Teamwork Quality in Software Engineering Education

A case study of the course IN2000 at the University of Oslo

Egwene Tegelaár

Thesis submitted for the degree Master of Science in Informatics: Programming and System Architecture
60 credits

Department of Informatics
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

June / 2020

Teamwork Quality in Software Engineering Education

A case study of the course IN2000 at the University of Oslo

Egwene Tegelaár

June 2020

2020

Teamwork Quality in Software Engineering Education

Egwene Tegelaár

http://www.duo.uio.no/

Trykk: Reprosentralen, Universitetet i Oslo

Abstract

Background: As the industry of software engineering often requires the employees to work in a team setting using agile methods, it is important to teach bachelor students of software engineering these skills. Courses that require a project allows students to collaborate in an agile team setting, utilizing the tools seen in the industry. This gives the students an opportunity to practice technical skills and soft skills, such as communication. Additionally, they can use the theory they have learned in a practical setting, making them better prepared to enter the industry. However, team-based courses are complex to execute, especially as a mandatory course where the number of enrolled students and teams is high.

Aim: The aim of this thesis is to investigate student teams, their teamwork quality, and the agile practices they utilize. Specifically, what characterizes a good student team and what affects their result.

Method: A case study utilizing mixed methods was conducted in a project-based software engineering course. Data was collected through the use of a survey, interviews, and other documentation like the project reports from the student teams. The study sample for the survey was 197 students and 5 teaching assistants. Interviews were conducted with 6 students and 2 teaching assistants.

Results: The results provided insight on several aspects of student teams; choice of project case, team composition, process models, and rating of teamwork quality. Additionally, the results revealed that there are several differences between the low- and high-performing teams, one of which is multidisciplinarity.

Conclusion: While communication and coordination within the teams are important, it is effort that has the highest correlation to the resulting project grade for the high-performing teams. For all teams, it seems that the project grade is more related to the level of success for the individual team members than it is to teamwork quality. The thesis provides four suggestions for practice; 1) *partly instructor-formed teams*, 2) *technical onboarding*, 3) *closer relationship between teaching assistants and teams*, and 4) *guidance on implementation of agile processes*.

Acknowledgements

What an adventure it has been, completing this thesis. Someone once told me to enjoy the process, as it in itself is a memory. While writing this master thesis has been challenging at times, I have enjoyed every minute of it.

I am forever grateful to my supervisors, Yngve Lindsjørn and Viktoria Stray, for all their guidance, help, and support throughout the process of writing this thesis. They saw my interest for IN2000 and suggested that the course and student teamwork could be the topic for this thesis. I couldn't have dreamt of a better topic! Thank you both for such a gift. I also wish to thank my fellow teaching assistants in IN2000, especially Steffen Almås and Erlend Stenlund. Working alongside you has been a joy, and your enthusiasm for the course has fired up under my own.

To my fiancé, Benjamin; no one has believed in me as fiercely as you (nor has anyone proofread this thesis quite as fiercely as you). Thank you for the support, patience, and take-away dinners.

Additionally, I want to thank my wonderful sister, who through this entire process has FaceTimed me frequently just to get an update on the latest word count. I appreciated it more than what it seemed. To my friends and family, thank you for allowing me to speak endlessly about this topic, and for your constant encouragement. I know I promised this would be the end of it, but I am afraid I lied.

Lastly, a huge thanks goes to the original IN2001 team, Bingr. My very first team. You all taught me so much and inspired me in ways I didn't even realize. Being a part of our team was the highlight of my bachelor studies.

Egwene Tegelaár
Oslo, June 2020

Table of Contents

List of Tables

List of Figures

List of Acronyms & Abbreviations

API = Application Programming Interface

IDE = Integrated Development Environment

Prosa = Programming and System Architecture

Design = Design, Use and Interaction

Digec = Digital Economics and Leadership

TWQ = Teamwork Quality

App = Application

1 Introduction

The topic for this master thesis is teamwork quality in software engineering education. To investigate this topic, I used a case study of the student teams participating in the mandatory bachelor course *IN2000 – Software Engineering with Project Work* (IN2000) at the University of Oslo.

Agile has become the de-facto way of working in the industry (Zaitsev et al., 2020, p. 1), and relies on heavily teamwork (Moe et al., 2008, p. 76). Thus, the importance of teamwork projects during the bachelor's degree is said to be essential for students of software engineering (Iacob & Faily, 2018, p. 163; Ju et al., 2018, p. 144). Teamwork projects can teach students how to collaborate in an agile team setting, utilizing skills and tools used by the industry to solve problems and develop a product (Iacob & Faily, 2018, p. 170). This will make them better prepared to enter the industry upon the completion of their degrees (Fioravanti et al., 2018, p. 806).

IN2000 was taught at full capacity for the first time in the spring semester of 2019. The year before, in 2018, a pilot for the course was held at a much smaller scale. The pilot had only voluntarily enrolled students. The objective of IN2000 is for the students to develop an Android application (App) in less than three months by working in teams of 4-6 students, and by utilizing different agile methods and tools. At first glance the course seems heavily focused on technology. While this is true, another central learning outcome of the course is to learn how to work together as a team. Therefore, it is essential to look at how the teams work together and the quality of the teamwork in a student project setting. This can be done with the help of teamwork models. While there are many teamwork models that can be beneficial to use in a student setting, like the "Big Five" by Salas et al. (2005), this thesis will look at the student teamwork through the TWQ model from Hoegl & Gemuenden (2001).

1.1 Motivation

Signing up as a single-course student for an introductory course to software engineering at the Department of Informatics in 2016, is the reason I am handing in my master thesis, four and a half years later. This course got me interested in the planning phase of the development process and teamwork in software engineering. Thus, leading me to pursue several courses in this particular area of software engineering at the Department of Informatics. This allowed me to take the pilot of the course IN2000 in 2018. The course taught me to put software engineering theory into practice, utilizing agile tools and processes, and working structured in a team setting towards a common goal. The course moved its focus from solitary task work to working together as a team over a period three months. This shift of focus allowed me to experience first-hand that there is more to software development than programming, and that in fact a large proponent of software development is teamwork. Learning how to work in a team is important because it is a vital tool used in software development companies.

I have had the privilege to work as a teaching assistant in IN2000 both in 2019 and 2020. This involvement has allowed me to take an active role in the evolution of the course through the co-creation of mandatory assignments, correction of the mandatory assignments, and by providing feedback. Being so involved in the course has been motivating.

1.2 Research Question

As there is a broad range of topics covered in IN2000, and several interesting research possibilities in regard to teamwork and the use of a large project in a mandatory course, deciding on a research question has been challenging. The focus of this thesis has been on teamwork and agile methods in software engineering education, which is reflected in the research questions. To attempt to answer these questions, I will draw on studies done on agile teams, teamwork models, and software engineering education

RQ1: What characterizes a high-performing team in a student project, and how does the communication and coordination within a team affect the result?

The first research question is concerned with the characteristics of a high-performing student team, and specifically if and how their communication and coordination within the team affects the result. Both communication and coordination are important in agile teams.

You communicate to coordinate, and coordinate through communication. This is especially the case with the daily stand-up meeting which can improve communication while functioning as way to coordinate team efforts (Stray et al., 2017, p. 274-275). However, communication is one of the most important factors in agile teams (Moe et al., 2008, p. 76; Cockburn & Highsmith, 2001, 9. 131).

RQ2: How is the teamwork quality in the student teams related to their project grade?

For the second research question, I will utilize two studies done on teamwork quality through the use of the TWQ model. The first was conducted by Hoegl & Gemuenden in 2001 and the second was conducted by Lindsjørn, Sjøberg, Dingsøyr, Bergersen, and Dybå in 2016. As the second study focused on agile teams, their results are what will be used when comparing student and professional teams. These studies showed that when the teamwork quality was highly rated within the teams, both the success of team members and team performance was rated higher as well (Hoegl & Gemuenden, 2001, p. 446). There is a separate construct for team performance in this model, but as this is a graded course in an educational setting I wish to investigate if teamwork quality is related to the project grade.

The research questions will be further explored in chapter 5 and 6.

1.3 Thesis structure

The rest of this thesis will be structured as follows:

Chapter 2 – *Background and related works*, provides an overview over current research in the areas of software engineering education and teamwork.

Chapter 3 – *Research Context*, covers the context of the research. This includes the structure and overview of IN2000 in 2019, how the course facilitated for agile development, and the team composition in the student projects.

Chapter 4 – *Research Methods*, covers the choice of methodology, how the data collection was conducted, and how the collected data has been analyzed.

Chapter 5 – *Results*, provides the findings and results from the collected data.

Chapter 6 – *Discussion*, here the background theory and the results will be combined in order to answer the research questions. In addition, it will go through the limitations of the thesis.

Chapter 7 – *Conclusion and future work*, provides concluding remarks and suggestions for further research.

2 Background and Related Works

2.1 Agile Teamwork

Katzenbach and Smith define a team as “a small number of people with complementary skills who are committed to a common purpose, set of performance goals, and approach for which they hold themselves mutually accountable" (2005, p. 3). A good team is one that has a strong feeling of unity and is motivated by the success of the team (Sommerville, 2016, p. 275). Table 1 provides an overview of the difference between a group of people who work together, and a team.

Group	**Team**
Strong clearly focused leader.	Shared leadership roles.
Individual accountability.	Individual and mutual accountability.
The group’s purpose is the same as the broader organizational mission.	Specific team purpose that the team itself delivers.
Individual work products.	Collective work products.
Runs efficient meetings.	Encourages open-ended discussions.
Measures its effectiveness indirectly by its influence on others (such as financial performance of the business).	Measures performance directly by assessing collective work products.
Discusses, decides, and delegates.	Discusses, decides, and does real work together.

Table 1: The difference between groups and teams from Katzenbach & Smith (2005, p. 4)

In software engineering, teamwork is of great importance, and it is especially emphasized in agile methods (Lindsjørn et al., 2016, p. 274; Stray et al., 2011, p. 146). The optimal team size for agile teams tends to be three to six members, however, there are variations to this number (Hoegl, 2005, p. 211; Sommerville, 2016, p. 274).

2.1.1 Self-Organizing Teams

The Agile Manifesto states in its principles for agile that self-organizing teams will provide "the best architectures, requirements, and designs" (Agile Manifesto, 2001). As such, self-organization characterizes not only agile teams, but also agile methods (Stray et al., 2011, p. 147; Cockburn & Highsmith, 2001, p. 132). They are often autonomous, which means that they usually have a high degree of independence, dedication and leadership (Patanakul et al., 2012, p. 734). There are three factors of autonomy that influence how much freedom a team has, these are; 1) *external autonomy* which denotes outside influence on the activities in the team, 2) *internal autonomy* which is concerned with how the team structures work and decisions, and 3) *individual autonomy* which covers how much freedom the team members have in organizing their own tasks (Moe et al., 2008, p. 78). Self-organizing teams are often seen as effective since they have the freedom to make decisions within the team, which in turn allows them to solve arising problems accurately and quickly (Moe et al., 2008, p. 77). However, companies experience difficulties implementing self-organizing autonomous teams. Some of the most common barriers are that the teams have too many dependencies to others, there is a lack of trust, and the team members do not have clear and common goals (Moe. et al, 2019). In addition, further investigation on whether self-organized teams actually provide better performance is required (Dingsøyr et al., 2016, p. 109).

Another central aspect of self-organized and autonomous teams is that the teams are cross-functional, also referred to as multidisciplinary teams. Cross-functional teams have team members with the necessary experience or skills for the work done in the team, who are also able to step in where needed (Parker, 2003, p. 4). The latter refers to the redundancy of functions, which means that "team members acquire multiple skills so that they are able to perform each other's jobs and substitute as the need arises" (Moe et al., 2008, p. 82). For example, a programmer can also do testing and work with the requirements. A team that is cross-functional in regard to the skills present in the team has an increased ability for self-organization (Hoda et al., 2013, p. 424), as the required skills are present, and they can utilize them to reach the goals of the team. Parker found that cross-functional teams were most effective in industries like the software industry "that value adaptability, speed, and an intense focus on responding to customer needs" (2003, p. 6). Hoda points out that there is a need to balance specialization and cross-functionality in the team, particularly in self-organizing teams (2013, p. 239).

2.1.2 Coordinating Mechanisms

Coordination is essential in agile teams, especially in self-organized teams (Zaitsev, 2020, p.1; Stray et al., 2019b, p. 111). It can be defined as the "efforts to manage resources, relationships and task interdependencies" (Zaitsev, 2020, p. 1). Coordination mechanisms can enable the team to manage these efforts and to work together more effectively by promoting team members' understanding of the tasks and available resources in their team (Salas et al., 2005, p. 559). Three coordinating mechanisms described in Salas et al. are 1) *shared mental models* – common understanding of the goal and how to reach it, 2) *closed-loop communication* – both sender and receiver of information acknowledge that it has been received, and 3) *mutual trust* – all team members look out for each other and the team (Salas et al., 2005, p. 565-570). Stray et al. differentiate between synchronization artifacts and synchronization activities as coordination mechanisms (Stray et al., 2019a, p. 7011).

Synchronization artifacts are tools which can facilitate coordination in a team (Zaitsev, 2020, p.1, 20). These tools are used to coordinate the teamwork and can manifest as tasks, communication tools, backlogs and Kanban boards to mention a few (Stray et al., 2019a, p. 7011). Internet based communication tools are being used more and more in both organizations and start-ups, one example of this is Slack (Stray et al., 2019b, p. 111). The benefit of using a communication tool is "increased transparency, team awareness, and informal communication" (Stray et al., 2019b, p. 111). In a study it was found that the ability to be able to edit and delete one's own posts in the communication tool might make it easier for team members to participate in online communication (Stray et al., 2019b, p. 112).

Another form of coordination, that pertains to the abovementioned coordination mechanisms and is an example of synchronization activities, is meetings. Communication is one of the most important factors in agile teams (Moe et al., 2008, p. 76; Cockburn & Highsmith, 2001, 9. 131), and often manifests as meetings. Here, activities connected to the tasks are coordinated and issues are often shared (Stray, 2018, p. 1). Meetings can be planned or unplanned. A study of software teams in large projects found that team members spent more time in unplanned coordination (unscheduled meetings and ad-hoc conversations), than they did in planned coordination (scheduled meetings) (Stray, 2018, p. 2). Communication on for example Slack is also a form of unplanned coordination, and it may be expected in teams that the team members are up to date on any discussion or communication on Slack (Stray et al., 2019b, p. 117).

2.1.3 Scrum, Kanban and ... Scrumban?

Because agile teams work with agile process models, it is important to add a little about what these are. There are three agile process models that are relevant for this thesis; Scrum, Kanban, and Scrumban. Other popular agile process models and methodologies, like XP, will not be covered.

Scrum is a “framework that involves and employs various processes and techniques to build something" (Stoica et al., 2016, p. 10). The team in Scrum is cross-functional and is “responsible for managing itself to develop software every Sprint" (Schwaber, 2004, p. 143). In addition to the team, there are two roles that are central, the *product owner* who represents the stakeholders and manages the *product backlog*, and the *scrum master* who facilitates scrum in the team (Schwaber, 2004, p. 142). The scrum practices include; the use of *sprints* which are time-boxed periods where work gets done, a list of prioritized requirements for the project in the *product backlog*, and the *sprint backlog* which is a list of the tasks to be completed during the sprint (Schwaber, 2004, p. 136, 142). In addition to these practices, a large part of Scrum is the use and implementation of meetings during a sprint, these are presented in table 2. Stray et al. found that it was particularly the practice of daily stand-up meetings “that distinguishes agile from non-agile teams” (Stray et al., 2017, p. 278), but also observed that non-agile teams are adopting the daily stand-up meetings.

Sprint Planning Meeting:	Planning the next sprint by selecting items from the product backlog that can be developed in the next iteration. Following this a sprint backlog is created by the team, which consists of the tasks and estimates for the tasks for the coming sprint.
Daily Scrum, or Stand-Up, Meeting:	Daily 15-minute meeting for the team where each team member goes through the three questions; w*hat have you done, what will you do, what are your obstacles*?
Sprint Review Meeting:	A demonstration of the functionality completed during the sprint, held at the end of the sprint.
Sprint Retrospective Meeting:	A meeting at the end of a sprint where the team discusses two areas in regard to the finished sprint; *what went well, what could be improved?* Improvements are discussed and actionable items are prioritized in the next sprint to make it better.

Table 2: Scrum meetings (Schwaber, 2004, p. 133-139)

While Scrum is a time-boxed process model that limits the work-in-progress for each sprint, Kanban is task-based and limits the work-in-progress for each state of the workflow (Stray, 2011, p. 153; Kniberg & Skarin, 2010, p. 15). Kanban "is based on Just-In-Time and Lean production systems" (Nikitina et al., 2012, p. 140), and is much less prescriptive than Scrum. The focus of Kanban can be summed up in the following three principle;

1) Workflow visualization – through the use of a Kanban board with columns representing each state in the workflow (Kniberg & Skarin, 2010, p. 15).

2) Limit work-in-progress – each workflow state limits the maximum number of tasks that may be in that state at any given time (Kniberg & Skarin, 2010, p. 4)

3) Measure lead time – by measuring how long it takes for a task to be completed the workflow can be optimized and predictable (Stoica et al., 2016, p. 12).

Scrumban is as the name suggests a hybrid process model comprised of elements from both Scrum and Kanban (Nikitina et al., 2012, p. 140). The focus of Scrumban is to optimize the workflow within a Scrum process, usually through implementing continuous flow instead of being driven by sprints (Reddy, 2015; Nikitina et al., 2012, p. 142). An organization or team can implement the continuous development flow from Kanban and at the same time keep beneficial elements from Scrum (Nikitina et al., 2012, p. 141). This is what the first conception of Scrumban was, a layering of "a Kanban system within a Scrum context" (Reddy, 2015). Reddy emphasizes that this is one way of seeing Scrumban, but that it has evolved into much more than this (2015).

2.2 Teamwork Models

One of the ways to understand teamwork is through the use of models. There are many teamwork models, and they focus on different aspects of teamwork. Some look at *team performance*, others at *team effectiveness*, and others still consider them together. In the teamwork quality construct by Hoegl & Gemuenden (2001) *team performance* is made up of team effectiveness and team efficiency, while in Salas et al. (2005) their model the "Big Five" is focused on team effectiveness *and* team performance.

The "Big Five" consists of five components that are often present in teamwork models; team leadership, mutual performance monitoring, backup behavior, adaptability, and team orientation (Salas et al., 2005, p. 570). A review conducted on team performance studies found that there were "five factors that particularly influence performance: team coordination, goal orientation, team cohesion, shared mental models, and team learning" (Dingsøyr et al., 2016, p. 106). The same review compared these five factors to the Agile Manifesto to see if and how they corresponded to the advice given to agile teams (Dingsøyr et al., 2016, p. 108). Another study points out that "team performance is complex, and the actual performance of a team depends not only on the competence of the team itself in managing and executing its work, but also on the organizational context provided by management" (Moe et al., 2010, p. 481). While each component in the "Big Five" model is seen as necessary for team effectiveness, the components can manifest in different ways in different team settings (Salas et al., 2005, p. 570).

This thesis uses the teamwork quality (TWQ) model by Hoegl and Gemuenden (2001) as used in Lindsjørn et al. (2016). It was chosen because it uses quantitative measurements to investigate how the parties involved in a team (team members, team leaders, and product owners) rate the teamwork, team members' success and team performance. The division of the model into constructs and subconstructs allows for investigation of the variables inside these constructs -such as communication, coordination and mutual support.

2.2.1 Teamwork Quality Model

The teamwork quality (TWQ) model investigates the collaboration in teams through focusing on the quality of its interactions (Hoegl & Gemuenden, 2001, p. 435-436). The hypothesis for this model is that “TWQ is positively related to the success of innovative projects" (Hoegl & Gemuenden, 2001, p. 439). While the original study by Hoegl & Gemuenden in 2001 was conducted on traditional software teams, Lindsjørn et al. conducted a replication of this study in 2016 that focused on the effect TWQ had on performance and team members success in agile teams (p. 275).

TWQ is the overall perception of how well the teamwork in a team is perceived. The TWQ is regarded through six subconstructs, detailed in table 3. Teams that collaborate well are assumed to practice behavior related to these subconstructs (Hoegl & G, 2001, p. 436). Teamwork quality is also connected to the constructs of *team performance* and *team member’s success*, which together can account for the success of a project. Constructs and subconstructs related to the model are described in table 3.

Data collection for this model utilizes a survey where the respondents rate statements on a lickert scale, and each statement is connected to a subconstruct. The number of statements for each construct and subconstruct can be seen in the parentheses in table 3. The respondents have one of three roles in the team; team member, team leader, or product owner. The team members will rate statements connected to all three constructs, while product owner and team leader will only rate statements connected to the construct *team performance*.

