs regarding the work-tools that the team was developing. The case team is referred to as a "team" throughout this paper, although in reality they more closely resemble a group in that not every team member is mutually dependent [36]. While the team's developers worked together in a mutually dependent manner that more closely resembled that of a "team", the non-developers operated more like support functions for the developers as they were not as involved in the daily work of every team member. For example, the architect did not share tasks with the rest of the team members, and were only grouped with them for social reasons. The two functionally responsible worked with testing functionality that were developed by the developers, but one of them were involved with the front-end worked while the other were involved with the back-end work.

4.3 Workflow

The team worked in terms of releases and sprints, where one release (or delivery) was divided into three week long sprints. Prior to the start of a new release "behovsfase" meetings, or requirement phase meetings, would be held in the team to present and discuss the requirements related to the release. Preplanning meetings were held during the last week of every sprint. During these meetings, the department's team-lead and the Scrum masters and functionals of every team in the department would plan which tasks should be worked on during the coming sprint, and give each task a priority. The first day of the new sprint each team then holds their own sprint-planning meetings, in which each team's tasks are prioritized and planned in a finer manner. As the tasks have already been prioritized during the pre-planning meetings, the main purpose of the meeting is for the team to gain a common overview of what the coming sprint will be like. If the team finds it necessary then they will shortly after also hold estimation meetings, in which each task is estimated in terms of how many work hours are required to complete it. Throughout the sprint the team holds daily stand-up meetings in order to keep each other up to date on their progress, and at the last day of the sprint the team holds a sprint retrospective where they report on what went well, what didn't, and which steps they can take to improve the next sprint.

4.3.1 Sprint-planning

The first Monday of every sprint a meeting was held in order to plan out which task would be worked on during the sprint. The team has a pre-defined "weight" capacity for the sprint, depending on how many developers are on the team, where one "weight" represents one week of work for a single developer. During this meeting, the team commonly go through their tasks, and each task is briefly explained, usually by one or two people, to the rest of the team. The tasks are presented to the team through notes on a board, where each note represent each task. The notes are sorted by the tasks' priority, and they are grouped according to where they are in the task workflow, which is one of the following stages: 1) "Open / In analysis", 2) "Ready for execution", 3) "In progress", and 4) "Awaiting dependency". The team commonly discuss each task briefly, and move the notes between the groupings depending on whether the team should work on it this sprint or not. Some of the tasks presented to them already have an estimation in terms of weights, and some tasks are given estimation during the meeting. However, the team may hold a meeting after the sprint-planning meeting in order to estimate tasks that do not have an estimation. The team is informed of which person has the role of Sheriff, and they are asked whether any of the team members wish to hold a KOMPIS-lecture in which they lecture a topic to the rest of the team.

4.3.2 Daily stand-up

Every day before lunchtime the team held a stand-up meeting. During the meeting, which was led by the Scrum master, every single team member gives a brief description of the task they are currently working on, whether they have any problems, and whether they need any help from their teammates. The meetings are scheduled to last 15 minutes, and its purpose is not to initiate complex discussions, but rather simply keep the rest of the team up to date with the work one is doing. If some of the team members realize that a topic requires further discussion, then they are encouraged to schedule time for this after the stand-up meeting is finished.

Despite the fact that only the developers functioned as a mutually dependent "team" while the entire team functioned more as a group, every one of the team's members attended the daily stand-up meeting, together with the architect and sometimes even the department's team-lead. Furthermore, the case-managers, for which the team were developing work-tools, were also invited to the daily stand-up meetings, and would occasionally attend, although usually they would be too busy with their own work. This is peculiar, because the practice of daily stand-up meetings were made for teams and not for groups, and are known to be less productive and satisfactory with a high number of attendants [59]. During a conversation with the team's Scrum master we were told that before the COVID-19 lockdown, there was some negativity related to the daily stand-up meetings as they would often get side-tracked and not be as effective as they could have been. However, more recently, there has been a shift of focus from reporting what was done yesterday to informing what each individual is working on in the present, in order to find out who needs help and who needs to schedule further discussion after the stand-up meeting is concluded. Furthermore, after the COVID-19 lockdown and the enforcement of working remotely from home, the Scrum master allowed the team some slack to socialize during these meetings, because some days the daily stand-up would be the only time where some team members met other people from work. Because of this there has been more positivity around the daily stand-up, and it has become more crucial function in order to reduce the perceived distribution of the team.

4.3.3 Sprint retrospective

The last Friday of the sprint the team holds a sprint retrospective in which they reflect on what went well and what didn't, in order to improve the next sprint. The team use a common board on which each team member are free to write notes and place them in either of three columns, named "What worked well", "What worked not so well", and "Points of action". One team member leads the meeting through its agenda, although it varies which team member is responsible for leading the retrospective meeting. During the first 15-30 minutes of the meeting all of the team members write their own notes and place them in either of the two first columns. These notes contain thoughts about what has been positive and what has been negative during the last sprint. When the team has finished writing notes the team commonly go through every note, although there seemed to be more focus on the notes within the "What didn't go well" column. For each note, the team discuss what the problem is, whether the team should do something to improve it, in which case they discuss what can be done. If the team decides to take action, a new note is placed in the "Points of action" column of the board, briefly describing what should be done, along with who should do it. When the team has discussed every note in the two first columns, brief documentation is written describing who attended the meeting, along with which points of action were decided upon, and who is responsible for each point.

4.3.4 Sheriff

Part of the team's areas of responsibility is the maintenance of the services and applications that fall under their part of the organization's domain. Every week a team member is appointed to the role of "Sheriff", meaning that they are responsible for monitoring the health status of the team's services. The Sheriff must regularly look through health check dashboards and system logs to detect any malfunctions, as well as read any incoming errors and alarms. If an error is detected, the Sheriff is then responsible for either fixing the error, or delegate the job to someone else in the team if they don't know how to fix it themselves.

4.3.5 KOMPIS

During every sprint planning meeting, the Scrum-master asks the team whether any of them wants to hold a KOMPIS lecture. If a team member has something they wish to share with the rest of the team, whether it's technical, domain-related, or something else, the team schedules a KOMPIS meeting in which the team member holds a lecture. Every member of the team is then invited, and the team members can choose for themselves whether to attend or not. During a conversation with the team's Scrum-master, we were told that new people would soon be joining the team, and so the team was planning to hold a KOMPIS meeting in which one of the team's functionally responsible would lecture about the team's domain.

4.4 Team virtualness

To evaluate the team's degree of virtualness, a conceptual framework was constructed in which a list of dimensions of virtualness, as summarized by Johnson et al. (2009) [35], were grouped into two groups, "Team distribution" and "Use of technology", as shown in Table 2 in Chapter 3.1. The complete list of dimensions of virtualness included geographic dispersion, percentage of time spent apart while working on a task, degree of physical distance, synchronicity of communication, temporality, diversity, electronic dependence, level of technology support, use of virtual communication tools, amount of informational value provided by those tools, and the use of computer mediated communication. To get a measure of each of these dimensions we used a combination of statements procured from interview transcripts, observed events during meetings as well as oral communication with the team's Scrum master during a feedback meeting.

4.4.1 Team distribution

The first group of dimensions of team virtualness, as described in the conceptual framework (see Chapter 3.1), were the dimensions related to team distribution, including geographic dispersion, percentage of time spent apart while working on a task, degree of physical distance, synchronicity of communication, temporality and diversity. We will present the results related to the team's degree of distribution by grouping them according to each dimension. The findings are summarized in Table 10.

Dimension of distribution	**Findings**
Geographic dispersion	All team members located in the same country and in (or near) the same city.
Degree of physical distance	Team members do not meet physically, with very few exceptions.
Percentage of time spent apart while working on a task	Approximately 100 percent
Synchronicity of communication	Video calls for meetings. Chat and audio or video calls for communication between individuals.
Temporality	The team worked iteratively. All team members worked the same "core hours" during the day, with a few exceptions.
Diversity	Team members live in the same country and speak the same language.

Table 10: Technologies used for team communication and collaboration.

In Chapter 3.1 we defined the dimension of geographic dispersion as the team's dispersion relative to countries and cities, while degree of physical distance was described as the effort required to physically meet the rest of the team. The informants were not asked directly where their current home address is, however all 5 of the informants said that they used to work in the organization's office located in a Norwegian city prior to the pandemic lockdown. These statements, supplemented by the contextual information provided by small talk during the observed meetings, made it seem safe to assume that the majority of team members are located in the same city, or at least close enough to it so that daily commuting is possible. Out of the 5 team members who were interviewed, all 5 of them reported that all of the team members had been working remotely from home since the onset of the COVID-19 pandemic, and that they did not have any meetings that the team attended physically since then. However, 4 informants reported that those who had functional roles had attended occasional meetings at their office. Two informants reported that the team had been offered to physically meet for one occasion, but both of the informants had chosen to stay at home. One of these informants said that this meeting was in a social non-work related context. One informant who joined the team in June 2020 said that they had only ever met one of the other team members in-person since they joined the team, and that they had only met the rest of the team members through digital communication. To summarize, a total of 3 out of 5 informants explicitly reported not having met the rest of the team in-person since the lockdown was initiated on march 13th 2020, while all 5 of them reported having worked from home during that time period.

All of the informants said that the team meetings take place using group video calls. These calls provided a high degree of communication synchronicity during the meetings held by the team. 4 of the informants expressed dissatisfaction with the conversation flow of the video meetings, in that small delays made it difficult not to talk over one another. We observed three meetings held by using group video calls, and observed highly synchronous conversation take place, although every once in a while one person would talk over another by accident. During the interviews the informants were asked how they communicate with each other outside of the team meetings. Every one of the 5 interviewed informants said that when they initiate communication with another team member, they usually send them a text-based chat message containing simple correspondences. These chat messages were sent using Microsoft Teams. If the receiver of the message were online on Microsoft Teams they would immediately get a notification through the application that someone had sent them a message, and they were free to respond at their convenience. All 5 informants also said that if discussions required more complex forms of communication, then the communicating team members would initiate an audio or video call, yielding high degrees of communication synchronicity for the duration of the call. To summarize, the actual video and audio calls between the entire team or individual team members had a high degree of synchronicity. Chat messages had a lower degree of synchronicity, and so initiating contact would usually not happen instantly.

1 informant mentioned during their interview that their organization had recently imposed a 7,5 hour work day limit on their employees, and so the team members' work days were mostly in sync. The team's Scrum master was asked to elaborate on this during the feedback meeting, and they said that due to economic reasons, the employees were asked not to work longer than 7,5 hours each day until the end of 2020. These restrictions seemed to be lifted at the turn of a new year, however. Nonetheless, the employees had defined times at which they were expected to be at work, directly translated from Norwegian to mean "core time". These were set hours in which the employees were expected to be present and available. Although employees were allowed to sometimes deviate from these times when planned in advance, the team members appeared to work mostly simultaneously during these set hours.

Every one of the informants who were interviewed said that they were either an employee of the team's employing organization, or they were a consultant from a Norwegian consultant company. None of the emerged information seemed to indicate any form of off-shoring, and none of the interviewed or observed team members appeared to be citizens of a country other than Norway. The team worked together and communicated in Norwegian, and it appeared that every team member knew the language.

4.4.2 Use of technology

The second group of dimensions of team virtualness, as described in the conceptual framework (see Chapter 3.1), were the dimensions related to the use of technology, including electronic dependence, level of technology support, use of virtual communication tools, amount of informational value provided by those tools, and the use of computer-mediated communication. The informants talked about all of these dimensions to some degree, including statements about which technologies were used to which purpose. Additionally, statements were made by all the informants regarding their own satisfaction with the team's use of technology, and those statements were also included and presented in this subsection.

Throughout the interview the informants were asked questions asking how the team hold meetings, how they communicate with each other, and how they document their work. Through their answers the informants all mentioned an array of technologies that are used for team communication and collaboration, including Microsoft Teams, Miro, Jira and a Wiki. The results are summarized in Table 11.

Meetings	**Other interactions**	**Documentation**
Teams - Video calls	Teams - Chat	Jira
Miro - Boards	Teams - Video calls	Wiki
	Teams - Channels	Miro - Boards

Table 11: Technologies used for team communication and collaboration.

All 5 informants reported that their meetings take place through video calls facilitated by Microsoft Teams, and one informant said that Microsoft Teams are the standard communication tool used by their organization. Furthermore, 3 of the informants also mentioned using Miro during meetings as a substitute to a traditional blackboard. According to the informants explanations and our observations, Miro is a digital tool where participants are able to create notes and figures which they are able to display on a board that may be divided in multiple sections. It is mainly used for planning-meetings as well as retrospective meetings, but all 3 of the informants mention using Miro for workshops as well. All 3 of these informants said that they are satisfied with Miro as a tool.

4 out of 5 informants experienced a loss of natural conversation flow during video calls facilitated by Microsoft Teams as opposed to in-person meetings, and explained that this is due to the tendency for multiple people to speak simultaneously. 2 of these informants also expressed an increased communication difficulty due to the loss of natural body language and mimic, as one would lose

the ability to maintain eye contact with the other attendants. Despite these difficulties, 4 out of 5 informants, including 3 of the informants who also reported experiencing difficulties with video calls, still reported that Microsoft Teams worked well as a substitute for in-person meetings, and that they had no major difficulties using it. 1 informant claimed that they preferred virtual meetings using Teams and Miro above in-person meetings, as they found it easier to maintain focus throughout the meeting. These results are shown in Figure 10.

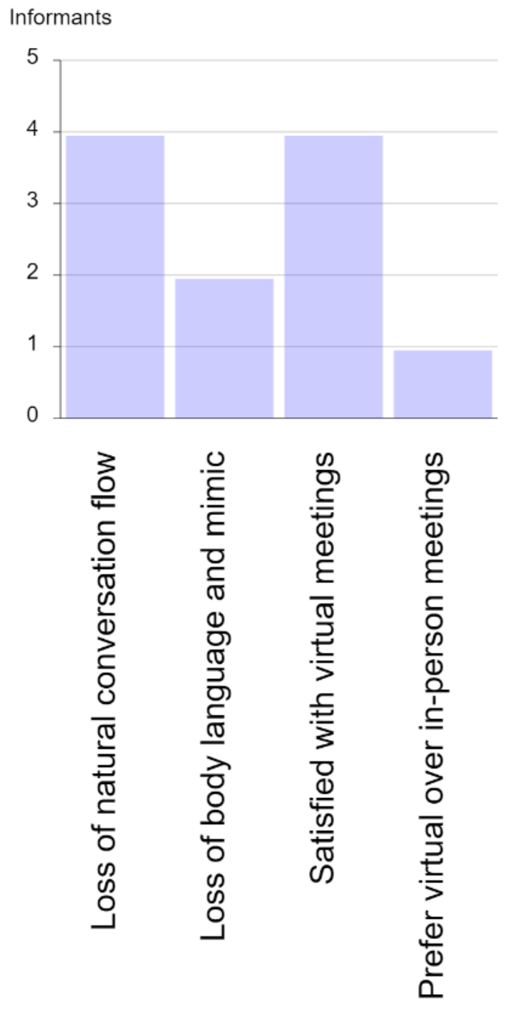


Figure 10: Satisfaction with use of technology during meetings.

The informants were asked how they interact with other team members outside of team meetings. All 5 of the informants replied that direct contact with another person is usually initiated through a text-based chat on Teams, and when discussions become more complex the conversation proceeds to become a video call. Furthermore, 2 informants mentioned the existence of several dedicated Teams channels for different subjects, which the entire team had access to. These subjects include production deliveries, code-reviews and testing. According to 1 of these informants, some of the Teams channels are also open to other teams and individuals within the organization, for example channels for the organization's various domains. Those channels were often used for inter-team collaboration.

When asked whether working virtually had an effect on their threshold for initiating contact with other team members, all 5 informants reported that they did not experience a higher threshold for initiating contact in a virtual setting. 3 informants actually reported experiencing a lower threshold for initiating contact with other team members in a virtual setting, as opposed to approaching them in-person in a co-located office, especially if they wanted to approach someone from another team, or someone whom they did not know. 2 of them expressed that the use of Microsoft Teams has decreased their own fear of disturbing their colleagues when reaching out, in that they were able to send a text-based message which the other person were free to reply to whenever they were available. The third informant appreciated the discretion of video calls between two parties, explaining that in an office-landscape everyone would be able to see if someone approaches another person to ask for help. 1 informant expressed frustration with the loss of ability to see whether the other person is actually present or away from their work station, as Teams lists a person as "available" until they've been away from their computer for a given amount of time. In spite of this frustration, this individual still reported that they did not experience a higher threshold for initiating contact. These results are shown in Figure 11.

When asked how the team documents their work, 4 out of 5 said that they use Jira as their main tool for documentation. The informants explained that Jira is a tool that maintains a task backlog where each individual task goes through a workflow where it is first created, next it goes through funcitonal analysis where requirements are specified. Then the task receives a priority as well as an estimation for work time required to completion, before it is placed in an iteration. Once placed in an iteration, the task goes through another workflow in which it is placed in a queue from which the developers may choose which task they wish to work on. Lastly, when the task has been completed, the result is deployed to production, and the Jira ticket is closed. We simply used the word "task" when describing the workflow, but in reality, a Jira ticket may represent a problem that needs to be solved, decisions that needs to be made, as well as questions that needs to be answered. Each ticket may include a description of its content, and users may write comments within the ticket in order to discuss the issue or provide help to the person who is currently working on it. Furthermore, each ticket has its own status describing where it currently is in the workflow. 3 informants also mentioned using their organization's own designated Wiki-page for storing documentation. During the retrospective meetings the team uses a Miro board in order to create items of action, and both the Miro board as well

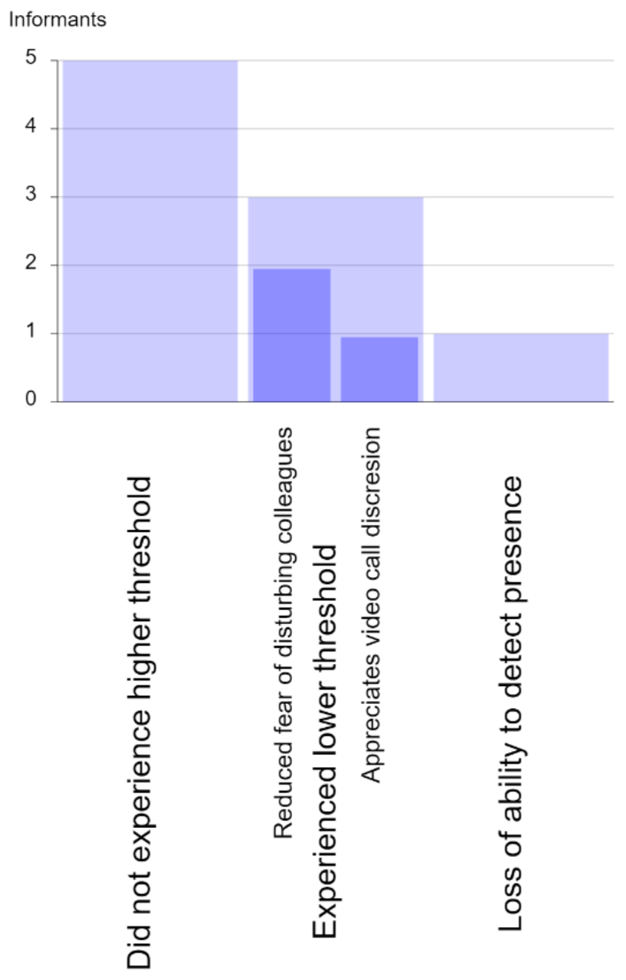


Figure 11: Experienced threshold for initiating contact with other team members.

as these items of action are stored in a Wiki page by filling in a template. 1 informant mentioned that the sprint planning meetings are documented in the Wiki in a similar manner, and 1 other informant mentioned that validations are documented in the Wiki. However, neither of these 2 informants provided further explanation of details.

