



An Exploration of Participation in Student Software Teams

How Programming Experience Affects Participation in Student Software Teams and Leveraging Repository Insights for Monitoring Participation

Merel Steenbergen

An Exploration of Participation in Student Software Teams

How Programming Experience Affects Participation in Student
Software Teams and Leveraging Repository Insights for
Monitoring Participation

by

Merel Steenbergen

to obtain the degree
Master of Computer Science
at Delft University of Technology,
Faculty of Electrical Engineering, Mathematics and Computer Science
to be defended publicly on Tuesday October 8, 2024 at 14:00.

Student number:

Project duration: February, 2024 – October, 2024

Thesis committee: Prof. dr. M.M. Specht, TU Delft, Responsible Professor
Dr. ir. E. Aivaloglou, TU Delft, Daily Supervisor
Dr. G. Iosifidis, TU Delft, Committee Member

Cover: Gource Visualization by The Alpha Blenders procedural
computer graphics

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Acknowledgements

I've always known that crunching numbers or writing code behind a desk for multiple hours a day might fulfil me for a while, but would not satisfy me in the long run. I've realized, increasingly in the past years, that the people around you make or break anything; I want to work with a diverse group of people. Teaching is one of my passions, but I know I do not want to teach full-time either. Centering my master's thesis around education was incredibly rewarding. So much so, that I even want to continue research in this field. I'd never imagined I'd love doing research, but here I am.

I did require some help to get to this point. For her amazing supervision and encouraging words, I'd like to thank Dr. Fenia Aivaloglou. She gave invaluable feedback and pushed me when I needed it. For helping me through my first steps in qualitative research, I want to thank Prof. Esther Medina-Ventura and Prof. Marcus Specht. They took a critical look at the observations I performed and the statistical tests I performed during this thesis, both of which I did not have as much experience with as I thought.

In my personal life, Julian has been my rock. He provided me with food and coffee when it became time to start crunching in the home stretch of this thesis. My parents and sister have been incredibly supportive as well, taking every call I sent their way, even though they live at the other side of the country. Furthermore, I have been blessed with a lot of great friends who checked in on me and allowed me to rant to rationalize.

I feel incredibly grateful for all the people in my life who supported me during these eight months. Coming back to my original statement: *The people around you make or break anything*. And they definitely made this thesis.

*Merel Steenbergen
Delft, October 2024*

Abstract

Group work is an integral part of Computing Education, but introduces problems when students either fail to participate enough or dominate, not allowing their team members to contribute. This research focuses on identifying the participation level of the students, allowing the teacher or teaching assistant who monitors the group throughout the project to identify and resolve problems. The software metrics that can be extracted from the code repository are often used in this monitoring process. Therefore, this study investigates the correlation between the software metrics and the participation level. Furthermore, this study aims to understand the correlation between programming experience and the participation level in the project.

To this end, four project groups were observed over several weeks, followed by interviews with their teaching assistants. A questionnaire assessed each student's programming experience and collected data about confounding factors. A selection of software metrics to collect was made based on the literature and the teaching assistant interviews. Per contributor, we collected and analysed the number of lines of code, amount of issue comments, amount of MR comments, amount of commits, cumulative cyclomatic complexity, maximum cyclomatic complexity, amount of pipelines triggered, and pipeline failure ratio.

Of the twenty students in the observed groups, eight were classified as high participation, and six were identified as low participation. No significant correlation was found between participation level and gender, age, nationality, GPA, or group familiarity. Programming experience also did not predict participation, suggesting that a lack of experience is not a valid reason for low engagement. Furthermore, the number of lines of code students wrote, the number of commits students made, and the number of comments that students posted on merge requests are correlated with the participation level. This means these metrics can be used for monitoring student software groups and can indicate more than just code contribution.

Contents

Acknowledgements	i
Abstract	ii
Nomenclature	v
1 Introduction	1
2 Background and Related Work	2
2.1 Group Roles	2
2.2 Social Loafing	3
2.3 Dominance	4
2.4 Team Diversity	4
2.5 Programming Experience	5
2.6 Software Metrics	5
3 Research Questions	7
3.1 Hypotheses	7
3.1.1 Programming Experience	7
3.1.2 Software Metrics	8
4 Methodology	9
4.1 Participants and Recruitment	9
4.2 Data Collection	10
4.2.1 Questionnaire Design	10
4.2.2 Observative Study	12
4.2.3 Teaching Assistant Interviews	12
4.2.4 Software Metrics	13
4.3 Data Analysis	14
4.3.1 Participation Level	14
4.3.2 Participant Characteristics	16
4.3.3 RQ1: Programming Experience	16
4.3.4 RQ2: Software Metrics	16
4.3.5 Effect Size	16
4.3.6 Qualitative Analysis	17
5 Results	18
5.1 Participation Level	18
5.2 Participant Characteristics	19
5.3 RQ1: Programming Experience	20
5.3.1 Effect Size	20
5.3.2 Qualitative results	21
5.4 RQ2: Software Metrics	22
5.4.1 Qualitative Results	22
5.5 Exploratory Findings	24
5.5.1 Member Familiarity	24
5.5.2 Dominance in Meetings	24
5.5.3 Interpersonal Dynamics	25

6 Discussion	26
6.1 Implications and Recommendations	26
6.2 Threats to Validity	27
6.2.1 Subjectivity of Participation Level	27
6.2.2 Participant Profile	27
6.2.3 Sample Size	28
6.2.4 Accuracy of Metrics	28
6.3 Recommendations for Future Work	28
6.3.1 Participant profile	28
6.3.2 Sample Size	29
6.3.3 Accuracy of Metrics	29
6.4 The Use of LLMs in this Thesis	29
7 Conclusion	31
References	32
A Questionnaire	36
B Teaching Assistant Interview Protocol	42

Nomenclature

List of Abbreviations

Abbreviation	Definition	Page number
CC	Cyclomatic Complexity	5
CCC	Cumulative Cyclomatic Complexity	5
CE	Computing Education	1
CS	Computer Science	1
FET	Fisher's Exact Test	16
GPA	Grade Point Average	10
LOC	Lines of Code	5
MR	Merge Request	12
SLTQ	Social Loafing Tendency Questionnaire	11
TA	Teaching Assistant	1
VCS	Version Control System	5

1

Introduction

Collaboration competencies are core skills across many study disciplines and are increasingly important now that knowledge is becoming more specialised. Every student in Computing Education (CE) will encounter group work during their studies. It allows students to practice effective forms of communication and collaboration. These will be useful in their careers, which often entail working in multi-disciplinary teams and conveying ideas and concepts to colleagues who are not educated in computing disciplines [1, 2, 3]. In fact, it has become an expectation from the industry that graduates from Computer Science (CS) programs possess strong group work abilities [4, 5].

While group work has many benefits, certain undesirable behaviours, such as lack of participation or poor communication, can hinder the learning outcomes. Detecting these behaviours early is critical for teaching students more effective collaboration strategies. However, detecting participation levels can be complex as it is an aspect of human behaviour, which cannot be measured by isolated actions. To determine the participation level, a consistent pattern in the behaviour of students needs to be detected. This study seeks to understand the key factors that influence student participation in group projects.