Construct	**Subconstruct**	**Description**
Teamwork Quality (38)	Communication (10)	Degree of frequent, spontaneous, and open communication.
	Coordination (4)	Degree of structured and synchronized efforts within the team. The team makes decisions; estimates, prioritizes, and delegates tasks in particular.
	Mutual Support (7)	Degree of team members helping and supporting each other in their work.
	Effort (4)	Degree of the team members effort on the tasks of the team.
	Cohesion (10)	Degree of interactions among the team members, and their motivation to maintain the team.
	Balance of Member Contributions (3)	Degree that team members contribute with their expertise.

Construct	Subconstruct	Description
Team member's success (8)	Work Satisfaction (4)	Degree to which team members are motivated to participate in future team projects.
	Learning (4)	Degree to which team members learn social, project management, technical, and creative skills.
Team performance (15)	Effectiveness (10)	Degree to which the team meets expectations regarding quality of the outcome.
	Efficiency (5)	Degree to which the team meets expectations regarding time, cost, and adherence to schedule and budget.

Table 3:TWQ constructs based on Lindsjørn et al. (2016, p. 276) and Hoegl & Gemuenden (2001, p. 437)

The empirical results from Hoegl & Gemuenden showed that there was a positive connection between the different measures of TWQ and project success (2001, p. 446). Project success is seen through the constructs of *team performance* and *team member success*. When both team performance and team member success were highly rated in a team, this would positively influence the six subcategories in the TWQ construct (Hoegl & Gemuenden, 2001, p. 446). This relationship is shown in figure 1. Lindsjørn et al. found that the TWQ variable with most effect on team performance among the agile teams was *mutual support* (2016, p. 281).

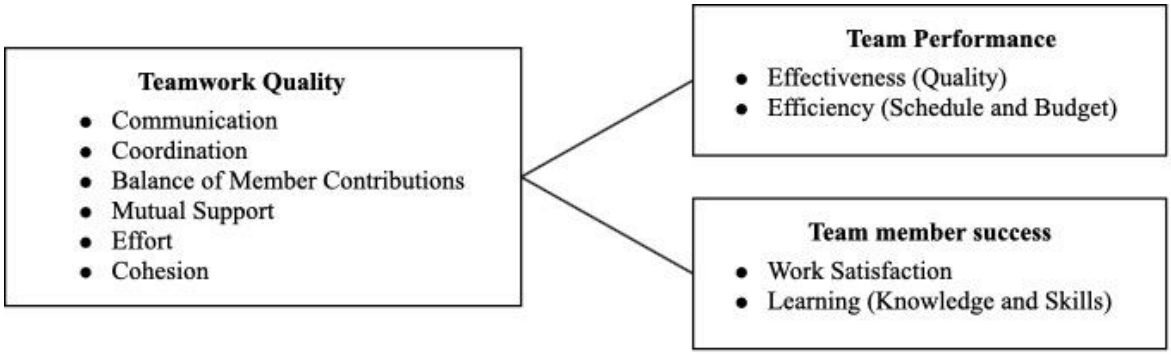

Figure 1: Teamwork Quality & Success (Hoegl & Gemuenden, 2001, p. 439)

Hoegl & Gemuenden found that those who work together on a project had a higher degree of agreement in their ratings (like team leaders and team members), than those who are further removed from one another (like team members and managers) (2001, p. 443). While this was seen in the study on the agile teams as well, Lindsjørn et al. experienced that there was a lower degree of agreement between the roles in the agile teams than in the original study (Lindsjørn et al., 2016, p. 279). The results from the two TWQ studies were similar, with the original study scoring slightly higher on the TWQ variables (Lindsjørn et al., 2016, p. 281).

Lindsjørn et al. had expected that the agile teams would both rate TWQ higher and that this would affect the team performance more, as there is a heavy focus on communication between the roles in agile teams (Lindsjørn et al., 2016, p. 279). One explanation to why this was not the case might be that agile teams today expect more from teamwork than what the teams in the original survey did (Lindsjørn et al., 2016, p. 281).

TWQ cannot entirely measure the quality of collaboration in a team; there are several other aspects in a project that may be of importance to team performance (Hoegl & Gemuenden, 2001, p. 446). However, the TWQ construct can be used to estimate the quality of teamwork within teams (Hoegl & Gemuenden, 2001, p. 446), which is an important factor for "improving team performance, especially for improving the quality of the team's product" (Lindsjørn et al., 2016, p. 282). The quality of a team's product is seen through the variable *effectiveness*. It may also provide insight into how the team members, and other key roles involved, perceives the teamwork and team performance.

2.3 Software Engineering Education

Software engineering education covers many areas and principles important to a student of Information Technology. For many new students there is a "misconception that software development in general is equivalent to coding" (Iacob & Faily, 2018, p. 163). Programming is essential in software engineering, but software engineering also covers everything taught to the students ranging from processes, requirements engineering, design, tools, planning, time estimation, testing, information security, communication, teamwork, version control, and more. The aim of software engineering education is for the students to understand the value of these other areas and skills that are necessary to successfully design, develop and maintain software. In this thesis the focus will be on software engineering education that provides the students with a project case that has to be solved within a team over a semester or more.

Several universities provide a mandatory introductory course to software engineering during the first or second year of their bachelor's degrees within Information Technology. This is especially useful when going from small programming tasks common to the first year of bachelor studies, to larger and more complex programming tasks in the final years (Sedelmaier & Landes, 2015, p. 418; Chatley & Field, 2017, p. 118).

2.3.1 Student Projects in Software Engineering Education

As software development is not a solitary pursuit, there is an inherent need for the development of collaborative skills in students of software engineering. Collaborative skills are one of the abilities referred to as soft skills, defined by Oxford English Dictionary as the "abilities which enable effective communication and social interaction with other people" (2020). In addition to problem-solving skills, graduates of software engineering need to possess the soft skills required for teamwork to be successful in the professional industry (Abad et al., 2019, p. 208). Being able to participate in a collaborative project in a team setting during the bachelor's degree is essential for both the students and the industry they plan on entering (Iacob & Faily, 2018, p. 163).

As a continuation of the introductory courses for bachelor students, it has become popular to offer students courses centering around a larger software development project spanning a semester or more. These courses focus on project-based learning and can close some of the gap between what is taught at universities and what is needed in the industry by bringing "students closer to practice by means of a real project (Fioravanti et al., 2018, p. 806). These project-based courses should be based on realistic problems that can simulate the professional settings of software engineering (Abad et al., 2019, p. 208). Table 4 provides an overview of the team size and project type among other things from research on student software development projects.

Study	Team size	Project	Length	Prerequisite	Year
Holmes et al. (2018)	3-10	Pre-existing	1 year	Nominated by home university	3rd-4th
Alperowitz et al. (2016)	8-10	From scratch	1 semester	-	-
Zorzo et al. (2013)	5-6	From scratch	2 years	-	-
Iacob & Faily (2018)	5-6	From scratch	2 semesters	A bachelor course in programming	2nd
Paasivaara et al. (2018)	7-9	-	1 semester	Completed first year	2nd
Chatley & Field (2017)	5-6	From scratch	1 semester	Completed previous years	3rd
Delgado (2017)	4-6	From scratch	1 semester	Software engineering, programming	2nd

Table 4: Projects in other software engineering courses

While none of the projects detailed in table 4 are identical, they carry much of the same reasoning and purpose behind them; having the students complete a software development project using "agile process models and using appropriate tool support for effective teamwork" (Delgado et al., 2017, p. 78). This provides the students with a hands-on experience that can be transferred to the industry.

In modern software engineering education, the opportunity for students to participate in a team project during bachelor's degree is crucial (Ju et al., 2018, p. 144). In these project courses the students have to draw on the knowledge they have acquired in previous courses and apply this theory to design, develop and/or maintain a piece of software in a team setting (Zorzo et al., 2013, p. 2). These software development project courses provide an arena for the students to practice their technical skills (Ju et al., 2018, p. 144), but because of the collaborative nature of such courses the students will also have to learn and adopt "the soft skills required for working as part of a team" (Iacob & Faily, 2018, p. 170).

Software development projects can teach the students the need for agile methods and structured processes in a way that only experiencing the software development process can (Iacob & Faily, 2018, p. 166; Delgado et al., 2017, p. 77). It also provides the students with many typical challenges related to real projects. These challenges can be collaborating with customers, figuring out the requirements for the software, working together as a team, and time management (Paasivaara et al., 2018, p. 51). The function of the software development projects is to motivate the students to see the importance for the use of software engineering methods and techniques, in addition to learning how to work in a team. These are areas that many students of software engineering view as both irrelevant and abstract prior to seeing them in action (Sedelmaier & Landes, 2015, p. 420. Iacob & Faily, 2018, p. 163). The reason for this might be that until students see and experience the practical application of the theory they are being taught, the theory is just that, theory (Fioravanti et al., 2018, p. 806).

There are significant gaps between what the industry needs and what is being taught at universities; agile methods, which are used in the majority of software development teams, is not adequately reflected in bachelor programs (Chatley & Field, 2017, p. 117-118), nor is teamwork in software engineering courses given enough importance (Iacob & Faily, 2018, p. 163). In a study it was observed that "current engineering curriculum is dominated by convergent, analytical work and passive knowledge acquisition" (Cropley, 2015, p. 163), which denies the students the opportunity to solve problems creatively.

While fundamental knowledge is important, students also needs to face situations where they need to draw on their collective fundamental knowledge to solve problems (Cropley, 2015, p. 163).

2.3.2 Team Composition in Education

In some courses the teams were self-organized, with 4 to 6 team members (Delgado et al., 2017, p. 79; Alperowitz et al., 2016), while in others they were formed by the instructors (Iacob & Faily, 2018, p. 169). The rationale behind instructor formed teams was that it would make the team setting more realistic, as in a professional setting the students would not be able to choose their own team (Iacob & Faily, 2018, p. 169). In a study it was observed that students were not overall happy with instructor only formed teams, causing discontent, lack of enthusiasm, and a belief among students that they would have performed better if they could have picked their own team (Iacob & Faily, 2018, p. 169). Another study found that the student formed teams performed only slightly better than the instructor formed teams (Løvold et al., 2020, p. 5), so a possible solution could be to go for a combination of student and instructor formed teams.

Another issue with student teams, especially when formed by the instructors, is lack of motivation or ambition among some team members. Fioravanti et al. found that it was an issue that some students "wanted to learn while others only wanted to complete a mandatory course" (2018, p. 810). This challenge with team composition is mirrored in other courses, where students felt that not all team members wanted to take an active part in the teamwork (Iacob & Faily, 2018, p. 169).

Instructor formed teams have the advantage that they can ensure multidisciplinary teams (Iacob & Faily, 2018, p. 169). However, other studies have shown that students to some degree can manage this themselves (Delgado et al., 2017, p. 77-78; Løvold et al., 2020, p. 3). Whether this is because the students can see the value of a multidisciplinary team or because they are encouraged to form multidisciplinary teams is unknown.

2.3.3 Challenges

While there are many challenges related to project-based courses, two central challenges are; 1) that project-based courses are complex to execute, and 2) there is a difference between student projects and real-life projects. Project-based software engineering courses often come with much complexity (Ju et al., 2018, p. 144). Part of this is because of "the *high number* of simultaneous projects and the *varying nature* of the projects" (Alperowitz et al., 2016, p. 323), especially if the course has a high number of enrolled students. Being able to give feedback to the teams is a challenge in larger courses with teamwork (Chatley & Field, 2017, p. 117). A solution for these issues is to use teaching assistants to help with the workload and act as mentors or supervisors for the student teams. These are often students who have previously completed the course and are familiar with the technology and structure (Krusche et al., 2017, p.92), but in some cases they are hired developers (Holmes et al., 2018, p. 32).

There will be an inherent difference between a student project in an educational setting and a professional team in a real-life work setting. To accommodate for student teams, Alperowitz et al. created a process model based on Scrum which is tailored for students working on the project part time (2016, p. 323). In most of the projects, the students are required to develop a product from scratch in a team consisting solely of other students. Thus, they do not experience what it is like to work with legacy code, having to adopt pre-existing processes and tools, or have a community of seasoned developers around them (Holmes et al., 2018, p. 32) Because the project cases often are curated for the specific setting of the course, they usually have no real outside customer. In the cases where Scrum is used, the role of the product owner is then covered by instructors or teaching assistants (Zorzo et al., 2013, p. 4). While this is a solution, the students will not experience how to communicate and coordinate with a real customer.

3 Research context

3.1 The Software Engineering Course

The research context of this thesis is the 20 ECTS software engineering course IN2000 "Software Engineering with Project Work" at the University of Oslo. This course will from now on be referred to as IN2000. The course was carried out for the first time in the spring semester of 2019, running from January to June. A pilot of the course had previous to this been completed in the spring semester of 2018. The pilot course is outside the scope of this thesis.

IN2000 is a project- and team-based software engineering coursed taught at bachelor levels at the Department of Informatics. The course has a strong focus on the use of modern methods, techniques and tools in software engineering. IN2000 can be seen as an extension of the mandatory introductory courses given in the first three semesters of the bachelor programs. The core introductory courses consist of programming, software engineering, and design principles to mention some. Many of the assignments in the introductory courses aim to provide the students with a general understanding of the fundaments of informatics, which is then built upon in later courses. By combining the learning outcomes from these courses, IN2000 teaches the students how to utilize what they have learnt in seemingly unconnected courses to complete an extensive system development project in a team.

The objective of the team project in IN2000 was for the students to develop a functioning Android application that met a set of requirements. This was to be achieved by working in an agile manner in a team setting over the course of twelve weeks.

For most of the enrolled students the project in IN2000 was their first encounter with teamwork and putting the theory of agile into practice. IN2000 takes place in the fourth semester for bachelor students enrolled in the three study programs 1) Programming and Systems Architecture (prosa), 2) Digital Economics and Management (digec), and 3) Design, Use and Interactions (design). For students enrolled in these study programs IN2000 is a mandatory course to complete their bachelor's degrees.

3.1.1 Learning Outcomes

The learning outcomes of IN2000 are focused on the non-programming aspects of Software Engineering, such as knowledge about agile principles, development methods, processes, and teamwork. The learning outcomes do not explicitly state anything about learning how to use Android Studio, the programming language Kotlin, or how to develop a mobile application. However, this can be seen in the context of learning outcome number 4 in table 5, that the students will be able to use professional system development methods, techniques, and tools.

After having completed IN2000 the student will:

1.	Have knowledge of the most important system development methods, including their strengths and weaknesses.
2.	Have knowledge of central processes and actors in project and teamwork that apply agile principles.
3.	Have knowledge of the following activities within system development: requirements collection and analysis, design, programming, testing, as well as maintenance and further development.
4.	Be able to use professional system development methods, techniques and tools.
5.	Have the competence to work in teams and the ability to reflect on your own and the team's work in system development projects.
6.	Have knowledge of the methods and principles of built-in safety and universal design.

Table 5: The learning outcomes of IN2000 (UiO, 2019)

The examination in the course took place at the end of the semester and consisted of both the delivery of the team project and a four-hour written digital exam. Each of these counted for 50% of the overall grade. The grade of the team project provided a common grade for all the team members, while the written digital exam provided an individual grade. Both the team project and the written digital exam had to be passed in the same semester, and in order to be eligible for the final written examination all mandatory assignments and presentations had to be approved. The course used grades from A to F, where A was the best possible grade and F was failed.

3.1.2 Structure

In 2019, IN2000 provided one to two lectures, each two hours long, every week for the first eight weeks. Attendance was only mandatory for the first lecture. The lectures set out to provide knowledge to the students in crucial themes for the course, they are presented in table 6. The lecture themes were changed in 2020, these changes are presented in chapter 3.4.

		Lecture
Week 1	1	Introduction to the course
	2	Introduction to the technology
Week 2	3	Teamwork, agile methodologies and project work
	4	Modelling and object-oriented principles
Week 3	5	Agile practices
	6	Secure System Development, Threat Modelling, and Built-in Privacy in
Week 4	7	Research Methods
	8	Requirements Handling
Week 5	9	Architecture and Technical Debt
	10	Basic Principles of Testing
Week 6	11	Main Architecture for Developing Android Apps
Week 7	12	How to Write a Long Report in a Team
	13	About the API from the Norwegian Meteorological Institute
Week 8	14	Universal Design

Table 6: Lecture structure spring 2019 (UiO, 2019)

Because IN2000 builds on the mandatory introductory courses, the lectures often provide a quick repetition of central aspects of the themes before focusing on the aspects that are specific to IN2000. In addition, there were five seminars each week for the entirety of the course, run by the teaching assistants in IN2000. Students could attend none to five seminars each week. During the first eight weeks of the course, the seminars covered the technical aspects required for the project work. They provided hands-on tasks for practice, tips and tricks for the technology, and help with the mandatory assignment. During the project the seminars functioned as a time when the teams could ask questions and receive help from teaching assistants.

The course had two mandatory assignments. The first was an individual technical assignment prior to the project start, and the second was a team-based project plan assignment two weeks into the project.

3.1.3 Admissions

In the spring semester of 2019, the admission to IN2000 was restricted to the students enrolled in one of the three bachelor programs in which the course was mandatory. This meant that in 2019 there were only students from Programming and Systems Architecture (prosa), Digital Economics and Management (digec), and Design, Use and Interactions (design). These bachelor programs had IN2000 as a mandatory course in their fourth semester for the completion of their bachelor's degree. In addition to being enrolled in one of the three bachelor programs mentioned above, the students needed to have completed the prerequisite courses prior to enrolling in IN2000. First, the students needed to have completed the introductory courses in object-oriented programming, which covered both Python and Java. Second, they were required to have completed the introductory course to software engineering. All of these courses took place during the first year of their bachelor programs. Third, the students had to complete an elective course in either algorithms, databases, user-oriented design, or intermediate interaction design.

While there are some elements of groupwork in several of the required courses leading up to IN2000 and other elective courses, several of the courses for the design bachelor are specifically focused on teamwork.

3.1.4 The Technology in IN2000

The technology used in the course was new to the vast majority of the enrolled students. In previous courses the students have been familiarized with both Python and Java, but without the use of an integrated development environment (IDE). In this course, the students have to learn a new programming language, Kotlin, and use it with the IDE Android Studio. This IDE is specifically aimed at developing high-quality Android apps and offers built in tools for developing for Android. The students are also introduced to, and required to use, the version control system Git and host their Git repositories on GitHub. Android Studio provides built in Git integration. In order to get the students familiarized with the technology prior to the project start they had to complete a mandatory assignment focused on Kotlin and Android Studio.

The abovementioned technologies and tools were mandatory to use in the project. Additionally, the students were encouraged to adopt other tools aid them in communication and coordination throughout the project. The retrospective tool Evetro was presented in a lecture, while other tools (such as Trello and Slack) were mentioned in the seminars. There were no recommended technologies or tools apart from the ones that were mandatory.

3.1.5 Project Cases

The project cases were specifically designed for the course and were based on the data available in the Application Programming Interface (API) provided by the Norwegian Meteorological Institute. For each case there was a short explanation of the case, a few suggested functionalities, and an overview of the relevant API resources for the given case. Depending on the project case, the students would use either api.met.no or frost.met.no, or both. The frost API provides historical data relating to weather. The projects came with no pre-made code, the teams had to develop their Android application from scratch.

Case 1 – The Weather at Sea: Weather forecasts on the sea ahead of time, based on speed, distance and position. The weather forecast includes wave heights and wind directions. This case requires the use of api.met.no.

Case 2 – Lightning Alert: A map based alert system for both lightning and thunderstorms showing the nearby lightning activity. Provide notifications for lightning in a user decided area, and when user is entering an area where lightning is forecasted. This case requires the use of both api.met.no and frost.met.no.

Case 3 – Flight Planning: Visual representation on map to show the weather conditions for specific airports and weather forecast between two airports. Includes the probability for weather that allows for visually flying (VFR) from an airport. This case requires the use of api.met.no.

Case 4 – Air Qualities in the Cities: Provides detailed information about current and expected air quality on a map. Notifications when the air quality is lower than a user-defined limit. This case requires the use of api.met.no and if desired frost.met.no can be used in addition.

Case 5 – The Changing Climate in Norway: Graphical presentation of both data on climate collected over the last 100 years, and the consequences of climate change in regard to weather. Can also include future forecasts based on different scenarios. This case requires the use of frost.met.no.

The teams could freely choose between five predefined project cases, and there was no limit placed on how many teams could choose the same case. All five project cases had three to four requirements to get the teams started. It was expected that the students would seek out more requirements for their project case as the first step of the project work. This would be done through surveys and interviews with the user group for their specific case.

3.2 Facilitation for Agile

There is a strong focus on agile methods in the course. From the first lecture the students are told that they are expected to follow and work with agile methodology in the project. One of the prerequisite courses for IN2000 was introduction to software engineering, a course which also put a lot of emphasis on agile methodologies and the difference between traditional methods in teamwork and agile methods.