5 Results

During the data analysis process described in Chapter 3.5, 5 interview transcriptions along with 3 written summaries of observed team meetings were coded with respect to the conceptual framework described in Chapter 3.1. From the analysis emerged several types of data. Firstly, a set of general statements emerged from the analysis of the interview transcripts, along with a number representing how many of the 5 informants expressed a statement of similar nature. These statements are visualized in bar graphs in this chapter, where the length of the bar represents how many informants expressed agreement with each statement. Moreover, some of the graph bars contain even smaller bars inside them. They represent how many informants not only expressed a statement similar to that of the larger outer bar, but also expressed a related and more specific statement, and is written with a smaller font and displayed with a stronger colour. From the written observation summaries, analysis procured descriptions of a set of observed events, grouped by topic. Some of these events described the protocol of and tools and practices used during the observed meetings. These event descriptions were used not only to outline a narrative of how these meetings are performed, but also to identify the use of technology as well as the use of Agile practices that are associated with the sharing of team knowledge. A different grouping described events in which the sharing of team knowledge was demonstrated either through socialization or through externalization and internalization. Lastly, there were groupings of event descriptions according to the different types of team knowledge as described in the conceptual framework (see Chapter 3.1), depending on which type of team knowledge was being shared.

This chapter is divided into two sections, each of which presents the results relevant for answering one of the research questions. Section 1 presents our findings regarding the team's team-knowledge sharing activities, including both mentions of activities by the informants as well as observed demonstrations of team-knowledge sharing. In Section 2 we present the quantified measure of shared team knowledge between the 5 interviewed informants.

5.1 Team-knowledge sharing activities

Several different approaches were used to construct a picture of which knowledge-sharing activities are used by the team, and data was used both from interviews, observations and the feedback meeting with the team's Scrum-master. In Chapter 3.1 we established three concepts which would serve to identify the sharing of knowledge. First, any use of Agile methodologies associated with increased team knowledge sharing such as Scrum or Extreme Programming, especially practices such as Scrum roles, daily stand-up, sprint and release planning, sprint retrospective, cross-functional teams, working in iterations and pair programming, were considered to be knowledge sharing activities. Furthermore, any mentioned or observed activities that involved the movement of knowledge, either through socialization or by externalization and internalization, were also included. During the interviews a range of questions were asked regarding the team's way of working, how they communicate and how they perform documentation. Due to the loose structure of interviews and open-ended nature of questions we received a wide array of responses, where some activities were mentioned more often than others. While the majority of identified activities were procured from the interviews, in-depth information about the various activities were retrieved both from interviews, observations and the feedback meeting. The full list of identified knowledge sharing activities are summarized in Table 12.

5.1.1 Agile practices

All 5 of the informants responded that their team is cross-functional, works iteratively, and that the team works using the Scrum methodology. 3 informants said that their sprints each last for three weeks. 4 informants mentioned the use of Scrum roles, meaning that the team has one designated Scrum master, as well as one team member who holds the role of product owner, in addition to the other team members of various cross-functional roles. However, from our feedback meeting we understood that the role of Scrum-master also includes responsibilities that are more traditionally labeled as "team-lead" responsibilities. See Chapter 4 for more details. According to all 5 informants, the team holds a stand-up meeting every day in which every team member provides a status of their work, and notifies the team of any problems they are facing. Although the stand-up meetings are not used for problem-solving, these meetings allow the team members to be aware of each other's issues, and to approach each other at a different time to help each other resolve their problems. All informants mentioned having sprint planning meetings at the beginning of each sprint, and 4 informants responded that they also do sprint retrospectives. These three meetings were observed, and more closely described in Chapter 4. 1 informant also said that the team also has processes in place for new features that could remind of release planning, although they did not use that name and they did not go in to much detail of the nature of these processes other than that the team has meetings in order to plan which overarching tasks should be done in order to complete the release, as well as establish requirements.

4 informants responded that they use task estimation, in which tasks are given a number representing how many work weeks are required for one developer to complete the task. 3 informants said that somewhat regularly they hold estimation meetings. 1 information said that at the beginning of a new project the team receives a package of Jira items, which will be roughly estimated at the beginning of the project. However, throughout the sprints the team will receive new Jira items, many of which lack an estimate, and so the estimation meetings serve to provide an estimation for those tasks. From the observations we learned that during the sprint planning meeting the team would get an overview of tasks, and if many of their tasks lacked an estimate they would schedule a meeting shortly after in order to estimate the tasks. These meetings, however, would only be held if there was a need for it. 1 informant responded that the estimations are based on the team members' own experiences, and 1 informant also said that during these meetings, the team members share their

Agile practices	**Externalization**	**Externalization and internalization**	**Socialization**
Cross-functional teams	Document task workflow in Jira	Training new team members	1-on-1 calls for clarification
Working in iterations	Document decisions and functional clarifications in Jira	KOMPIS-lecture	Daily coffee breaks
Sprint and release planning	Document meeting attendants in Wiki	Architecture meetings	Meeting discussions
Task estimation	Document validations in Wiki	Create retrospective points of action	Discuss retrospective points of action
Prioritized tasks	Document retrospective points of action in Wiki	Using MIRO boards for workshops	
Scrum-roles	Document meeting Miro board in Wiki	”Behovsfase”	
Sprint retrospective		Clarification meetings	
Daily stand-up meetings		Discussions in Jira	
		Ask for code-reviews in Teams channel	
		Functional clarifications in Teams channel	

Table 12: Summary of all team-knowledge sharing activities mentioned during interviews.

content-related knowledge by providing an explanation of why they provide a given estimation on a task. The result of these meeting are increased sharing of task-related team knowledge.

All 5 informants mentioned the use of a prioritized queue for tasks, facilitated by Jira. 2 of the informants mentioned that the product owner is at least partly responsible for setting the task priorities. By combining what we learned during the observations along with various statements from the interviewed informants, we learned that there is one Jira item per task, and the Jira item contains a priority of either 1000, 2000 or 3000, where a higher number means a higher priority. When the developers choose a new task to start on, they are presented their task backlog in which the Jira items are sorted by priority, and they are inclined to choose from the tasks with highest priority.

5.1.2 Activities mentioned during interviews

Aside from the Agile practices, the interviewed informants mentioned an array of other activities that involved the sharing of team knowledge. These involved activities of externalization such as various types of documentation, activities of both externalization and internalization such as lectures and training, as well as activities of socialization.

1 informant said that the team sometimes hold something called "kompis" (buddy) meetings, in which one team member holds a lecture about a specific topic, and the other team members are invited to listen. The topics lectured may be both technical but also non-technical, for example about the organization's domain. 2 informants mentioned the "behovsfase" meetings in which the team discuss their tasks' requirements, and 1 informant said that they found them to be useful because through these meetings they gain knowledge of why a task is to be solved a certain way. Furthermore, 1 informant said that team members sometimes hold meetings in which architecture is explained to the other members of the team, without going much into detail about these meetings. 2 informants said that when required, the team will hold clarification meetings about specific topics, but did not elaborate on details. 1 informant said that new members require some amount of training when first joining the organization and the team. They did not specify the nature of this training, however. 3 informants said that the team holds daily coffee breaks in which the team members are allowed to socialize, although it is unclear whether any team knowledge is shared during these breaks as they are not focused on work and hold no structure. It is included, however, as socialization in general tends to facilitate the sharing of tacit knowledge.

4 informants said that they externalize task-related team knowledge such as task and problem descriptions by using Jira. 2 informants elaborated on how exactly Jira was used, and their combined explanation said that Jira is a tool that systematizes task knowledge. Through Jira, the team can share information about specific tasks, such as a problem description, requirements, suggestions to solutions, priority in the task queue, estimation of required work time, which team member created the task, which team member is working on

the task, as well as the task's status. Every team member has access to the Jira items, and so if one needs information about a specific task, they may look it up in Jira. If they have questions that the Jira ticket does not answer, then they can look up who created the task and ask them directly for clarification. When a team member starts working on a task they will label the Jira item as "In progress", while marking themselves as the person working on that item. Every sprint the team will be working on a group of Jira items, and these items are presented to the team members as a queue sorted by task priority. 2 informants said that whenever functional clarifications or decisions are made regarding a task, the team tries to document it within the Jira item, although they evaluate whether they find it necessary for every case.

Another knowledge management system used by the team are the organization's Wiki pages. 3 informants said that they use a Wiki page in order to document information about sprint planning meetings and retrospective meetings, such as who attended them, what was discussed and which points of action has been decided upon. 1 informant said that the Wiki page contains templates for the sprint retrospective and sprint planning meetings, and that the team are supposed to fill this templates during those meetings and store them in the Wiki. For both of these meetings the team also make a screenshot of the resulting Miro board and upload it. In the retrospective, the selected points of action are also documented here, along with the person responsible for each point. 1 informant said that the team also document validations in Wiki, without elaborating what these validations do.

1 informant said that the team has a dedicated Teams channel for code reviews and functional clarifications. In this channel, team members can notify the rest of the team that they would like a code-review on their work, and other team members can respond to let them know they will do it. The channel is also used for functional clarifications, in that a team member can notify that a task needs functional clarification, and other team members may respond by clarifying.

3 informants said that during various workshops they use Miro as a tool. This allows all of the team members to express themselves in a systematized manner. During the sprint retrospective, the team uses Miro board to express their own feelings about the previous sprint through notes. During the sprint planning meeting, the Miro board contains a board containing a single note representing each Jira item the team would be working on during the coming sprint.

All of the informants said that they find it easy to approach others on the team if they need help with something, and said that they regularly contact each other to ask for help, usually through a text chat which often proceeds to become a voice or video call. In other words, all 5 informants indicated that 1-on-1 sessions of knowledge sharing are common.

5.1.3 Activities observed during sprint retrospective

We observed a sprint retrospective meeting with the team, during which we noted any events that involved the sharing of team-knowledge between the team members. The meeting seemed to be a facilitator of knowledge sharing through discussion, especially focused around problems encountered by the team, solutions to these problems, as well as the team's norms. The findings, which are summarized in Table 13, were grouped by mode of conversion (i.e. socialization, externalization, and externalization and internalization).

Socialization	**Externalization**	**Externalization and internalization**
Discuss positives with previous sprint	Expressing positives and negatives on Miro board	Explaining notes on Miro board
Discuss negatives with previous sprint	Expressing points of action on Miro board	Explaining technical issues to teammates
Discuss solutions to problems	Document points of action in Wiki	
Discuss team norms	Document Miro board in Wiki	
Discuss "health check" points	Document meeting attendants in Wiki	
Remind team of meeting scope	Document "health check"	

Table 13: Summary of all team-knowledge sharing activities observed during retrospective.

At the beginning of the meeting, and empty Miro board was opened with three columns named "What went well", "What didn't go well", and "Points of action", and the team members all started filling the two first columns with notes. Each note represented an externalized expression of a positive or negative aspect with the previous sprint. When the team was finished writing notes, they went through each note together in a manner where the author of the note often explained what they tried to expressed through the note, and the team would discuss the issue and possible solutions to problems. During these discussions it sometimes occurred that team norms would be questioned or new norms would be suggested, and the team would discuss it further. Sometimes technical issues would be brought up, and a team member would have to explain its details to the rest of the team.

Towards the end of the meeting, as the team finished discussing the notes, a Wiki page was brought up in which they documented who attended the meeting as well as the selected points of action along with their responsible team members. Then a screenshot was taken of the Miro board, and this was uploaded to the Wiki as well. Lastly, the team brought up a "health check" website, which contained a set of points that were either green or yellow. The team discussed whether any of these should change color, before they did some small talk and concluded the meeting.

5.1.4 Activities observed during sprint planning

As we observed a sprint-planning meeting with the team, we noted every observed event that included the sharing of team knowledge through socialization, externalization, or both externalization and internalization. The results are summarized in Table 14.

The objective of the sprint planning was to organize a set of Jira items in order to decide which tasks would be worked on during the sprint. As the meeting started, the Scrum-master brought up a Miro board in which a set of Jira items were presented through notes on the board. The board was divided into four columns, named "Open / In analysis", "Ready for execution", "In progress" and "Awaiting other issues", and each of these columns were filled with a set of Jira items representing pending tasks. Which column a task was placed in represented its status within the task workflow. Furthermore, the tasks were colour coded by priority and whether the item needs estimation. They were also tagged with the name of the person responsible for it. Moreover, written notes could be added to each task item on the board throughout the meeting, and seemed to serve as a guide to summarizing the meeting at the end.

Before starting, the Scrum-master brought up the Wiki pages in order to fill in the meeting attendants. The Wiki page also contained a number representing the sprint capacity. Then the team members were asked whether anyone would like to hold a KOMPIS lecture during the sprint, which the team discussed for a short while and a few suggestions were brought up. The KOMPIS lecture was part of the Wiki page's template, in which they would document any upcoming lectures. Although the team decided not to hold any lectures this sprint, it still facilitated the sharing of team knowledge in the form of expertise location through socialization. The team would also be notified who is Sheriff during the coming sprint.

The Scrum-master proceeded to lead the meeting in a manner in which the team collectively went through each Jira item on the board. Usually, one or two people would start by explaining the Jira item by sharing their task-content knowledge. They would explain the problem, and sometimes also suggest strategies to solve those problem. Some tasks were divided into sub-tasks, and a team member would have to clarify exactly which sub-task the team was responsible for doing. Some tasks were dependent on other tasks, and a team member would have to clarify this as well. Some of the tasks were already being worked on by the team, and the responsible team member would have to elaborate on

Socialization	**Externalization**	**Externalization and internalization**
Discuss potential KOMPIS lectures	Organize Jira items on Miro board by workflow status	Explain Jira item (task content)
Discuss task content	Color coding Miro task notes by priority	Explain potential task strategy
Discuss task progress	Add meeting notes to Miro board	Explain task dependencies
Discuss task estimation	Comment Jira item with task strategies	Explain task's sub-tasks
Discuss task strategies	Set weight on Jira item	
Schedule meetings	Document potential KOMPIS lectures in Wiki	
Notify who is Sheriff	Document meeting attendants in Wiki	
Discuss which tasks to include in sprint	Document sprint capacity in Wiki	
Discuss whether an estimation meeting is required		
Remind team of meeting scope		

Table 14: Summary of all team-knowledge sharing activities observed during sprint planning.

Socialization	**Externalization**	**Externalization and internalization**
Discuss problems		Explain previous task
Discuss problem strategies		Explain current task
Discuss task progress		Explain encountered problems
Discuss daily activities		Suggest problem strategies
Schedule time to collaborate		Express need for assistance
Remind team of meeting scope		Express need for clarification

Table 15: Summary of all team-knowledge sharing activities observed during stand-up.

their progress. A subset of the tasks were straightforward to explain, while others prompted more in-depth discussion within the team, such as what a task actually entails, which strategies are best, and which estimation weight a task should have. These discussions could lead to changes in the Jira item, such as commenting task strategies or set a new estimated weight. Throughout the meeting the team would get a common overview of which tasks should in fact be worked on during the coming sprint, and which should wait. They would also discuss whether an estimation meeting is required, based on how many of the tasks on the board lack an estimate. In the end they would summarize the meeting, and schedule new meetings if they were required.

5.1.5 Activities observed during stand-up

As we observed a sprint-planning meeting with the team, we noted every observed event that included the sharing of team knowledge through socialization, externalization, or both externalization and internalization. The results are summarized in Table 15.

No additional tools were used other than Microsoft Teams in order to facilitate the communication, and we observed no acts of pure externalization of team knowledge. The meeting was led by the Scrum-master, who asked the team members, one at a time, what they had been doing since the last stand-up meeting, what they would be working on today, and whether they had encountered any problems. In some cases the Scrum-master also asked the team member about their progress of their current task, if they had been working on the same task for a while. Every team member systematically explained what they had been working on since last time and what they were working on now. If they had encountered any problems they would briefly elaborate on this, and express the need for assistance or clarification from other team members when needed. In some cases one or two people provided the clarification or suggestions for problem strategies to the person in need. In other cases a discussion was prompted within the team about the problem and possible solution strategies. If the discussions grew complex, the involved parties would schedule time to collaborate to solve the problem, and carry on with the meeting.