In this work, we hypothesise that some students claim less programming experience as the reason to participate less in the project. Research suggests that programming experience only impacts the performance in the first introductory CS course [6, 7]. However, participation in a group project is not measured solely using the code contributions, but also includes collaborative tasks such as planning and asking for help. Therefore, the first research question is: ‘How is the level of participation related to prior programming experience in student software development project groups?’

A tutor or Teaching Assistant (TA) often monitors group projects since the study programmes are so large that a single teacher cannot monitor all groups simultaneously [8]. This can lead to inconsistency in the intensity of guidance and grading. Therefore, it would be preferable to have more objective measures to detect the participation of the students working on the project. One of the main advantages of software development projects is the abundance of data to use for detection. If the code is stored in a shared repository, this can provide insights into developer behaviour. The metrics that are gathered from code repositories are already used for monitoring and grading student software development projects [9, 10, 11, 12]. This raises the second research question: ‘What is the correlation between the participation level per student and the software metrics in software development projects?’

By addressing these questions, this study aims to contribute to a deeper understanding of what affects participation in group projects and how it can be measured, ultimately leading to recommendations for monitoring students in project groups. Chapter 2 will go deeper into the literature related to these topics. Chapters 3 reiterates the research questions and poses hypotheses. Chapter 4 provides an extensive explanation of the methodology of gathering and analysing the data. Chapter 5 shows the results, which Chapter 6 discusses and finally Chapter 7 concludes this thesis.

2

Background and Related Work

Group work is a key part of CE. It has positive effects on programming skills and, although to a lesser extent, on course satisfaction [13, 14]. This effect is enhanced in interdisciplinary projects since computer science can often be applied to other subjects. Teachers should leverage these connections so students know what they will do, learn, and contribute. Aligning these expectations fosters joint progress in multidisciplinary projects [15].

In the early days of using computers in classrooms, researchers and teachers expressed concerns about the potential negative impact of computers on students' learning. They expected that students would focus on the computer instead of collaborating and thus learn less while working in groups with computers. However, it was found that using a computer when working in small groups does not impact the understanding of course topics and material [16]. Computers have become an integral part of modern classrooms, and collaborative work is suggested to have a positive impact on programming skills. Coding together (pair programming) before working on problems individually allows students to learn more complex content [17].

Human behaviour is complex, so many factors impact collaboration. The following sections will explain factors that affect group work in computing education, providing the foundation for the research questions.

2.1. Group Roles

Effective collaboration is influenced by the roles that individual members adopt in the team [18]. Strijbos and de Laat [19] have identified eight participative stances, or roles, regarding group work as shown in Figure 2.1. These are divided into four stances appearing in large groups (>7) and four appearing in small groups. As can be seen, two aspects other than group size have an impact on these stances: Orientation and effort. This research will focus on smaller groups of five students, so only the four stances for smaller groups are relevant. They are the Captain, the Over-rider, the Free-rider, and the Ghost.

The Captain is usually the leader of the group and keeps the group in mind while working on the project. The term Captain implies that there can only be one in the group, but this is not true. Captains are effective collaborators and a team composed fully of Captains will likely be effective in group projects. The Free-rider tries to get a benefit, such as a passing grade, with as little effort as possible. The Ghost is a passive and absent person, either due to disinterest or external circumstances. It can be difficult to see whether someone is a ghost, since they might exhibit behaviour of another role when they are present in the group, but are too often absent to unambiguously assign that role. Finally, the Over-rider tries to control the group process and final product by pushing the other group towards their proposals. These will be used as a basis for defining group roles in this research, which will be discussed in detail in Section 4.

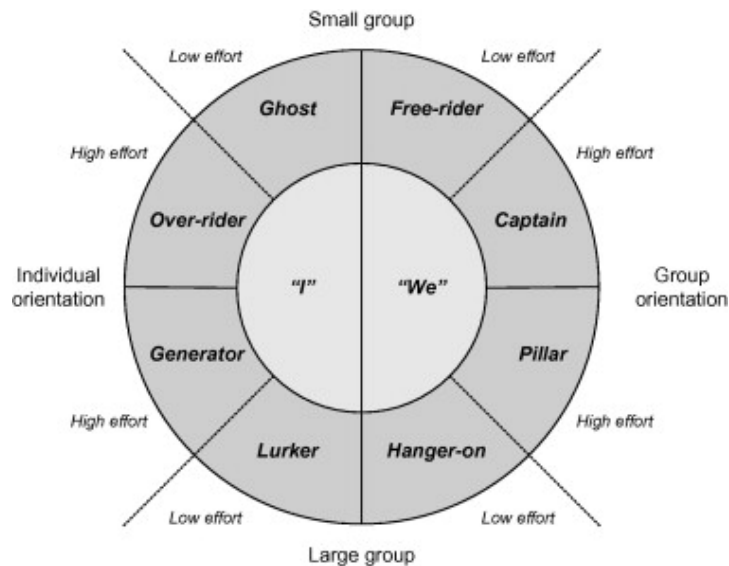


Figure 2.1: Eight participative stances, by Jan-Willem Strijbos and Maarten F. de Laat, 2009 [19]

Notably, the only positively connotated role is the captain, the other three imply a negative impact on group work. Excessive or insufficient effort can result in dominance or social loafing, respectively. Both significantly affect collaborative work; students have appointed social loafing as the main source of reluctance to work in groups [20, 21]. The following sections will discuss these phenomena and their impact on group work.

2.2. Social Loafing

Maximilien Ringelmann, a French agricultural engineer, coined the term 'social loafer' around the late 1800s when he conducted research involving students pulling a rope. He found that the students pulled harder when alone than in a group. This is called "The Ringelmann Effect" [22]. It appears in group work quite often, even in the present day and is referred to as 'social loafing' or 'free-riding'. An important distinction is that social loafing is not necessarily putting less effort into a task than team members; it means putting less effort into a group task than into individual tasks. Free-riding and social loafing are synonymous and will be used interchangeably in this thesis.

The presence of a social loafer severely negatively influences the group work experience of the other team members, as is often reflected in course evaluations. Smaller groups, smaller project scopes, self-selected groups and regular peer reviews are strategies to reduce social loafing, therefore increasing student satisfaction with their peers, which in turn positively influences students' perception of grade fairness [23, 24, 25]. Students often lack effective approaches to address challenges within team projects, and letting them work together in groups is insufficient to teach these skills [26, 27]. Collaborative strategies and skills have to be actively taught and integrated into course design. Preferably, these strategies are taught to all students, but some groups might require more guidance or intervention by course staff [28]. Weekly surveys are a valuable and suitable tool to identify those groups. These surveys have been designed to proactively identify teams that may not be on track to meet learning goals [29].

Some students show strategies for dealing with social loafers that are not effective. A common pitfall is that they overly prioritise the final product, leading them to take over the work of social loafers. Especially in STEM courses, there is a more significant emphasis on the final product, as that is often delivered to a client. It is believed to be better to teach students how to provide constructive, non-accusatory feedback to deal with social loafing [30, 31]. Students can learn from professionals, who use the team infrastructure to solve problems by assigning earlier deadlines, breaking the project into smaller chunks, and reassigning responsibilities. These strategies have been proven effective in avoiding and solving the problem of social loafing to an extent [28].