The course was for many students their first meeting with teamwork and agile in practice. Because of this, the course needed to facilitate for agile. In the beginning of the course, this was done through incorporating agile into all the lectures and showing the connection between the topic and agile. The students were thoroughly introduced to process models, especially Scrum, and agile planning during the first weeks of the course.

3.2.1 Working Agile in the Teams

Scrum was the main process model covered in the lectures, with both Kanban and XP being mentioned as well. The students were expected to utilize agile practices throughout their project and decide on a process model to follow. There was, however, no required way of working agile in the course. As part of working in an agile manner is being able to identify which methods and practices will benefit the team, this decision was left up to the students.

The second mandatory assignment was aimed at helping the students with planning their project and to start thinking about how to execute the project in an agile manner.

The teams were encouraged to reach out to the teaching assistants for guidance. In this assignment the teams presented their overarching project plan, with proposed sprint lengths and tasks designated to specific sprints. The assignment also asked how often the team met, what the temporary requirements to the project case were, and a summary of a retrospective meeting. The teams were also asked about their process model, and why they believed this would benefit the team.

Regardless of which process model the student teams chose, the instructors and teaching assistants strongly recommended the teams to meet physically as often as they could. Early in the semester the students were presented the four meeting types typically seen in scrum; sprint planning, daily stand-up, sprint review, and retrospectives. Ideally, the teams would incorporate all of the meetings to some degree to get a feel for them and then decide to continue with them or not.

When considering the project cases, it would have been easy to add comprehensive lists of requirements for the project cases described in section 3.1.5. A long list of requirements detailing exactly how the application should function and look would make the project unrealistic, and the students would have been deprived of important lessons of software engineering. Instead, the decision was to offer project cases that were both open for interpretation but also in need of further work before the programming could start. This forced the students to consider user groups for their applications, the requirements, how the architecture would ideally be, and design mock-ups early in the project. By allowing the students to do this themselves, they had to make an overarching estimate of how much time was required to finish tasks, prioritize tasks, see dependencies in their workflow, and allow for change of plans when things took longer time than expected or when there were issues that needed to be resolved.

3.3 Team composition

In the first lecture of the course, the students were presented with two options for assembling teams. They could either assemble a team on their own or be placed in a team by the instructor. There were three factors the students were asked to consider when assembling teams: First, if everyone in the team had the same level of ambition when it came to project grade, and if they had similar capacity for the workload in terms of weekly hours spent on the course. Second, if there were any team members who could not meet at fixed times due to work, volunteering, a heavy course load that semester, or other reasons. Third, if there was variation in both the study programs and the genders of the team members.

All students had to fill out a spreadsheet to either register their self-assembled team or to register that they wished to be placed in a team. The students who assembled their own teams had to register the names, usernames, and bachelor programs of their team members. The students who wished to be placed in a team had to respond to two additional questions; 1) what their desired grade in the course was, and 2) if there were any specific times that they were unable to meet with their team. Once the teams were established, git repositories on GitHub were created for them. Here both the instructors and the teaching assistants had viewing rights, which enabled them to follow the progress of the teams.

3.4 Changes made to the course

While the focus of this thesis is on the students enrolled in IN2000 in the spring semester 2019, it is worth noting some of the changes that has been made to the course from 2019 to 2020. The changes to the course were made from our experience with running the course in 2019, student feedback, and through findings related to this thesis. By the time this thesis was handed in these changes had been in effect for six months. There are three main changes;

1) The teams are primarily created by the instructors.

2) An increase in teaching assistants for the course.

3) Stronger focus on the technical aspects of the course in the beginning of the semester.

To make the course more realistic in terms of real-life work situations it was decided that the teams for the 2020 execution of IN2000 were to be primarily created by the instructors.

The same conclusion has been reached in other courses, like the course discussed by Iacob & Faily (2018 p. 169). The aimed team size was 5 to 6 students. We decided to allow for up to 3 students to sign up together, and the rest of the team would be chosen by the instructors. The students could also sign up alone. When signing up for the teamwork project all students had to specify which study program they were enrolled in and which grade they were aiming for. There was also room for additional information where students could specify if they worked a lot, took extra courses, could only meet during evenings or weekends, or if there was another person they wished to work with. This feature was used by some students to create their own teams entirely, but it was decided that we would not fight these students on this. However, only five teams out of 42 ended up being entirely student-made.

There were some challenges in terms of getting an even distribution of students from different study programs and of different genders in all the teams. In the future the team composition could be done entirely at random by the instructors. This would allow for a more even distribution of both study programs and lead to more multidisciplinary teams.

Due to this new way of creating the teams for the course, a kick-off day was introduced. As 37 of the 42 teams were created by the instructors, this was meant as a helping hand for the teams to meet and start thinking about the project. The focus was on team building exercises as several teams most likely had team members that did not know each other from before. In the interviews several of the students expressed that one of their main challenges was getting started, and one of the teaching assistants experienced that one team didn't start until three weeks into the project period.

Another change made was getting more teaching assistants for the course. In 2019 there were five teaching assistants who each had responsibility for following up seven to eight teams during the duration of the course and holding a two-hour seminar each per week. The number of teaching assistants was increased to nine in 2020. This increase led to a decrease in the number of teams each teaching assistant was responsible for. In 2020, they were responsible for following up four to five teams each. This would allow the teaching assistants to follow up their teams more closely, while the teams would receive responses to their questions quicker. It was also decided that the teams that were lacking students from one of the study programs would get a supervisor with proficiency in that field. For example, the teams with no designers got a teaching assistant who is a designer, and the teams with only one or two programmers got a teaching assistant who is a programmer.

The five two-hour weekly seminars continued in 2020, but this time with two teaching assistants for each seminar.

The last change made was a stronger focus on the technical aspects of the course in the beginning of the semester. In 2019 there was only one lecture at the beginning of the semester that introduced Android Studio and Kotlin, with the rest of the technical teachings being covered in the weekly seminars. In 2020, a technical lecture-series was added at the beginning of the semester, consisting of four lectures focused on the technology used in the course. This is shown in table 7.

		Lecture
Week 1	1	Introduction to the course
	2	The Basics of Android Studio and Kotlin
Week 2	3	More on Android Studio
	4	More on Kotlin
Week 3	5	API, data formats, HTTP-requests and Proxy-servers
	6	Teamwork, agile methodologies and project work
Week 4	7	Agile practices
	8	Basic Principles of Testing
Week 5	9	Secure System Development
	10	Modelling and object-oriented principles
Week 6	11	Architecture and Technical Debt
	12	From Theory to Practice – the project from A to Z
Week 7	13	Application Programming Interface (API)
	14	Development of Android apps and use of patterns
Week 8	15	Universal Design
	16	Evaluation Method / Research Methods

Table 7: The changed lectures for 2020 (UiO, 2020)

The weekly seminars summarized the lectures, and more technical tasks were created for the students to complete either in the seminars or on their own. In 2019 there was only one mandatory technical assignment before the start of the project, in 2020 a second mandatory assignment was added that focused on API's. The goal behind this assignment was that every student enrolled in the course would know how to retrieve information through an API call. To reflect these changes, it was also decided that more points would be rewarded for the code in the project than in 2019.

To investigate how these changes to the course might have affected the teamwork quality, a TWQ-survey has been conducted for the students enrolled in the course the spring semester of 2020. However, due to the drastic measures invoked at the University of Oslo to contain the spread of the COVID-19 there are several considerations that must be taken into account when analyzing the data. The University closed its campus in the beginning of March, sending both students and employees home, and moved all teaching and project work to digital platforms. In addition, the course became a “pass/fail” graded course, with no project points to use for comparison.

4 Research Methods

4.1 Research Design

A case study was chosen as the research methodology for this thesis. This research methodology is useful when approaching research questions that look at "initiatives or innovations to improve or enhance learning or teaching" (Case & Light, 2011, p. 191). Case studies do not always set out to prove or disprove a certain hypothesis or theory and are much more exploratory in nature. In contrast to some other methodologies the focus is rather to get new insights on a specific case and gather empirical "evidence about what is going on" (Robson & McCartan, 2016, p. 150). This is particularly the case for exploratory case studies where the goal is to find out what is happening (Runeson & Höst, 2008, p. 135). Based on this, a case study was a good match for this thesis, as the aim is to study a contemporary phenomenon; namely teamwork in student teams. This is done in the real-life context of a software engineering course using mixed methods for data collection (Robson & McCartan, 2016, p. 150). The framework for research design in this thesis is outlined in table 8.

Component	
1. Purpose	The case studied was a software engineering course, with a larger software development project and teamwork. This thesis seeks to understand how teamwork functions in student teams, and what characterizes the high-performing teams. The results will be able to provide insight on how to facilitate high-performing teams, and how to conduct a large software engineering course with a semester-long teamwork project.
2. Theory	This thesis is guided by theory from software engineering education, agile teamwork and teamwork models.
3. Research Questions	The research aims to offer insight on the characteristics of the high-performing teams in a student project, and if the quality of teamwork is related to the overall project grade.

Component	
4. Data collection	Mixed methods were used to collect data. The data collection consisted of a survey, semi-structured interviews, and documents, including the project reports from the teams. The use of methodological triangulation will ensure the trustworthiness of the data (Rogers & Preece, 2011, p. 225).
5. Study sample	Data was collected from students eligible to take the exam in the course in the spring of 2019, and its teaching assistants. First, a survey was conducted including most of the students and all of the teaching assistants. Then interviews were conducted with a small selection of students from different teams and with the senior teaching assistants.

Table 8: The framework for research design in this thesis (Robson & McCartan, 2016, p. 72-73)

4.2 Survey

To collect quantitative data for this thesis I have used the survey from Lindsjørn et al. (2016). There are several advantages to conducting surveys, for instance that they provide a straightforward approach to study attitudes and beliefs, and allow for data standardization (Robson & McCartan, 2016, p. 248). However, there are some disadvantages that are relevant for this survey; respondents might not respond accurately, and the data is affected by characteristics such as the respondent's memory or motivation (Robson & McCartan, 2016, p. 248).

The survey was slightly altered to accommodate for the difference in the settings of the respondents; the respondents of the original survey were professional agile teams, while the respondents in my survey are students who have not had a customer nor dealt with budgets in regard to their project work. The survey questions are listed in Appendix I. The survey was cross sectional, as the respondents answered based on their recent experience with teamwork in the course (Kitchenham & Pfleeger, 2008, p. 68). The survey took approximately 15 minutes to answer.

4.2.1 Study sample

The study sample consisted of two groups; the students enrolled in the course in the spring of 2019, and the 5 teaching assistants who acted as supervisors for the teams. Out of the approximately 200 enrolled students, 196 participated in this study.

The questions were aimed at three roles; team members, scrum masters, and product owners. The students filled the first two roles, with one scrum master selected from each team based on responses on the background question of the survey. The product owner segment of the survey was answered by the teaching assistants. It should be noted that the involvement of the teaching assistants in the teams varied greatly, which makes this part of the data less reliable.

4.2.2 Data collection

The data collection from the students was conducted during the student's final presentations of their projects. This took place over two weeks in the middle of May 2019. The presentations were held in the period between project completion and the final written exam. At each presentation the survey was presented, and its purpose explained to the students. It was emphasized that their responses would not have any effect on their final grade in the course, and that they would be anonymous. All students who were present at the presentations responded to the survey. The survey was handed out on paper, answered individually, and was then collected at the end of the presentations. The teaching assistants received the survey after assessing the student projects, in the beginning of June 2019.

The students were asked to rate their personal agreement to the statements in the survey on a lickert scale from strongly disagree (1) to strongly agree (5), and to not contemplate what the rest of the team would respond to them as the survey was to be completed individually.

4.2.3 Data Analysis

The data was analyzed using the built-in formulas and graphs for illustration in excel. The latent variables and variables used in the analysis is shown in table 9.

Answered by	Latent variables	Variables	Items
Team Members	TWQ	Communication	10
		Coordination	4
		Mutual Support	7
		Effort	4
		Cohesion	10
		Balance of member contribution	3
	Team member's success	Work Satisfaction	4
		Learning	4
Team Members Scrum Masters Product Owner	Team Performance	Effectiveness	10
		Efficiency	5

Table 9: Latent variables and variables (Lindsjørn et al., 2016, p. 278)

After the data was collected as described in section 4.2.2. the responses from the survey's lickert scales were entered numerically into excel, where *strongly disagree* represented 1 and *strongly agree* represented 5. Some items in the survey had to be rephrased, and others had to be

coded in reverse. Both of these are highlighted in Appendix I.

Responses pertaining to the scrum master role were filtered out from the answers of the team members. There were 19 instances where more than one person in a team viewed themselves as scrum master. In these cases, the first registered response indicating the scrum master role has been selected as scrum master for the team. Once this was done, only the two latent variables referring to team performance were regarded for the 39 scrum masters. The data was upon entry to excel entered into three different datasets with regard to roles; team member, scrum master, and product owner. The responses were connected across all datasets by using the team number as an identifier.

The results were grouped by subconstructs from the TWQ model. In this section subconstructs will be referred to as variables. For each response, the average was calculated of the ratings given to each of the statement belonging to the different variables, as shown in table 9. This meant that for the variable *communication*, the average would be calculated using the ratings of its 10 connected statements. Appendix I provides a detailed overview over which statements the different variables (subconstructs) consisted of. This was done for each of the responses in the three datasets.

Following this the averages in the 4th dataset was calculated on a "team-by-team" basis, where the team was viewed as the unit of analysis. First the average was calculated using the results from each team in dataset 1, this entailed calculating the average rated response to for example *mutual support* from all of the team members in a given team. For dataset 2 and 3 this was not necessary to do as there for each team was only one scrum master and one product owner. Table 10 provides an overview of the datasets. A few of the surveys were partly incomplete upon delivery. These have been included in the survey and have been taken into account in the analysis. The data in the survey has been analyzed calculating statistics and comparing mean values; analyses that are suitable for datasets with some incomplete responses (Kitchenham & Pfleeger, 2008, p. 88-89).

Dataset:	**Responses:**	**Content:**
1. Team Member Responses	196	Teamwork Quality Team Member's Success Team Performance
2. Scrum Master Responses	39	Team Performance
3. Product Owner Responses	39	Team Performance
4. Aggregated Responses on Team Level	39	Mean of responses from dataset 1-3

Table 10: Overview of datasets

The 4th dataset has been the one frequently used for analysis of the collected data in this thesis. To find the Pearson's correlation coefficient between datapoints I wrote a script in Python using the libraries pandas and matplotlib. The script is presented in appendix V. The correlations have been interpreted using a conventional approach outlined in Schober et al. (2018, p. 1765).

4.3 Interviews

To collect qualitative data for this thesis, 8 semi-structured interviews were conducted. The use of an interview is a "flexible and adaptable way of finding things out" (Robson & McCartan, 2016, p. 286) as it allows for modification of both wording of question and the order questions are asked. Most importantly, interviews allow for follow-up questions that were not planned in advance and generate valuable insight into the topic being examined; this is common in semi-structured interviews (Rogers & Preece, 2011, p. 220-230).

The aim of the interviews was to get in-depth information from a selection of the people involved in the course. The interviewees were involved in either the team projects as students, or guidance of the teams as teaching assistants. The interviews where held over Zoom approximately a year after the project ended in 2019.

4.3.1 Study Sample

To find interview subjects among the students who had taken the software engineering course in 2019, a request to participate in an interview focusing on their experiences with the course was sent out to a few students. The selection of students to interview was based on the following criteria; first that they represented different teams, and second that they had passed the course. In the end, six students were interviewed. This is only a small selection of both students and teams. However, the students interviewed had a good representability in regard to study program, gender, and team composition.

Out of the six interviewees, three were enrolled in the *prosa* study program and three were enrolled in the *design* study program. I was unfortunately not able to get in touch with a student enrolled in the *digec* study program. The interviewees were well-balanced in terms of gender, three of the interviewees were men and three were women.

The teaching assistants were selected based on their seniority as teaching assistants within the course, with the added benefit that they themselves had participated in the pilot version of the software engineering the year before, in 2018. Two teaching assistants were interviewed, both teaching assistants are still involved with the course in 2020.

Interviewing six out of 197 students is not representative. The answers of the interviewees can therefore only provide insight in the interviewees own personal experience with the course, their team, and the project work. Even if it is not representative, the interviews can shed light on different aspects and experiences with the course. This was after all the aim of the interviews.

4.3.2 Data Collection

The interviews were conducted over a two-week period at the end of April and the beginning of May 2020. These are shown in table 11. Each interview took about 39 minutes, varying from 32 to 42 minutes.

Interviewee	Team	When	Duration
Student 1	Team A	April 2020	42.21
Student 2	Team B	April 2020	39.24
Student 3	Team C	April 2020	40.24
Student 4	Team D	May 2020	36.34
Student 5	Team E	May 2020	39.41
Student 6	Team F	May 2020	44.16
Teaching Assistant 1	Supervised team D and E.	May 2020	32.08
Teaching Assistant 2	Did not supervise any of the interviewees.	May 2020	39.35

Table 11: Overview of the interviews

The interviews were semi-structured. Interview guides were prepared prior to the interviews with open-ended questions. I prepared two interview guides, one for the students and one for the teaching assistants (see Appendix II and Appendix III). The interviewees did not gain access to the interview guides. The questions functioned as the starting point for the conversation and the as a guide for the interviews to make sure central topics were covered. A semi-structured interview allows for asking additional questions in order to go into greater detail, and for the interviewee to add other information (Robson & McCartan, 2016, p. 285).

During the interview thorough notes were taken. These notes included the word-by-word answers and quotes from the interviewees. When additional clarification was required after the interview, I reached out to the interviewee either via email or messenger.

4.3.3 Data Analysis

The interviews were analyzed using thematic analysis, as described by Braun and Clarke (2006). They define thematic analysis as a method of identifying, analyzing and reporting patterns within data, which are called themes (Braun & Clarke, 2006, p. 6). A theme reflects something significant about data that is able to answer the research question. The final set of themes should structure the data and represent it in a meaningful way.

Following each interview, the notes and citations were examined to search for obvious patterns, and to check if there was need for clarification from the interviewee. The first elements that were sorted into categories were those concerning background information, like the team size, team members, grade, and study program of both the interviewee and his or her team members. The interviews were not transcribed in the strict sense of the word, but the benefit of holding interviews digitally over Zoom is that one is able to type quickly while the interviewee is speaking, while still being present with the interviewee.

The interview guides provided the central themes, leading to most of the information from the interviews already being sorted into various categories. The central themes were slightly different for the students and the teaching assistants, and due to their different roles in the course they have been analyzed separate from each other.

For each interview I went through the transcript in search of patterns and repetitions within the central themes. Statements were color-coded based on their content. After all interviews had been color-coded, the statements were grouped together based on the different colors. These groupings then became the basis for sub-themes, where some sub-themes consisted of several groupings. The central themes and sub-themes for the interviews are listed in table 12.

Interviewee Role	Central Theme	Sub-Theme
Student	Team Project	Communication Coordination Mutual Support Effort Motivation Member Contribution
	The Course	Teamwork Learning Impression Worklife
Teaching Assistant	The Teams	Role Teams
	The Course	Important Aspect Student Learning Challenges

Table 12: Central themes and sub-themes in interview

For the student interviews, the content of the groupings was analyzed against the six subconstructs of TWQ, they are detailed in table 3, in section 2.2.1. These subconstructs became the sub-themes for the central theme *team project*. However, the term *cohesion* did not capture the responses in the interview, instead a sub-theme called *motivation* was added. After this analysis each sub-construct had at least three groupings connected to it, these groupings were then named based on their content. The central themes of the student interviews can be seen in figure 2 and figure 3.