5.2 Shared Team Knowledge

The second research question, **R2:** "How much overlap is there between the individual team members' team knowledge?" was dependent on two pieces of information: 1) which types of team knowledge should be measured, and 2) which metrics should be used to measure it. The framework by Faegri et al. (2016) was used to determine which types of team knowledge should be measured, and these types are summarized in Table 3 of Chapter 3.1. To measure the overlap of the team's held team knowledge, *similarity of team knowledge* [13] was used as a metric to quantify the team's shared team knowledge. Interview transcripts were analyzed as described in Chapter 3.5.6, in which specific statements made by the informants displaying any of the selected knowledge types were compared, and statements that were similar were combined into more general statements. The result was a set of general statements related to each of the primary knowledge types described in Chapter 3.1. Attached to each of these statements were a number representing the number of informants who expressed a statement of similar nature. These statements and their number of agreeing informants will be represented in this chapter, grouped by the team-knowledge category under which each statement fall (i.e. goal-related, team-related, process-related and task-related). Each set of related statements will be presented in a bar graph, visualizing how many informants expressed each general statement. Furthermore, if a statement was made by a set of informants, and a subset of those informants express a more specific statement about the same subject, then it will be displayed as a bar graph of stronger colour and smaller font within the original bar graph.

5.2.1 Goal-related team knowledge

When asked about the team's overarching goals, 4 informants said that they have a goal related to development. Out of these 4 informants, 3 of them specified that their overarching goal is to develop tools used by the organization's internal case managers, and 2 of them said that their goal is the development of any currently ongoing project. We grouped these two statements within one that we named "Development" because while talking to one of the informants we learned that their currently ongoing project is in fact a tool they are developing for the case-managers. Furthermore, 4 informants mentioned maintenance as one of their overarching goals. 2 of these informants further specified the maintenance of their technical ground structure, or, their previously existing back-end services. The remaining 2 informants specified the maintenance of existing applications. One of these informants elaborated by describing an existing application that worked as a user interface for their customers. 3 informants said that one of the team's goal is to ensure that the customer and data delivered by the team's services is correct. While talking to 2 of these informants we learned that the team's domain require a high degree of data integrity, and that errors could cause problems for their customers and members. A single informant responded that one of their goals are to correct incoming errors, and 1

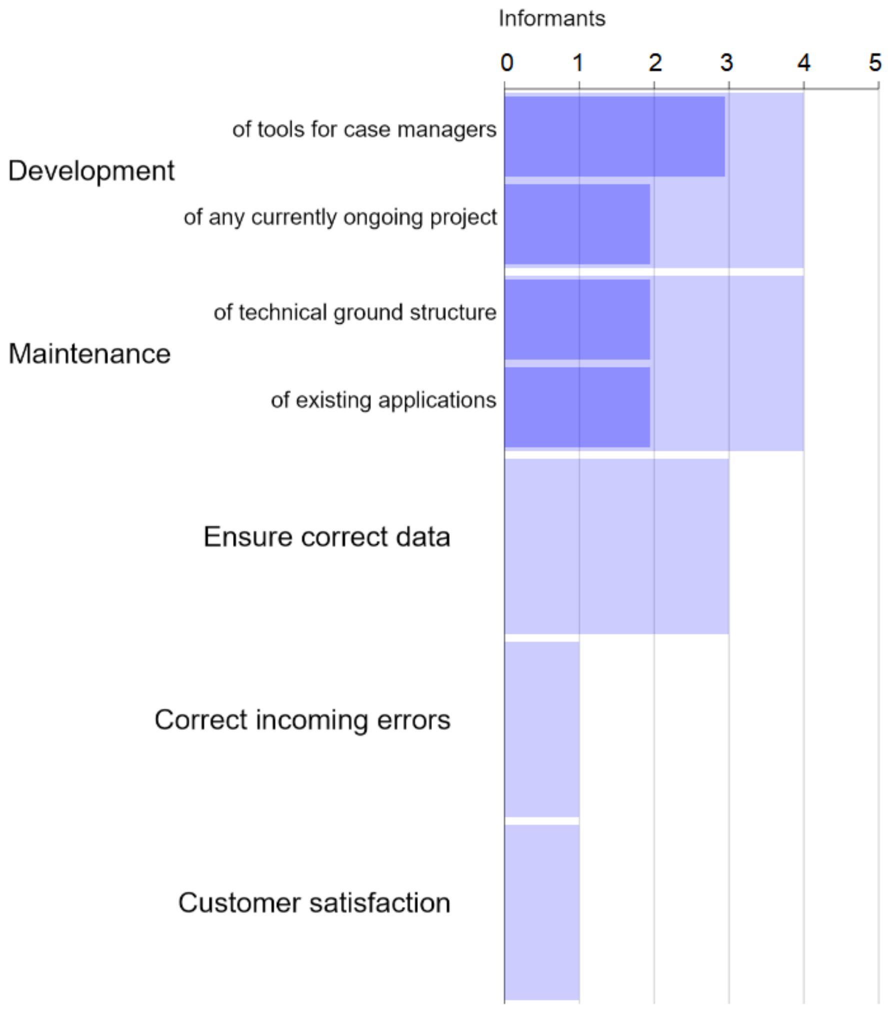


Figure 12: Overarching team goals according to the informants.

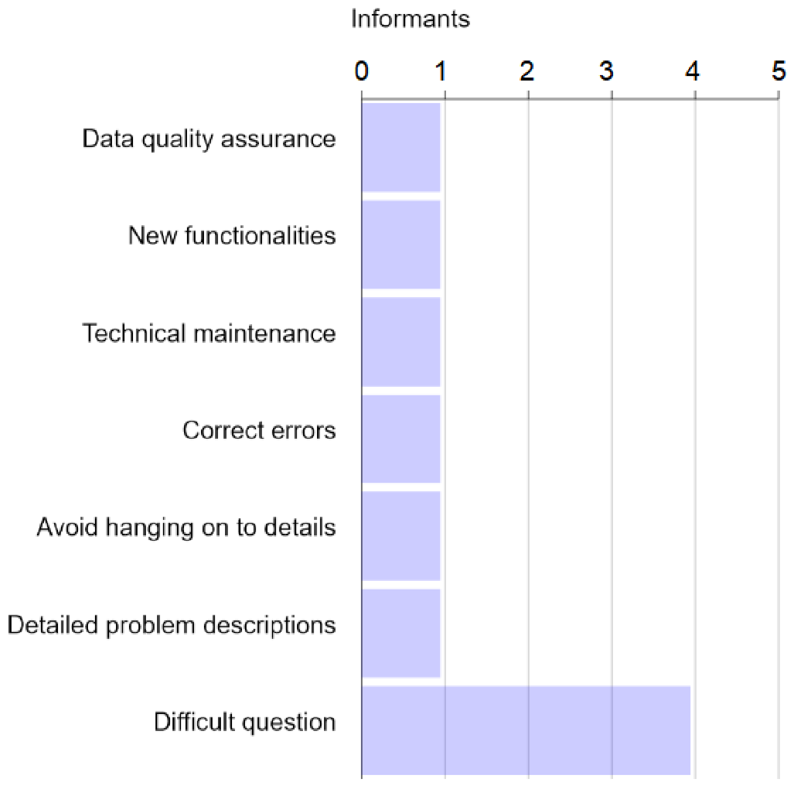


Figure 13: Strategic consensus according to the informants.

informant mentioned customer satisfaction as a goal. The informants' responses regarding the team's overarching goals are summarized in Figure 12.

When asked what is the strategy to reach the overarching goals, the informants' responses were scattered, and no two informants responded the same thing. The result were different responses, each mentioned by a single informant. These responses were to ensure the integrity and quality of the data managed by the team, to add new functionalities (without going into detail as of which functionalities should be added), to continuously perform technical maintenance, to correct existing errors, to avoid spending too much time and energy on details and focus on the big picture, and to provide detailed descriptions of problems and solutions. 4 out of 5 informants expressed that the question was difficult to answer. When we presented these results during the feedback meeting, we got the impression that these are questions that one usually do not think about, and therefore it is not common to explicitly share knowledge about it.

5.2.2 Team-related team knowledge

The informants were asked who is on the team, and which roles they have. For reference, the list of team members and roles provided to us by the team's scrum master lists the following team roles: 2 functionally responsible, 1 architect, 1 functional architect, 1 UX-designer, 1 product owner, 1 team lead, and 6 developers, where 1 of the developers had the role of Scrum master. Out of the 6 developers, 2 were back-end developers, 1 were front-end developer, and

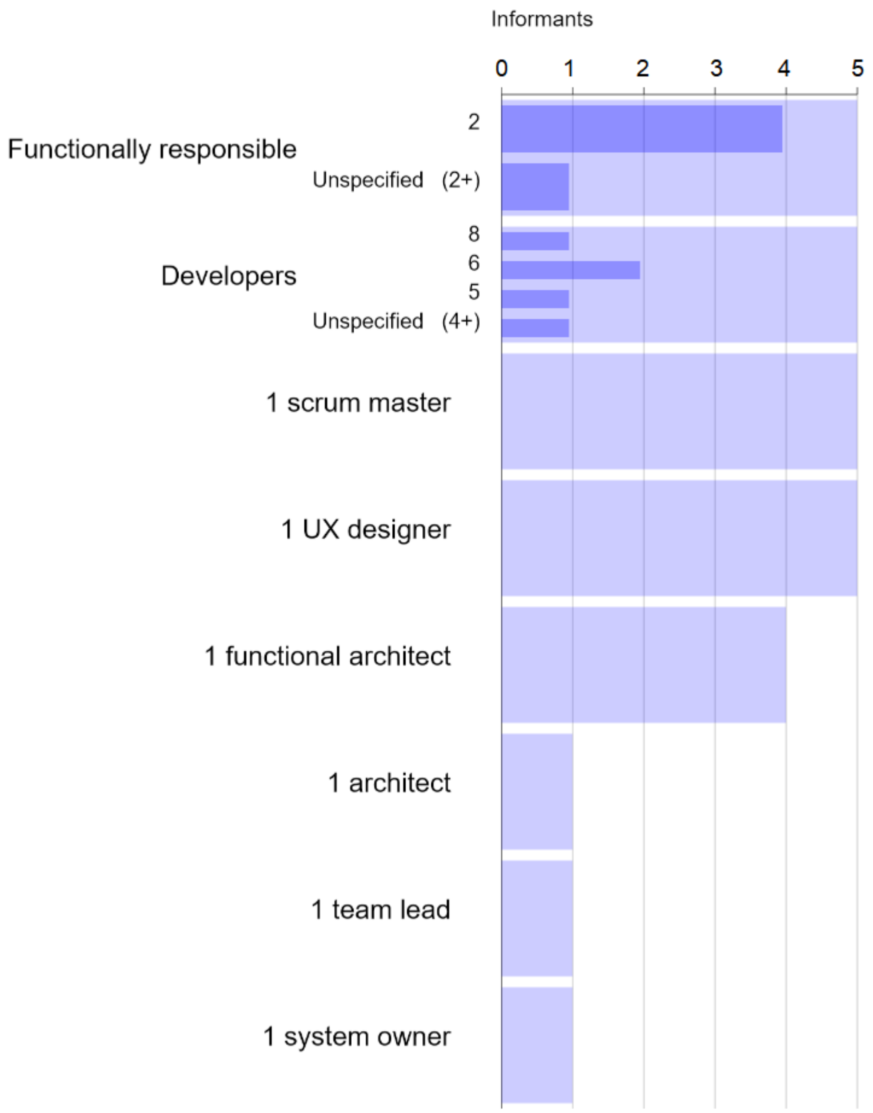


Figure 14: Amount of team members belonging to each role, according to the informants

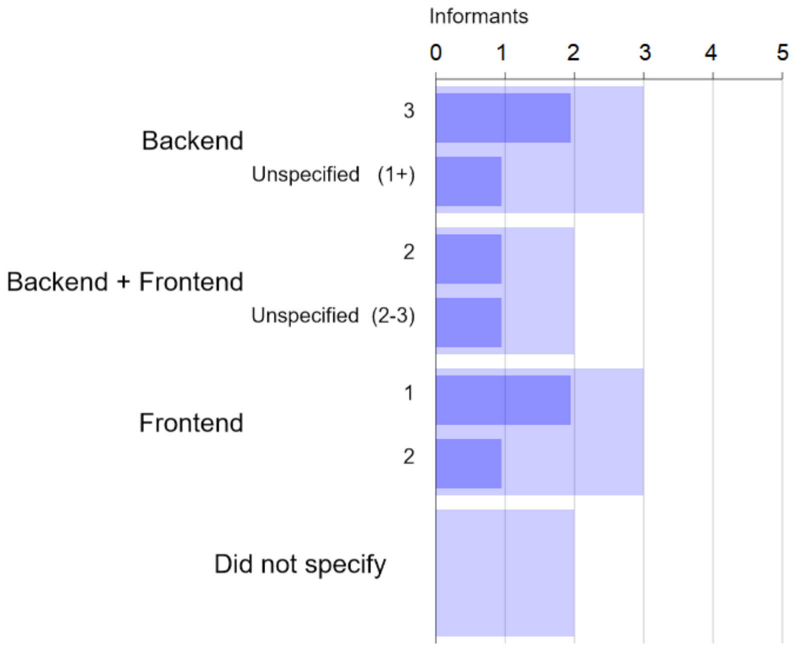


Figure 15: Amount of team members belonging to each developer sub-role, according to the informants

3 people worked on both back-end and front-end.

All 5 informants responded that the team has 1 Scrum master and 1 UX designer, and 4 informants responded that they have 1 functional architect. 4 out of 5 team members responded that they have exactly 2 functionally responsible, but 1 out of 5 did not specify any exact number, except that there at least 2. The informants responded somewhat differently on how many developers are on the team, where 2 responded six, 1 responded eight and 1 responded five. 1 of the respondents did not specify how many developers were on the team, but through their phrasing we could deduct that they were talking about at least 4 different people. It should be noted that it was the same individual who did not specify in both cases. Furthermore, 1 informant mentioned an architect, 1 informant mentioned a team lead and 1 informant mentioned a system owner. These results are summarized in Figure 14.

3 out of 5 informants also specified the roles of the developers when listing who is on the team. 2 informants said that they have three back-end developers, while 1 informant did not specify an exact number, but that they have someone who works back-end. 1 informant said that they have two developers who work both back-end and front-end, while 1 informant said that they have two or three developers who work both. 2 informants responded that they have one front-end developer, while 1 informant responded that they have two front-end developers. These results are summarized in Figure 15 Note that the informant who did not specify an exact number for the former two roles is the same informant who did not specify on how many functionally responsible and developers they have.

We presented these results during the feedback meeting and learned that after the initiation of the lockdown, several new members had joined the team. Many of the new members had non-developer roles, meaning that not only had they never met the rest of the team in-person, but neither did they work interdependently with all the team's developers. The Scrum master said this could be a possible explanation as to why there is a scatter on the responded number of developers on the team. We were also told that 1 of the developers worked purely front-end, while the rest of the developers work both front-end and back-end, but that many of them consider themselves as pure back-end developers because they don't like working on front-end.

When presenting these results during the feedback meeting, we learned several things. First we learned that after the initiation of the lockdown, several new members had joined the team. Many of the new members had non-developer roles, meaning that not only had they never met the rest of the team in-person, but neither did they work interdependently with all the team's developers. The Scrum master said this could be a possible explanation as to why there is a scatter on the responded number of developers on the team. Secondly, we learned that only 1 of the developers were considered to be purely front-end, while the rest are able to do both front-end and back-end work. Some of the developers, however, considered themselves to be purely back-end because they prefer back-end over front-end, but were able to take front-end tasks as well should the need arise. For this reason, the Scrum master themselves would place all the developers except for one into the "front-end + back-end" group. Furthermore, we learned that while there are technically two people listed as members of the team with the role of architect and team-lead, whether they could actually be defined as part of the team can be questioned. The architect were a new employee and were grouped with the team so that they would have a group of colleagues with which they could socialize at work. The team-lead, as described in Chapter 4, had more of a project manager role, and were not directly involved in the day-to-day work of the team. Lastly, we learned that the department has had many recent changes of product owners because there have been several different deliveries, and many domains have also changed product owners during an ongoing release as well. At one time during the observed sprint-planning someone asked whether a particular person were the system owner, and said that they change so frequently. Out of our three observed meetings, the system owner attended only one, which was the stand-up.

The informants were also asked whether they have a face and name connected to each role that they summarized, and all 5 responded yes. Furthermore, the informants were asked to name a field in which they themselves feel that they lack expertise, and later asked whether they knew someone on the team who holds such expertise. All 5 informants responded that they know of someone on the team that they can approach if they need help in that particular field.

5.2.3 Task-related team knowledge

When asked about which tasks the team is currently working on, all 5 informants responded that they are currently working on a project which we refer to as "Project P1", which is an application used by the organization's internal case managers. According to 3 of the informants, "Project P1" is a task in which

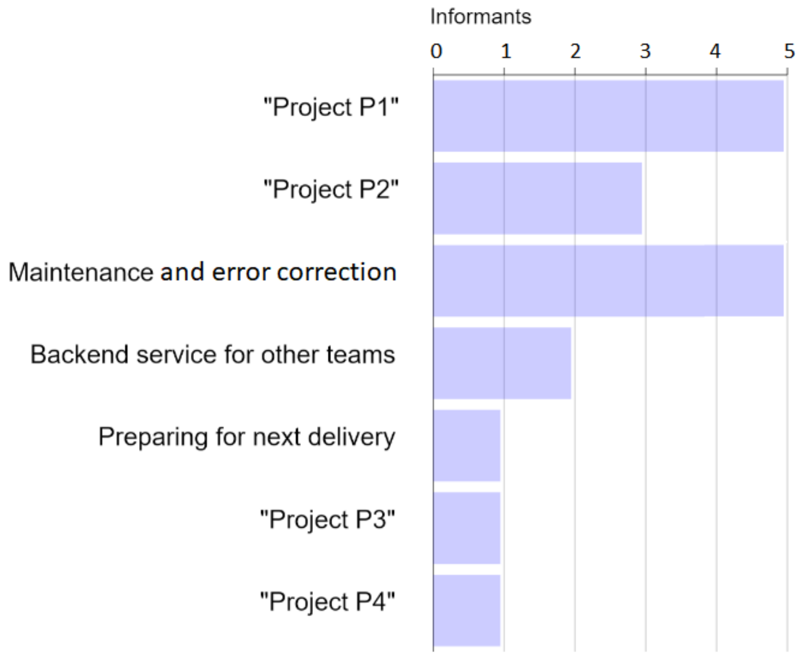


Figure 16: Task content: Which tasks the team is currently working on, according to the informants.

the team is re-implementing an existing application because the old one is built with Flash, which is being deprecated. 2 informants also explained that the Team is working on the front-end part of this application, while another team is providing the back-end. 3 out of 5 informants responded that the team is also working on a task which we refer to as "Project P2". 2 of these informants explained that this task involves logic around their customers rights to access data or their terms of agreement should there be any fusions or fissions between organizations. We found it relevant to add that the 3 informants who responded "Ensure correct data" were the same 3 informants who responded "Project P2" as one of the team's current tasks. All 5 informants also mentioned that the team is currently working on maintenance of existing applications, although only 1 informant specified what kind of applications are being maintained. This informant mentioned that existing Flash applications that will be replaced in the future still needs to be maintained for a while longer.