This reflex of taking over the work of social loafers is natural. The social compensation hypothesis dictates that people will do the opposite of social loafing when they expect co-workers or peers to perform poorly on important tasks [32, 33]. This means that they will put more effort into the group task to compensate for the expected lack of quality work by others. This effect can lead to dominant behaviour that can also be detrimental to the collaborative process.

2.3. Dominance

Dominance is seen as the opposite of social loafing, but this does not mean it positively affects collaboration. Some students take control of the process or the final product and prevent their team members from contributing. They try to maintain control over the results and compliance of their team members [34]. These students are called dominant or over-riders and can be as bad for the collaborative process as social loafers. It has been shown that dominance is a major obstacle when it comes to the creativity of virtual teams [35]. It is important to note that leadership and dominance are not the same. Although a leader can be dominant and vice versa, it is not necessarily the case. Leadership is associated with positive behaviours enabling the team members to work effectively, while dominance is about control [36].

In this research, this phenomenon will be referred to as over-riding, since dominance is not consistently flagged as a negative trait across the literature. Dominance is a subcategory of extraversion, defined by striving for superiority, control, and influence over others [37]. Extraverted and therefore more dominant students in group meetings affect the peer reviews. Most students are consistent when giving feedback to themselves and others. However, the least dominant student awards significantly more points and contributions to others than to themselves, while this may not reflect the actual situation [38].

2.4. Team Diversity

Other factors can influence the team performance in CE projects. Effective group members exhibit certain traits, such as willingness to work together, attitude towards uncertainty and other group members, and conscientiousness [37]. An important trait is emotional stability, which refers to a lack of anxiety and nervous tendencies. Self-esteem is a key contributor to emotional stability and is synonymous with self-efficacy. Other than self-efficacy, mathematics ability and comfort level are success indicators in CE [39].

Additionally, team diversity plays a significant role in team performance. However, diversity only benefits teams in which members focus on learning, also called a learning approach orientation [40]. Another approach is the performance approach orientation, which is a focus on demonstrating competence by outperforming others, associated with the over-rider role. Furthermore, member familiarity and willingness to share new information are positively correlated with team performance [41]. This is confirmed by Driskell et al. who describe openness to experience and the flexibility to adjust to different situations as important traits for successful team members [37]. While agreeableness is a beneficial feature, students should be wary of becoming naive and letting ineffective team members control their progress. Others should be approached with trust and given the benefit of the doubt, but if they do not reciprocate that trust should not be continued. Other research recommends forming groups based on student learning styles and characteristics [42, 43].

In addition, gender and programming attitudes have an impact on critical thinking skills and programming attitudes. Girls generally have higher critical thinking skills than boys in middle school [44]. However, they often have a more negative attitude towards programming, which can significantly affect their performance and experience when learning programming. While girls might be a valuable asset for problem-solving in group work, they might perform worse and enjoy the process less than their male counterparts where programming is concerned.

Later on, during their first year at the university, many female students in men-majority teams struggle with a lack of confidence and interpersonal obligations more than their male counterparts. They take on more non-technical tasks and their satisfaction with the collaborative work is lower [45]. Men in men-only teams also suffer from a lack of confidence, while this was not found in men in men-majority teams [46].

2.5. Programming Experience

Unsurprisingly, it has been proven that having previous programming experience is a significant success factor in introductory Computer Science (CS) courses. However, this effect is neutralised in subsequent CS courses [6, 7]. Gender does play a role in the success in these courses. Female students without programming experience perform similarly to their male counterparts, but female students with programming experience routinely outperform male students with experience in these courses [7].

Most of the research in this field focuses on measuring the effect of programming experience on performance in terms of grade or final product [47]. In group projects, difficulties students face are mostly related to technical skills; Team members lack experience or one group member takes control because they are highly skilled [14]. Since it has already been shown that dominance in group projects can come from a place of being a domain expert [34], it is likely that programming experience affects group dynamics as well. However, usually, the effect of the group projects on learning programming is measured. This plays a big part in one of the research questions.

2.6. Software Metrics

Most software development projects use a Version Control System (VCS) such as Git, which tracks previous software versions and allows developers to work on features separately. The repositories where the code is stored can provide us with a plethora of metrics about the collaborative process. Information about developer behaviour, such as often being the first to reply to a thread can provide valuable insights. These metrics, along with source code metrics such as Lines of Code (LOC), can be extracted from the VCS automatically and can represent developer productivity in the project when used appropriately [48]. However, Campbell's law applies: The more important a metric is in social decision-making, the more likely it is to be manipulated [49]. In this context that means that grading solely based on metrics would lead students to contribute to just the metrics instead of participating to the project in a meaningful way. An example would be adding more whitelines to the code to increase the amount of LOC, which is not related to the quality of the project or the collaboration.

In coursework, these metrics can be analysed to see if the task distribution among the team members is equal. This can assist with grading [9] and it creates a basis for setting expectations that better match the software industry [10]. It can also be used to identify different teamwork styles to understand habits [11] or as a tool that can be used for guidance during the group process [12].

Combining different metrics about the source code to measure developer contributions and productivity has been a popular research topic. The most used and most researched metric is Lines of Code (LOC), presumably because it is easily extracted. It has been shown to be correlated with the course grade [50]. The LOC metric can be made more accurate by removing white lines, removing lines which solely contain punctuation, and correctly attributing lines to the original author after refactoring [9, 51]. However, it has also been shown that these revisions might not really be necessary and any form of LOC can provide enough information about the developers [52]. Another metric that is often used is Cyclomatic Complexity (CC) [53, 54], which is a qualitative measure indicating the complexity of the code. This can be measured as Cumulative CC (CCC), the maximum CC per function, or the average CC added per contributor. In general, adding much complexity to the code in total is seen as a positive thing, but having high CC for one method is negative. This implies that the code should be refactored into multiple methods to enhance maintainability [55]. In fact, some research suggests that complexity metrics are strongly correlated with the LOC, and therefore LOC is the only metric necessary to predict development effort [56]. Still, students feel that none of the metrics should be used in isolation, and we need to be aware of the impact of using them [57].

Since programming assignments offer specific types of data, they offer unique monitoring techniques. This impacts grading as it allows for the inclusion of quality measures such as code quality, but can also introduce problems in groups of varying levels of prior programming experience. Students stress that it is most important that the setup and grading of group assignments are transparent and tailored to the learning goals. As long as these criteria are met, students think it is important teachers use all available and relevant information as assistance for monitoring and evaluation [58].

Since CS courses tend to have many students, manually checking all repositories for lack of contributions can be infeasible. Therefore, these metrics are used as a selection tool to flag groups that will be kept under closer supervision by course staff. The threshold value of these metrics that students need to adhere to, such as a certain amount of lines of code or percentage of contribution, is usually based on previous experience and estimations instead of research.