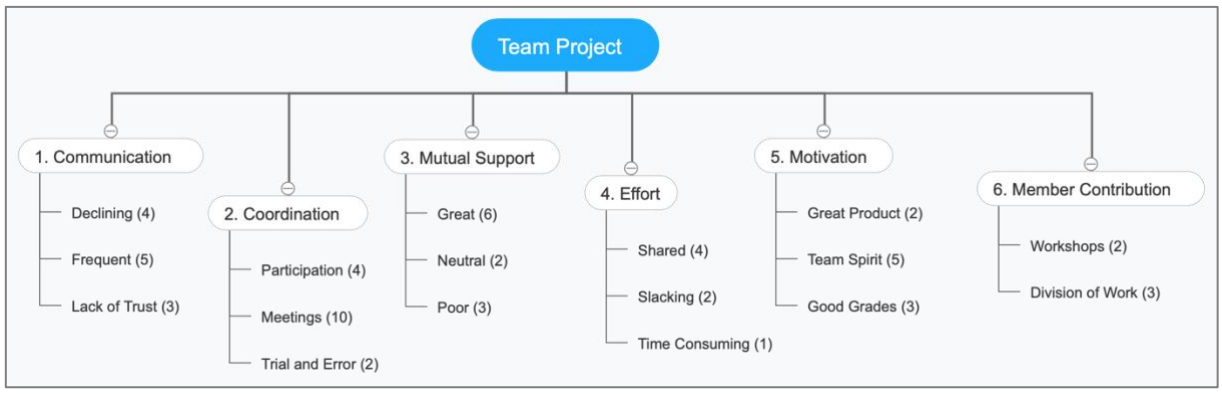

Figure 2: Sub-themes for "team project" from the student interviews

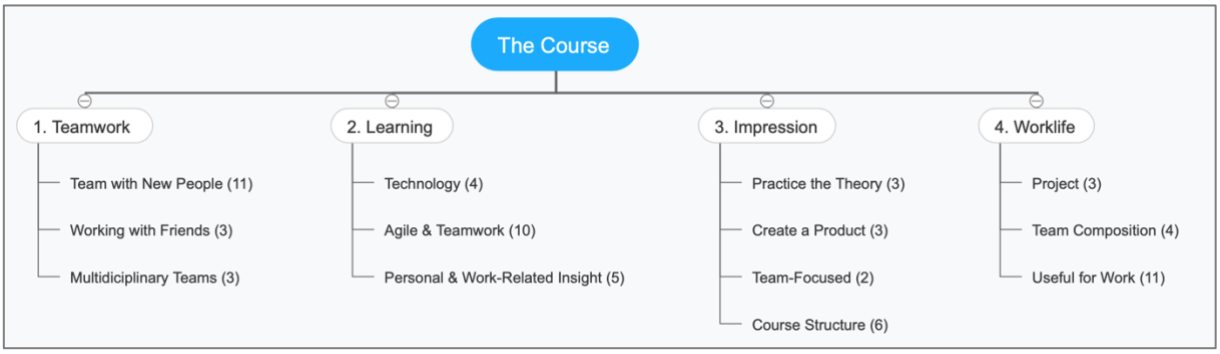

Figure 3: Sub-themes for "the course" from the student interviews

All interviews were conducted in Norwegian, and the citations were later translated to English for use in this thesis. This may have caused meaning to get lost or changed in translation. Translating the citations also runs the risk that I might have provided the citations a different meaning than what the interviewee intended.

4.4 Documents

In my work on this thesis I have relied on the use of several types of documents to understand the research context. The documents used for my research are presented in table 13. I have used the documents as a supplement to the survey and the interviews.

Documentation	**Description**
Documents	Team Reports Mandatory Assignments Project Grades Lecture Slides
Presentations	Lectures Student Presentations
Pictures	Illustration of Android applications

Table 13: An overview of the collected documentation

The team reports have particularly aided my understanding of how the teams worked together. There were 39 team reports, one from each team in the course. The reports were analyzed using thematic analysis, as described in section 4.3.3.

5 Results

5.1 Project Cases

"The project cases were well thought out, not just thrown together.
There was a careful consideration behind them,
and it was possible to solve them over one semester"
- Student 4

The student teams chose freely between five predefined project cases, described in section 3.1.5. The cases were perceived as well thought out by the students, and as such it was expected that there would be an even distribution of the cases chosen by the teams. However, out of the five project cases there was one project case that stood out; case 4. Air Quality in the Cities was chosen as a project case by 56.4% of the teams. This made it the undoubtedly most popular case compared to the others, as show in table 14.

Project Case Spring 2019:	**Number of teams:**	**% of teams:**
Case 1: The Weather at Sea	5	12.8%
Case 2: Lightning Alert	7	18%
Case 3: Flight Planning	2	5.1%
Case 4: Air Quality in the Cities	22	56.4%
Case 5: The Changing Climate in Norway	3	7.7%

Table 14: Project cases and how many teams chose them.

Air Quality in the Cities was the project case for five out of the six interviewed students. In the interviews it was pointed out that being able to create a useful product for a clearly defined user group was important. One student said that "*it was exciting to see that you could develop a product for a user group and have your product and ideas confirmed through the collection of data*" (Student 2), while another said that after the project "*you have something useful you can have on your phone to show people*" (Student 6).

There are three elements that account for the popularity of Air Quality in the Cities;

1) The data used for the Air Quality case provides updated measurements on a consistent basis, in contrast to particularly case 2, Lightning Support. This made the students perceive Air Quality as being easier to work with.

2) The potential user group for the application was perceived as being larger for Air Quality than for the other cases. It was also a focus point that this would make it easier for the teams to get in touch with potential users for data collection and user testing.

3) The students were eager to develop an app they could potentially benefit from, as they have a relationship with air quality by living in a city.

The project grade counted for 50% of the total grade in the course, the criteria for evaluation of the team project can be seen in Appendix IV. The distribution of grades based on the different cases is shown in figure 4. There was no team that scored lower than a C. The average project grade was B, with a point average of 42.07. As case 4 was the most popular case, it is no surprise that this is also the case with the majority of the grades.

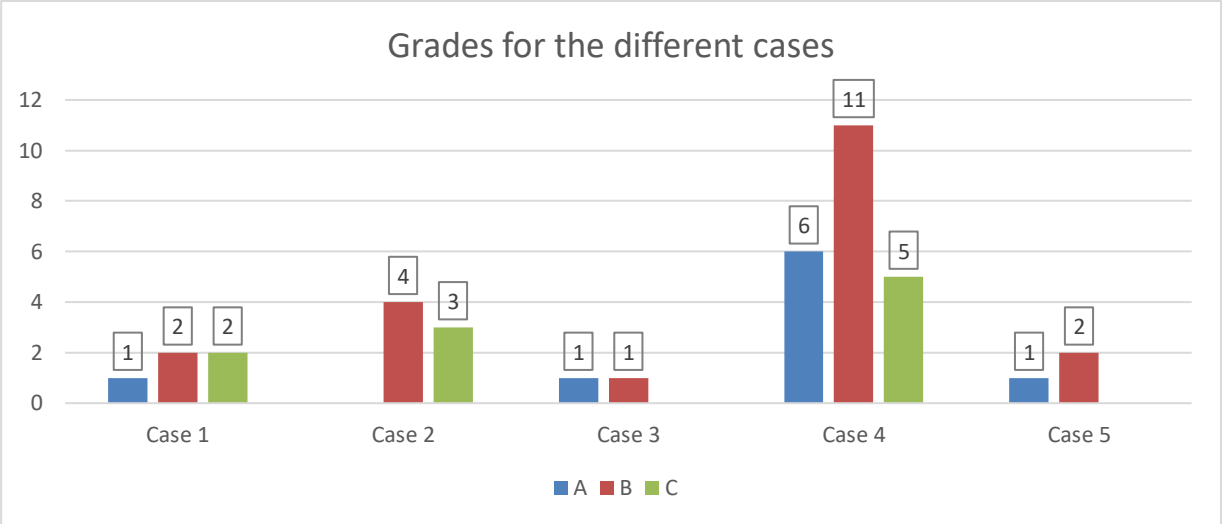

Figure 4: Distribution of alphabetical grades based on case

By looking at the average points given for the projects, we can see a clear difference between the project cases in terms of how the teams that chose them performed. This is shown in figure 5. Based on the average number of points achieved, case 2 (*Lightning* Alert) did the worst, scoring on average 2.5 points below the course average while case 3 (*Flight Planning*) did the best scoring on average 3.43 points above the course average.

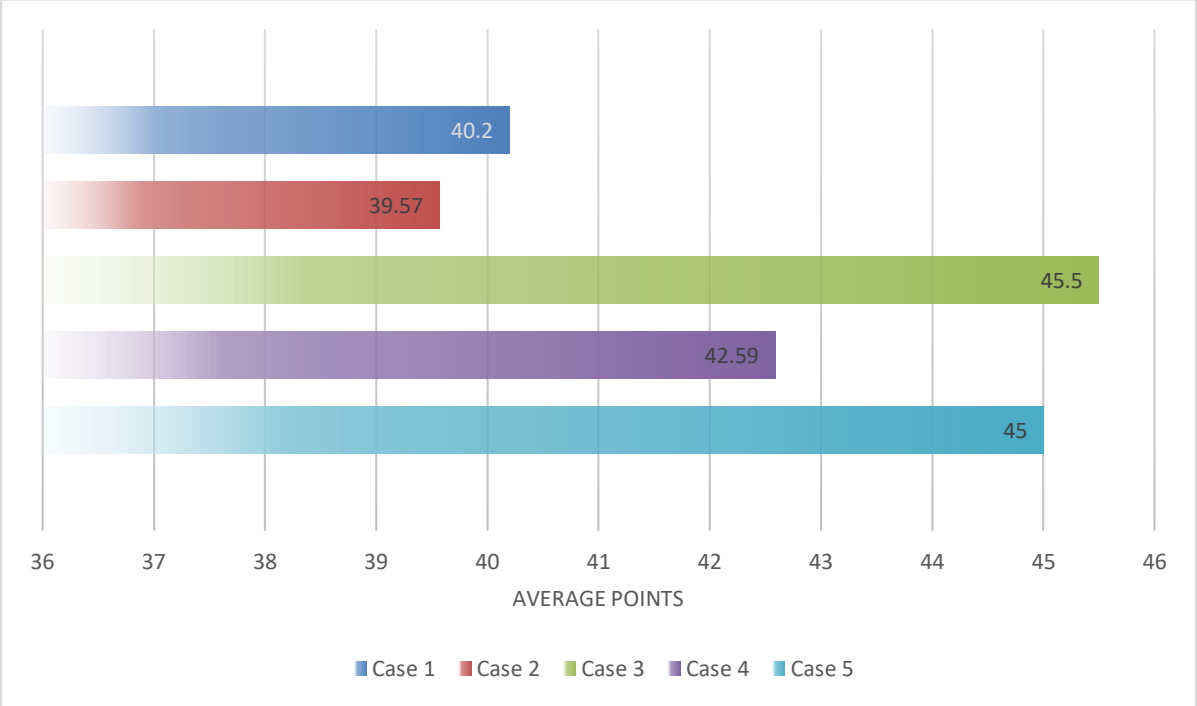

Figure 5: Average points for each project case

5.2 Team composition

"In a team where you don't know anyone you get the opportunity to test new roles and redefine how you work"
- Student 6

In 2019 there were 39 teams in the course, consisting of four to six team members. Team 1 to 30 were assembled by the students themselves, while team 31 to 39 were created by the instructors. The average team size was 5 team members. The quote above explains one student's reasoning behind choosing an instructor formed team. Table 15 shows the distribution of team sizes for both student-formed teams and for instructor-formed teams.

	Four TM	Five TM	Six TM
Instructor formed	3	4	2
Student formed	6	15	9
Total	9	19	11

TM = Team Members

Table 15: Team size in project

As the majority of the teams in the course were student formed, the instructors did not interfere with the forming of the teams, nor how diverse they were in regard to study programs and genders. However, the students were told to consider these factors when forming their teams. The difference in diversity by study program between the instructor formed teams and the student formed teams can be seen in table 16. Teams with more than one study program present are considered multidisciplinary.

	One program	Two programs	Three programs
Instructor formed	4	4	1
Student formed	12	18	0
Total	16 - (41%)	22 – 56.5%	1 – 2.5%

Table 16: Distribution of study programs in the teams

I did not intend to look at the effect of gender in the student projects, however, one teaching assistant made an observation that the teams that performed best had a variety of study programs, regardless of their gender distribution. He said, "*the gender of the team members didn't really affect the result, one of the all-girls teams had a variety of study programs and did really well, while one of the all-boys teams were all from the same study program and did really bad*" (Teaching Assistant 2). As such, I had to know if this was the case in the course. His observation is reflected in my findings as well, presented in table 17.

Team Composition	2G-2/3P	2G-1P	1G-2P	1G-1P
Points	44.07	39.37	43.22	40

G = Gender(s), P = Program(s)

Table 17: Team composition and points

Table 18 shows the distribution of grades to team size, and the average score for the teams of the different sizes. Among the 11 teams consisting of six team members, none got a grade below B. Their average points scored on the project here is also higher than the teams consisting of four or five team members. This could indicate that a team consisting of six team members is more likely to do well in the project.

However, these numbers alone do not say anything about why the teams consisting of five team members did worse than both the teams of four and six team members. Through my findings in table 17, there seems to be a link between how multidisciplinary a team is and their project result. Out of the teams with six team members, 81.8% of them were multidisciplinary, for the teams with four team members this number is 55.5%, while for the teams with five team members less than half of the teams are multidisciplinary at 47.3%. The majority (52.6%) of the teams with five team members consisted only of team members from one study program, and that can account for their lower overall score.

Team size	A	B	C	Average points
4	2	4	3	42
5	3	9	7	40.6
6	4	7	0	44.6

Table 18: Alphabetical grade distribution based on team size

5.2.1 Team Composition of Interviewee Teams

Table 19 provides an overview of the interviewed students. Column "ID" is the identification for the students interviewed. When quotes are used from the interviews, they will be attributed using the ID. The column "Program" refers to the study program of the interviewed student, and "Size" refers to the team size. "Programs in team" refers to which study programs the team consisted of, "Gender in team" refers to the gender distribution in the team.

ID	Program	Team	Size	Programs in team	Genders in team	Team formation	Project grade
1	Design	A	6	2	1	Unknown	B
2	Prosa	B	5	2	1	Friends	B
3	Prosa	C	4	2	2	Friends/Unknown	B
4	Design	D	5	2	2	Unknown	B
5	Prosa	E	6	2	2	Friends/Unknown	A
6	Design	F	4	3	2	Instructor	B

Table 19: Overview of student interviewees

The teams of the interviewed students were formed in different ways. The formation was not only different in terms of the teams being formed by the instructor or the students themselves; there were also different approaches to team-formation among the student-formed teams. The column for "team formation" in table 19 covers how the team was created.

Friends refer to teams that consisted only of members from the same friend group. *Unknown* refer to teams where the interviewee did not know anyone beforehand, and where the students actively sought out a team without any of their friends. In the interview, student 1 said that consciously choosing a team with strangers led to more diversity in the team in terms of both study programs and also interests. *Friends/Unknown* is a combination of the two previous categories. Here the student knew one of the other team members beforehand, and they found the other team members from outside their pool of acquaintances. The interviewees who had this formation of their teams said that it felt safe being with someone they knew, but that working with strangers taught them much about teamwork and helped them get the most out of the project. The final category is *instructor*, where the entire team was created by one of the course instructors. Student 6 said "*I would choose an instructor-made team again*". All of the interviewed students came from multidisciplinary teams. However, three of the teams were homogeneous in terms of gender while the remaining three had at least one team member of each gender.

5.3 Process models

> *"It was positive to practice the theory by being in a software development process. You learn how different tools work, what Scrum and Kanban can look like in practice, and the significance all of this this has on the development of a product"*
>
> - Student 2

The students were encouraged to try out different ways of working and adjusting the process models to fit the needs of their team. This led to a variation in how agile was practiced in the teams, and which process models they chose. The process models chosen by the teams are shown in table 20. The most popular process model was a hybrid between Scrum and Kanban; *Scrumban*. In terms of the average result from the teams, there was little difference in regard to which process model was chosen.

Process Model:	Number of teams:	Average points:
Scrum	17	42.3
Kanban	1	42
Scrumban	21	41.8

Table 20: Process models used by the teams

The advantages to choosing Scrumban as process model was reported to be due to the flexibility this approach added. The teams that chose Scrumban implemented it as follows;

1) Kept the scrum meetings and to a large degree the scrum roles.
2) Planned sprints with sprint backlog and goals for the sprint.
3) Used a Kanban board for backlog.
4) Focused on work-in-progress and worked on a task until it was done.
5) Stronger focus on flow as time was sometimes hard to plan in a student setting.

5.4 Meetings

"We met often, which I think was the key for us having such good communication throughout the project"
- Student 5

From the beginning of the course the students were encouraged to work together and meet each other often. For some teams this was the key to successful teamwork. The students were presented with four different types of meetings in Scrum along with an explanation of their uses and benefits in an early lecture. On average, the teams reported that they met 2.65 times per week during the project. By investigating the project reports, we can see which meetings were incorporated in the teams. This is presented in table 21.

Meeting Type:	Number of teams that used it:
Retrospective	36
Daily Stand-Up	29
Sprint Planning	26
Sprint Review	12

Table 21: Meetings in the teams

Several of the teams that utilized both retrospective meetings and sprint planning meetings did this once a week as one longer meeting. At the end of a sprint a retrospective meeting would be held, and after short break they would continue by planning for the next sprint.

One team explained that their reasoning for doing this was because the retrospective meeting was still fresh in their mind, enabling them to more consciously plan the next sprint.

Daily stand-up meetings were used by 29 teams, accounting for 148 students. The ratings for daily stand-up meetings from the survey showed that 76.5% of all the students were positive with the use of daily stand-up meetings in their project, with the mean rating being 4.1. The responses of 4 and 5 were coded as “positive” and responses of 1 and to were coded as “negative”. Through investigation of the student reports it became clear that the student teams did not use the stand-up meeting daily, but rather focused on holding in several times per week. There were no distinctive data on how many times per week the teams held the stand-up meeting.

In addition to meetings, the teams used different tools for communicating and coordinating their work. The most popular tools were; Trello, Slack, Google Drive, Evetro, Facebook Messenger, GitHub Projects, and Discord. I have not investigated how these coordination artefacts were used in the teams, nor how the teams dealt with dependencies or decided on who did what by when.

5.5 Teamwork Quality

"Figuring out how to work with people coming from different fields is a really valuable experience. And through this, you learn more about yourself and how you work in a team"
- Student 1

Investigating the quality of teamwork in student teams can provide understanding of how the students practice and experience teamwork. As said by student 1, it can be a valuable experience to learn how to work in a team setting with different people. The results from the survey are presented in table 22.

Latent Variable	Rater	Variable	No. of Items	Mean	Std. Dev.
Teamwork Quality (TWQ)	Team member	Communication	10	4.17	0.37
		Coordination	4	4.05	0.40
		Mutual support	7	4.42	0.37
		Effort	4	3.86	0.65
		Cohesion	10	4.26	0.45
		Balance of member contribution	3	4.24	0.41
Team member's success	Team member	Work satisfaction	4	4.31	0.43
		Learning	4	4.39	0-47
Team Performance	Team member	Effectiveness_TM	10	3.86	0.42
		Efficiency_TM	5	3.81	0.60
	Scrum Master	Effectiveness_SM	10	3.86	0.65
		Efficiency_SM	5	3.85	0.74
	Product Owner	Effectiveness_PO	10	3.9	0.72
		Efficiency_PO	5	3.71	0.87

Table 22: Descriptive statistics of the investigated variables

Investigation of the correlations were done in python using Pearson's correlation coefficient. The investigation was done by first looking for correlations between all variables, and then for correlations between the latent variables. The correlations are interpreted according to Schober et al. (2018, p. 1765), shown in table 23.

Observed Correlation	Interpretation
0.00 – 0.10	Negligible correlation.
0.10 – 0.39	Weak Correlation.
0.40 – 0.69	Moderate correlation.
0.70 – 0.89	Strong correlation.
0.90 – 1.00	Very strong correlation.

Table 23: Interpretation of correlations (Schober et al., 2018, p. 1765)

Strong correlations are in the tables marked in green, moderate correlations pertaining to the project point in the tables are marked as yellow.

5.5.1 Correlations in variables

First, I investigated the correlations between all the variables in the survey and the project points. This resulted in table 24 and figure 6. Out of the variables related to TWQ, *mutual support* was strongly correlated to *communication* (0.87) and *coordination* (0.85).