5 informants said that the team has tasks related to maintenance and error correction of existing services and applications. 1 informant specified by saying the team has tasks related to the maintenance of back-end services used by other teams. 2 informants responded that the team is developing back-end services, or end-points, to the front-end application developed by another team. 1 of these informants said that out of all the team members, there are two who are currently working on developing these endpoints. They both explained that the other team requires data on a specific format, and that their own team are responsible for developing end-points that provide that data. 1 informant re-

sponded that the team is currently preparing for starting their next delivery (i.e. release), and that many from the team are involved, including the functionals, architect, interaction designer and ”the whole gang”, as they said themselves.

1 informant said that one of the team’s members are working on a task which we refer to as ”Project P3”. They did not provide many details, except that there are continuously changes happening within that project, and so the team member in question are never out of work. 1 informant responded that the team is involved in a task referred to as ”Project P4” without going into detail as of what that task entails. The latter informant also expressed that they were uncertain of their team’s degree of involvement in said ”Project P4”. During the observed retrospective we noticed that one of the notes on the board of ”what went well” was a workshop that had the same name as ”Project P4”, but we learned nothing about its specifics other than that people were happy with its workshop. These informants’ responses regarding the team’s current tasks are summarized in Figure 16.

The informants were asked how their current tasks affect the environment around them, such as end-users and other teams. 4 informants responded that the ”Project P1” task directly affects the workflow of their end-users, which are the organizations internal case managers. The tool that the team is developing will allow the case managers to continue doing their work when their existing tool becomes deprecated. Out of these 3 informants, 1 specified that the re-implementation of their existing tool will bring improvements that will ease the workflow of the case managers; 1 informant specified that any unforeseen error may prevent the case-managers of performing their work; and 1 informant responded that the team working on the back-end for the ”Project P1” task are affected by the work done by their own team, in that their own team are dependent on the other team’s services in order to make their own deliveries, and so the other team is often pushed to prioritize maintaining those services.

2 informants explained that the work that the team does on the ”Project P2” task affects their customers’ workflow in that it determines what information each customer has access to about their members. The nature of the task is to provide each customer access to the member information which they are permitted to see, and restrict the customers from seeing member information that they’re not supposed to see. Earlier, when 1 of these informants were asked which tasks the team were working on, they provided an explanation of ”Project P2” in which they implied that the task affected the privacy of their members as well. However, since it was only implied and not a direct answer to the question asked, we chose not to include it in the results. Nonetheless we mention it as a side-note for transparency’s sake.

2 informants explained that the work they do on the back-end services used by other teams directly affect those other teams in that they are responsible for what data is provided to those teams. Although they did not explain what kind of data they serve to other teams, they both explained that some other teams are dependent on the back-end services that they provide and maintain. 2 informants explained that the maintenance work done by the team affects the quality of services that they provide, without specifying which services they

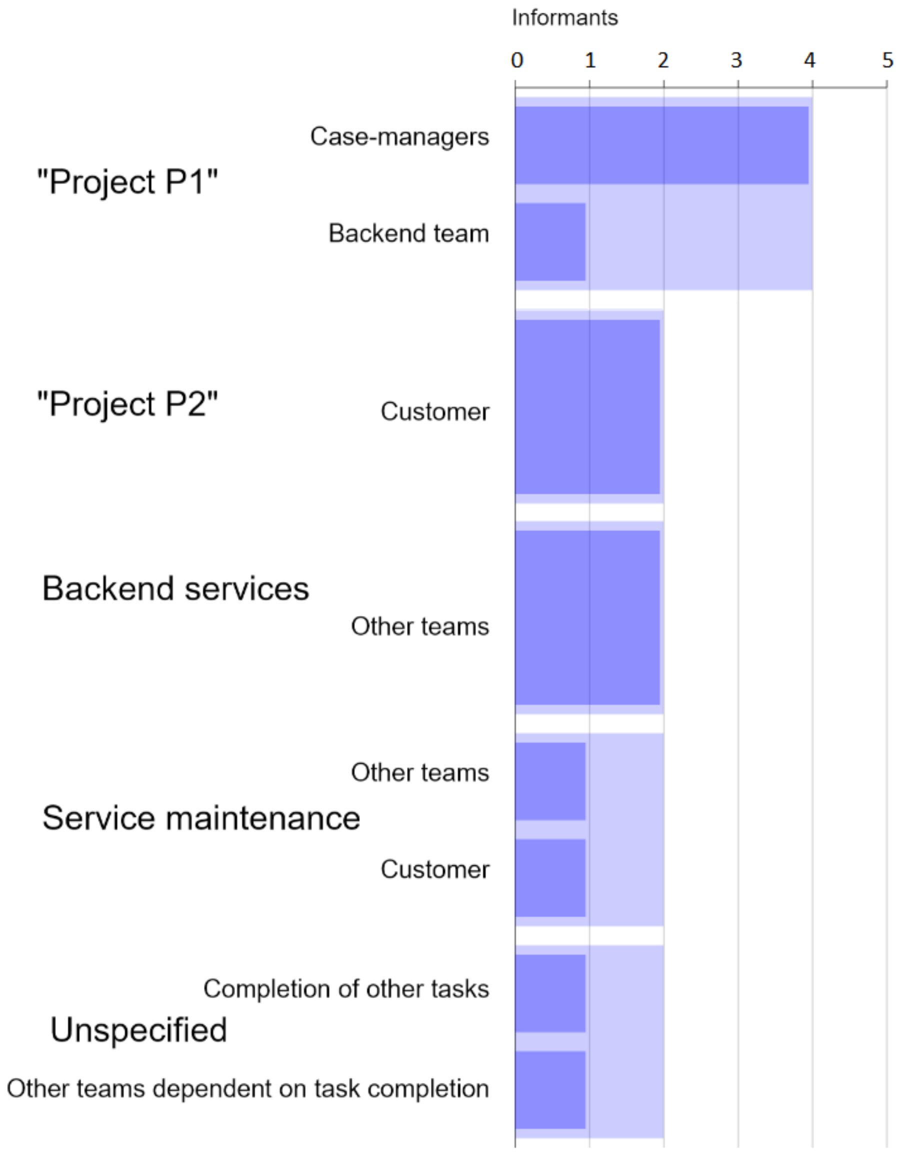


Figure 17: How the team's tasks affect the teams environment according to the informants.

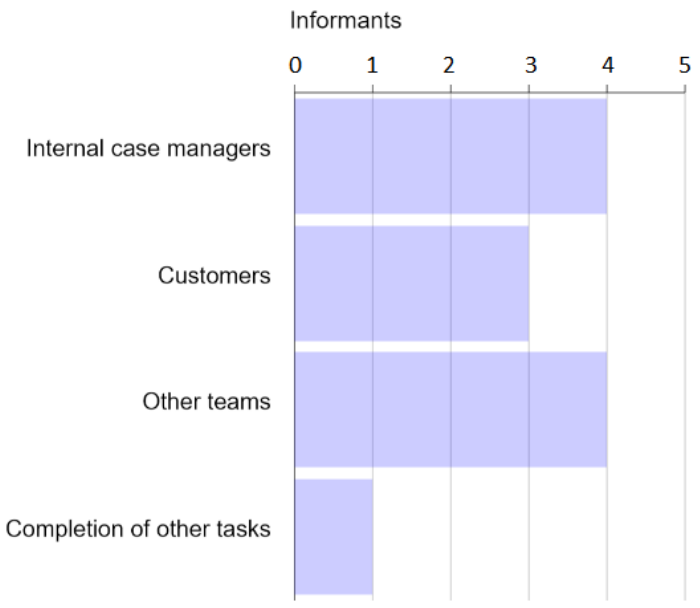


Figure 18: Which actors are affected by the team's current tasks, according to the informants.

were talking about. From these 2 informants, 1 informant were talking about services used by other teams, while 1 informant were talking about services used by their customers.

Lastly, 2 informants talked about how some of their tasks affect their environment without specifying which tasks they were talking about. Of these 2 informants, 1 informant said that the teams have tasks in which other teams are dependent on their completion, and 1 informant said that the completion of tasks directly affect the progress of the team's own project, in that failure to complete a task on time may further delay the completion of other project tasks. These results are summarized in Figure 17.

The informants' responses were additionally grouped together by which actors were said to be affected, regardless of which task they were affected by. This created a rough picture of which actors were affected by the total of work done by the team, according to the informants' responses. When grouping the responses this way, a total of 4 informants responded that their organization's internal case managers were affected by some task that the team is working on. 3 informants responded that their customers were affected, and 2 informants responded that their members were affected. 4 informants responded that other teams within the organization were affected, while 1 informant responded that the completion of the team's other tasks were affected. These results are shown in Figure 18.

The informants were asked whether there were any dependencies between their current tasks or their sub-tasks. 3 informants said that within the work that the team was currently doing on "Project P1", there were a few sub-tasks that were dependent on one anothers' delivery. 1 informant responded that there were no dependencies within the "Project P1" task, because the dependencies that they previously had were solved and delivered last week (note that all of the team members were interviewed during a 3-day span within the same week). 2 informants responded that within the "Project P2" task, there

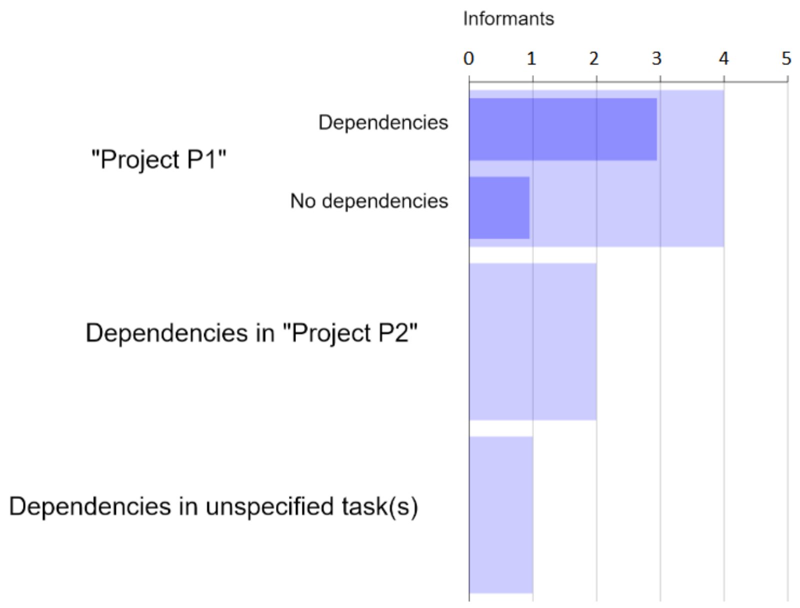


Figure 19: Task content: Which tasks have dependencies, according to the informants.

were currently sub-tasks that were related and dependent on one another. 1 informant explained that out of the work the team was currently doing, there were existing tasks or sub-tasks which were dependent on one another, but they did not specify which tasks were dependent. These results are shown in Figure 19.

When the researcher queried about the team's task strategies, he phrased the interview question as ”how will the team perform those tasks?”. All 5 informants responded by instead describing the team's task allocation strategies, and only 2 informants also described the team's task strategies. 1 of these informants responded that to perform the ”Project P1” task, the team would have to create new endpoints and try to keep the least amount of logic in the actual front-end application. 1 informant responded that with the ”Project P2” task, the team would have to attempt to reproduce errors within a test environment. 1 informant responded that with maintenance tasks, when their users report errors, the team had to identify the next plan of action by recognizing whether the error is previously known, and whether to create a new issue or simply add it to an error log and close it. The remaining 3 informants did not specify any task strategies. 3 out of 5 informants also responded that they found the question to be difficult. These results are shown in Figure 20.

All 5 informants provided some description of the team's task allocation

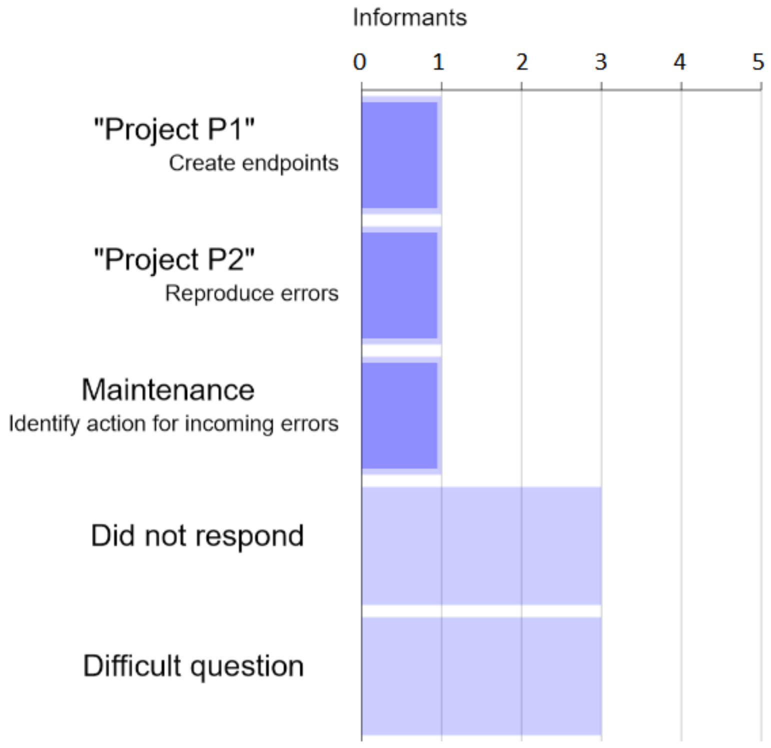


Figure 20: Task Strategies: What are the strategies for performing the team's current tasks, according to the informants.

strategies, and all 5 informants provided a description of how the team's developers, specifically, allocated tasks between them. All 5 informants explained that the first factor deciding who takes which task is a prioritized queue of tasks from which the developers pick one from the top. The prioritized task queue was the main factor, although not the only factor, deciding who works on which task. 4 informants responded that priority aside, which tasks a developer worked on was also decided by their skills, and if they did not possess the skills required to perform the highest prioritized task, then they were free to choose the next one in the queue. 3 informants responded that the developers also had a role division between them, deciding who works on which kinds of tasks. These 3 informants also provided examples of such role divisions, where 2 informants divided between dedicated front-end and back-end developers, while 1 informant divided by which applications or services they mostly worked on. Furthermore, 1 informant responded that when someone has worked on a task for a very long time, the team tries to alternate tasks so that the person doesn't grow tired of that task while also ensuring that more than a single person gains knowledge of that field. 1 informant said that they usually checked whether any of the other team members needed help with their task before starting on a new one. These results are shown in Figure 21.

2 of the informants were not developers, and 1 of them explained that there were dedicated tasks that they worked on, and that those tasks were naturally allocated to them if it fell within their area of responsibility. While the other

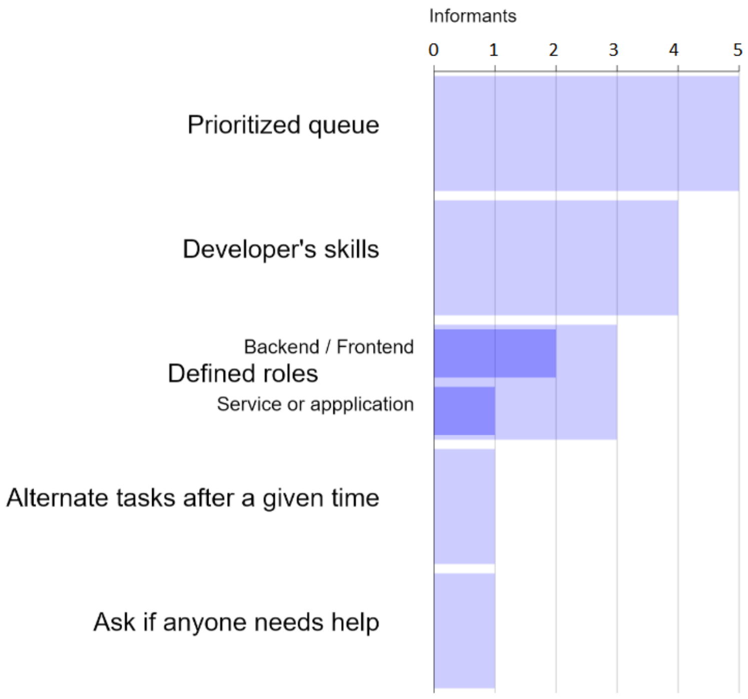


Figure 21: Task allocation strategies: How developers allocate tasks between them, according to the informants.

informant who were not a developer didn't explicitly say so when asked, it was heavily implied throughout the interview that the same applies to them, as they are the only person in the team with their given role, and their role clearly defines their area of responsibility.

5.2.4 Process-related team knowledge

The informants were asked to list which meetings the team held regularly, which meetings the team held irregularly or spontaneously, as well as any other interaction that happens between team members. The informants were also later asked to describe the team's protocol for meetings that they hold on a regular basis, however it should be noted that the interviewer accidentally skipped the latter question in one of the interviews.

When asked to list the team meetings that were held on a regular basis, all 5 informants mentioned the daily stand-up meeting. When later asked to describe the protocol of the stand-up meetings, 4 out of 4 responding informants said that every team member gives a short status of what they've been working on and let the team know if they have problems or need help. 2 informants said that the stand-up meeting is also used for small talk between team members, although they try to keep the duration of the meeting to a minimum. 1 of these informants explicitly said that the meeting typically lasts for 15 minutes, without being asked for duration.