Falsely flagging some students as social loafers is not a substantial issue since this will result in a tutor observing the group more closely. However, false negatives can be detrimental to the group process as undetected social loafers significantly impact the experience of their group members. Currently, the software metrics are already used to estimate student contribution and identify social loafers. However, there has not been much research towards the correlation between these metrics and the actual participation level of the students.

3

Research Questions

The literature regarding programming experience has found contrasting results. On the one hand, multiple sources mention that the amount of programming experience affects the results in introductory programming courses, but not anymore in any subsequent courses [6, 7]. On the other hand, being dominant in a group can come from a place of domain expertise, which could indicate programming expertise in software development projects [34]. From this inconsistency, the first research question arises:

RQ1 How is the level of participation related to prior programming experience in student software development project groups?

Software metrics are often used to assist in the monitoring and grading process in software development groups. There has been much research towards ensuring the accuracy of these metrics [9, 50, 51, 55]. However, it has not been researched how these metrics reflect the participation level of students. Therefore, the second research question in this thesis is:

RQ2 What is the correlation between the participation level per student and the software metrics in software development projects?

The participation level is based on participation in project meetings and judgement by the TA guiding the project. The TA has insight into the code contributions, which implicitly impacts the participation level assigned to the students. The participation level will be measured as a categorical variable with three levels: Low, medium, and high.

3.1. Hypotheses

Taking into account the literature of the previous section and experiences from group work, hypotheses for these research questions arise. These will be used to guide the analysis and help determine the correlations in the data.

3.1.1. Programming Experience

As mentioned in the background, both age and gender can affect dominance and thus participation level [34]. Other factors that can influence the group process and dynamics are performance in previous courses, diversity and expectations, and group member familiarity [40]. The confounding factors found in the literature will be taken into account when collecting data. The null hypothesis and alternative hypothesis are formulated as follows, where H_1 is the main hypothesis about RQ1 and the others are used to rule out confounding factors:

H_{1_0} Programming experience is not related to the participation level in the project.

H_{1_a} Programming experience increases as the participation level in the project increases.

$H_{0_{Age}}$ Age is not related to the participation level in the project.

$H_{a_{Age}}$ The participation level in the project increases as age increases.

$H_{0_{Gender}}$ Gender is not related to the participation level in the project.

$H_{a_{Gender}}$ The participation level in the project is larger for students who are the dominant gender in their group.

$H_{0_{GPA}}$ GPA is not related to the participation level in the project.

$H_{a_{GPA}}$ The participation level in the project increases as the GPA increases.

$H_{0_{nationality}}$ Nationality is not related to the participation level in the project.

$H_{a_{nationality}}$ Teams with more international students will have higher participation level students.

$H_{0_{mf}}$ Member familiarity is not related to the participation level in the project.

$H_{a_{mf}}$ The participation level in the project increases as member familiarity increases.

3.1.2. Software Metrics

Virtually an infinite number of combinations of metrics could be collected to analyse. Therefore, a selection has to be made of metrics to collect and analyse. The null hypothesis is as follows:

H_{2_0} The metrics are not correlated with the participation level.

The alternative hypotheses are based on the individual metrics, so they are listed after the selection of metrics has been made, which is done in Section 4.2.4.

4

Methodology

Three things need to be collected in order to answer the research questions: The participation level, the programming experience and the software metrics. Multiple data sources were used to collect the participation level: Observations, interviews with the TAs monitoring the groups, and a questionnaire. This questionnaire was also used to gather the programming experience and confounding factors as identified in the background. The selection of software metrics to collect was based on the TA interviews and the literature. They were collected from the GitLab repository, which contains the code the students wrote.

4.1. Participants and Recruitment

Gaining insights into the dynamics of Software Development teams can only be done by observing them. To facilitate this research, students of the Delft University of Technology were recruited as participants, specifically students enrolled in the project course CSE2000 Software Project, which is worth 15 ECTS¹. Participation in the research was voluntary and four groups out of roughly ninety joined the study. These groups each consisted of five undergraduate students pursuing a Bachelor's degree in Computer Science and Engineering, with the majority being in their second year of study. This means all students should have experience with object-oriented programming, some experience with functional programming, and should know about problem-solving strategies. Furthermore, they have all completed a software development project in their freshman year where they have encountered Scrum, software testing, and GitLab.

Before the course started, all available projects were listed on an online forum. The students replied to the five projects they found most interesting, either as a group or individually. The projects were divided using an assignment algorithm that attempts to assign the projects according to preference. Students who replied to projects individually were grouped based on project preference. In the end, most of the groups are self-selected and some are randomly assigned based on project preference.

Over the course of ten weeks, these students carry out a project to solve a real-world problem with software development, commissioned by a client and guided by a coach and TA from the university. The student team will have to show problem and risk analysis, requirement engineering, implementation, testing and validation, and collaboration. The code the students write will be stored in a GitLab repository², which is managed by and shared with the course coordinators. During the project, the students will have weekly meetings with the TA for guidance, who will be monitoring their code repository as well. They will be learning technical writing, teamwork, and responsible computer science. These skills are shown during the midterm presentation in week five, the final presentation in week ten, and in the final report. Furthermore, Scrum³ is used as an agile collaboration technique, using the GitLab issue board functionality as a Kanban board. The final grade for the course is composed of 15% final report, 20%

¹https://studiegids.tudelft.nl/a101_displayCourse.do?course_id=64444

²<https://about.gitlab.com/>

³<https://www.scrum.org/resources/what-scrum-module>

design, 35% product, 20% process, and 10% presentation. The grade is normally assigned to a full group, except in cases where individual adjustments are in order as determined by the teacher, based on input from the TA, peer reviews, and code contributions. The course includes two moments for peer review, both anonymously and non-anonymously, but these are formative and are not explicitly be used for grading. However, the TA might use the peer reviews as a sign that intervention is necessary and implicitly partially base grades on them.

Unfortunately, the ethics approval to start participant recruitment was only obtained after the project had already started. Therefore, participant recruitment took place from the third week of the project. This was done by advertising the research after the mandatory sessions and asking students if they would like to join. The first data collection took place in week six out of ten of the project and ended in the beginning of week nine. After the project ended, in week ten, interviews with the TA have been conducted.

4.2. Data Collection

Three tools were used in the data collection: A questionnaire, observations, and interviews. First, the questionnaire is described. Then, the setup of the observation study is described. Then, the TA interview protocol is discussed. Finally, the selection and collection of the software metrics are explained. Section 4.2.4 also includes the alternative hypotheses to H_{20} .

4.2.1. Questionnaire Design

A questionnaire was created with three goals: (1) to determine the programming experience of the students, (2) to get a profile of the students, so possible confounding factors that were found in the background can be tested for effect on the participation level, and (3) to provide an extra basis for determining the participation level. The next sections will explain the three aspects of the questionnaire. The full questionnaire can be found in Appendix 7. It was created in the TU Delft environment of Qualtrics and included the consent form detailing which data would be collected. It was communicated that completing the questionnaire would take roughly 10 minutes. Still, not all participating students immediately filled in the survey after receiving it, some required (multiple) reminders. Before the survey was administered to the participants, it has been reviewed by peers of the researcher to gather feedback. Based on this, some corrections in spelling and grammar were made. Clarifications that were added to the questions of the standardised instruments are discussed below. The survey was administered to the participants in the first week of data collection, corresponding to the sixth week of the project.