Variables	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
(1) Communication															
(2) Coordination	0.77														
(3) Mutual Support	0.87	0.85													
(4) Effort	0.76	0.78	0.74												
(5) Cohesion	0.78	0.64	0.77	0.80											
(6) Balance of m.con.	0.72	0.74	0.78	0.80	0.69										
(7) Work satisfaction	0.82	0.62	0.72	0.79	0.86	0.64									
(8) Learning	0.35	0.13	0.26	0.39	0.63	0.29	0.69								
(9) Effectiveness (TM)	0.62	0.46	0.52	0.50	0.61	0.48	0.70	0.59							
(10) Efficiency (TM)	0.66	0.67	0.63	0.68	0.63	0.62	0.69	0.33	0.74						
(11) Effectiveness (SM)	0.48	0.49	0.53	0.45	0.39	0.55	0.45	0.09	0.48	0.47					
(12) Efficiency (SM)	0.35	0.44	0.40	0.33	0.24	0.33	0.34	0.01	0.40	0.52	0.76				
(13) Effectiveness (PO)	0.31	0.37	0.24	0.42	0.29	0.32	0.49	0.41	0.60	0.55	0.27	0.35			
(14) Efficiency (PO)	0.21	0.37	0.18	0.31	0.17	0.18	0.26	0.12	0.50	0.52	0.33	0.47	0.79		
(15) Project Points	0.25	0.13	0.20	0.28	0.34	0.24	0.53	0.51	0.39	0.40	0.08	0.17	0.64	0.38	

Table 24: Correlations looking at all variables and the project points

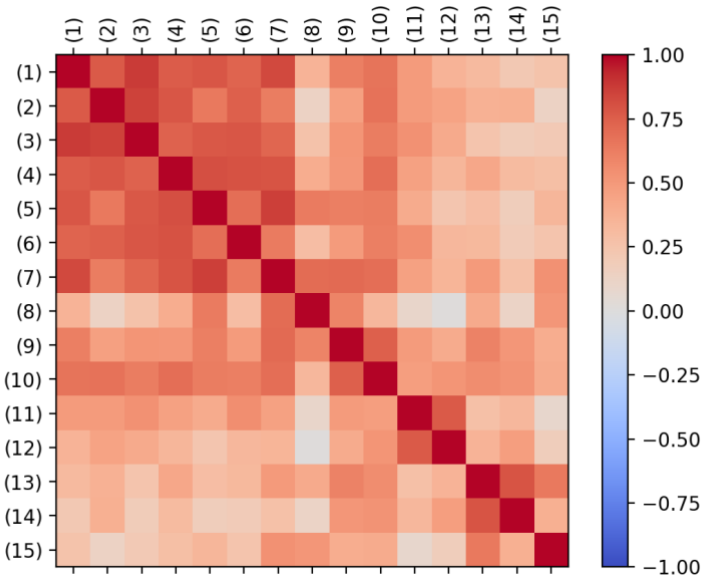

Figure 6: Correlations looking at all variables and the project points

5.5.2 Correlation in latent variables

I then looked at the latent variables and the project points, which resulted in table 25 and figure 7. Here we see that *TWQ* and *team performance* as rated by the team members were strongly correlated.

Variables	1	2	3	4	5	6
(1) TWQ						
(2) Team Members' Success	0.66					
(3) Team Performance (TM)	0.72	0.65				
(4) Team Performance (SO)	0.48	0.25	0.54			
(5) Team Performance (PO)	0.34	0.35	0.61	0.41		
(6) Project Points	0.28	0.56	0.42	0.14	0.52	

Table 25: Correlations looking at the latent variables and the project points

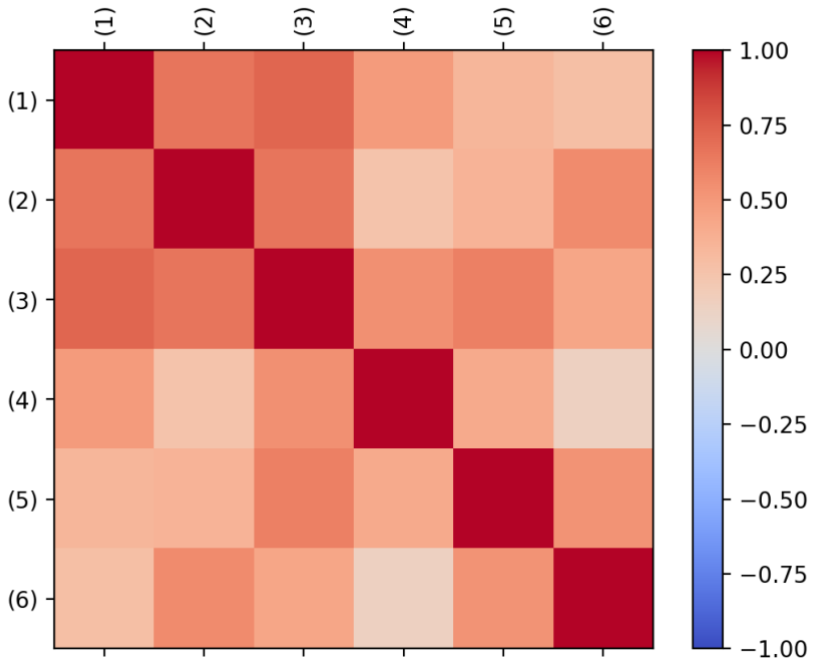

Figure 7: Correlations, looking at the latent variables and the project points

5.6 High- and low-performing teams

The high-performing teams are the ones that scored from 46 to 50 points on their projects, while the low-performing teams are the ones that scored from 30 to 35 points. The lowest performing teams received the same alphabetical grade as the teams scoring from 36 to 40 points. One of the teaching assistants pointed out that a common denominator for the high-performing teams was that they had a high process compliance, meaning that what they said they would do corresponded to what they actually did. The difference between the high- and low-performing teams is shown in table 26.

	High-Performing Teams	**Low-Performing Teams**
Average Points	47.2	31.6
Average Team Size	5.2	4.66
Average Weekly Meetings	3.15	2
Percentage Multidisciplinary Teams	66.66%	0%
Process Model	1. Scrumban (78%) 2. Scrum (22%)	1. Scrumban (100%)
TWQ - Rating	4.48	4.20
Team Member's Success - Rating	4.74	3.89
Team Performance - Rating	4.24	3.42

Table 26: Difference between high- and low-performing teams

While table 26 shows the difference in ratings on the TWQ construct, figure 8 shows how these ratings manifested in the different variables. In this figure, the ratings from product owner, as rated by the teaching assistants, are not included. The high-performing teams rated all of the variables in the TWQ survey higher than the overall average. *Effort* and *efficiency rated by team members* were rated significantly higher by these teams than the overall average. The low-performing teams followed the curve for all teams for TWQ ratings. They rated *learning* and *efficiency rated by the scrum master* significantly lower than the overall average.

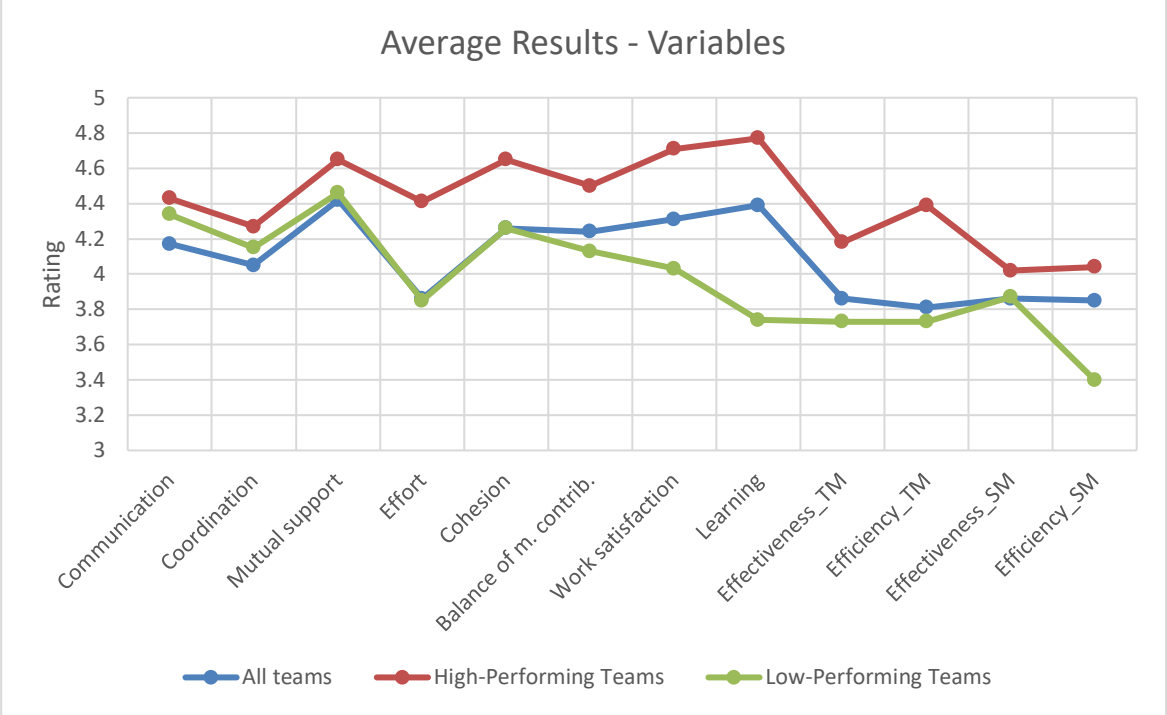

Figure 8: Survey results, variables for all teams and the high- and low-performing teams

When comparing these two grouping of teams based on the ratings of the TWQ variables, five variables stand out; *effort*, *work satisfaction*, *learning*, *efficiency* (team member), and *efficiency* (scrum master). These variables are rated at least 0.5 points higher by the high-performing teams.

TWQ

While both the high- and the low-performing teams rated their *communication*, *coordination* and *mutual support* in a similar manner (less than 0.20 in difference between ratings), there is a larger difference when it comes to *effort*. The high-performing teams rated *effort* 0.56 higher than the low-performing teams.

Team members' success

The variables related to team members' success shows the largest differences between these teams. Here the high-performing teams rated both *work satisfaction* and *learning* higher than the low-performing teams, with 0.68 and 1.04 points respectively.

Team performance

The low-performing teams rated effectiveness and efficiency lower than the high-performing teams. This was the case for both raters (team members and scrum master). The difference is

however highest in regard to efficiency, where the high-performing teams rated it 0.66 (team members) and 0.64 (scrum master) points higher.

In terms of the correlation between the various TWQ constructs, the results are quite different for the high- and low-performing teams. While *Team Members' Success* has a very strong correlation to *TWQ* among the high-performing teams, the correlation is weak among the low-performing teams. For the high-performing teams there is a there is a negligible to weak correlation between the project points and any of the TWQ constructs. However, the project points are strongly correlated to all of the TWQ constructs for the low-performing teams. This is shown in figure 9 and table 27.

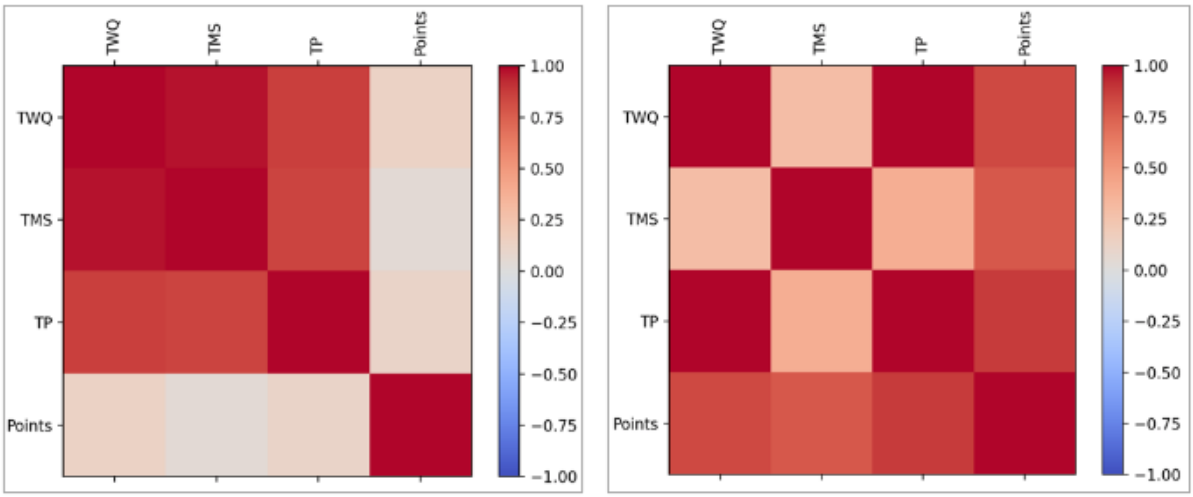

Figure 9: Correlations, high-performing (left) & low-performing (right)

Variables		**1**	**2**	**3**	**4**
(1)	TWQ		0.24	0.99	0.82
(2)	Team Members' Success (TMS)	0.97		0.39	0.77
(3)	Team Performance (TP)	0.85	0.84		0.88
(4)	Points	0.12	0.04	0.11	

Table 27:Correlations, high-performing (blue) & low-performing (orange)

Further investigation of the high-performing teams shows that there are very strong correlations among the TWQ variables, presented in figure 10 and table 28. Here the variables pertaining to team members' success (learning and work satisfaction) and team performance (effectiveness and efficiency) were aggregated.

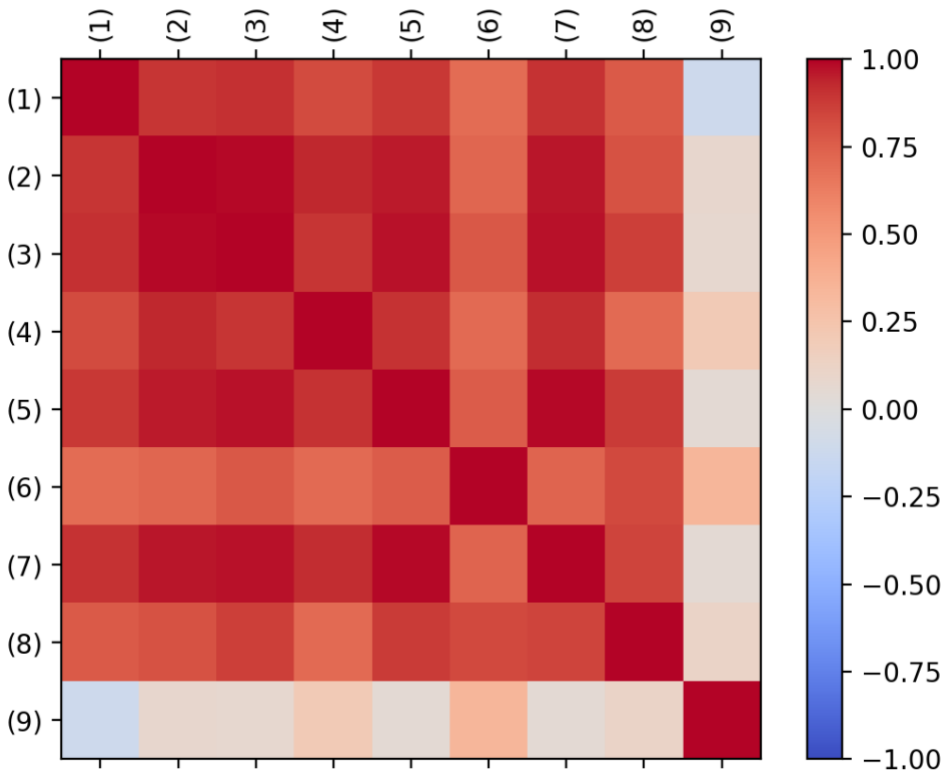

Figure 10: Correlations high-performing teams

Variables		1	2	3	4	5	6	7	8	9
(1)	Communication									
(2)	Coordination	0.89								
(3)	Mutual Support	0.91	0.98							
(4)	Effort	0.82	0.93	0.89						
(5)	Cohesion	0.88	0.95	0.97	0.90					
(6)	Balance of m.con.	0.70	0.72	0.78	0.70	0.76				
(7)	Team Member's Success	0.89	0.96	0.97	0.92	0.98	0.73			
(8)	Team Performance	0.76	0.80	0.86	0.70	0.87	0.83	0.84		
(9)	Project Points	-0.11	0.08	0.07	0.20	0.04	0.33	0.04	0.11	

Table 28: Correlations high-performing teams

5.7 Summary of results

There are several interesting results detailed in this chapter. Some of them are directly linked to the research questions, while others can provide more understanding of the student teams and their context. The results can be summarized as follows;

1) A case that provides an easily defined and accessible user group is more popular among student teams. Additionally, students seem to want to create something that is relevant for themselves.

2) Multidisciplinary teams perform better than non-multidisciplinary teams, regardless of how they were formed (by students or instructors).

3) Scrumban was a popular process model, adopted by most of the student teams due to the added flexibility that suited a student context.

4) Retrospective meetings were the one form of meeting held in nearly all the teams throughout the project.

5) In regard to RQ2, there seems to be but a weak correlation between TWQ and project points. However, there is a moderate correlation between the project points and team member's success. Students could have a positive experience with the teamwork in their team without necessarily scoring high in terms of project points, and vice versa.

6) Considering RQ1, there are certain elements that characterize high-performing teams; they are usually multidisciplinary, have on average 5.2 team members, and meet on average 3.15 times per week. The average for all teams in the course was 5 team members and 2.65 weekly meetings. In addition, they rate *TWQ, Team Members' Success,* and *Team Performance* higher than the average team. The high-performing teams rate the TWQ variables *effort, work satisfaction* and *learning* much higher than the low-performing teams. The result of the high-performing teams, referring to the project points, has a negligible to weak correlation with both communication and coordination as rated in the survey.

6 Discussion

6.1 Teamwork Quality

Through investigating the interactions within a team, the hypothesis of the TWQ model is that "TWQ is positively related to the success of innovative projects" (Hoegl & Gemuenden, 2001, p. 439). What is deemed a "success" when it comes to a student project will vary, both for the students and for the instructors. Some of the ways success of a student project can manifest are as follows; 1) the team produced a functioning app, 2) all team members passed the course, 3) the team got a good project grade, 4) the app developed was fantastic, 5) the teamwork was a positive experience for the team members, and/or 6) the team members learned a lot throughout the project. As for number 1 and 2, I can with certainty say that all student projects were a success. While I view performance as the points awarded the teams for their project, I have no clear answer as to what success should be viewed as in this context. As such, when discussing success, the definition will be that the team created a functioning app.

Previous TWQ surveys have investigated professional traditional and agile teams, while this thesis investigates student teams engaged in a semester-long project in a software engineering course. When comparing my results to previous results, I will focus on the agile teams. One of the central learning outcomes of this course was for the students to obtain knowledge of central processes and actors in agile teamwork. The project for the student teams should therefore be viewed as the arena where they got to practice agile and teamwork; aspects of software development that until this course had only been theoretical. Therefore, student teams cannot be viewed as agile in the same sense as the professional teams.

The TWQ model has several benefits for investigating student teams; 1) it does not take a lot of time to investigate several areas, 2) it can show what teams on average think on the same statements, 3) it's possible to survey a large cohort and thus get a larger study sample, 4) because it has so many subconstructs we can see "inside" the constructs, for example, there was only a small difference between the TWQ scores for the high- and low-performing teams, but there was a major difference between how they experienced effort, work satisfaction and learning. Knowing that these elements are important for a team to perform well we can try to tailor aspects of the course to accommodate for this.

The teamwork quality survey on agile teams had clear distinctions between the different roles in the agile teams (Lindsjørn et al., 2016). This allowed for providing the roles with slightly different surveys. This distinction was not present in the student projects, as some did not use the product owner role in the team and others rotated on who acted as scrum master or product owner for a specific sprint. In my survey the teaching assistants responded as the product owner for the teams they had supervised, but their involvement with the different teams varied. This needs to be taken into account when comparing the results from the student teams with the agile professional teams.

Another difference between the surveys is that while the respondents from the professional teams came from a wide range of domains, the respondents from the student teams all worked with the same weather API's provided by The Norwegian Meteorological Institute to create a mobile application. The students also had a common background from having taken compulsory courses in both programming and software engineering during their first three semesters.

Figure 11 displays the average values of the variables used in the survey from both student and professional teams. Overall, we see that the students rate the variables of TWQ and team member's success higher than what the professional teams did. A possible reason is that since this was the first teamwork the majority of the students engaged in, they had nothing else to compare it to. As such, they might have had lower expectations for the teamwork and therefore rate the teamwork higher than what it actually was when it exceeded these expectations. Lindsjørn et al. also points to the presence of implicit models as a possible explanation for variations in the ratings (2016, p. 280-281). Implicit models imply that "if team members consider TWQ to be high, they may also consider performance to be high" (Lindsjørn et al., 2016, p. 280-281). They also point to the tendency where team members seem to rate performance highly even if it was not, simply because there was a good social dynamic in the team (Lindsjørn et al., 2016, p. 280-281).