All 5 informants also listed sprint-planning meetings as one of their regular

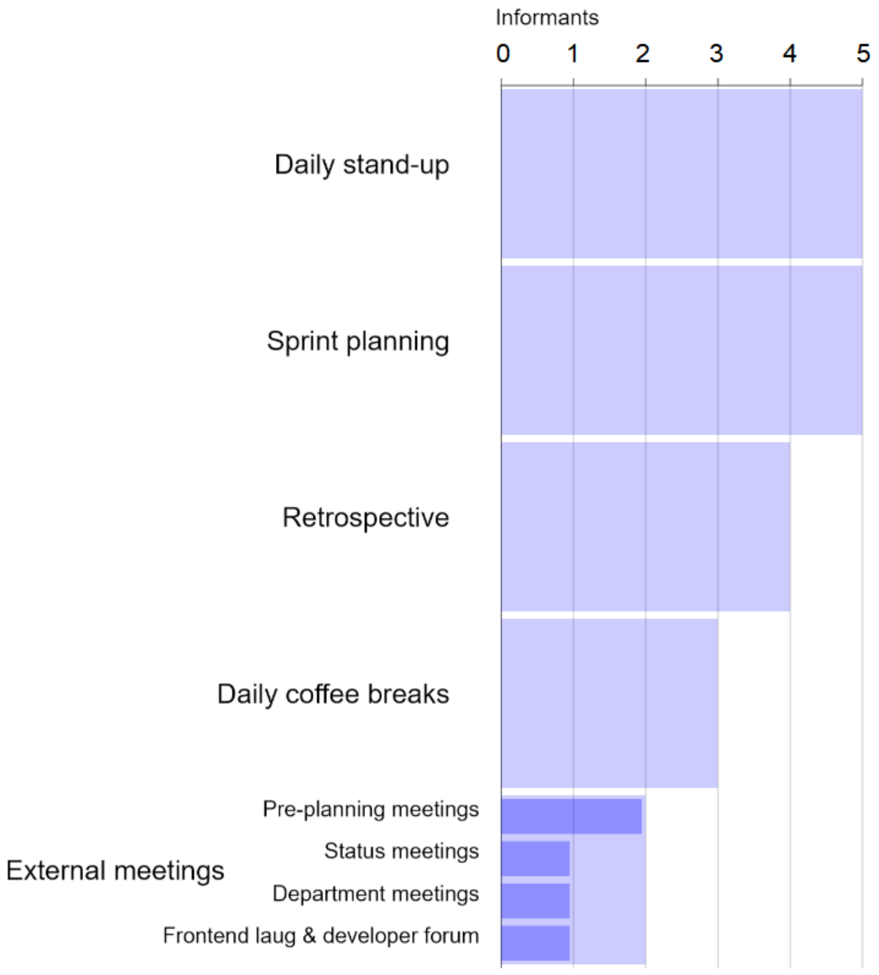


Figure 22: Team interaction mental models: Which meetings the team holds regularly, according to the informants.

meetings, and although they were not asked for it's frequency, 4 informants said that the meetings take place every 3 weeks, at the start of a new sprint. When asked to describe the protocol for the sprint planning meetings, 4 out of 4 responding informants said that the team gets a common overview of what work is to be done during the next sprint, and 3 of these informants said that they do this by going through a prioritized queue of tasks that has been prepared beforehand. 2 out of these 4 informants also said that during the sprint planning meeting, the team make an assessment of which tasks can actually be included in the sprint, and which tasks need more preparation.

4 informants mentioned retrospective meetings as one of their regular meetings, and all 4 informants said that the retrospective meetings were held as frequently as sprint planning meetings, that is, every three weeks. 3 out of 4 informants described the meeting protocol, and said that the team first summarize the previous sprint, as in what worked well and what didn't, and later they create a few points of action to improve these issues for the coming sprint.

3 informants mentioned that the team had daily coffee-breaks at a scheduled time, which offered the team members to meet in a more casual setting. Furthermore, 2 informants mentioned an array of meetings that were held externally to the team itself, which some members of the team attended regularly. Out of these external meetings, 2 informants mentioned pre-planning meetings in which a team-lead and all scrum-masters from the organization's teams planned the coming sprint before the teams themselves held their own internal sprint planning meetings. 1 informant mentioned dedicated status meetings for the team's various projects in which a selected few members reported to management about the projects' status. 1 informant mentioned department-meetings where all the organization's teams presented what they had been working on. 1 informant mentioned a meeting called "frontend laug" in which developers from the various teams discussed frontend, as well as the "developer forum" in which developers discussed backend. These meetings were held once every sprint. These results are shown in Figure 22.

When asked whether the team held any meetings irregularly or spontaneously, all 5 informants said that they do hold meetings for various purposes whenever there is a need for it. 4 informants mentioned estimation meetings as an example, in which the team attempts to estimate how much time is required to complete each individual task from a set of given tasks that require estimation. 2 informants mentioned "behovsfase" meetings, but both provided rather brief descriptions. 1 informant said that at the onset of a new release, the functional architect and UX designer present requirements related to the release in order to discuss their feasibility and cost with the rest of the team. The other informant said that during these meetings the team discuss which requirements they have and why they have them. The latter informant also expressed that these meetings are useful because attendants gain knowledge about not only what the requirements are, but why they are needed. Furthermore, 2 informants said that the team held clarification meetings whenever there were subjects in which the team needs clarification, but the structure of these meetings were highly variable. 1 informant mentioned meetings with end-users who

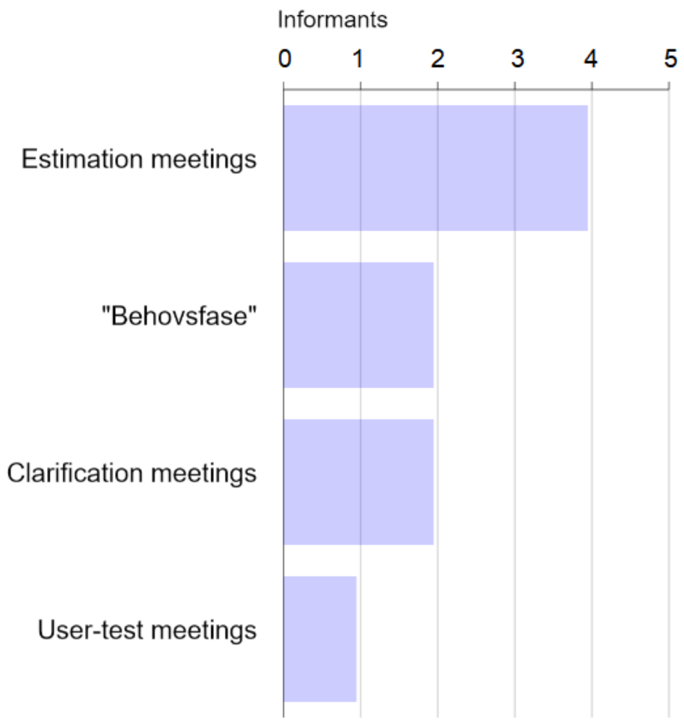


Figure 23: Which meetings the team holds irregularly, according to the informants.

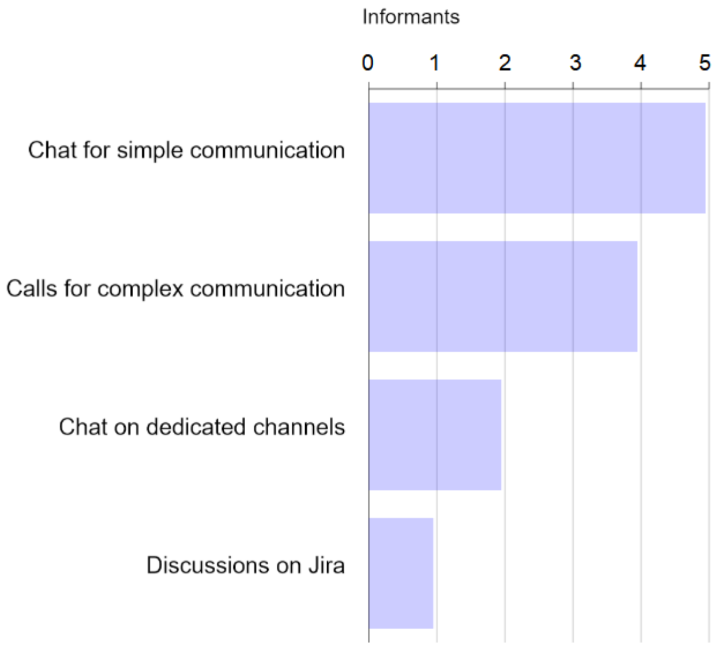


Figure 24: What type of interactions happen outside of meetings, according to the informants.

were testing the systems being developed by the team, saying that because the end-users don't understand the technical terms as well it is important to communicate properly to make sure they understand each other. These results are shown in Figure 23.

The informants were then asked whether they would communicate with other team members outside of the team meetings, and in which case they were asked to elaborate on how that communication would take place. All 5 informants said that if they had simple questions to another person then they would send them a text-based chat message on Teams, and if the correspondence required a more complex form of communication then they would initiate an audio or video call through Teams. While all 5 informants mentioned using text-based chat for communication, 2 of these informants also specified that the team had dedicated Teams channels for various purposes in which several team members could participate. 1 informant said that some discussions about specific tasks would take place textually within the Jira item of question in order to maintain documentation of the discussions. These results are shown in Figure 24.

5.2.5 Overlap measures

The initial calculated overlap measures based on the total received responses is shown in Table 16. These are the overlap measures prior to any filtering or combination of responses. Based on the researchers own interpretation of the results, some of these responses were filtered due to either being unspecific or irrelevant, or combined due to being similar. The resulting overlap measures after filter and combination is shown in Table 17. After filtering and combination, the total of the remaining responses yielded a numeric average of 2.5 and a moderate overlap measure of overall team-related knowledge.

Within the responses related to overarching team goals, we noticed that two statements possibly referred to the same thing. These statements were "Development of tools for case managers" and "Development of any currently ongoing project". During the feedback meeting we learned that the development of tools for case managers *were* the team's currently ongoing project. And so we argue that these statements may be combined, yielding a numeric average of 2.2 instead of 2.0.

The team membership knowledge was considered low-to-moderate for all roles, and low for the developer roles. There was, however, one informant who did not specify an exact number for several of the roles they mentioned. If all of the unspecified responses are filtered away, then the numeric average for all roles becomes 2.5, yielding a moderate degree of overlap as opposed to low-to-moderate. However, by filtering unspecified responses, the developer roles were only specified by 2 out of 5 informants, which is not enough to make any conclusions about the team's knowledge about the developer's roles. Thus we choose to disregard the responses related to developer roles due to lack of responses.

Within the responses about which actors were affected by each task, two of the responses did not specify which task they were talking about. If these

Knowledge type	**Numeric average**	**Label**
Overarching team goals	2.0	Low-to-moderate
Strategic consensus	1.0	Low
Team membership (all roles)	2.3	Low-to-moderate
Team membership (developer roles)	1.4	Low
Current tasks	2.6	Moderate
Actors affected (grouped by task)	1.6	Low
Actors affected (total)	3.0	Moderate
Task dependencies	1.8	Low
Task strategies	1.0	Low
Task allocation strategies	2.3	Low-to-moderate
Team interaction (regular meetings)	2.8	Moderate
Team interaction (irregular meetings)	2.3	Low-to-moderate
Team interaction (outside of meetings)	3.0	Moderate

Table 16: Initial overlap measures of each measured knowledge type prior to filtering of unspecified or irrelevant statements, or combination of similar statements.

two unspecified responses are filtered out, the numeric average becomes 1.8 as opposed to 1.6. The numeric average of the responses grouped by actor instead of task becomes 3.3 as opposed to 3.0. If the unspecified answer related to task dependencies is filtered out, then the task dependency overlap becomes 2.0 as opposed to 1.8, yielding a low-to-moderate overlap as opposed to low.

Within the task-allocation strategies, there was one response, "Ask if anyone needs help", that was only mentioned in a by-sentence without elaborating in detail. The informant implied that this was more of a personal practice than a general practice, and thus we argue that this response can be filtered due to irrelevance. 3 informants also mentioned "Defined roles" as a task allocation strategy, in which 2 referred to developer's roles as either back-end or front-end, and 1 referred to the developer's roles as to whether they were working on a back-end service or an application. Throughout the interviews the words "back-end" and "services" were often used interchangeably, and so were the words "front-end" and "application". For this reason we believe that these three responses all referred to the same thing, and thus can be grouped into a single statement named "Defined roles". After filtering and grouping responses, the numeric average of task allocation strategies becomes 3.3 as opposed to 2.3, yielding a moderate degree of overlap as opposed to low-to-moderate.

5 of the received responses related to interaction models of regular meetings were related to inter-team meetings. However, the question was aimed at intra-team meetings only, as is reflected by the phrasing of the question, and thus we argue that the statements related to "External meetings" may be disregarded, yielding a numeric average of 4.3 as opposed to 2.8, and a high overlap measure as opposed to moderate. One of the responses related to interaction models outside of meetings mentioned Jira discussions. Discussions on Jira is in the form of comments on specific tasks, has a very low synchronicity. It would not be strange if the majority of informants did not consider Jira comments to be a form of interaction, but rather documentation. In fact, a total of 4 informants mentioned the use of Jira for documentation, of which 2 mentioned using Jira for some type of discussion. For this reason we argue that the response may be considered irrelevant and thus filtered, yielding a numeric average of 3.7 as opposed to 3.0, and an overlap measure of moderate-to-high as opposed to moderate.

Knowledge type	**Numeric average**	**Label**
Overarching team goals	**2.2**	Low-to-moderate
Strategic consensus	1.0	Low
Team membership (all roles)	**2.5**	**Moderate**
Current tasks	2.6	Moderate
Actors affected (grouped by task)	**1.8**	Low
Actors affected (total)	**3.3**	Moderate
Task dependencies	**2.0**	**Low-to-moderate**
Task strategies	1.0	Low
Task allocation strategies	**3.3**	**Moderate**
Team interaction (regular meetings)	**4.3**	**High**
Team interaction (irregular meetings)	2.3	Low-to-moderate
Team interaction (outside of meetings)	**3.7**	**Moderate-to-high**

Table 17: Overlap measures of each measured knowledge type after the filtering of unspecified or irrelevant statements, and combination of similar statements. The highlighted values are the ones affected by filter and combination.

6 Discussion

The objective of this study was to provide empirical data of the shared team knowledge and knowledge sharing activities of a virtual software development team, in order to fill a gap of empirical research on this particular topic. Our case was a cross-functional software development team in which the team members all worked remotely from home, and through observed meetings and in-depth interview of 5 team members we sought to answer two questions: **R1:** "Which team-knowledge sharing activities are used by the team?", and **R2:** "What is the overlap measure of the team's shared team knowledge?".

After presenting the results to the aforementioned questions, the purpose of this chapter is to provide context and meaning to the results, examine their links to previous research, while illuminating possible weaknesses. Section 1 of this chapter discusses and evaluates the team's degree of virtualness. Section 2 discusses the findings of knowledge sharing activities while tying them to which types of team knowledge are likely being shared during these activities. Section 3 evaluates the overlap measures of various types of team knowledge and discusses possible causes and weaknesses of the results. In Section 4 we tie the previous sections together to evaluate the findings in relation to each other, and link our findings to previous research.

6.1 Team virtualness

To evaluate the team's degree of virtualness, we will consider the team's work situation and practices in terms of the degrees of virtualness as summarized by Johnson et al. (2009) [35]. These dimensions were geographic dispersion, percentage of time spent apart while working on a task, degree of physical distance, synchronicity of communication, temporality, diversity, electronic dependence, level of technology support, use of virtual communication tools, amount of informational value provided by those tools, and computer-mediated communication.

All of the team members were located in the same country and in or near the same city, but were dispersed in the manner that they were all located in their own separate homes. However, if we evaluate the teams degree of geographic dispersion with respect to country and city borders, it can be considered to be low. Although the team was not highly dispersed geographically, one could argue that they were still highly dispersed in a more practical sense, however, if we take their perceived distance into account [29]. Although the team members were located within commuting distance of each other, they were not allowed to physically meet, and so they might as well have been located on their own separate islands. For this reason one could argue that the team has a high degree of physical distance. Team members are never co-located while working on the same task, with the only exception being a single social gathering in a non-work setting, as well as a handful of meetings in which only a few of the team members gather in the organization's office building. In other words, the team spends close to one hundred percent of their work time apart. For this reason, all communication between team members took place through virtual tools, yielding a non-optimal synchronicity of communication in most of the cases. During scheduled meetings the team would all gather in a Teams video call with a relatively high communication synchronicity for the duration of the meeting. However, small delays seemed to be somewhat of a problem during these meetings, as it caused people to talk over and interrupt one another. For this reason we considered the synchronicity of communication during meetings to be moderate-to-high instead of high.

Communication outside of meetings usually took place through text-based chat through Teams. While it can be argued that text-based communication generally have a low degree of synchronicity, especially with technologies like e-mail where one have to actively check for new e-mails. The team used Microsoft Teams for their daily work, however, meaning that team members seemed to be logged into teams mostly throughout the day, so they were likely to get a notification as soon as someone sent them a message. There was also a very low overhead for sending someone a text-based message through chat compared to for example e-mail. The results also showed that text-based chat was only used for simpler correspondences, while more complex queries were often made by making video or audio calls through teams. We concluded that the communication synchronicity or lower for text-based chat than for audio and video calls, but due to the way in which the chats are used to initiate contact that may later lead to audio or video calls, we'd argue that their synchronicity of communication outside of the meetings is moderate, as opposed to low. Thus the overall synchronicity of communication of the team can be considered to be moderate.

The team members all spoke Norwegian and seemed to be Norwegian citizens, and so their degree of diversity could be considered as low. Since they were all located in the same time-zone their business hours were the same, and they had set "core-hours" during which employees were expected to be present at work. These factors contributed positively to the team's degree of temporality. Additionally, they worked in an iterative manner with a set of regular meetings, such as the daily stand-up, sprint planning and sprint retrospective. An iterative way of working may provide a team with a rigid structure that maintains the team's common pace of working. As such we would consider the team's temporal dispersion to be low.