Student Profile

Age, gender, performance in previous courses, nationality, and member familiarity were identified as confounding factors in Section 2 [34, 41, 37]. To measure these, they have been included in the questionnaire. The first factor is age, for which respondents were asked to provide an integer value. The second is gender, where a text box was used instead of a dropdown menu to accommodate the wide range that students may identify as. The same was applied to nationality, allowing students born in one country, but raised in another to include both in their answers. Furthermore, a question about the students' Grade Point Average (GPA) has been included. A multiple-choice format with ranges was chosen since many students do not know their GPA by heart and calculating it would be time-consuming. Familiarity with group members was also assessed using multiple-choice options, while course familiarity was measured with a simple yes/no question.

Programming Experience

A standardised survey, developed by Feigenspan, Kästner, Liebig, Apel, and Hanenberg [59], has been adopted to evaluate participants' programming experience. This survey was created using insights from existing research on assessing programming experience. It has been tested in a controlled experiment with 128 students and was found to be reliable, even when relying on self-assessment. This indicates that self-evaluation of programming capabilities, as opposed to using a programming test or peer-reviewing, does not significantly affect the results when using this tool.

The survey asks the students about their experience with specific programming languages. Originally, these were Java, C, Haskell, and Prolog, as can be seen in Figure 4.1. Since Prolog is not part of the current curriculum, but Python, Javascript and C++ are, the Prolog question has been replaced by

questions about these three languages. Furthermore, a clarification has been added that the size of software projects is defined as the number of lines of code, since that was unclear in the original survey question. The question about the size of professional projects was slightly rephrased, based on peer feedback stating it was unclear if this question referred to students' own contributions or the total size of the project.

Source	Question	Scale	Abbreviation
Self estimation	On a scale from 1 to 10, how do you estimate your programming experience?	1: very inexperienced to 10: very experienced	s.PE
	How do you estimate your programming experience compared to experts with 20 years of practical experience?	1: very inexperienced to 5: very experienced	s.Experts
	How do you estimate your programming experience compared to your class mates?	1: very inexperienced to 5: very experienced	s.ClassMates
	How experienced are you with the following languages: Java/C/Haskell/Prolog	1: very inexperienced to 5: very experienced	s.Java/s.C/s.Haskell/s.Prolog
	How many additional languages do you know (medium experience or better)? How experienced are you with the following programming paradigms: functional/imperative/logical/object-oriented programming?	Integer 1: very inexperienced to 5: very experienced	s.NumLanguages s.Functional/s.Imperative/s.Logical/s.ObjectOriented
Years	For how many years have you been programming?	Integer	y.Prog
	For how many years have you been programming for larger software projects, e.g., in a company?	Integer	y.ProgProf
Education	What year did you enroll at university?	Integer	e.Years
	How many courses did you take in which you had to implement source code?	Integer	e.Courses
Size	How large were the professional projects typically?	NA, <900, 900-40000, >40000	z.Size
Other	How old are you?	Integer	o.Age

Integer: Answer is an integer; Nominal: Answer is a string. The abbreviation of each question encodes also the category to which it belongs.

Figure 4.1: The Programming Experience Questionnaire, by Eirini Kalliamvakou et al. (2009) [59]

Social Loafing Tendency Questionnaire

While searching for measurement instruments for participation level, the Social Loafing Tendency Questionnaire (SLTQ) was the most similar to what was needed [60]. This tool has been validated by research and has been tested for reliability. It can be used to predict social loafing, which means that students who score high on social loafing tendency will likely show significantly less effort in a group task than in an individual task. This part of the questionnaire contains seven statements about group work that students answer on a Likert scale of 1 (strongly disagree) to 5 (strongly agree) which can be found in Figure 4.2. The SLTQ was added to the questionnaire to support the subjective assignment of participation level to the students. First, some of the answers are reversed to manage the use of negation. Then, the answers to the statements are added to each other and scaled to a score between 0 and 1.

-
1. In a team, I am not indispensable
 2. In a team, I will try as hard as I can^a
 3. In a team, I will contribute less than I should
 4. In a team, I will actively participate in the discussion and contribute ideas^a
 5. In a team, it is okay even if I do not do my share
 6. In a team, it does not matter whether or not I try my best
 7. In a team, given my abilities, I will do the best I can^a
-

Figure 4.2: The original statements of the SLTQ, by Xiangyu Ying et al. (2014) [60]

Two adjustments have been made to the SLTQ. The first statement contained a double negation 'In a team, I am not indispensable', which was changed for clarity. The first negation has been removed and its scoring has been reversed to account for this single instead of double negation. Furthermore, the literature shows that social loafers have a group orientation and need the group to get their desired result; a high grade [19], while dominant students have trouble trusting their group members and may prefer individual work over group projects [36]. Therefore, the final statement 'I prefer working alone over working in a group' was added. This statement is reversely evaluated, meaning that a higher score on this question, indicating a preference to work alone, decreases the tendency to be a social loafer.

4.2.2. Observative Study

Discovering group dynamics without observing the group process firsthand is complex. Therefore, in this research, four groups of five students each were observed to determine their participation levels. This section will discuss the experimental setup of the observation study. Information about the processing and analysis of this data can be found in Section 4.3.1.

Experimental Setup

In the course the participants are taking, groups have one meeting per week with their Teaching Assistant (TA) and work autonomously the rest of the week. The group meetings with the TA can provide insight into the progress and state of the final product. However, these TA meetings often serve as a checkpoint for the group and can be quite rehearsed. Most groups will organise collaboration sessions in which they will work together on their project.

Scrum is used as the collaboration framework, which includes sprint planning and sprint retrospectives in which task distribution is discussed and team reflections are made, which are more interesting in the context of discovering group dynamics than the TA meetings. Therefore, the collaboration sessions excluding the TA were chosen for observations. These observations were naturalistic, meaning the researcher sat in on the meetings to observe and record but did not intervene. The groups were offered a room on campus to work together as compensation for participating in the research, but none took this offer. Some groups opted to collaborate online rather than on campus. In that case, the audio of the online meeting was recorded. The main researcher was present during these sessions to record the audio and annotate (non-)verbal communication. The meetings took between 30 minutes and 1.5 hours each.

4.2.3. Teaching Assistant Interviews

To assess participation levels, the data collected from the observations was complemented with interviews with the TAs monitoring the participating groups. This was done because observing groups is quite time-intensive. A single researcher is performing the data collection and analysis, so a more scalable approach to gathering data is needed. Additionally, given the small sample, it is preferable to have multiple sources confirming the participation level, especially since it is a subjective measure.

Another approach would be to directly ask the students what they think their participation level is. However, relying solely on self-reporting could potentially introduce bias into the results, especially since the levels have positive and negative connotations. The course instructor suggested that the anonymous peer reviews conducted throughout the course can offer valuable insights. However, accessing those for research introduces complexities in the consent process due to the platform infrastructure used to gather them, making it infeasible to use the peer reviews for this study.