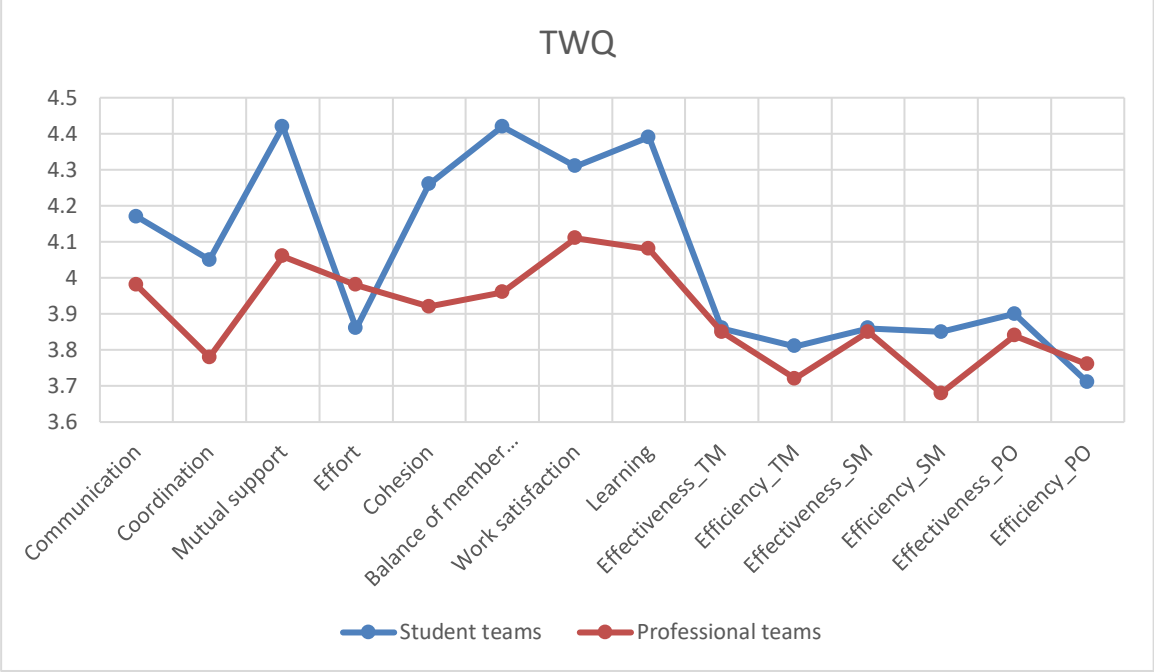

Figure 11: TWQ results from student and professional teams

The two highest rated TWQ variables from the student teams are *mutual support* and *balance of member contribution*. Lindsjørn et al. found that *mutual support* affected team performance the most out of all the TWQ variables (2016, p. 281). A possible explanation for this is that the agile teams have no leader, making mutual support an important factor of agile teamwork. Mutual support is seen through how the team members support each other and their work, but also entails "quick resolution of conflicts, constructive discussions, respect for suggestions and contributions made by other team members, the ability to reach consensus, and good cooperation" (Lindsjørn et al., 2016, p. 281). It is then interesting that there is a large drop in the student ratings of *effort*, which is the lowest rated TWQ variable among the student teams. As *effort* had a strong correlation to both *mutual support* and *balance of member contribution*, shown in table 24 in section 5.5.1. I would have expected that the high rating of *mutual support* and *balance of member contribution* meant that *effort* would also be highly rated.

Figure 12 shows the standard deviation between the responses of both student and professional teams. While the results in figure 11 does not follow a similar curve, it is interesting that the standard deviation does. With *effort* once again being an exception, where the standard deviation among the students is 0.31 higher than among the professional teams. In the student teams this might indicate that some students felt like they had to carry the majority of the workload, thus rating the overall effort as low, while those contributing less were happy with the overall effort of the team.

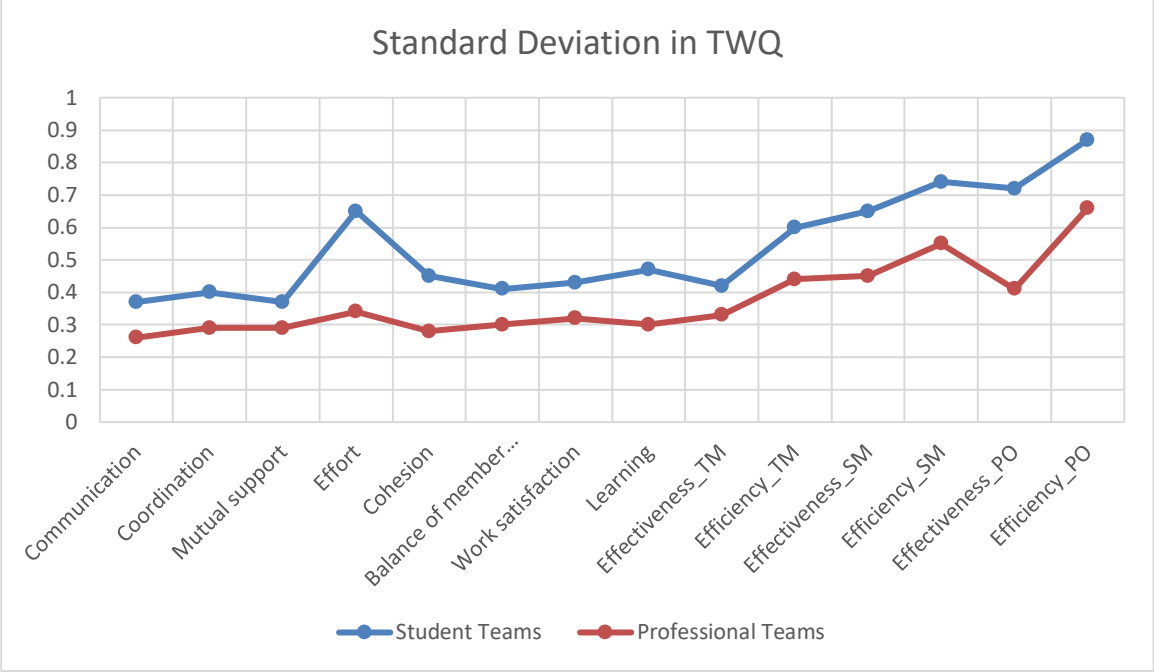

Figure 12: Standard Deviation between student and professional teams

A contributor to the high standard deviation and lower score for *effort* might be because the team members in the student teams have different approaches to the course and the project. It is a mandatory course, and not a course chosen by the students based on interest. This could mean that some students just want to pass the course, and thus put in the minimum effort, while others are engaged and want to do well (Fioravanti et al., 2018, p. 810).). This was a concern also voiced by one of the interviewees; "*One downside to having the course be mandatory for everyone is that you don't know if people are actually motivated to take the course or even find it interesting. Which is just absurd since the course is so close to the work life most of us will be entering*" (Student 6). This was a challenge observed in other teamwork-focused software engineering courses as well (Iacob & Faily, 2018, p. 169). A team consisting of team members with different levels of ambition and team orientation can be a challenge for goal orientation and team cohesion (Salas et al., 2005, p. 570; Dingsøyr et al., 2016, p. 106). I found that *effort* was strongly correlated to all of the other 5 TWQ variables, and that of *work satisfaction*.

In section 5.5 correlations between the variables of the survey done on the student teams were presented. There is a strong correlation between *work satisfaction* and *effectiveness* as rated by the team members, shown in table 24. The latent variable *team performance* as rated by the team members in table 25 has a strong correlation to *TWQ*. This is interesting because the quality of teamwork can affect team performance, specifically the product created by the team (Hoegl & Gemuenden, 2001, p. 446; Lindsjørn et al., 2016, p. 282).

In regard to RQ2, there seems to be but a weak correlation between *TWQ* and project points. However, there is a moderate correlation between *team member's success* (which holds *work satisfaction* and *learning*) and project points. This could indicate that the teams that were happy with the product of their teamwork and who felt they learned a lot, would receive a better grade on their project, or vice versa.

With an average rating of 4.17 on the TWQ elements in the survey, most students seemed pleased with their teamwork. Among the lowest scoring teams, the TWQ was even slightly higher than the overall average, at 4.20. In other words, the students can have a positive experience of their teamwork even if the graded results are not great. Table 26 showed the difference in average ratings on the three constructs of the TWQ model for high- and low-performing teams. While I could not see a strong correlation between team member's success or team performance and TWQ, the numbers indicate that when these two constructs were highly rated, the TWQ was higher rated as well. This corresponds with what Hoegl & Gemuenden showed in their investigation (2001, p. 446).

6.2 Agile Teamwork

One of the learning outcomes of the course IN2000 is to "have the skills to work in teams and the ability to reflect on your own and the team's work in system development projects" (UiO). The importance of this learning outcome is underlined by Chattel, "teamwork is an essential component of any software engineering program and is a key skill that many employers look for when hiring graduates" (2017, p. 120).

Katzenbach & Smith asked the question "what makes the difference between a team that performs and one that doesn't?" (2005, p. 2). To be able to answer this question in the context of this thesis, we first need to consider what it means for a team to perform in the course.

Performance can be seen through process compliance, highly rated performance constructs in the TWQ survey, or the project points. This thesis did not observe how the teams worked throughout the project and relies on their project reports for information on this matter. Because of this I cannot look at performance as process compliance in this setting. The presence of both implicit models and the tendency to rate performance highly if there was a good social dynamic in the team (Lindsjørn et al., 2016, p. 280-281), makes the measure of *team performance* in this setting indicative at best. Therefore, performance is viewed through the lens of the project points awarded to the team by the instructor on the basis of the evaluation criteria in Appendix IV.

6.2.1 Agile teams in the course

As agile teams are characterized by self-organization, it was important for the course to practice this throughout the projects (Stray et al., 2011, p. 147; Cockburn & Highsmith, 2001, p. 132). Apart from some teams being formed by the instructors, they were not meddled with by the instructors or teaching assistants. This allowed for a high degree of autonomy, specifically external autonomy (Moe et al., 2008, p. 78). While the teams had high external autonomy, referring to how free the teams were to organize their own tasks, the challenge for the student teams was achieving internal autonomy (Moe et al., 2008, p. 78). Internal autonomy is seen through how the teams structured their project, tasks, communication and decisions. One particular challenge for internal autonomy was too high individual autonomy, as this would cause the team to not be included in decisions or execution of tasks central to the project. From the interviews, it seemed like most of the teams struggled with some level of individual autonomy where one or two team members did not include the rest in their tasks or decision-making. This led to confusion regarding what was actually completed, a lot of work being done twice, and a lack of trust. Autonomous, self-organizing teams require trust and common goals, when this is not present the teamwork becomes challenging (Moe. et al, 2019).

An important aspect of self-organizing teams is that they have the necessary skills present in the team and that the team members are able to contribute to more than one area of the work (Parker, 2003, p. 4; Hoda et al, 2013, p. 424). This is referred to as cross-functionality.

I found that the teams that were cross-functional, meaning that they had a combination of study programs, received the highest amount of average points for their project, at 44.07 points in teams with two genders, and 43.22 points in teams with one gender. The teams with only one study program, meaning a lack of cross-functionality present, performed poorer regardless of the gender distribution in their team at 39.37 (two genders) and 40 (one gender) average points.

A possible explanation for this difference is that the teams with members from only one study program did not have the necessary skills present in their team (Parker, 2003, p. 4), or that they were unable or unwilling to step in or acquire the missing skills (Moe et al., 2008, p. 82). When the required skills are present, it is easier for a team to accomplish their goal and respond to changes and challenges as they arise throughout the project (Parker, 2003, p. 6; Moe et al., 2008, p. 82).

6.2.2 Process Models

A central element in the course was encouraging the students to work with process models seen in the industry, like for example Scrum. The use of a process model is important for organizing and structuring the project work, as it provides a framework for development (Stoica et al., 2016, p. 10). In a previous mandatory introductory course, the students had been presented with different process models and the difference between traditional and agile development. However, the first time the students had to apply this theory in practice was in the course discussed in this thesis.

The utilization of a process model in the project was seen as comprehensive and unclear, especially at first. Several teams reported that they felt a strict adherence to any process model was difficult in the course setting, as they were all aimed at professional cross-functional teams who work full time (Schwaber, 2004, p. 143). It was especially pointed out that Scrum was overwhelming with its meetings and backlogs, and that there was a lack of guidance in regard to the scrum roles of scrum master and product owner. In the course there was one team that chose to use only Kanban, dismissing all the meetings or roles from Scrum entirely. While they did have one team member who identified as a scrum master for the team, they state in their report that they believed Scrum to be too formal a process model for such a small and short project, which would ultimately stand in the way of them making changes quickly.

In another project-based software engineering course they incorporated a modified version of Scrum, named Rugby, which was "adapted to account for part-time developers" (Alperowitz et al., 2016, p. 323). Having a process model designed for a student-project context might enable an easier transition for students to go from not having worked with process models to incorporating them into their work. However, it should be noted that guidance in the use of process models was available to the students; but they would be required to reach out to the teaching assistants themselves.

While Scrum was the process model most focused on in the lectures, this was not the most used process model among the teams. Scrum was chosen by 17 teams. However, the majority of the teams incorporated Kanban elements into their Scrum process models. This layering of Scrum with Kanban results in the hybrid process model Scrumban (Reddy, 2015). This process model was chosen by 21 teams. The reason behind the popularity of Scrumban in the course might be because this was the first time the students worked with agile processes in a project. As such, adding elements from Kanban, like the focus on flow of tasks, visualize the workflow and limiting work-in-progress, could allow for optimization of their process and provide a better overview of the tasks (Reddy, 2015; Nikitina et al., 2012, p. 142; Kniberg & Skarin, 2010, p. 15). However, some teams did not fully utilize any process model, and one team had a process more akin to waterfall than agile. When no clear process model was used there was a tendency to label the process as "Scrumban" with no further explanation as to why or how. As described by several teams in their project reports, this was due to inexperience in the use of process models in projects as this course was their first meeting with it.

6.2.3 Coordination through tools and meetings

Three coordinating mechanisms described in Salas et al. are 1) *shared mental models* – common understanding of the goal and how to reach it, 2) *closed-loop communication* – both sender and receiver of information acknowledge that it has been received, and 3) *mutual trust* – all team members look out for each other and the team (Salas et al., 2005, p. 565-570)

In projects such as the one the students were engaged in through this course, coordination of tasks and resources is essential (Zaitsev, 2020, p.1; Stray et al., 2019b, p. 111). Coordination in teamwork can take many forms, like described in section 2.1.2, and is essentially the synchronization of the team efforts.

All of the teams utilized some form of synchronization artefact, or rather tools, to coordinate the teamwork (Stray et al., 2019a, p. 7011). The most popular tools for coordination in the student projects are shown in table 29.

Tool name:	**Used by:**	**Function:**
Trello	24 teams	Keep track of tasks, both project backlog and sprint backlog, visualization of workflow.
Slack	20 teams	Internal communication in the team, can have different channels for different topics, integration of github and bots like standuply (automatic stand-up meeting), and direct messages.
GoogleDrive	15 teams	Organizing and overview of resources, surveys, and work on the project report collaboratively.
Evetro	9 teams	Tool used for retrospective meetings.
Messenger	8 teams	Chat client used to coordinate meetings and questions.
GitHub Projects	6 teams	Kanban board on GitHub to streamline and automate workflow.
Discord	5 teams	Voice chat with screen sharing.

Table 29: Project tools

The use of a synchronization artefact to visualize the work progress, like for example Trello or GitHub Projects, has the possibility to enhance the team's common understanding of what they collectively are doing and how they can reach their goal. With a sprint backlog the use of Trello can show how many of the tasks are done or in progress, allowing the team to assess how to reach their sprint goal if they are lagging behind. This pertains to the shared mental models described by Salas et al. (2005, p. 565-570). Slack and messenger were the two most popular communication tools among the teams, allowing them to quickly get in touch with each other regarding questions, updates or just to chat. Stray et al. found that one benefit of using a communication tool like Slack was the increased transparency as the information in the channels was available to all of the team members (Stray et al., 2019b, p. 111). This in turn has a positive effect on the team awareness (Stray et al., 2019b, p. 111), and the communication within the team. I have not investigated how the coordination artefacts were used in the student teams, nor how they dealt with dependencies. However, it would be an interesting area for more research to be conducted. Having a clearer idea of what types of synchronization artefacts are successful and useful in student teams can help set forth guidelines for these kinds of courses in the future.

With all but one of the teams choosing either Scrum or Scrumban, nearly all the teams used the meetings particular to Scrum. These are sprint planning meeting (used by 26 teams), daily stand-up meeting (used by 29 teams), sprint review meeting (used by 12 teams), and sprint retrospective meeting (used by 36 teams) (Schwaber, 2004, p. 133-139). The meetings function as synchronization activities (Stray et al., 2019, p. 7011). The students were presented with these meetings prior to the project start and advised to utilize them in their projects.

There is however an ongoing discussion as to how beneficial some of these meetings are in a professional setting (Stray et al., 2020).

Setting attainable goals is seen as important for the teams, both in terms of being able to realize the goals and to be motivated to do so (Katzenbach & Smith, 2005, p. 4). This could be translated to that the teams that consciously set goals for each sprint and had goals that required the collective effort of the entire team would be more motivated to reach these goals. The high-performing teams reported that a good sprint planning was critical for the quality of the sprint, and 26 out of 39 teams actively used this meeting throughout the project. Stray et al. point out that the success of these planning meetings is dependent on the product owner (Stray et al., 2011, p. 159). The teams and projects did not have a product owner, to some degree the teaching assistants could fill this role but for the most part the teams had to organize themselves without one.

Daily stand-up meetings require daily attendance which, in the context of the course, was not plausible. Through the reports we see that some teams discarded this meeting, while others moved it online using the communication tool Slack with an automatic stand-up bot, called standuply. The purpose of the daily stand-up is to update all the team members on what they are doing (Stray, 2017, 274), and when unable to meet in person this digital solution covers this purpose. In the background section of the TWQ survey the students rated their experience with daily stand-up. This showed that 76.5% of the students were happy with stand-up meetings, and that it had a positive effect on process, team spirit and effectiveness of the team.

The largest challenge with stand-up meetings was to find a time when all team members were available, this challenge caused several teams to discard the meeting all together. There were other teams who decided to hold the stand-up meetings through digital channels such as slack, messenger or discord. Some teams reported that they found the stand-up meeting redundant as they saw each other every day anyway, this was often the case for the teams that consisted solely of students from the same friend group. Others felt the meeting was incompatible with the course setting, as the course only accounted for two thirds of the course load of that semester.

Nearly all of the teams held retrospective meetings on a regular basis, at 36 out of 39 teams. One possible reason behind the popularity of the retrospective meeting is that it was part of a mandatory assignment for the team at the beginning of the project. That nearly all of the teams utilized this meeting throughout their project indicates that it was seen as valuable even after this assignment. Several teams report that the retrospective meetings really helped them figuring out why something wasn't working the way they expected, and in other cases it was in the retrospective meetings that all team members finally understood what had really been going on during the sprint. To reduce the number of meetings, it was not uncommon to combine the retrospective with the sprint-planning for the next sprint.

As meetings are a form of communication, it is important to note that some of the student teams might not see the need for specific meetings, as their preferred way of communication and coordination may be more unstructured. One study found that "ideas and contributions are usually shared, discussed, and evaluated with other team members more quickly and efficiently in informal communication than in formal communication" (Lindsjørn et al., 2016, p. 275). This might be the case for some of the student teams.

6.2.4 Are the teams really agile?

Were all the teams in IN2000 actually teams? And if they were, were they agile? These two questions are important to consider, but it is equally important to consider them against the reality that this is the first teamwork the majority of the students, who are only on the second year of their degree, has engaged in. Both teamwork and agileness are two aspects that are part of the learning outcomes to the course, meaning that it is intended that the students learn this during the course.

In other words, some teams may have been groups who did not successfully apply agile principles, but they most likely learned much on these topics from the course.

Information from the survey results, project reports, and interviews point to some teams displaying characteristics more in line with that of a group rather than that of a team. There was a particular tendency to lean towards individual accountability, delegation, and individual work products in some of the teams, these are characteristics more closely related to groups than teams (Katzenbach & Smith, 2005, p. 4).

However, most of the teams showed characteristics connected to that of a team. They shared leadership roles by rotating on the scrum master role, they had both individual and mutual accountability, worked together often, and communicated frequently (2005, p. 4). The last three characteristics might lead to a strengthened feeling of mutual support in the teams. Table 30 shows some of the group vs. team characteristics from Katzenbach & Smith (2005, p. 4), with a short explanation of how this manifested in the student teams.

Group	**Manifestation**	**Team**	**Manifestation**
Strong clearly focused leader.	One strong leader, or no one organizing the teamwork.	Shared leadership roles.	Rotated on the Scrum Master role.
Individual accountability.	High individual autonomy, team not included in decisions regarding the project.	Individual and mutual accountability.	Internal autonomy, the team make decisions and coordinate the teamwork together.
The group's purpose is the same as the broader organizational mission.	Focus on passing the course, continue the education, graduate.	Specific team purpose that the team itself delivers.	Focus on creating a useful application, winning a prize for best app, receiving a good grade, learn a lot.
Individual work products.	Works together seldom, divides the work so it has little to no dependency on other team members.	Collective work products.	Work together often, tasks are dependent on more than one team member.
Runs efficient meetings.	Few and short meetings, to the point.	Encourages open-ended discussions.	Engages in several kinds of meetings to improve product and process.
Discusses, decides, and delegates.	Connected to individual work products.	Discusses, decides, and does real work together.	Connected to collective work products and mutual accountability.