As for the dimensions of virtualness that were related to the use of technology, many of those dimensions seemed similar in nature, and so our results about these dimensions were not as clear-cut as with the dimensions related to distribution. Nonetheless, they still provided enough information to describe these dimensions if we consider the context and the overall results as a whole, instead of individual findings. For example, our case is a software development team where each team member work remotely. Their work station is a computer, and it is needed for any collaboration with the rest of the team. For this reason alone it is possible to claim that the team's electronic dependence is quite high. All of their workflows are facilitated through various technologies, making the team's level of technology support high as well. All of their communication takes place over Microsoft Teams, either through voice or video calls, or through text-based chats, and thus we conclude that their use of virtual communication tools, as well as their degree of computer-mediated communication, is high. As for the informational value provided by those tools, it seemed like the team depended on them for collaboration and coordination, making their informational value quite high.

Dimension of virtualness	**Degree**
Geographic dispersion	Low
Degree of physical distance	High
Percentage of time spent apart while working on a task	High
Synchronicity of communication	Moderate
Temporality	Low
Diversity	Low
Electronic dependence	High
Level of technology support	High
Use of virtual communication tools	High
Informational value provided by virtual communication tools	High
Computer-mediated communication	High

Table 18: The team's degree of virtualness.

Table 18 summarizes our interpretations of the team's dimensions of virtualness. Out of 11 dimensions, 7 of them were considered to be high, 1 was moderate, and 3 were considered to be low. If each dimension weighed equally we would simply declare the team's overall degree of virtualness to be moderate-to-high. However, this largely depends on the definition of team virtuality, and the weights of each of its factors. Unfortunately, there are disagreements about how team virtualness should be defined and evaluated [31], and so pinning down an exact quantitative measure of the team's degree of virtualness becomes a challenge that falls outside the scope of this paper.

6.2 Knowledge sharing activities

While we did not directly observe any explicit sharing of goal-related knowledge, several activities were mentioned during the interviews. Two types of meetings mentioned, "Behovsfase" and architectural meetings, focus on the sharing of goal-related knowledge, although these meetings are not held very often. Moreover, it is likely that working with the product owner in a cross-functional team enables this kind of knowledge sharing [11].

"Behovsfase" meetings revolve around not only discussing the requirements of a coming release, but also the reasoning of the various requirements. From the information we could gather around these meetings, it is likely a facilitator of knowledge sharing related to the team's overarching goals, as it helps team members learn why they do the things that they do. Moreover, it is the springboard for dialogue between the developers and the non-developers, such as the product owner, architects and designers, providing a more complete picture and making sure the different parties understand each other. These meetings, however, are only held at the beginning of a new delivery, and thus don't occur very often. Architectural meetings, as one informant mentioned, are held more as the need arises, and revolve around bridging the team's tasks with the domain. From what we could gather, these meetings also facilitate sharing of goal-related knowledge in the form of explaining the domain (overarching goals), and architecture (goal strategies). We have no information about how often these meetings are held, but are inclined to assume it is not very often because these types of meetings were only mentioned once by a single informant.

While the team may not often hold meetings with the purpose of sharing goal-related knowledge, some of their Agile practices may still contribute. Through iterative work cycles and Scrum roles the product owner is frequently brought close to the developer, and are able to steer them in the direction of the goal during throughout the development process [11, 68]. Task prioritization enables the product owner to inform the development team of which tasks are most important to achieve the most value, and priorities may be re-evaluated each sprint as the development evolves [10]. By having cross-functional teams there are frequent interactions between developers and non-developers such as architects, enabling a flow of information between those with roles that require more goal-related knowledge, and those with roles that require more technical knowledge [11].

Nonetheless, we did not observe any instances of goal-related knowledge sharing, mostly because our observed meetings' purposes are mostly to share other types of knowledge, such as team-related and task-related. For more complete data, we would also have liked to observe a "behovsfase" meeting as well as an architectural meeting, but none of these meetings were being held for the duration of this project. It is likely that had we been able to observe any of these meetings we would have a more complete picture of the team's sharing of goal-related knowledge, which, as it is now, is somewhat lacking.

One recurring observation of the sharing of team interaction mental models is the collective use of the Wiki pages throughout the meeting. Although the Wiki is used for documenting various aspects of the different meetings, the meetings are documented by following a template that the team collectively fill in throughout the meeting. These templates serve as an agenda for what will be done throughout the meeting, and may help inexperienced team members know the purpose and plan of the meeting. At least, from the researcher's own point of view as an outsider, the collective use of the templates was helpful for this purpose. During the observed meetings we also saw demonstrations of sharing of process-related knowledge in the form of reminding the team of the purpose and scope of this meeting in order to get them back on track, and sometimes suggesting possible meetings to bring up the topic, or suggest that the discussing parties schedule a later time for that discussion. Other than using Wiki templates and reminding the team of the meeting scope, we did not observe any explicit sharing of team interaction mental models. However, it seems reasonable to assume that this is a type of knowledge that will be learned naturally through experience as a member of the team. 1 informant said an amount of training is required for new team members, and although we were not given any details about this training, it is not unlikely that some introduction to the team's way of collaborating is in place. If a team member is present on Microsoft Teams, then they will be invited to team meetings as they are scheduled, and thus they will know which meetings are being held that they should attend.

A lot of shared process- and team-related knowledge was observed during the sprint retrospective, where the team members not only shared their thoughts and opinions about the team's overall situation, but also open discussions about the team's practices and whether they should be changed. If someone perceived that a team norm was not optimal, this meeting allowed them to bring it up for discussion with the rest of the team. If the team decided on changes to their norms, then these decided changes would be externalized as points of action in the Wiki. Overall, the sprint retrospective appeared to us as the largest facilitator of team-norm related knowledge sharing. At the end of the meeting the team would collectively discuss "health check" points, which is a documentation of the team's overall status in terms of resource needs for upper management. This discussion served to share a mental model of the team's needs.

Although sprint-planning meetings mainly focused on the sharing of task-related knowledge, some sharing of team-related knowledge was observed as well. The meeting starts by discussing which KOMPIS lectures should be held, and this discussion facilitates sharing of knowledge related to expertise location. Although the KOMPIS meetings themselves don't focus on a specific type of knowledge, they revolve around general knowledge sharing, and also help team members get insight of the lecturing person's expertise. The meetings are superficially documented in the sprint-planning meeting's Wiki page, although it is unclear how frequently the Wiki pages from such meetings are actually accessed and consumed.

Daily stand-ups facilitate the sharing of a team membership model in that every team member give a brief description about their daily activities. While this may seem trivial, the case team was quite large, and when the members do not meet in-person on a regular basis it may be difficult to keep track of who is who. Overall, we did not observe many activities that revolved around the sharing of team membership knowledge, and so we argue that the daily stand-up meetings are crucial for this purpose.

From the explanations provided by the data, Jira can be described as a knowledge management system for task-related knowledge. A majority of the team's documentation processes seemed to revolve around Jira and the management of task-related team knowledge. The system was not only used for externalization, but also actively used by the developers for things such as task description, task allocation and clarifications related to the task's requirements and strategies.

The team's sprint planning practices also largely contribute to the sharing of task-related team knowledge. A part of the sprint planning meeting's protocol is that for each task planned for the sprint, a person with knowledge about the task will share it with the rest of the group, often prompting discussions. The results of these discussions may be externalized in Jira, where it will later be consumed by the person who works on the task. Furthermore, task estimation meetings seems to serve as a facilitator of shared task strategies, as they involve team members sharing their knowledge about how a task should be solved and how long it might take.

While the daily stand-up is not a platform for problem solving, it facilitates the sharing of task knowledge on a more superficial level in that team members report on problems they're having, and other team members may provide possible solution strategies. Moreover, it serves as a gateway to further discussions after the stand-up meeting in smaller groups specific to certain problems. It was also used for brief clarifications with the other team members.

6.3 Shared team knowledge

The overlap measure of shared team knowledge was found to be on the lower end of moderate, and we believe a significant factor to this result is the non-interdependence of several team members. During our feedback meeting we discussed whether the team by definition was, in fact, a team, or whether they were a group. We learned that while the team's developers functioned as a team, that is, they were mutually dependent, it would seem that the entire team functioned more as a group, meaning that different people could work on different items in parallel [36]. In this section we discuss some of these findings in more detail.

6.3.1 Goal-related team knowledge

The within overarching team goals, result showed two statements that were both responded by only a single informant each, namely "Correct incoming errors" and "Customer satisfaction". We consider the fact that the former statement were responded by a team member whose tasks and responsibilities differ from the other respondents. Due to the team's structure more resembling a group than a team, the team members could have responsibilities and tasks that did not overlap with the rest of the team's, making them have a different perspective on the overarching goal of the team. This would explain why there is some spread in the results, and it could explain why this individual considered error correction to be an overarching team goal, while the other respondents did not. For the latter statement, "customer satisfaction", it should be noted that the informant who expressed this statement answered very thoroughly on the question, and mentioned a higher number of goals compared to the other informants.

The statements regarding the team's strategic consensus were highly scattered, and almost all the informants found the question to be difficult. The scatter in results could be caused by several factors: First, it could simply be that the team had a low degree of knowledge overlap on goal strategies. This interpretation seemed plausible due to the feedback received when presenting the results during the feedback meeting. We were told that these are abstract questions that one tend to not think about in one's daily work, and that the team members were more concerned with the concrete tasks that are ahead of them. Secondly, since the question was phrased in a manner where the informants were first asked for the team's overarching goals, and then after they were asked about the team's strategy to reach those goals, the resulting degree of strategic consensus is dependent on the degree of agreement of the team's overarching goals. In other words, it becomes unlikely that the degree of strategic consensus will be higher than the agreement on the team's overarching goals.

6.3.2 Team-related team knowledge

In short, there was a high agreement that the team had 2 functionally responsible, 1 Scrum master, 1 UX designer and 1 functional architect. There was some scatter as to how many developers were on the team, ranging from 5 to 8 with one unspecified answer. The architect, team-lead and system owner were mentioned by only a single informant each, meaning that the majority most likely did not consider them to be part of the team.

When listing the members of the team there was one informant who did not specify an exact number of people belonging to each role. For example, this individual said that they had a plural of functionally responsible, at least 4 developers, at least 1 back-end developer and either two or three developers who work both back-end and front-end. The phrasing of the question was "Who is on the team, and which roles to they have?", and it does not explicitly ask for an exact number, even if that is what the researcher were implicitly asking for. Neither was the informant asked to specify exact numbers during the interview. These mistakes were caused by the inexperience of the researcher as an interviewer, and could have been avoided had the question been phrased to explicitly ask for specific numbers, or if the researcher probed for specific numbers during the interview. As we proceed to interpret the results, however, all we can do is keep this fact in mind and consider the likelihood that an unspecified number does not necessarily mean that the informant does not know, but rather that they did not know that that's what was being asked for. For this reason we chose to disregard their response, as their actual knowledge has not been collected.

There was some scatter in the responses about how many developers are on the team, where 1 informants said eight, 1 said five, 2 said six and 1 did not specify but mentioned at least 4 different people. During the feedback meeting we learned that a possible explanation for this was that after the lockdown was initiated, the team had gained several new team members who, at the time of the interviews, had never met the rest of the team in-person. Moreover, some of these new members had non-developer roles, meaning that, due to the structure of the team (i.e. as a group in which the developers worked as a team), they did not work very interdependently with all the developers.

Out of the three informants who specified the developers' roles as either front-end, back-end or both, there was also some scatter, and one of the respondents did not specify exact number of people who worked either back-end or both. During the feedback meeting we learned that the team only has one front-end developer, while the rest of them, in practice, are able to work both front-end and back-end. However, some developers avoid front-end work if they can and thus consider themselves back-end developers only. In other words, the developers' role definitions seemed to be somewhat unclear, with the exception of one developer having the role as pure front-end. This could be a possible explanation for the scatter. However, with only three respondents specifying, the sample size is very thin, especially when one informant did not specify exact numbers, and so the conclusions we can draw from those results will be weak. For the 2 informants who did not specify, it is difficult to know whether they did not know the developers' roles, or whether they simply did not know that it was relevant information. The phrasing of the question did not specify exactly how specific their responses should be, which is likely the cause as to why some informants specified about the developers' roles, while others didn't.

The team architect, team-lead and system owner were all mentioned by only one informant each. For the architect is likely that the majority of team members simply don't consider them as technically part of the team. They were grouped with them for social reasons and did not work directly with the team's tasks. However, during our observations, the person who were listed as architect in our received list of team members seemed to be present at all of the observed meetings (i.e. stand-up, retrospective and sprint-planning), and actively participating in discussions. What we make of this, is that the architect is an active participant in common group meetings and activities, but that their responsibilities and tasks do not overlap with the rest of the team, and therefore they are not considered technically as a team member. Similarly to the architect, it is likely that only a single person mentioned the team-lead because the majority do not consider them as part of the team. As we described in Chapter 4, the team-lead acted more as a project or department manager responsible for several teams, and therefore is not involved in the team's day-to-day work.

As for the system owner, it seemed like the team has had several recent changes of system owners, and that it was simply hard to keep track. During the retrospective we observed that one person asked whether a particular person is the system owner, confirming that some team members may find it hard to keep track of who the system owner is. However, the system owner seemed to participate in the daily stand-up meetings, so they appeared to be at somewhat involved in the team's daily activities. It seems improbable that the majority of team members were not aware of their presence, and so it must be questioned whether the informants considered the system owner role as part of the team or not.

6.3.3 Task-related team knowledge

When asked which tasks the team are currently working on, the informants responded with an array of various tasks. All informants were in agreement that the team was working on "Project P1", which was the replacement of the work tools for the organization's internal case managers. They were all also in agreement that the team had continuous tasks related to maintenance and error correction of existing services and applications. This seems to correlate with the informants responses about the team's overarching goals. 3 informants responded "Project P2" as one of the team's tasks, which was the task related to their customers' rights to access member data in the case of fusions or fissions between organizations. The fact that the 3 informants who responded "Project P2" as one of the team's current tasks were the same 3 informants who responded "Ensure correct data" on the team's overarching goals further increases our belief that the team's responses about the team's overarching goals were highly related to the team's current tasks.

While there was a general consensus about the tasks related to "Project P1" and "Maintenance and error correction", there is only a moderate consensus about "Project P2", and a low consensus for the rest of the tasks. These responses, similarly to the responses regarding overarching goals, could be due to the fact that the team by definition more resembles a group than a team. With different people working on different things without being mutually dependent of one another, it is more difficult to be aware of each others tasks than if they were working on the same projects. For example, the information who responded "Project P3" said that it is only a single person who worked on those tasks. Similarly, there were only two team members who were working on the tasks related to developing end-points for other teams. The spread in the responses regarding the team's current tasks is thus likely explained by the fact that different team members are working on different non-interdependent tasks.

For two of the tasks that only had 1 respondent each, namely "Preparing for next delivery" and "Project P4", it may very well be that that the other informants don't consider it a "task" in the same sense as the other responses. The informant who responded "Preparing for next delivery" said that many of the team members are involved, or as they said it, "the whole gang". It seems unlikely that the team were not aware that this preparation was taking place. Moreover, during the observed retrospective, the team expressed that they were happy with the recent workshop, which had the same name as "Project P4", and so the team seemed to be aware of its existence. Furthermore, the informant who responded with "Project P4" expressed their uncertainty about the team's involvement in that project. Neither "Preparing for next delivery" nor

"Project P4" was mentioned any more in the informants' responses regarding task content, strengthening our belief that the informants did not consider them significant enough for further discussion.

While there was a high agreement that the "Project P1" task affected the case managers, there was an overall low degree of overlap in which actors were affected by each task. A potential cause is that these results are dependent of and will inherit the scatter of the responses related to the team's current tasks.

There was some disagreement about whether there were any task dependencies in "Project P1". While 3 informants seemed to agree that the project currently had dependent task, 1 informant said that the tasks' dependencies had been resolved last week. The interviews were all conducted in the middle of the same week within a 3-day span, so this was an actual disagreement and not just a change of circumstances. It could be that either of the groups who responded that there was or wasn't dependencies in "Project P1" were fully informed on the status quo, possibly because all the team members don't work interdependently. 2 out of 3 informants who talked about "Project P2" also talked about it having dependencies, so although the number 2 is low in and of itself, it is a moderate amount compared to how many informants mentioned "Project P2" in the first place.

Similarly to the responses about the goal strategies, a majority of the informants found the questions about task strategies to be difficult, and although all of them responded, only two of them actually provided responses related to task strategies, while all of them also (or only) described the team's task allocation strategies. None of the responding informants said the same thing regarding the team's task strategies, mostly because they discussed the strategies of different tasks. It is possible that the phrasing of the question was too vague, and instead of "how will the team perform those tasks?" it might have been better with something along the lines of "what is the strategy for performing those specific tasks?". Yet, we can't help but notice the similarities between the graphs of goal strategies and task strategies, making it possible that the spread in responses is not inherently caused by the vagueness of the question. Again we refer to the findings that the team functions more as a group. It wouldn't be strange if a person did not know how a specific task is to be solved is they are not directly involved in solving it.

6.3.4 Process-related team knowledge

The bare results yielded a moderate overlap in interaction models related to regular meetings, low-to-moderate overlap related to irregular meetings, and a moderate overlap related to communication outside of meetings. However, as we will show in the coming paragraphs, we argue that a set of responses may be disregarded, yielding a moderate-to-high overlap in regular meetings and communication outside of meetings.

The informants had a moderate-to-high degree of overlap in their answers about which meetings the team holds regularly, with the exception of the mention of a set of meetings that take place externally to the team. However, the question was aimed at intra-team meetings only, and thus we can disregard the responses regarding inter-team meetings. With irregular meetings, there was a high agreement that estimation meetings are held when ever there is a need for it, but a lower overlap on the remaining three meetings that were mentioned, namely "Behovsfase", clarification meetings, and user-test meetings, yielding an overall low-to-moderate overlap in irregular meetings.

As for interactions that happen outside of meetings, there was a high agreement that simple correspondences are done through Teams chat, and often serve as a gateway for initiating more complex communication through video and audio calls. Fewer informants mentioned discussions on Teams channels and on Jira, however. It is peculiar that only 2 people mentioned the Teams channels, because they were described as being used often. It is possible, however, that other respondents who mentioned Teams chat were also referring to the use of channels, without explicitly saying so. The fact that "Jira" was not a more frequent response is likely due to the phrasing of the question, which was as whether the team members interact outside of meetings, and in which case how those interactions take place. Discussions on Jira is in the form of comments on specific tasks, has a very low synchronicity. It would not be strange if the majority of informants did not consider Jira comments to be a form of interaction, but rather documentation. In fact, a total of 4 informants mentioned the use of Jira for documentation, of which 2 mentioned using Jira for some type of discussion. For this reason we chose to disregard the "Discussions on Jira" response, yielding a moderate-to-high overlap in interactions outside of meetings.