The TAs, who have access to the peer reviews and meet with the groups weekly, can use their insights to provide a relatively accurate depiction of the participation without divulging any student-specific information. Hence, after the final week of the course, TA interviews took place in favour of the other approaches. During this interview, the TAs were asked some contextual questions to determine their experience with monitoring project groups and any problems they had identified in the groups. Then, they were then asked to assign one role to their students; captain, over-rider, free-rider or ghost. Furthermore, they were also asked which software metrics they use to monitor and assess the groups, this information was used when selecting the metrics to collect to answer RQ2. The interview protocol can be found in Appendix B. The audio of the interviews was recorded and transcribed after the meeting. The results were compared among TAs and with the results from the observation study, which is discussed in Section 4.3.1.

4.2.4. Software Metrics

To answer the second research question, it needs to be determined which metrics to gather and how to gather them. The students work with GitLab as a VCS during their project, from which metrics can be extracted. Multiple tools exist to extract various metrics, so data is abundant. A selection of metrics was created based on the literature and the TA interviews.

Metric Selection

Since the goal of this research is to be applicable in practice, both considerations from the literature and the realities of the current situation have to be taken into account. Currently, the university uses its own tool that allows the course staff to compare members of a group and all groups of that year's project course. It shows the total amount of commits, issues created, merge requests (MR), comments, and an overview per group. This way, it is easier to identify groups that might be slacking, meaning that they may require more guidance. Notably, these metrics all focus on quantity and can thus easily be cheated. Students have been known to contribute to improve their metrics, while the goal should always be to improve the product and process as a group [57, 49]. This is more difficult with qualitative metrics, since improving the quality of the code is cannot be done by making empty commits.

As reflected by the literature, the most used metrics are Lines of Code (LOC) [50, 51, 52, 56] and Cyclomatic Complexity (CC) [53, 54, 55]. LOC is a quantitative measure which is suggested to be a good indication of the amount of contribution. However, it is an easy metric to cheat, since refactoring code is an easy task that can quickly rack up the amount of lines a student wrote. CC is a qualitative metric that indicates the complexity of the methods written by a student. There are multiple ways of approaching this metric. Cumulative Cyclomatic Complexity (CCC) measures the sum of the complexities of each function written by a student. This way, it is easy to see if someone just wrote constructors, getters, and setters, or if they actually contributed logic to the project. However, having a very high CC per function indicates low-quality code. Because they are often used and easily collected, LOC, CCC and maximum CC per contributor were all collected. When collecting the LOC, the amount of commits was automatically collected.

TAs mentioned using the activity in Git as a metric in their interviews. Therefore, the number of comments on issues and merge requests were also collected. Involvement in the code of other students seems to be a good indicator of the participation level of the students. Furthermore, since one TA mentioned pushes as valuable data points for measuring participation, the pipeline success ratio has been measured. To do that, the amount of pipeline runs triggered per contributor had to be collected first, which has been included as an additional quantitative measure.

The metrics that have passed consideration, but have not been collected in this research are: The amount of files changed, the number of MRs created and merged, test coverage, documentation quality, modularity, coupling, cohesion, the estimated time spent and the quality of the MR comments. The MR metrics were not collected because they are even easier to cheat than LOC. Documentation quality cannot be automatically measured, nor can the quality of the MR comments. One difficulty that had to be faced was that students could choose their own programming languages for their projects. This meant that measuring test coverage, modularity, coupling, and cohesion would require multiple tools. That, in combination with the fact that they are difficult to measure per contributor, led to the decision not to collect them. Since one of the flagged behaviours is 'invests a lot of effort and time in the collaborative task', it would be valuable to measure the time spent on the project. However, there is no way to measure the time spent accurately enough to use for grading [9].

One TA mentioned the MR comments as a good indicator of the participation level, specifically the quality of those comments. They expressed that many comments consisted of meaningless messages such as 'LGTM' (looks good to me). However, analysing the quality of comments is not an automated process and is time-consuming. Therefore, it was not performed in this study, but it will be discussed in Section 6.3

Ultimately, a combination of quantitative and qualitative metrics has been collected. The metrics collected per contributor are lines of code, amount of issue comments, amount of MR comments, amount of commits, cumulative cyclomatic complexity, maximum cyclomatic complexity, amount of pipelines triggered, and pipeline failure ratio. Below, the null hypothesis $H2_0$ is reiterated and the alternative hypotheses are formulated:

H2₀ The metrics are not correlated with the participation level.

H2₁ Students with a high participation level write more lines of code than those with a low participation level.

H2₂ Students with a high participation level commit more often than those with a low participation level.

H2₃ Students with a high participation level trigger more pipelines than those with a low participation level.

H2₄ Students with a high participation level have a higher ratio of successful pipelines than those with a low participation level.

H2₅ Students with a high participation level commit more cumulative cyclomatic complexity than those with a low participation level.

H2₆ The maximum cyclomatic complexity of code written by students with a high participation level is higher than code written by those with a low participation level.

H2₇ Students with a high participation level comment more on issues than those with a low participation level.

H2₈ Students with a high participation level comment more on merge requests than those with a low participation level.

Metric Collection

Previously, GitInspector⁴ was used in the course to show statistics such as percentage of contributions, percentage of comments and LOC. However, this tool is no longer being maintained, so the university decided to discontinue its use. The research on LOC contradicts itself on the meaningfulness of filtering the LOC for white lines [9, 52]. However, the consensus is that it cannot hurt, so GitReporter was used to collect this LOC [9]. This tool attributes the lines of code that were edited slightly to the original author, reducing the effect of students 'cheating the metrics', however it does not take boilerplate code into account. All other metrics have been collected using the GitLab API⁵ and a simple Python script.

4.3. Data Analysis

All this raw data had to be processed and analysed. First, the protocol for determining the participation level of the students will be described. Then, the statistical tests used to test for correlations are described. Finally, the analysis of the qualitative data is discussed briefly.

4.3.1. Participation Level

The observed meetings resulted in roughly 12 hours of audio recordings. They were first transcribed to transform them into text that could be used for processing. Subsequently, all transcript fragments containing student behaviour associated with a group role were labelled. The labels that were used in the behavioural analysis to identify the participation level are based on the research into group roles by Strijbos and de Laat. The roles as they define them should not be used to assess group behaviour but to "assist teachers and students in recognising certain patterns in group member behaviour and enable them to respond to these behaviours" [19, p. 10]. Most research in this field focuses on group composition, but in this thesis, the roles will be used to understand the group dynamics to help teachers respond to them and monitor groups better. Table 4.1 shows the behaviours observed from the different roles, which have been used as the labels during data processing.

Strijbos and de Laat originally defined four participative stances that appear in smaller groups; The captain, over-rider, free-rider, and ghost. However, since this thesis is based on a small number of participants nuances between the roles might be difficult to differentiate. Therefore, it was decided to remove a dimension from the wheel as shown in Figure 2.1. If the orientation (individual/group) is removed from the equation, the remaining dimension is the effort. This has been defined as the participation level throughout this thesis. As can be seen in Figure 2.1, the captain and over-rider are high-participation roles and the free-rider and ghost are low-participation roles.