Table 30: Manifestation of group and team characteristics (Katzenbach & Smith, 2005, p. 4)

One of the learning outcomes for the course states that the students will have the competence to work in teams and have the ability to reflect on their own and the team's work in the project (UiO, 2019). Another one is that they will be knowledgeable in the processes and actors that apply agile principles, and to be able to use professional methods, techniques and tools (UiO, 2019). When talking about agile and its principles, the Agile Manifesto is often brought up as a guide, but to what extent did these principles apply to the students in this course?

*"**Individuals and interactions over processes and tools**"* (Agile Manifesto, 2001).

It can be hard for students to focus on interactions over processes and tools when these processes and tools are brand new to them and the use of them are required in a graded course. However, the number of weekly meetings and the high rating the teams gave *mutual support* indicate that they did indeed manage to focus on interactions.

*"**Working software over comprehensive documentation**"* (Agile Manifesto, 2001).

We required a report of the project, which can be seen as comprehensive documentation as it had to be more than 40 pages long. However, the students were also required to have working software to present. Because the report was much more important in terms of their overall project grade, this could be seen as negative and that the course itself does not follow the agile manifesto.

*"**Customer collaboration over contract negotiation**"* (Agile Manifesto, 2001).

There were no real customers in the project, and neither was there any contract negotiation. However, customer collaboration in the student projects could be seen as the involvement of the user group in development, and contract negotiation as the team agreeing on what their application should consist of and what their focus should be.

*"**Responding to change over following a plan**"* (Agile Manifesto, 2001).

Students had to create a project plan, but it was emphasized that this plan was more of an overview over how many sprints they had and to function as a "guideline". As students, they will undoubtedly have had to respond to change, as they take other courses, and several have part-time jobs that needs to be taken into account.

Through this lens it seems as in the effort to teach the students how to be agile the course structure might have rendered them less so, especially though the focus on processes, tools, and comprehensive documentation. However, the course should be considered as an arena where the students both experience and learn about agile and how to utilize it. As so much of the course is new to the students, it will require time to become familiar with all aspects of it.

6.3 Software Engineering Education

There is a misconception among bachelor students of software engineering that software development equals programming (Iacob & Faily, 2018, p. 163). Even after introductory courses into the various fields of software engineering, like processes, design, programming, requirements engineering, this misconception is prevailing (Sedelmaier & Landes, 2015, p. 420; Iacob & Faily, 2018, p. 163). By introducing the students to the theoretical aspect of software engineering education early in their bachelor's degrees, they are then capable of utilizing this knowledge in practice when faced with a larger team project (Sedelmaier & Landes, 2015, p. 418; Chatley & Field, 2017, p. 118). Several of the interviewed students in this thesis pointed to the practical application of the theory they had previously learned as one of the most eye-opening areas of the course. There have been several studies on project-based learning and the use of a larger teamwork project in software engineering education, some of which are outlined in section 2.1. However, there seems to be few courses like this with a large number of participating students where all students work on the same type of project case utilizing the same basic technologies and API's. This sets IN2000 apart.

The project courses in software engineering most often require the students to draw on knowledge and theory they have acquired during their first semesters (Zorzo et al., 2013, p. 2). Particularly the importance of process models and good communication skills in software engineering might best be understood in a context where they need to be put into practice. This allows the students to draw on the fundamental knowledge and apply it to solve the problem at hand (Cropley, 2015, p. 165). One student said, "*you don't really know the importance of a structured process until you're in the middle of it*" (Student 4)). Another said, "*we have learned about what Scrum and agile development is, but you really do need to test in in practice to actually understand it*" (Student 5).

6.3.1 Student Projects

Teamwork projects during the bachelor's degree are essential for students to not only sharpen their problem-solving and technical skills (Ju et al., 2018, p. 144), but also to develop the soft skills required in teamwork (Abad et al., 2019, p. 208; Iacob & Faily, 2018, p. 163). In addition to this, projects can aid in providing a semi-realistic development setting where the students experience common challenges to teamwork, thus bridging the gap between what the students are taught and what the industry is expecting from them (Abad et al., 2019, p. 208; Fioravanti et al., 2018, p. 806; Paasivaara et al, 2018, p. 51). In all of my six student interviews this was a central theme that came up; the usefulness of the course. The students saw its relevance to industry, stating that "*the benefit of the course is that what you learn is transferrable to other courses and to work settings"* (Student 1). Another of the interviewees even called for more teamwork and project courses in the bachelor's degree, particularly for the students studying programming and system architecture.

Like several of the other courses, IN2000 onboarded the students during the first few weeks by introducing technology, tools and repetition of concepts from the introductory courses (Iacob & Faily, 2018, p. 166, Krusche et al., 2017, p. 94). The onboarding also consisted of a mandatory technical assignment. Chattel observed that when assignments are not mandatory or assessed, students tend not to do them (2017, p. 125). The feedback on the mandatory assignment was that there should have been more of them, and that they were useful; "I really liked that there was a mandatory assignment where everyone had to familiarize themselves with Android Studio and Kotlin" (Student 5).

Most of the studies on projects in software engineering education focused on having a variety of cases and technologies in the student projects (Delgado et al., 2017; Paasovaara, 2018, p. 51, Iacob & Faily, 2016, p. 164), others required the teams to utilize the same set of technological tools and APIs in their projects (Krusche et al., 2017). The course discussed in this thesis falls under the latter category. A high variety of cases and technologies increases the complexity of the course, especially when there are a high number of teams (Alperowitz et al., 2016, p. 323). This complexity can be managed by establishing a framework of tools, cases and technologies that applies to all the teams. The common framework makes it more manageable for instructors and teaching assistants to assess and assist the student teams, especially as teaching assistants can be brought in from among the students who have previously taken the course (Krusche et al., 2017, p.92; Holmes et al., 2018, p. 32).

In the course discussed in this thesis, the teaching assistants are for the most part selected from the previous iterations of the course.

It has been observed that when the teams were able to choose and define their projects freely, there was an increase in engagement and motivation for the projects (Delgado et al., 2017, p. 85). This engagement and motivation manifested as ownership for the product created in the team. While IN2000 did not provide for entirely open cases, this can be an explanation to why so many teams chose case 4, Air Quality, in IN2000; it was relevant to them.

6.3.2 Team Composition

There are many ways to form a team in software engineering projects; student-formed, instructor-formed, or a combination (Delgado et al., 2017, p. 79; Alperowitz et al., 2016; Iacob & Faily, 2018, p. 169; Løvold et al., 2020, p. 5). In the course discussed in this thesis the students could choose if they wanted to create their own team or be assigned a team by the instructors.

The benefits of instructor-formed teams are that they provide a realistic setting as in a professional setting one would not be able to choose one's own team, and that the instructors can ensure multidisciplinary teams (Iacob & Faily, 2018, p. 169). A third benefit for instructor-formed teams is that it ensures that no team can start their project and teamwork until all students have been assigned a team, thus ensuring an equal start. In IN2000 it was reported that several of the student-formed teams began structuring their teamwork and preparing for the project weeks, and sometimes even months, before the project was set to start. This put the students in the instructor-formed teams at a disadvantage as they were informed about their team less than three weeks before the project was set to start. While there are many factors that come into play in teamwork, this advantage could explain some of the differences in results between the student- and instructor-formed teams in this course. The downside to instructor-formed teams is that the team members have different expectations and motivations for the project; some wants to do well while others just want to pass the course (Fioravanti et al., 2018, p. 810), which can result in instructor-formed teams performing poorer than the student-formed teams (Løvold et al., 2020, p. 4).

I have found that while there is a small difference in results between student- and instructor-formed teams (where the student-formed teams perform better), the difference between multidisciplinary and non-multidisciplinary teams is much larger. As seen in table 31 there is still a difference between student- and instructor formed teams; but only if the teams are multidisciplinary. Non-multidisciplinary teams score fairly evenly regardless of how they were formed.

	Multidisciplinary	**Non-multidisciplinary**
Instructor formed	41.2	39.7
Student formed	44.4	39.6

Table 31: Difference between instructor- and student-formed teams

One way to allow students some freedom to choose their teammates, while maintaining some level of realism in terms of team composition, is to combine the two approaches. This way students could sign up with one or two students they know from before, and then they will be matched with other students by the instructors. This could safeguard the need for trust and a feeling of security, which is highlighted as both a coordination mechanism and a requirement for a functioning team (Moe. et al, 2019; Salas et al., 2005, p. 565-570).

In the interviews, the students who had a team comprised of both friends and new people reflected that they had the best of both worlds with this method. Two of the interviewees expressed some regret as to how they chose their team. The first one concerned being on a team with just your friends; "*if I had to do the course over, I'd only be on a team with one or two people that I knew and have the rest be new people*" (Student 2). The second one concerned the combination of study programs in the team, especially in a team where the majority of the students were designers; "*I'd focus more on having a multidisciplinary team as opposed to the one I had where we were all but one from design. Working with a multidisciplinary team can be intimidating in the beginning but would probably be better in the long run*" (Student 4). Student 4 did not consider their team to be multidisciplinary due to the majority being students from design.

6.3.3 Challenges with Student Projects

One of the main challenges in student projects is the requirement that the students have to work with process models designed for full-time developers. While the students were told that they could implement the processes and tools from agile in a manner that was effective to them, it may have been difficult to know exactly what this meant. Figuring out what an adapted process should look like and what needs to be a part of it can be particularly difficult. Using a process model specifically aimed at the part-time developer, which the student teams are, could help the process for the teams considerably (Alperowitz et al., 2016). While the students did state that the processes felt to comprehensive for a course, they did not request an adaptation of the process models. This is likely because they do not know that these exist. The students also expressed a desire for the teaching assistants taking a more active role with the individual teams. One student said that "*what would have been really helpful would have been to have a session with the teaching assistant at the beginning to guide us with sprint planning. We didn't know how to plan a project in the beginning, or even how the process was supposed to work, because all we knew was the theory of it and not how to actually do it*" (Student 2). While the concepts of processes and methods were re-introduced at the beginning of the course, this did not provide enough guidance as to how to apply them in practice.

In some project-based software engineering courses, a central part of the course is the involvement of industry and real customers (Alperowitz et al., 2016; Holmes et al., 2018). This provides an added value in the course, as the student teams have to learn how to collaborate with an entity outside their own team and a real product owner. IN2000 did not involve the industry in this manner but did include the Norwegian Meteorological Institute through the use of their API's. Courses that include industry partners tends to be much smaller than IN2000. A course with approximately 200 enrolled students divided into 39 teams makes it difficult to get industry partners involved while maintaining the same level of learning for all teams. Providing a real customer for the students to work with would add more realism to the course and the project. However, this was not something mentioned by the students as a "lack" in the course. Students have reported that the situation felt as realistic as it could, considering its context and constraints.

The interviews provided more insight in this and the benefits of having a course such as this in their bachelors' degree. "*This is the most engaging course I have had, and it confirmed that technology is something I want to pursue*" (Student 5), "*I have talked about my experiences from the course in job interviews, and the response has been very positive*" (Student 1), *"I think it is very positive to have worked on projects like this, like, when you're looking for a job, it also shows that you are able to work with agile methods"* (Student 3).

6.4 Characteristics of a high-performing team

One of my research questions was concerned about what characterizes a high-performing team in a student project, and if the communication and cooperation within a team affected their result. The result refers to the project grade.

The characteristics of a high-performing team in this course are, perhaps unsurprisingly, that they were multidisciplinary, met and communicated frequently and consistently throughout the project, and coordinated their work effort through both meetings and tools (such as Trello and Slack). All high-performing teams utilized three of the meetings particular to Scrum; sprint planning, stand-up and retrospective. Their general assessment of these meetings was;

1) A good sprint planning was critical for the quality of the sprint.

2) Stand-ups had a positive effect on process, team spirit and effectiveness.

3) Retrospective meetings played a central role in identifying which areas that worked particularly well for the team and which didn't.

Having the abovementioned characteristics, however, does not guarantee for a high-performing team. Through investigating the ratings of the TWQ survey and correlations between variables, a more nuanced view of what characterizes a high-performing team in a student setting emerged. Overall, the high performing teams rated all aspects of the survey higher than the average.

While communication is strongly correlated to all of the variables from the survey, it is negatively correlated to the project points. Both *communication* and *coordination* are very strongly correlated to *mutual support*, while *effort* is very strongly correlated to *coordination*,

cohesion and *team member's success*. This seems to indicate that these elements were important to each other.

Mutual support denotes the how much the team members have helped and supported each other throughout the project. Coordinating frequently with the rest of the team seems to be central for generating mutual support in the team. Through regular meetings that focus on support, improvement, or both, all team members are up to date on the progress and challenges in the team. Meetings like the daily stand-up or retrospective can aid in this and ensure that the team looks out for everyone in the team. Behavior like this relates to the coordination mechanism *mutual trust* (Salas et al., 2005, p. 565-570); not only do the teams trust that tasks are being completed, everyone is heard and involved in decision-making. This in turn can lead to the team members trusting each other to contribute on both tasks and team cohesion. A lack of trust is one of the most common barriers for self-organizing teams (Moe. et al, 2019).

A cross-functional team will have team members who both has the necessary skills and the ability to acquire new skills to help out the team and team members when necessary (Parker, 2003, p. 4). This ability and willingness to help each other can foster a positive social environment within the team, which in turn may positively affect both *mutual support* and *cohesion.* These two variables are very strongly correlated. A team with highly rated *cohesion* demonstrates a positive social environment in the team, a collective commitment to the project, and pride in the team and the team efforts (Hoegl & Gemuenden, 2001, p. 438). Without these factors, there will be difficulties achieving a highly rated *TWQ* (Hoegl & Gemuenden, 2001, p. 438). The interviewee from one of the high-performing teams stated that "*when everyone on a team is bringing their all to do well, you want to do the same and be a positive addition to the team*", and continued by stating that the most important part of the teamwork was to "*be of help and to contribute on the project"* (Student 5). This student explained that all team members contributed when needed. If they needed an extra hand for design or programming purposes, they would shift around tasks to make it possible, demonstrating the advantage of a cross-functional team (Parker, 2003, p. 4; Hoda et al., 2013, p. 424). Any issues related to the progress of the project these would be detected at the stand-up meeting which allowed them to deal with it following the meeting. In other words, they demonstrated the ability to self-organize to deal with the situations that arose throughout the project (Hoda et al., 2013, p. 424).

The success of the team members is very strongly correlated to the team's cohesion. This could indicate that teams that interacted with each other regularly and had a common motivation or goal for the project (good grade, fantastic app, winning a prize), also had more satisfied team members in terms of work and learning. For these teams, *team member's success* was very strongly correlated to *TWQ*. This could indicate that the teams were self-organizing and managed to coordinate and communicate like mentioned in "Agile Teams". The results indicate that the high-performing teams were actual teams; they were committed to the project, had clear goals, and took collective ownership of the project (Katzenbach and Smith, 2005, p. 3). The success of not only the project, but also of the team members, was central. While I have not investigated the "Big Five" in particular for this thesis, it appears that the high-performing teams incorporated all of the five components (Salas et al., 2005, p. 570).

While there is no considerable correlation between project grade and *communication* or *coordination* in my findings, all correlations regarding TWQ are higher when investigating the high-performing teams. The importance of these two aspects within a team will likely increase throughout the project period as well (Paasivaara et al., 2018, p. 56), meaning that if both communication and coordination is established early it will be easier for the team to progress with the project later on. This might suggest that while these two factors alone were not directly related to the project grade, teams that had a high rating of both did better overall.

To conclude, while project grade shows no strong or even moderate correlation to any of the TWQ constructs for the high performing teams, the presence of highly rated TWQ constructs and sub-constructs are indicative of a high-preforming team. In addition to the characteristics of being multidisciplinary, communicating and coordinating often through meetings and tools, utilizing the Scrum meetings, the high-performing teams have a high level of mutual support, cohesion, work satisfaction and learning. These elements are strongly or very strongly correlated to both communication and coordination.

6.5 Suggestions for Practice

There are undoubtedly challenges related to conducting a project-based course with approximately 200 enrolled students. Based on the findings from the research done for this thesis, and the experience from both students and teaching assistants with the course, this section will give some suggestions for practice. Some suggestions have already been implemented in the 2020 iteration of the course (described in section 3.4).

6.5.1 Instructor-formed teams

While the student-formed teams did perform slightly better than the instructor-formed teams, I would suggest not allowing for fully student-formed teams. Combining the aspects of student- and instructor-formed teams can be done by allowing the student to sign up with one or two classmates to then be matched with other students by the instructors. This will aid in giving the students a slightly more realistic team situation, as "*in reality, it is very rare that you are on a team with only your friends*" (Student 5), while also giving the instructors the ability to ensure the teams are multidisciplinary. Additionally, this will avoid some teams starting the project weeks before it officially starts. The size of the teams should be 5-6 students, as this ensures that the team will be able to complete the course if one of their team members decides not to take the course.

6.5.2 Technical onboarding

At the beginning of the course there needs to be a clear and coherent onboarding of both the technical aspects of the course, but also the theory necessary to complete the course as intended. The technical onboarding should be covered in the lectures very early, which will allow the students to familiarize themselves with it before the project start. We have had success with having mandatory technologies for the project; Android Studio, Kotlin, Git and GitHub. Another suggestion is providing the students with a list of communication and coordination tools which can be useful in their project. Examples of such tools are Trello and Slack.

6.5.3 Closer Relationship Between Teaching Assistants and Teams

Teaching assistants supervise the student teams and can provide assistance with the technical aspects of the project, the implementation of processes, tools, and theory. Therefore, it is important to facilitate for a closer relationship between teaching assistants and the teams they supervise. This can be done by keeping the maximum number of teams per teaching assistant at 5 and require all teams to meet with their teaching assistant at least once during the project period.

Additionally, having *enough* teaching assistants should be a goal for the course administration. What "enough" entails will vary, but a rule of thumb is that in a large project-based course where the teaching assistants will supervise the teams, there should be one teaching assistant for every 25 students.

Lastly, to accommodate for some teams having a lack of team members from one of the study programs, it can be beneficial to provide them with a supervisor with proficiency in that field. While the goal is for all teams to be multidisciplinary, there might still be an uneven distribution of study programs within some teams.

6.5.4 Guidance on Implementation of Agile Processes

The process models presented are not tailored to the student context, and as such they can be difficult to approach and maintain for a brand-new team with little to no experience in the field. While part of the project in this course was for the teams themselves to decide on which agile processes and practices to adopt, it could be beneficial to provide guidelines as to what this should entail. More than half of the student teams investigated in this thesis adopted the process model Scrumban, where they combined elements from Scrum and Kanban to work in their context. Providing a guide of what *should* be included in the processes will likely provide a better experience for all students.

6.6 Limitations

My master thesis relies on the research done by one of my master advisors, Yngve Lindsjørn, and I study a course where I have been, and still am, very invested. The pilot course of IN2000 was my favorite course during my bachelor's degree, and watching it evolve has been a highlight to my time at the University of Oslo. This makes me prone to unconscious bias, but also enthusiasm for the topic.

Because of my involvement with the course I am on a first-name basis with all of the students and teaching assistants interviewed, some of whom I know particularly well. In 2019 I also knew several of the students enrolled in the course through work or volunteering in the student community. These two factors may be a limitation to this thesis, as some might have responded what they thought I was looking for, both in the interview and the survey. The possibility of student bias is a consideration we need to have in general when investigating students in a course setting. Even when told that the responses would not be handled until after the final results for the course were published the students might have responded what they think we want to hear. Or rather, responding with what they believed to be the "correct" answer to a survey which was interested in their *experiences*.

Much of the data in this thesis relies on student perceptions. The survey asks for the student's perception on 61 statements, and the team reports detail what the teams wanted us to know about their project. The latter may be a disadvantage, as what was written in the team report and what was actually done in the project might differ.

The use of the TWQ model influences my focus, results and consequent discussion. Had I used a different team model, for example the "Big Five", the results might have provided different insight.

This is a case study of a single occurrence of a software engineering course. This thesis can therefore only report on what happened in this specific course in that specific year. Investigation of the student teams from 2020 may yield different results. By investigating the student teams in this course over a longer period of time, we would likely be able to either generalize or draw more concrete conclusions in regard to teamwork in student teams. I would encourage this investigation.

7 Conclusion and Future Work

This master thesis has attempted to answer two research questions. These were concerned with the teamwork in student projects in a bachelor level software engineering course. The attempt to answer these questions has been done through the use of a case study of the software engineering course IN2000, where data has been collected through a survey, interviews, and relevant documents.

RQ1: What characterizes a high-performing team in a student project, and does the communication and cooperation within a team affect the result?

High-performing teams seem to be characterized by usually being multidisciplinary and having a higher number of team members and meetings than the average team. They rate *TWQ*, *team member's success,* and *team performance* higher than other teams, with a very strong correlation between *TWQ* and *team member's success*. The result of the team is considered as the project points they received. However, the project points have only a negligible to weak correlation with communication and coordination as rated in the survey. While these two variables are not correlated to the project points, the high-performing teams rate them higher than the average team, indicating that they were important for the overall achievement. From the results it seems that it is the variable *effort* that affects the result. The high-performing teams rated *effort* 0.56 points higher than both the average for all teams and for the low-performing teams. Additionally, *effort* is very strongly correlated to *coordination*, *cohesion*, and *team member's success* for the high-performing teams, more so than for the average team.