6.4 Summarizing the discussion

We suggested that several of the low scores of knowledge overlap were possibly caused by the team's structure as a group instead of the formal definition of a team, in which every team member is mutually dependent [36]. These suggestions would be in agreement with previous findings showing that dependence is positively associated with knowledge sharing [51].

The team's goal-related knowledge appeared to be in correlation with their task-related knowledge. In other words, their idea of the team's overarching goals were tied to which ever tasks were before them, possibly due to the team's frequent sharing of task-related knowledge compared to their relatively infrequent sharing of goal-related knowledge. With this focus on tasks, the informants were able to provide a rich set of data about their task-related team knowledge, while the data regarding their goal-related team knowledge was a bit narrower. Likewise, the data showed a rich set of task-knowledge sharing activities, and very few goal-knowledge sharing activities. It must be noted that a rich data provides greater opportunity to create an overlap than lacking data, and it must be considered that the data collection methods may have been inadequate in collecting data related to goal-related knowledge and knowledge sharing. For example, a sprint-planning was observed, which is heavily task focused, while no "behovsfase" meeting was observed, which have more emphasis on goals. Nonetheless, the overlap in the informants' knowledge of overarching team goals seemed to be somewhat similar to their overlap in which tasks they are working on. Similarly, their overlap in goal strategies was similar to their overlap in task strategies.

Out of the measured team-related knowledge we only procured measures of team membership knowledge, and none of team member models and expertise location. The main reason for this is that mapping such knowledge would require lengthy interviews, and so knowledge types had to be prioritized. In hindsight we take notice that the majority of observed sharing of team-related knowledge is not of the measured type (i.e. team membership), but rather the unmeasured types (i.e. expertise location and team member model). The only observed sharing of team membership knowledge was when each team member briefly described their daily activities, and this sharing was not explicit enough to give an updated overview of the exact role of each team member. We only measured a moderate degree of team membership knowledge, as there seemed to be some disagreement how many developers were on the team, and whether the architect, team-lead and system owner were considered to be part of the team. While the overlap in team membership knowledge is not optimal, it is not a complete picture of the teams overlap in team-related team knowledge, and under different circumstances it would be desirable to measure expertise location and team member model as well.

The largest portion of identified knowledge sharing activities were related to the sharing of task-related knowledge, and of most significance was possibly the sprint-planning and the use of Jira as a task-knowledge management system. The overlap score of task allocation strategies was the 3rd highest score of all the knowledge types, possibly because task allocation is almost completely facilitated by Jira's prioritized task queue. Task descriptions were considered moderate, and this type of knowledge was brought up every sprint planning by visually sharing a list of Jira items, supported by explanations and discussions. Moreover, task descriptions were shared during daily stand-up, as people described what they were working on. However, the informants had a low overlap of task strategies, which is a type of knowledge that is both managed by Jira and discussed during sprint-planning and partly in daily stand-ups, so it can be questioned whether these activities alone made a difference. It must be considered, however, that there is a significant possibility that up to 3 of the informants misunderstood what the researcher was asking for when asking about task strategies during the interviews, and so the internal validity of the overlap of task strategies suffers.

Knowledge related to task dependencies scored a low-to-moderate overlap, and we only observed sharing of knowledge related to task dependencies during the sprint-planning, where some Jira items would be sorted under "awaiting dependencies", and when team members verbally explained dependencies. Although task-dependency knowledge would be externalized by uploading a screenshot of the Miro board to Wiki, it is unlikely that this knowledge was accessed frequently by the team members in the same way as the Jira task queue. Externalization alone does not constitute knowledge sharing unless another party internalizes the knowledge, and it is possible that sharing of task-dependency knowledge simply wasn't frequent enough to yield a higher degree of sharedness. In a similar fashion, we observed no sharing of knowledge related to which actors are affected by specific tasks, correlating with the low overlap in which actors were affected by which tasks. Aside from the task-strategies measures, the positive link between task knowledge managed by Jira and overlap measure of task knowledge would be in agreement with previous findings that suggest the use of technology for knowledge management improves knowledge processes such as knowledge sharing [1].

There was a high overlap in interaction models for regular meetings, which is also a type of knowledge that is partly managed by a knowledge management system (i.e. the Wiki) through active documentation. The overlap score for interaction models of irregular meetings was low-to-moderate, and none of the informants mentioned any documentation of these meetings. These results are in agreement with previous findings that usage of Wikis as a knowledge management system leads to improved sharing of knowledge [43]. It must be noted, however, that just because the informants did not mention it, it doesn't necessarily mean that the documentation did not take place. But it does suggest that the informants were unaware of such documentation. The interaction models outside of meetings scored moderate-to-high without being mentioned in any documentation, but this is likely a type of knowledge that is learned through experience in working with the team, and possibly through training.

7 Conclusions

Due to an emergent need for a high degree of virtuality in software development teams paired with a lack of empirical studies of such cases, this study set out to describe a case of a single virtual software development team. The aim was to describe their degree of virtuality, their knowledge sharing activities, and their overlapping degree of shared team knowledge. The study was conducted through qualitative and semi-quantitative analysis of observations, interviews and a feedback meeting with the team's Scrum master.

The team's virtualness was measured in terms of 11 different dimensions related to distribution and use of technology, where 7 of them were high, 1 moderate, and 3 were low. More specifically, the team had a high degree of technology usage, while the dimensions related to distribution were overall moderate due to their low degree of diversity, geographic distribution and temporal dispersion. With a vast disagreement in literature as to how virtuality is defined and measured [31], we chose not to explicitly label the overall degree of virtuality, but rather describe each of the identified dimensions of virtuality [35]. The hope is that this transparency enables method generalisability that is independent of which measure and definition of virtuality is used.

The team used a variety of Agile practices that has been linked to increased knowledge sharing, such as various Scrum practices, cross-functional teams and an iterative work cycle. They used Jira as a knowledge management system for task-related knowledge, and a Wiki for the management of process-related and some team-related knowledge. We observed much sharing of team-norm knowledge during sprint retrospective, and much sharing of task-related knowledge during sprint planning. Both of these meetings involved usage of Miro to externalize knowledge. During the daily stand-up be observed some sharing of team-related knowledge and some sharing of task-related knowledge on a superficial level. We did not observe any sharing of goal-related knowledge, and only a few infrequent goal-knowledge sharing activities were mentioned during the interviews.

The team's overlap score of team knowledge was on the lower end of moderate, with higher scores related to team interaction at regular meetings and outside of meetings and task allocation strategies between the developers, and lower scores related to goal strategies and task strategies. We discussed these results with the team's Scrum master during a feedback meeting and learned that a significant amount of the spread may be linked to the non-interdependence of the team members in a large team, which would be in agreement with previous research claiming dependence is positively linked with knowledge sharing [51]. The types of knowledge that were managed through knowledge management systems generally scored higher than the types that didn't, with the exception of task strategies. These results, too, are in agreement with previous research finding that usage of knowledge management systems is positively linked with knowledge sharing [1, 43].

7.1 Limitations

The most important limitation to this study is that the results are not, and are not supposed to be, generalisable. It is a descriptive case study, and so the goal is to describe various aspects with a single case, and the aspects of a general population. Although we explored possible links and explanations of the results in the discussion chapter, these explanations are highly specific to the case, and although some findings may be in agreement with previous findings, the goal of the study was not to confirm or reject previous research.

The most significant threat to internal validity was likely the low number of informants interviewed. In order to have a more correct overlap measure of shared team knowledge it would be required to interview the entire team, which unfortunately we were not given the opportunity to do. Instead we interviewed a sample of about half the team and were required to generalise the result from that sample to the whole team. The biggest downside to this was that in the cases where several of the informants did not respond, the results were too weak to draw any conclusions and in some cases the results were discarded.

Due to a restriction in interview time, we chose not to attempt to map the team's knowledge related to expertise location, team member models and team norms. We concluded that mapping these types of knowledge would require a large amount of time, and with with the low number of informants it was possible that the data would not be complete. For this reason the overlap measure in those three knowledge types were not measured, making our overlap measure of team knowledge somewhat incomplete.

The types of observed meetings were also not complete, especially because none of the observed meetings involved sharing of goal-related knowledge. To get a complete picture of the team's knowledge sharing activities it would be required to observe requirement phase meetings, architectural meetings and possibly estimation meetings as well. Unfortunately, these meetings were not held for the duration of data collection.

7.2 Implications and future research

The main contribution of this paper is empiric material that, hopefully, future researchers may use alongside other empirical studies to increase our common knowledge about knowledge sharing in virtual software development teams. We believe that recent events have contributed to a normalization of working from home, and thus it is also our belief that it is highly relevant to study how to increase the performance of highly distributed virtual teams. Especially in the software development industry, which is highly knowledge-intensive, this is partly achieved by researching how to increase the team's knowledge sharing. For this reason we have two main propositions for future research.

First, a single empiric case hardly contributes to the common knowledge base on it's own. It is only in combination and comparison with other empiric studies that the findings have further utility. For this reason we suggest that future researchers perform similar case-studies, describing the virtuality and knowledge sharing behavior of software development teams, as well as the success of these knowledge sharing behaviours.

Next, once there is an established base of empirical studies, we suggest that systematic reviews are performed to investigate the relationships between knowledge sharing behaviour, degree of shared knowledge, and team virtualness. More specifically, we believe it would be interesting to see whether virtualness acts as a moderator between knowledge sharing activities and the degree of shared knowledge. Or in other words, whether certain types of knowledge sharing are less effective when performed in a more virtual setting.

References

[1] Maryam Alavi and Dorothy E Leidner. “Knowledge management and knowledge management systems: Conceptual foundations and research issues”. In: *MIS quarterly* (2001), pp. 107–136.

[2] Maryam Alavi and Amrit Tiwana. “Knowledge integration in virtual teams: The potential role of KMS”. In: *Journal of the American Society for Information Science and Technology* 53.12 (2002), pp. 1029–1037.

[3] Mohammad Alsharo, Dawn Gregg, and Ronald Ramirez. “Virtual team effectiveness: The role of knowledge sharing and trust”. In: *Information & Management* 54.4 (2017), pp. 479–490.

[4] Yanti Andriyani. “Knowledge Management and Reflective Practice in Daily Stand-Up and Retrospective Meetings”. In: *International Conference on Agile Software Development.* Springer, Cham. 2017, pp. 285–291.

[5] Yanti Andriyani, Rashina Hoda, and Robert Amor. “Understanding knowledge management in agile software development practice”. In: *International Conference on Knowledge Science, Engineering and Management.* Springer. 2017, pp. 195–207.

[6] Giovanni Asproni. “An introduction to scrum”. In: *Software Developer’s Journal* 6 (2006), pp. 1–10.

[7] Swati Kaul Bhat, Neerja Pande, and Vandana Ahuja. “Virtual team effectiveness: an empirical study using SEM”. In: *Procedia Computer Science* 122 (2017), pp. 33–41.

[8] Finn Olav Bjørnson and Torgeir Dingsøyr. “Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used”. In: *Information and Software Technology* 50.11 (2008), pp. 1055–1068.

[9] Angel Cabrera, William C Collins, and Jesus F Salgado. “Determinants of individual engagement in knowledge sharing”. In: *The International Journal of Human Resource Management* 17.2 (2006), pp. 245–264.

[10] Lan Cao and Balasubramaniam Ramesh. “Agile requirements engineering practices: An empirical study”. In: *IEEE software* 25.1 (2008), pp. 60–67.

[11] Thomas Chau and Frank Maurer. “Knowledge sharing in agile software teams”. In: *Logic versus approximation.* Springer, 2004, pp. 173–183.

[12] Sharolyn Converse, JA Cannon-Bowers, and E Salas. “Shared mental models in expert team decision making”. In: *Individual and group decision making: Current issues* 221 (1993), pp. 221–46.

[13] Nancy J Cooke et al. “Measuring team knowledge”. In: *Human factors* 42.1 (2000), pp. 151–173.

[14] Jonathon N Cummings. “Work groups, structural diversity, and knowledge sharing in a global organization”. In: *Management science* 50.3 (2004), pp. 352–364.

[15] Thomas H Davenport and Varun Grover. “Knowledge management”. In: *Journal of management information systems* 18.1 (2001), p. 3.

[16] Thomas H Davenport, Laurence Prusak, et al. *Working knowledge: How organizations manage what they know.* Harvard Business Press, 1998.

[17] Vida Davidavičienė, Khaled Al Majzoub, and Ieva Meidute-Kavaliauskiene. “Factors Affecting Knowledge Sharing in Virtual Teams”. In: *Sustainability* 12.17 (2020), p. 6917.

[18] Ana Ortiz De Guinea, Jane Webster, and D Sandy Staples. “A meta-analysis of the consequences of virtualness on team functioning”. In: *Information & management* 49.6 (2012), pp. 301–308.

[19] Norman K Denzin and Yvonna S Lincoln. *The Sage handbook of qualitative research.* sage, 2011.

[20] Esther Derby, Diana Larsen, and Ken Schwaber. *Agile retrospectives: Making good teams great.* Pragmatic Bookshelf, 2006.

[21] James H. Dulebohn and Julia E. Hoch. “Virtual teams in organizations”. In: *Human Resource Management Review* 27.4 (2017). Virtual Teams in Organizations, pp. 569–574. ISSN: 1053-4822. DOI: `https://doi.org/10.1016/j.hrmr.2016.12.004`. URL: `http://www.sciencedirect.com/science/article/pii/S1053482216300961`.

[22] Michael Earl. “Knowledge management strategies: Toward a taxonomy”. In: *Journal of management information systems* 18.1 (2001), pp. 215–233.

[23] Christof Ebert and Jozef De Man. “Effectively utilizing project, product and process knowledge”. In: *Information and Software Technology* 50.6 (2008), pp. 579–594.

[24] Tor Erlend Fægri, Viktoria Stray, and Nils Brede Moe. “Shared knowledge in virtual software teams: A preliminary framework”. In: *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE).* IEEE. 2016, pp. 174–178.

[25] Ali Yahya Gheni et al. “FACTORS AFFECTING GLOBAL VIRTUAL TEAMS’PERFORMANCE IN SOFTWARE PROJECTS”. In: *Journal of Theoretical and Applied Information Technology* 92.1 (2016), p. 90.

[26] Shahla Ghobadi. “What drives knowledge sharing in software development teams: A literature review and classification framework”. In: *Information & Management* 52.1 (2015), pp. 82–97.

[27] Shahla Ghobadi and Lars Mathiassen. “Perceived barriers to effective knowledge sharing in agile software teams”. In: *Information systems journal* 26.2 (2016), pp. 95–125.

[28] Egon G Guba, Yvonna S Lincoln, et al. “Competing paradigms in qualitative research”. In: *Handbook of qualitative research* 2.163-194 (1994), p. 105.

[29] Dorina C Gumm. “Distribution dimensions in software development projects: A taxonomy”. In: *IEEE software* 23.5 (2006), pp. 45–51.

[30] Paul Hendriks. “Why share knowledge? The influence of ICT on the motivation for knowledge sharing”. In: *Knowledge and process management* 6.2 (1999), pp. 91–100.

[31] M Reza Hosseini et al. “Evaluating virtuality in teams: A conceptual model”. In: *Technology Analysis & Strategic Management* 27.4 (2015), pp. 385–404.

[32] Samireh Jalali and Claes Wohlin. “Global software engineering and agile practices: a systematic review”. In: *Journal of software: Evolution and Process* 24.6 (2012), pp. 643–659.

[33] Sirkka L Jarvenpaa and D Sandy Staples. “The use of collaborative electronic media for information sharing: an exploratory study of determinants”. In: *The Journal of Strategic Information Systems* 9.2-3 (2000), pp. 129–154.

[34] Sirkka L Jarvenpaa and D Sandy Staples. “Exploring perceptions of organizational ownership of information and expertise”. In: *Journal of management information systems* 18.1 (2001), pp. 151–183.

[35] Stefanie K Johnson, Kenneth Bettenhausen, and Ellie Gibbons. “Realities of working in virtual teams: Affective and attitudinal outcomes of using computer-mediated communication”. In: *Small Group Research* 40.6 (2009), pp. 623–649.

[36] Jon R Katzenbach and Douglas K Smith. “The discipline of teams”. In: *Harvard business review* 83.7 (2005), p. 162.

[37] Brenda Killingsworth, Yajiong Xue, and Yongjun Liu. “Factors influencing knowledge sharing among global virtual teams”. In: *Team Performance Management* (2016).

[38] Bradley L Kirkman and John E Mathieu. “The dimensions and antecedents of team virtuality”. In: *Journal of management* 31.5 (2005), pp. 700–718.

[39] Richard Klimoski and Susan Mohammed. “Team mental model: Construct or metaphor?” In: *Journal of management* 20.2 (1994), pp. 403–437.

[40] Heeseok Lee and Byounggu Choi. “Knowledge management enablers, processes, and organizational performance: An integrative view and empirical examination”. In: *Journal of management information systems* 20.1 (2003), pp. 179–228.

[41] Hsiu-Fen Lin. “Effects of extrinsic and intrinsic motivation on employee knowledge sharing intentions”. In: *Journal of information science* 33.2 (2007), pp. 135–149.

[42] Lowell Lindstrom and Ron Jeffries. “Extreme programming and agile software development methodologies”. In: *Information systems management* 21.3 (2004), pp. 41–52.

[43] Ann Majchrzak, Christian Wagner, and Dave Yates. “Corporate wiki users: results of a survey”. In: *Proceedings of the 2006 international symposium on Wikis*. 2006, pp. 99–104.

[44] John E Mathieu et al. "The influence of shared mental models on team process and performance." In: *Journal of applied psychology* 85.2 (2000), p. 273.

[45] Matthew B Miles, A Michael Huberman, and Johnny Saldaña. *Qualitative data analysis: A methods sourcebook*. Sage publications, 2018.

[46] 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.

[47] Sarah Morrison-Smith and Jaime Ruiz. "Challenges and barriers in virtual teams: a literature review". In: *SN Applied Sciences* 2 (2020), pp. 1–33.

[48] James Newkirk. "Introduction to agile processes and extreme programming". In: *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*. IEEE. 2002, pp. 695–696.