⁴<https://github.com/ejwa/gitinspector>

⁵<https://docs.gitlab.com/ee/api/rest/>

Behaviour	Role			
	Captain	Over-Rider	Free-Rider	Ghost
	High		Low	
Enforces (group) deadlines	x			
Tries to find consensus about how to approach the task	x			
Speaks in positive tone	x			
Contributions span more than simply a focus on the product	x			
Tries to keep the group on track	x			
Expresses desire for group building or socialising	x			
Invests a lot of effort and time in the collaborative task	x	x		
Assigns tasks to other group members	x	x		
Pushes other group members to adopt their proposal/idea/approach		x		
Has a focus on the final product		x		
Kick-starts collaborative activities		x		
Refers back to their proposals that were not used		x		
Takes a leading role in the composition of the final product		x		
Does not contribute much to meetings unless prompted			x	
Does not take on tasks eagerly (except maybe trivial tasks)			x	
Expresses interest for a high grade			x	
Promises they will do a task 'soon'			x	
Is urged to do as promised			x	
Fails to deliver on promises			x	
Can have a more positive experience compared to their group members			x	
Low participation in comparison to team members			x	x
Uses external factors as an excuse for not contributing			x	x
Ignores messages				x
Misses meetings				x
Does not take up (many) tasks				x
Contributions are unrelated to the discussion				x
Contributions are a reflection of own interests and problems				x

Table 4.1: Labels used for behavioural analysis

After labelling, the transcripts were split per group member to create individual overviews of communication. Then, the messages were reviewed in the context of each other in an attempt to determine behavioural patterns. This ensures that the behaviour is persistent and not just exhibited in one accidental instance. Since this is a solo project, only one researcher was performing this data processing and labelling. Whenever questions, discrepancies, or ambiguous data were encountered, it was discussed with a secondary researcher and the work was checked to ensure objectivity as much as possible.

After collecting all the data, it was combined to get the results. First, the participation level assigned by the observation study and the participation level assigned by the TA interview were combined. Most of the judgements matched. However, since there is no way to deal with disagreement objectively, the students for which the TA judgement of participation level mismatched the judgment of the observation have not been included in the analysis.

The SLTQ score is not used in the initial assignment of the participation level, but is used to confirm the results of the classification of the participation levels. A t-test has been used to find the correlation, after a Shapiro-Wilk test and Levene's test have been performed to test for normality and equality of variances, respectively. The hypotheses corresponding to these tests are:

$H_{0_{SLTQ}}$ There is no correlation between the SLTQ and the participation level in the project.

$H_{a_{SLTQ}}$ Students with a low participation level will have a high SLTQ score and vice versa.

4.3.2. Participant Characteristics

Since gender, nationality and GPA are categorical in the questionnaire and the sample size is small, a Fisher's Exact Test (FET) has been used to determine the correlation between these features and the participation level. Since member familiarity and GPA both have four categories, leading to a 2x4 contingency table, a variation of the FET was needed for feasibility. The Freeman-Halton extension was used in both cases.

However, not only gender is a confounding variable according to the literature, but the most dominant gender in the group can affect dominance as well [34]. Therefore, each student has received an extra label to indicate whether they are of the most dominant gender in their project group, which has been tested for correlation with the participation level using an additional FET.

Age is entered in the questionnaire as an integer, so it has been tested using a Shapiro-Wilk test for normality, a Levene's test for equality of variances, and, based on the results of these, a t-test or Mann-Whitney U test for correlation with the participation level. Additionally, the GPA has been converted to an integer using the median value of the range the students answered to confirm the results of the previous FET and tested with the same tests as the age for correlation with the participation level.

4.3.3. RQ1: Programming Experience

First, the results from the questionnaire were converted to a programming experience score. This was done by taking the answers to the two significant questions as indicated by the original paper [59]. Their participant population is similar to the one in this study, so the correlations should transfer. The relevant questions are: 'How do you estimate your programming experience compared to your classmates?' and 'How experienced are you with the following programming paradigm: Logical Programming?'. Both are answered on a Likert Scale from 1 (very inexperienced) to 5 (very experienced). These answers are added to each other and the result is scaled to a score between 0 and 1.

Significance tests were used to reveal any statistically significant differences between the programming experience score of high-participation and low-participation students. Since we are looking for the effect of a categorical variable on a numerical variable, logistic regression has been performed to do this.

4.3.4. RQ2: Software Metrics

To determine if there is a correlation between the participation level and the metrics, t-tests have been performed. Each metric was considered separately since there were not enough samples to perform ANOVA on the metrics as a whole. First, a Shapiro-Wilk test is used to test for normality. If that is found, Levene's test is used to test for equality of variance. In case of normality and equality, a students' t-test is used, in case of normality but not equality, a Welch's t-test is used, in case there is no normality, a Mann-Whitney U test is performed.

4.3.5. Effect Size

The effect size represents the change, measured in standard deviations, between the averages of two groups. This does not tell us if the correlation found is significant, but it does tell us how relevant it is. Therefore, Cohen's d has been calculated for the programming experience and software metrics [61]. In general, finding a Cohen's d of 0.2 indicates a small effect size, 0.5 indicates a medium effect size and 0.8 implies a large effect [61, p. 24]. This way, it is ensured that the research does not find a significant effect that would not have any implications in the course in real life. Since the sample size is quite small, the effect size can increase quite rapidly. Finding a significant correlation with a small effect size might mean the effect is too small to consider useful.

4.3.6. Qualitative Analysis

The observation data is rich with information beyond the participation level. Throughout the behavioural analysis, any remarks related to programming experience and software metrics were tagged along with the other labels. From this list of quotes, recurring patterns associated with experience and metrics can be identified. Furthermore, the TA interviews were cross-compared to find discrepancies in their assessments. Any additional patterns in behaviour were identified from the transcriptions of group interactions and examined in detail. Special consideration was given to the students for which the participation level determined by observation and the participation level assigned by the TA misaligned.

5

Results

This Chapter discusses the results of the statistical tests as described in the previous Chapter. First, the participation level assignment will be discussed. Then, the confounding factors are analysed. Then, the results of the tests directly relating to the research questions are described. These include the effect size and results from qualitative analysis relating to the research questions. Finally, some exploratory findings from the qualitative analysis are discussed.

5.1. Participation Level

Multiple TAs implied that some students did not fit a single role or showed behaviour belonging to two of the roles with the same effort (captain/over-rider & ghost/free-rider). Some students did not exhibit many of the behaviours flagged, or did exhibit flagged behaviour, but roughly in the same amounts for each role. It was concluded that a fifth role might be necessary when looking at the roles as participative stances. The TAs called it the worker or the implementor: A student who does their assigned tasks in time, but does not stand out in collaboration and is not a leader. These students could be assigned 'captains', but arguably, the name 'captain' implies leadership and does not fit these students. This situation corresponds to a medium participation level; the student does not show much behaviour that can be tagged but is too involved to be a ghost.