RQ2: How is the teamwork quality in the student teams related to their project grade?

Teamwork quality has only a weak correlation to the project points. This indicates that even if the project did not receive a high grade, the students still had a positive experience with both their team and the teamwork. This should be a goal in any teamwork course, to provide the students with a positive experience. *Team member's success* has a moderate correlation to the project points for the student teams. As this construct includes the variables *work satisfaction* and *learning* this indicates that the teams that were satisfied with the product of the teamwork, and where the team members felt they learned a lot, would receive more points for their project.

There is as of yet not much research done on teamwork in student projects, especially on bachelor level degrees (Iacob & Faily, 2018). Teamwork is a vital skill to possess for students of software engineering, as it is integral to the work life (Ju et al., 2018, p. 144). Combining teamwork with agile practices adds more value to the course, as this is the de-facto way of working in the industry (Zaitsev et al., 2020, p. 1). It was beneficial to investigate the teamwork through the lens of the TWQ model, as it allows us to see inside the various constructs.

Through this master thesis I have barely scratched the surface of a vast and exciting topic. My data has been rich and filled with interesting insights, but there is a limit to how much can be included in a master thesis. By collecting data from the 2020 iteration of the course, I will be able to contribute to this area further. It will be particularly interesting to see if the findings from the 2019 iteration are mirrored in 2020. For now, I have presented a list of 4 suggestions for practice;

1) *Partly instructor-formed teams* – Ensuring multidisciplinary teams and a realistic setting, while at the same allowing up to three students to sign up for a team together.

2) *Technical onboarding* – Establishing a foundation for all of the students in regard to the technical requirements of the course early should be a focus. This will allow the students to familiarize themselves with the technology and ensure that when the project begins everyone on a team has the same technical foundation.

3) *Closer relationship between teaching assistants and teams* – Fewer teams per teaching assistant will free up time and resources to establish closer relationships between the teaching assistants and their teams. A project-based software engineering course should focus on having enough experienced teaching assistants.

4) *Guidance on implementation of agile processes* – All of the students had taken the introductory course in software engineering and were familiar with some of the terms from agile. However, there is a big difference between theory and practice, and the theory they are taught focus on professional teams. Guidance on what should be included in the process and how to implement it would likely be very beneficial.

This topic would benefit from a more thorough approach. To better be able to determine the learning outcomes of the course and how the teamwork progress and evolves, it could be advised to collect data from the students prior to the project work to use as a preliminary dataset. The collection of qualitative data through interviews from a larger selection of the enrolled students would likely provide better and more representative understanding of the student experiences in a course like this. When investigating a software engineering course, the best data might not be the quantitative as it does not always allow for personal opinion and experiences (Alperowitz et al., 2016, p. 326). This can cause valuable insight to be lost. In addition, the TWQ survey for students in a course such as this could benefit from a longitudinal study, which would allow for more targeted adjustments to the course in regard to the importance of soft skills such as communication, mutual support, and teamwork.

I believe courses that use larger projects and teamwork will become more popular and vital in the future. Thus, I also believe that this is an area of software engineering that warrants more investigation.

Reference list

Abad, Z. S. H., Bano, M., & Zowghi, D. (2019). How Much Authenticity can be Achieved in Software Engineering Project Based Courses? *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 208–219. https://doi.org/10.1109/icse-seet.2019.00030

Agile Manifesto. (2001). Manifesto for Agile Software Development. Accessed from https://agilemanifesto.org

Alperowitz, L., Dzvonyar, D., & Bruegge, B. (2016). Metrics in Agile project courses. *Proceedings of the 38th International Conference on Software Engineering Companion - ICSE '16*, 323–326. https://doi.org/10.1145/2889160.2889183

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Case, J. M., & Light, G. (2011). Emerging Research Methodologies in Engineering Education Research. *Journal of Engineering Education*, *100*(1), 186–210. https://doi.org/10.1002/j.2168-9830.2011.tb00008.x

Chatley, R., & Field, T. (2017). Lean Learning - Applying Lean Techniques to Improve Software Engineering Education. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, 117–126. https://doi.org/10.1109/icse-seet.2017.5

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer*, *34*(11), 131–133. https://doi.org/10.1109/2.963450

Cropley, D. H. (2015). Promoting creativity and innovation in engineering education. *Psychology of Aesthetics, Creativity, and the Arts*, *9*(2), 161–171. https://doi.org/10.1037/aca0000008

Delgado, D., Velasco, A., Aponte, J., & Marcus, A. (2017). Evolving a Project-Based Software Engineering Course: A Case Study. *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, 77–86. https://doi.org/10.1109/cseet.2017.22

Dingsøyr, T., & Dybå, T. (2012). Team effectiveness in software development: Human and cooperative aspects in team effectiveness models and priorities for future studies. *2012 5th International Workshop on Co-Operative and Human Aspects of Software Engineering (CHASE)*, 27–29. https://doi.org/10.1109/chase.2012.6223016

Dingsøyr, T., Fægri, T. E., Dybå, T., Haugset, B., & Lindsjørn, Y. (2016). Team performance in software development: research results versus agile principles. *IEEE software*, *33*(4), 106-110. https://doi.org/10.1109/ms.2016.100

Fioravanti, M. L., Sena, B., Paschoal, L. N., Silva, L. R., Allian, A. P., Nakagawa, E. Y., … Barbosa, E. F. (2018). Integrating Project Based Learning and Project Management for Software Engineering Teaching. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 806–811. https://doi.org/10.1145/3159450.3159599

Hoda, R., Noble, J., & Marshall, S. (2013). Self-Organizing Roles on Agile Software Development Teams. *IEEE Transactions on Software Engineering*, *39*(3), 422–444. https://doi.org/10.1109/tse.2012.30

Hoegl, M. (2005). Smaller teams–better teamwork: How to keep project teams small. *Business Horizons*, *48*(3), 209–214. https://doi.org/10.1016/j.bushor.2004.10.013

Hoegl, M., & Gemuenden, H. G. (2001). Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept and Empirical Evidence. *Organization Science*, *12*(4), 435–449. https://doi.org/10.1287/orsc.12.4.435.10635

Holmes, R., Allen, M., & Craig, M. (2018). Dimensions of experientialism for software engineering education. *Proceedings of the 40th International Conference on Software Engineering Software Engineering Education and Training - ICSE-SEET '18*, 31–39. https://doi.org/10.1145/3183377.3183380

Iacob, C., & Faily, S. (2018). Redesigning an undergraduate software engineering course for a large cohort. *Proceedings of the 40th International Conference on Software Engineering Software Engineering Education and Training - ICSE-SEET '18*, 163–161. https://doi.org/10.1145/3183377.3183381

Ju, A., Fu, X., Zeitsoff, J., Hemani, A., Dimitriadis, Y., & Fox, A. (2018). Scalable Team-Based Software Engineering Education via Automated Systems. *2018 Learning With MOOCS (LWMOOCS)*, 144–146. https://doi.org/10.1109/lwmoocs.2018.8534590

Katzenbach, J. R., & Smith, D. K. (2005). The discipline of teams. *Harvard Business Review*, *83*(7), 162.

Kitchenham, Barbara A. & Pfleeger, Shari L. (2008). Personal Opinion Surveys. In Shull, F., Singer, J., & Sjøberg, D. I. K. (Eds.). *Guide to Advanced Empirical Software Engineering* (p. 63-92). London, United Kingdom: Springer.

Kniberg, H., & Skarin, M. (2010). *Kanban and Scrum - Making the Most of Both*: C4Media Incorporated.

Krusche, S., Bruegge, B., Camilleri, I., Krinkin, K., Seitz, A., & Wobker, C. (2017). Chaordic Learning: A Case Study. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, 87–96. https://doi.org/10.1109/icse-seet.2017.21

Lindsjørn, Y., Sjøberg, D. I. K., Dingsøyr, T., Bergesen, G. R., & Dybå, T. (2016). Teamwork quality and project success in software development: A survey of agile development teams. Journal of Systems and Software, volume 122, p. 274-286. DOI: 10.1016/j.jss.2016.09.028

Løvold, H., Lindsjørn, Y., & Stray, V. (2020). Forming and assessing student teams in Software Engineering courses. Unpublished manuscript. Department of Informatics, University of Oslo.

Moe, N. B., Dingsøyr, T., & Dybå, T. (2008). Understanding Self-Organizing Teams in Agile Software Development. *19th Australian Conference on Software Engineering (Aswec 2008)*, 76–85. https://doi.org/10.1109/aswec.2008.4483195

Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology*, *52*(5), 480–491. https://doi.org/10.1016/j.infsof.2009.11.004

Moe, N. B., Stray, V., & Hoda, R. (2019). Trends and Updated Research Agenda for Autonomous Agile Teams: A Summary of the Second International Workshop at XP2019. *Agile Processes in Software Engineering and Extreme Programming – Workshops*, 13–19. https://doi.org/10.1007/978-3-030-30126-2_2

Nikitina, N., Kajko-Mattsson, M., & Stråle, M. (2012). From scrum to scrumban: A case study of a process transition. *2012 International Conference on Software and System Process (ICSSP)*, 140–149. https://doi.org/10.1109/icssp.2012.6225959

Oxford English Dictionary (2020) Soft Skills. Available at: https://www.oed.com

Paasivaara, M., Vodă, D., Heikkilä, V. T., Vanhanen, J., & Lassenius, C. (2018). How does participating in a capstone project with industrial customers affect student attitudes? *Proceedings of the 40th International Conference on Software Engineering Software Engineering Education and Training - ICSE-SEET '18*, 49–57. https://doi.org/10.1145/3183377.3183398

Parker, G. M. (2003). *Cross-functional teams: Working with allies, enemies, and other strangers*. John Wiley & Sons.

Patanakul, P., Chen, J., & Lynn, G. S. (2012). Autonomous Teams and New Product Development. *Journal of Product Innovation Management*, *29*(5), 734–750. https://doi.org/10.1111/j.1540-5885.2012.00934.x

Reddy, A. (2015). *The Scrumban [r] evolution: Getting the Most Out of Agile, Scrum, and Lean Kanban*: Addison-Wesley Professional.

Robson, C., & McCartan, K. (2016). *Real World Research* (4th ed.). Hoboken, NJ, United States: Wiley.

Rogers, Y., Sharp, H., & Preece, J. (2011). *Interaction Design* (3rd ed.). Hoboken, NJ, United States: Wiley.

Runeson, P., & Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8

Salas, E., Sims, D. E., & Burke, C. S. (2005). Is there a “Big Five” in Teamwork? Small Group Research, 36(5), 555–599. https://doi.org/10.1177/1046496405277134

Schober, P., Boer, C., & Schwarte, L. A. (2018). Correlation Coefficients. *Anesthesia & Analgesia*, *126*(5), 1763–1768. https://doi.org/10.1213/ane.0000000000002864

Schwaber, K. (2004). *Agile Project Management with Scrum*. Amsterdam, Netherlands: Amsterdam University Press.

Sedelmaier, Y., & Landes, D. (2015). Active and Inductive Learning in Software Engineering Education. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 418–427. https://doi.org/10.1109/icse.2015.174

Sommerville, Ian (2016). Project Management. In D. Sjøberg and Y. Lindsjørn (eds), *Selected Chapters from Software Engineering* (p. 229-258). Essex: Pearson Education Limited.

Stoica, M., Ghilic-Micu, B., Mircea, M., & Uscatu, C. (2016). Analyzing Agile Development – from Waterfall Style to Scrumban. *Informatica Economica*, *20*(4/2016), 5–14. https://doi.org/10.12948/issn14531305/20.4.2016.01

Stray, V. G., Moe, N. B., & Dingsøyr, T. (2011). Challenges to Teamwork: A Multiple Case Study of Two Agile Teams. *Lecture Notes in Business Information Processing*, 146–161. https://doi.org/10.1007/978-3-642-20677-1_11

Stray, V., Moe, N. B., & Bergersen, G. R. (2017). Are Daily Stand-up Meetings Valuable? A Survey of Developers in Software Teams. *Lecture Notes in Business Information Processing*, 274–281. https://doi.org/10.1007/978-3-319-57633-6_20

Stray, V. (2018). Planned and unplanned meetings in large-scale projects. *Proceedings of the 19th International Conference on Agile Software Development Companion - XP ’18*, 1–5. https://doi.org/10.1145/3234152.3234178

Stray, V., Moe, N. B., & Aasheim, A. (2019a). Dependency Management in Large-Scale Agile: A Case Study of DevOps Teams. *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 7007–7016. https://doi.org/10.24251/hicss.2019.840

Stray, V., Moe, N. B., & Noroozi, M. (2019b). Slack Me If You Can! Using Enterprise Social Networking Tools in Virtual Agile Teams. *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, 111–121. https://doi.org/10.1109/icgse.2019.00031

Stray, V., Moe, N. B., & Sjoberg, D. I. K. (2020). Daily Stand-Up Meetings: Start Breaking the Rules. *IEEE Software*, *37*(3), 70–77. https://doi.org/10.1109/ms.2018.2875988

UiO. (2019). IN2000. Available at: https://www.uio.no/studier/emner/matnat/ifi/IN2000/index.html

UiO. (2020). IN2000. Available at: https://www.uio.no/studier/emner/matnat/ifi/IN2000/v20/index.html

Zaitsev, A., Gal, U., & Tan, B. (2020). Coordination artifacts in Agile Software Development. *Information and Organization*, *30*(2), 1–23. https://doi.org/10.1016/j.infoandorg.2020.100288

Zorzo, S. D., de Ponte, L., & Lucredio, D. (2013). Using scrum to teach software engineering: A case study. *2013 IEEE Frontiers in Education Conference (FIE)*, 455–461. https://doi.org/10.1109/fie.2013.6684866

Appendix I – Survey Questions

Background Questions (14)	
General Background (6)	1. Team number 2. Age 3. Gender 4. Highest Completed Education 5. When was your first encounter with software engineering? 6. When was your first encounter with programming?
Team Information (5)	7. Approximately how many hours per week did you spend on the project? 8. Approximately how many times per week did your team meet during the project? 9. How happy are you with the use of stand-up meetings? 10. To what degree have you acted as a Scrum Master in the project? 11. What has been your main function in the team?
Programming Skills (3)	12. How would you rate your own programming skills? 13. How would you rate the programming skills of your teammates? 14. How would your teammates rate your programming skills?
Teamwork Quality (38)	
Communication (10)	1. There is frequent communication within the team 2. The team members communicate often in spontaneous meetings, phone conversations, etc. 3. The team members communicate mostly directly and personally with each other 4. Little communication in the team goes through central persons ** 5. Relevant ideas and information relating to the teamwork is shared openly by all team members 6. Important information is kept away from other team members in certain situations * 7. In the team there are conflicts regarding the openness of the information flow * 8. The team members are happy with the timeliness in which they receive information from other team members 9. The team members are happy with the precision of the information they receive from other team members 10. The team members are happy with the usefulness of the information they receive from other team members
Coordination (4)	11. The work done on subtasks within the team is closely harmonized 12. There are clear and fully comprehended goals for subtasks within our team 13. The goals for subtasks are accepted by all team members 14. There are conflicting interests in our team regarding subtasks/subgoals *
Mutual Support (7)	15. The team members help and support each other as best they can 16. If conflicts come up, they are easily and quickly resolved 17. Discussions and controversies are conducted constructively 18. Suggestions and contributions of team members are respected 19. Suggestions and contributions of team members are discussed and further developed 20. The team is able to reach consensus regarding important issues 21. The team cooperate well

Effort (4)	22. Every team member fully pushes the teamwork 23. Every team member makes the teamwork their highest priority 24. The team put(s) much effort into the teamwork 25. There are conflicts regarding the effort that team members put into the teamwork ∗
Cohesion (10)	26. The teamwork is important to the team 27. It is important to team members to be part of the team 28. The teamwork has had a special significance for the team ** 29. The team members are strongly attached to the team 30. All team members are fully integrated in the team 31. There were many personal conflicts in the team ∗ 32. There is mutual sympathy between the members of the team 33. The team sticks together 34. The members of the team feel proud to be part of the team 35. Every team member feels responsible for maintaining and protecting the team
Balance of member Contribution (3)	36. The team recognizes the specific characteristics (strengths and weaknesses) of the individual team members 37. The team members contribute to the achievement of the team's goals in accordance with their specific potential 38. Imbalance of member contributions cause conflicts in our team ∗
Team members' Success (8)	
Work Satisfaction (4)	39. So far, the team can be pleased with its work 40. The team members gain from the collaborative teamwork 41. The team members will like to do this type of collaborative work again 42. We are able to acquire important know-how through this teamwork
Learning (4)	43. We consider this teamwork as a technical success 44. The team learn important lessons from this teamwork 45. Teamwork promotes one personally 46. Teamwork promotes one professionally
Team Performance (15)	
Effectiveness (10)	47. Going by the results, this teamwork can be regarded as successful 48. All demands we have set for the project have been realized ** 49. From the project case description, the team has reached its goals ** 50. The performance of the team increases the understanding of methods in software engineering ** 51. The teamwork result is of high quality 52. The instructors are happy with the quality of the result of the teamwork ** 53. The team is satisfied with the teamwork result 54. The product produced in the team, requires little rework 55. The product proves to be stable in operation 56. The product proves to be robust in operation
Efficiency (5)	57. The instructors are satisfied with the progress of the teamwork ** 58. Overall, the team works in an efficient way ** 59. Overall, the team works in a time-efficient way 60. The team is within schedule 61. The team stays within the scheduled time **

* Item was coded reversed
** Item was rephrased for this thesis

Appendix II – Interview Guide Students

Part	Question
Introduction	Present myself and the purpose of the interview Inform about anonymity and confidentiality
General	What was your team number? What was your project case? What is your main occupation (student or employed)? If employed: What is in your experience the main differences between this student project and work projects?
Teamwork and Project	How would you describe the communication in your team? How frequently did you communicate in your team? What form did the communication take (meetings, chats, on a whim)? How was information shared within the team? How did your team coordinate on tasks? How was your common understanding of the tasks and subtasks? How did you reach agreement on what should be done and by when? How did your teammates contribute in the project? How did you experience your contribution in the team? Can you explain if and how your team helped and supported each other? What was the common response when someone needed help? How would you describe the effort of the team in the project? How did you share the workload? How was the team and project prioritized by the team members? What was the motivation of the team? How would you describe the team spirit?
Course in General	What do you think about the way teams were formed in 2019? What were the pros and cons to the way teams were formed? Can you explain whether or not you got the support you needed from either your team or teaching assistant? In what area of the course did you learn the most? How have you applied this in later courses and/or work? What is your overall impression of the course? What could have been better in the course?
Closing	Is there anything you would like to add? Thank the interviewee for taking the time to talk to me

Appendix III – Interview Guide TA

Part	Question
Introduction	Present myself and the purpose of the interview Inform about anonymity and confidentiality
Teams	How many teams did you have responsibility for in 2019? Can you explain how your communication with the teams was? How was the diversity in your teams in terms of study program and gender? What, in your opinion, defined the high-performing teams? What, in your opinion, defined the low-performing teams?
Course	How would you describe your role in the course? What was your most common activity in the course? How did you act as a Product Owner? What were the challenges in the course in 2019? How has action been taken to improve these challenges from 2019 to 2020? What is the most important aspect of this course? What do you think/hope the students have learned?
Pilot to the Course	What was your role in the pilot to this course? What was the greatest "take away" from the pilot course? What were the pros and cons to the way teams were formed?
Closing	Is there anything you would like to add? Thank the interviewee for taking the time to talk to me

Appendix IV – Evaluation Criteria

Criteria	% of score
Title, abstract, team presentation and introduction	4
User documentation	11
Requirements analysis, modelling, object-oriented design, patterns	15
Technical product documentation	15
Testing and test documentation	8
Process documentation and reflection on software development process	19
Overall impression, language, context	12
References, sources, appendices	4
Product and functionality	12

Table from Løvold (2020)

Appendix V – Python Script

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Read data and calculate correlation
data = pd.read_csv('averages.csv', sep=';', index_col=0)
correlation = data.corr()

# Plot data
figure = plt.figure()
axes = figure.add_subplot()
matrix = axes.matshow(correlation, cmap='coolwarm', vmin=-1, vmax=1)
figure.colorbar(matrix)

# Add ticks
ticks = np.arange(0, len(data.columns), 1)
axes.set_xticks(ticks)
plt.xticks(rotation = 90)
axes.set_yticks(ticks)

# Labelling ticks
axes.set_xticklabels(data.columns)
axes.set_yticklabels(data.columns)

plt.show()
```