[49] Tuyet-Mai Nguyen. "Four-dimensional model: a literature review in online organisational knowledge sharing". In: *VINE Journal of Information and Knowledge Management Systems* (2020).

[50] Judith Orasanu. "Shared mental models and crew decision making". In: *Princeton, NJ* (1990).

[51] Jun-Gi Park and Jungwoo Lee. "Knowledge sharing in information systems development projects: Explicating the role of dependence and trust". In: *International Journal of Project Management* 32.1 (2014), pp. 153–165.

[52] William B Rouse and Nancy M Morris. "On looking into the black box: Prospects and limits in the search for mental models." In: *Psychological bulletin* 100.3 (1986), p. 349.

[53] Ioana Rus, Mikael Lindvall, and S Sinha. "Knowledge management in software engineering". In: *IEEE software* 19.3 (2002), pp. 26–38.

[54] Linda Schweitzer and Linda Duxbury. "Conceptualizing and measuring the virtuality of teams". In: *Information systems journal* 20.3 (2010), pp. 267–295.

[55] Graeme Shanks et al. "Guidelines for conducting positivist case study research in information systems". In: *Australasian Journal of Information Systems* 10.1 (2002).

[56] Darja Smite, Marco Kuhrmann, and Patrick Keil. "Virtual teams [Guest editors' introduction]". In: *Ieee Software* 31.6 (2014), pp. 41–46.

[57] Darja Šmite et al. "Empirical evidence in global software engineering: a systematic review". In: *Empirical software engineering* 15.1 (2010), pp. 91–118.

[58] D Sandy Staples and Jane Webster. "Exploring the effects of trust, task interdependence and virtualness on knowledge sharing in teams". In: *Information systems journal* 18.6 (2008), pp. 617–640.

[59] Viktoria Stray, Dag IK Sjøberg, and Tore Dybå. “The daily stand-up meeting: A grounded theory study”. In: *Journal of Systems and Software* 114 (2016), pp. 101–124.

[60] Bart Van den Hooff et al. “Knowledge sharing in knowledge communities”. In: *Communities and technologies*. Springer. 2003, pp. 119–141.

[61] Sheng Wang and Raymond A Noe. “Knowledge sharing: A review and directions for future research”. In: *Human resource management review* 20.2 (2010), pp. 115–131.

[62] Jun Wei, Wei Zheng, and Mian Zhang. “Social capital and knowledge transfer: A multi-level analysis”. In: *Human Relations* 64.11 (2011), pp. 1401–1423.

[63] Karl M Wiig. “Knowledge management: an introduction and perspective”. In: *Journal of knowledge Management* (1997).

[64] Jessica L Wildman et al. “Team knowledge research: Emerging trends and critical needs”. In: *Human Factors* 54.1 (2012), pp. 84–111.

[65] John R Wilson and Andrew Rutherford. “Mental models: Theory and application in human factors”. In: *Human Factors* 31.6 (1989), pp. 617–634.

[66] Jen-Her Wu and Yu-Min Wang. “Measuring KMS success: A respecification of the DeLone and McLean’s model”. In: *Information & management* 43.6 (2006), pp. 728–739.

[67] Robert K Yin et al. “Design and methods”. In: *Case study research* 3.9.2 (2003).

[68] Shi Zhong, Chen Liping, and Chen Tian-en. “Agile planning and development methods”. In: *2011 3rd International Conference on Computer Research and Development*. Vol. 1. IEEE. 2011, pp. 488–491.

A Appendix: Interview Guide

Introduksjon

Takke personen for å stille til intervju

Presentere meg selv

- Navn og studie
- Masteroppgave om kunnskapsdeling i virtuelle team

Kort om intervjuet

- Varighet ca 60 min

Dette er ikke en eksamen av dine kunnskaper

- Hvis du får et spørsmål du ikke vet svaret på, er det greit å si «vet ikke»

Samtykkeskjema

- Deltakelse er frivillig
- Du kan når som helst trekke deg i ettertid
- Du kan velge å avstå fra å svare på spørsmål
- Intervjuet vil anonymiseres

Be om samtykke for opptak (Sett på ved tillatelse)

Bli kjent

Kan du fortelle meg litt om jobben / stillingen din?

Datainnsamling

Kan du fortelle litt om teamet du er i nå?

Jobber dere kun fra hjemmekontor, eller møtes dere noen ganger fysisk?

Hvilket medium brukes for møter?

Hvordan opplever du å bruke dette mediet for møter?

Holder teamet noen faste møter?

Hva gjør dere på de faste møtene?

Bruker teamet å holde noen møter utover de faste møtene?

Er alle møtene like nyttige for deg?

Deltar du på alle møter som blir planlagt?

Hender det at du interagerer med andre på teamet utenfor planlagte møter? I så fall, hvordan kommuniserer dere da?

Hvordan dokumenterer teamet arbeidet som blir gjort?

Hvem er det som er med på teamet ditt, og hvilke roller har de?

Hvilken rolle har du?

Finnes det noen områder hvor du føler har mer ekspertise enn de andre på teamet?

Finnes det noen områder du har lite kunnskap om?

Vet du om noen på teamet som sitter med kunnskap om dette området?

Hvordan opplever du å kontakte andre på teamet for å spørre om slikt?

Hva er det teamet ønsker å oppnå med arbeidet dere gjør?

Har du noen personlige mål som du ønsker å oppnå?

Føler du at de andre på teamet deler det samme målet som deg?

Hvilket arbeid skal utføres for at teamet skal nå disse målene?

Hvilke oppgaver jobber teamet med nå?

Hvordan skal disse oppgavene utføres?

Hvordan fordeler dere disse oppgavene (eller deler av oppgavene) mellom hverandre?

Er noen av disse oppgavene eller deloppgavene avhengig av hverandre?

Hvordan vil disse oppgavene påvirke miljøet utenfor teamet (dvs. f. eks. brukere, andre team hos SPK)

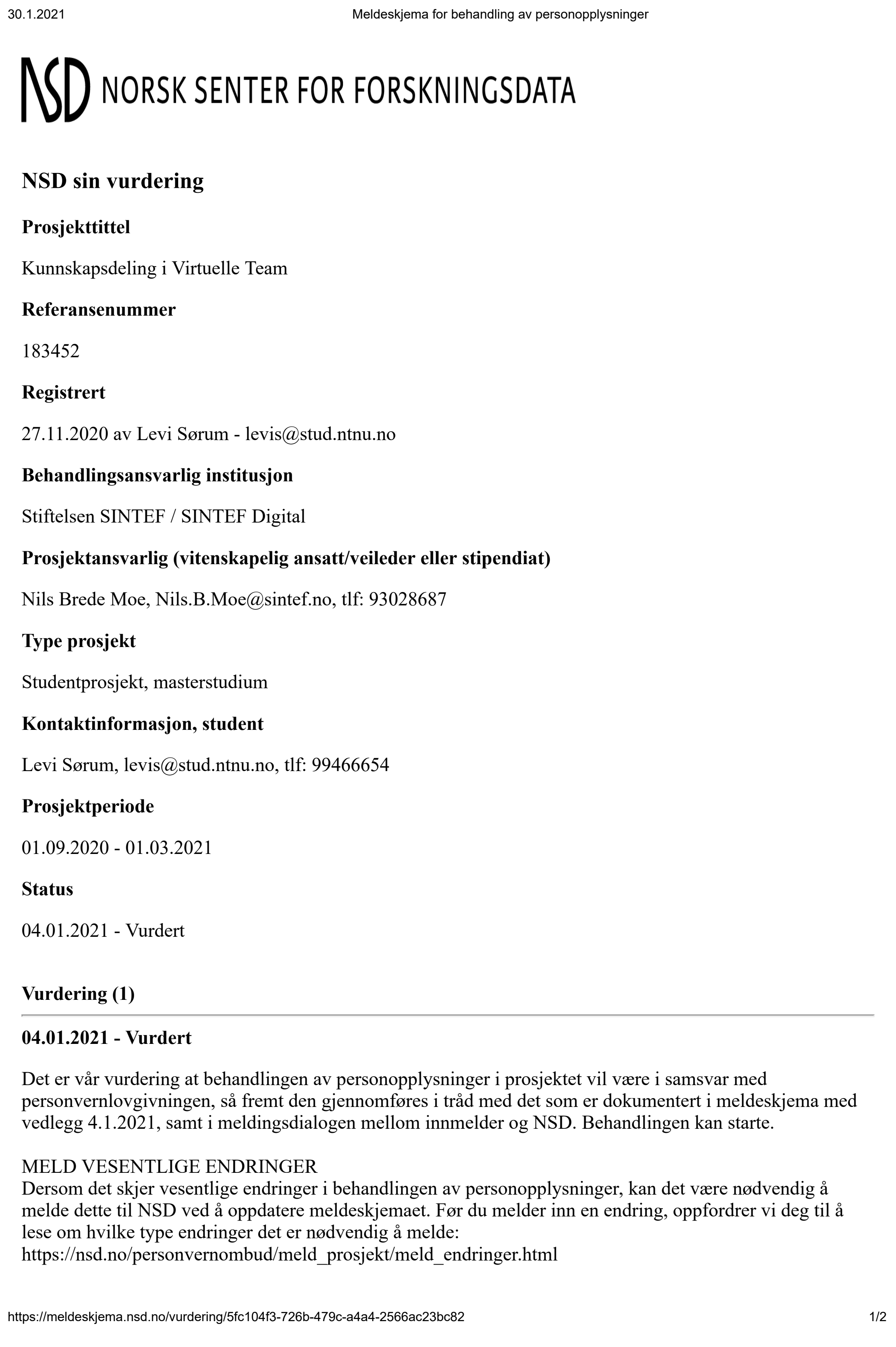
Avslutning

Er det noe mer du ønsker å legge til som du føler bør være med, som vi ikke allerede har snakket om?

Har du noen spørsmål til meg?

Takke for deltakelse

B Appendix: Approval from NSD



30.1.2021 Meldeskjema for behandling av personopplysninger

NORSK SENTER FOR FORSKNINGSDATA

NSD sin vurdering

Prosjekttittel

Kunnskapsdeling i Virtuelle Team

Referansenummer

183452

Registrert

27.11.2020 av Levi Sørum - levis@stud.ntnu.no

Behandlingsansvarlig institusjon

Stiftelsen SINTEF / SINTEF Digital

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)

Nils Brede Moe, Nils.B.Moe@sintef.no, tlf: 93028687

Type prosjekt

Studentprosjekt, masterstudium

Kontaktinformasjon, student

Levi Sørum, levis@stud.ntnu.no, tlf: 99466654

Prosjektperiode

01.09.2020 - 01.03.2021

Status

04.01.2021 - Vurdert

Vurdering (1)

04.01.2021 - Vurdert

Det er vår vurdering at behandlingen av personopplysninger i prosjektet vil være i samsvar med personvernlovgivningen, så fremt den gjennomføres i tråd med det som er dokumentert i meldeskjema med vedlegg 4.1.2021, samt i meldingsdialogen mellom innmelder og NSD. Behandlingen kan starte.

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til NSD ved å oppdatere meldeskjemaet. Før du melder inn en endring, oppfordrer vi deg til å lese om hvilke type endringer det er nødvendig å melde:
https://nsd.no/personvernombud/meld_prosjekt/meld_endringer.html

https://meldeskjema.nsd.no/vurdering/5fc104f3-726b-479c-a4a4-2566ac23bc82 1/2

Du må vente på svar fra NSD før endringen gjennomføres.

TYPE OPPLYSNINGER OG VARIGHET
Prosjektet vil behandle alminnelige kategorier av personopplysninger frem til 1.3.2021.

LOVLIG GRUNNLAG
Prosjektet vil innhente samtykke fra de registrerte til behandlingen av personopplysninger. Vår vurdering er at prosjektet legger opp til et samtykke i samsvar med kravene i art. 4 og 7, ved at det er en frivillig, spesifikk, informert og utvetydig bekreftelse som kan dokumenteres og som den registrerte kan trekke tilbake.

Lovlig grunnlag for behandlingen vil dermed være den registrertes samtykke, jf. personvernforordningen art. 6 nr. 1 bokstav a.

PERSONVERNPRINSIPPER
NSD vurderer at den planlagte behandlingen av personopplysninger vil følge prinsippene i personvernforordningen om:
- lovlighet, rettferdighet og åpenhet (art. 5.1 a), ved at de registrerte får tilfredsstillende informasjon om og samtykker til behandlingen
- formålsbegrensning (art. 5.1 b), ved at personopplysninger samles inn for spesifikke, uttrykkelig angitte og berettigede formål, og ikke viderebehandles til nye uforenlige formål
- dataminimering (art. 5.1 c), ved at det kun behandles opplysninger som er adekvate, relevante og nødvendige for formålet med prosjektet
- lagringsbegrensning (art. 5.1 e), ved at personopplysningene ikke lagres lengre enn nødvendig for å oppfylle formålet

DE REGISTRERTES RETTIGHETER
Så lenge de registrerte kan identifiseres i datamaterialet vil de ha følgende rettigheter: informasjon (art. 13), innsyn (art. 15), retting (art. 16), sletting (art. 17), begrensning (art. 18), underretning (art. 19), dataportabilitet (art. 20).

NSD vurderer at informasjonen som de registrerte vil motta oppfyller lovens krav til form og innhold, jf. art. 12.1 og art. 13.

Vi minner om at hvis en registrert tar kontakt om sine rettigheter, har behandlingsansvarlig institusjon plikt til å svare innen en måned.

FØLG DIN INSTITUSJONS RETNINGSLINJER
NSD legger til grunn at behandlingen oppfyller kravene i personvernforordningen om riktighet (art. 5.1 d), integritet og konfidensialitet (art. 5.1. f) og sikkerhet (art. 32).

For å forsikre dere om at kravene oppfylles, må dere følge interne retningslinjer og eventuelt rådføre dere med behandlingsansvarlig institusjon.

OPPFØLGING AV PROSJEKTET
NSD vil følge opp ved planlagt avslutning for å avklare om behandlingen av personopplysningene er avsluttet.

Lykke til med prosjektet!

Kontaktperson hos NSD: Lasse Raa
Tlf. personverntjenester: 55 58 21 17 (tast 1)

C Appendix: Letter of invitation

Vil du delta i forskningsprosjektet «Kunnskapsdeling i Virtuelle Team»?

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å få bedre kunnskap om hvordan kunnskapsdeling fungerer i virtuelle team. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål
Studiet er et case-studie for å undersøke hvordan et team av programvareutviklere som arbeider sammen virtuelt deler kunnskap mellom hverandre. Vi ønsker å få innsikt i hvilke metoder teamet benytter for å dele kunnskap med hverandre, samt vurdere effektiviteten av disse. Teamet vil observeres gjennom 3 digitale prosjekt møter etter avtale med leder for teamet, og hver enkelt deltaker vil bli intervjuet én gang. Studiet gjennomføres i relasjon til en masteroppgave som skrives i samarbeid med NTNU og SINTEF.

Hvem er ansvarlig for forskningsprosjektet?
SINTEF Digital og Instituttet for Datateknologi og Informatikk ved NTNU er ansvarlige for forskningsprosjektet.

Hvorfor får du spørsmål om å delta?
Du er utvalgt til å delta i denne studien fordi du er en person av høy interesse som jobber som en del av et tverrfaglig virtuelt team. Det er totalt 13 personer som får denne henvendelsen, og alle er en del av det samme teamet.

Hva innebærer det for deg å delta?
Hvis du velger å delta i prosjektet innebærer det at du blir observert ved tre forskjellige anledninger, samt at du stiller deg disponibel til intervju. Intervjuet vil ta 45-60 minutter, og lydopptaker vil bli brukt etter samtykke. Intervjuet består av spørsmål om din rolle i teamet, hvordan du kommuniserer med teamet, og din oppfattelse av teamet og dets arbeidsmetoder. Observasjon av møter vil bli skrevet referat om, hvor hovedtemaer som noteres er teamets arbeidsmetoder under møter.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle dine personopplysninger vil da bli slettet. Du kan også velge å avstå fra å delta i prosjektmøtet samt svare på enkeltspørsmål under intervju. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg, eller dersom du avstår fra å svare på spørsmål.

Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger
Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket. Masterstudenten samt veileder og prosjektansvarlig ved SINTEF er de eneste som vil ha tilgang til dine opplysninger. Opplysningene vil bli oppbevart på en kryptert server hos SINTEF. All data vil bli anonymisert ved publikasjon, og det vil ikke være mulig å identifisere enkeltpersoner som har deltatt i studien.

Hva skjer med opplysningene dine når vi avslutter forskningsprosjektet?
Prosjektet skal etter planen avsluttes 18. februar 2021. All data som ikke har blitt anonymisert vil bli slettet når studien er avsluttet.

Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:
- innsyn i hvilke personopplysninger som er registrert om deg, og å få utlevert en kopi av opplysningene,
- å få rettet personopplysninger om deg,
- å få slettet personopplysninger om deg, og
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger.

Hva gir oss rett til å behandle personopplysninger om deg?
Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra SINTEF har NSD – Norsk senter for forskningsdata AS vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Hvor kan jeg finne ut mer?
Hvis du har spørsmål til studien, eller ønsker å benytte deg av dine rettigheter, ta kontakt med:
- NTNU, Institutt for datateknologi og informatikk ved masterstudent Levi Sørum (levis@stud.ntnu.no), telefon: 994 66 654
- SINTEF Digital, Avdeling for Software Engineering, Safety and Security ved Nils Brede Moe (nils.b.moe@sintef.no), telefon: 930 28 687
- Vårt personvernombud: NSD – Norsk senter for forskningsdata AS (personverntjenester@nsd.no), telefon: 555 82 117

Hvis du har spørsmål knyttet til NSD sin vurdering av prosjektet, kan du ta kontakt med:
- NSD – Norsk senter for forskningsdata AS på epost (personverntjenester@nsd.no) eller på telefon: 55 58 21 17.

Med vennlig hilsen

Nils Brede Moe Levi Sørum
(Forsker/veileder)

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet *[sett inn tittel]*, og har fått anledning til å stille spørsmål. Jeg samtykker til:

☐ å delta i observasjon
☐ å delta i intervju

Jeg samtykker til at mine opplysninger behandles frem til prosjektet er avsluttet

----
(Signert av prosjektdeltaker, dato)