This medium participation level is not taken into account for the final testing for two reasons. First of all, the high and low participation levels of students both affect group dynamics and can become problematic. Too much participation could lead to dominance, while too little participation can indicate social loafing. Second, there were only two students marked as medium participation after the disagreements had been filtered, which is not enough to analyse as a separate category.

Out of twenty participating students, seven were identified as low-participation by the observation study, three were identified as medium-participation, and ten were assigned the high-participation label. The TAs identified six low-participation students, four medium-participation students, and ten high-participation students. After combining these results, eight students were assigned the high-participation label, two were assigned medium-participation, six were assigned low-participation level, and four were inconclusive. Since the medium-participation category contains only two students, it is too small to show in the figures. The risk of self-identification is too large. Therefore, the results are only shown for the high-participation and low-participation students.

The SLTQ has been used to support the division into participation levels. According to the hypothesis H_{aSLTQ} , these variables should be negatively correlated, meaning a high SLTQ score indicates a low participation level and vice versa. A students' t-test was performed to test this hypothesis, since the samples are normally distributed as demonstrated by a Shapiro-Wilk Test ($p = 0.347$). The t-test resulted in a p-value of 0.003 ($t(12) = -3.451$), meaning we must reject H_{0sltq} . In other words, students with a high tendency to become social loafers have been assigned a low participation score and vice versa. A boxplot showing the distribution of SLTQ scores can be found in Figure 5.1.

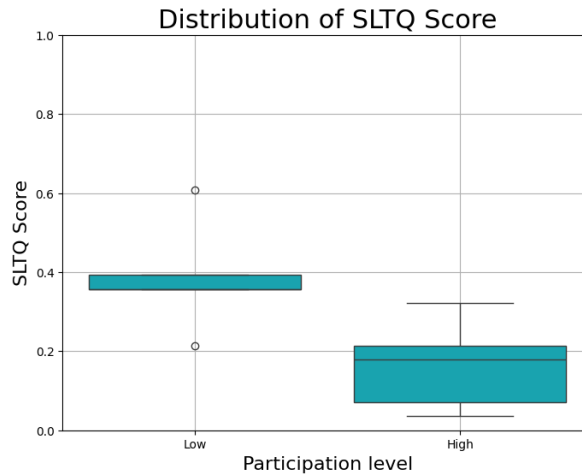


Figure 5.1: The distribution of the SLTQ score in participation level groups.

5.2. Participant Characteristics

In Chapter 2, age, gender, experience with other courses, and member familiarity were identified as confounding factors for the participation level. These were all collected from the questionnaire, which was completed by eighteen participants, of whom 5 were female and 9 were male. The average age was 20.9 years with a standard deviation of 1.64. The data was distributed normally as indicated by a Shapiro-Wilk test ($p = 0.083$) and the variances were equal as shown by Levene's test ($p = 0.166$). A student's t-test showed no correlation between the age and the participation level ($p = 0.169$). A visual representation of the data can be found in the boxplot in Figure 5.2. This means we can accept $H0_{Age}$.

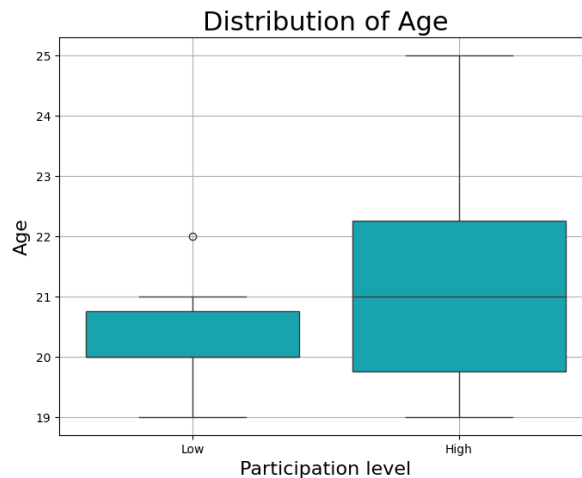


Figure 5.2: The distribution of the age in participation level groups.

A Fisher's Exact Test (FET) showed no correlation between the dominant gender in the group and the participation level ($p = 0.085$). Table 5.1 shows the number of participants who were the dominant gender in the group per participation level. Interestingly, there were no low-participation students who were not of the dominant gender in the group. Another FET showed no correlation between the gender, regardless of the dominant gender in the group, and the participation level ($p = 0.301$). Table 5.2 shows the contingency table this test was based on. Because of these results, we accept $H0_{Gender}$.

Part. level \ Dom. gender	Yes	No
	Low	6
High	4	4

Table 5.1: Contingency Table: Dominant gender in the group and participation level.

Part. level \ Gender	Male	Female
	Low	5
High	4	4

Table 5.2: Contingency Table: Gender and participation level.

Three more FETs were performed to determine the correlation between the participation level in the project and the nationality, member familiarity, and GPA. For member familiarity and GPA, the contingency tables have not been included in this thesis, since the frequencies in the tables would be quite low and all the data combined could have led to self-identification by research participants. The contingency table for nationality can be found in Table 5.3. Nationality was split into Dutch and International in this case, since the sample size was too small to take into account any other countries. No correlation was found between the nationality and the participation level ($p = 0.627$), so we accept $H_{0_{nationality}}$.

Participation level \ Nationality	Dutch	International
	Low	2
High	4	4

Table 5.3: Contingency Table: Nationality and participation level.

No correlation was found between the participation level and member familiarity ($p = 0.600$) and neither was it found between GPA and participation level ($p = 1.000$). Since the GPA had been collected in intervals, another test was included where the average value of these intervals was used as the GPA of the students and then a t-test was performed. The data was distributed normally ($p = 0.073$) and the variances were equal ($p = 0.709$). Still, no correlation was found with a students' t-test ($p = 0.372$). Therefore, we accept $H_{0_{GPA}}$ and $H_{0_{mf}}$.

One more participation characteristic was collected: Course familiarity. However, since all students were new to this course and none of them were taking it as a resit, no tests could be performed to determine its effect on the participation level. In this scenario that does not matter since the population is homogenous.

5.3. RQ1: Programming Experience

No significant difference was found between the programming experience score of the low-participation students ($M = 0.4888$, $SD = 0.1510$) and the high-participation students ($M = 0.4025$, $SD = 0.1464$), which can already be deduced when looking at Figure 5.3. The data were tested for normality using the Shapiro-Wilk test, which indicated no significant deviation from normality ($p = 0.0957$). Logistic regression revealed that programming experience is not an indicator of participation level ($p = 0.341$). Then, when testing for any other correlation, a students' t-test revealed no significant difference between the two groups, $t(12) = -0.045$, $p = 0.5174$. This means we accept H_{1_0} .

5.3.1. Effect Size

In addition to the t-test, the effect size was calculated using Cohen's d to assess the practical significance of the difference in participation levels. The effect size is $d = -0.0261$, which suggests that the difference between the two groups is negligible. This further supports the finding that programming experience has little to no practical impact on participation levels in this sample.

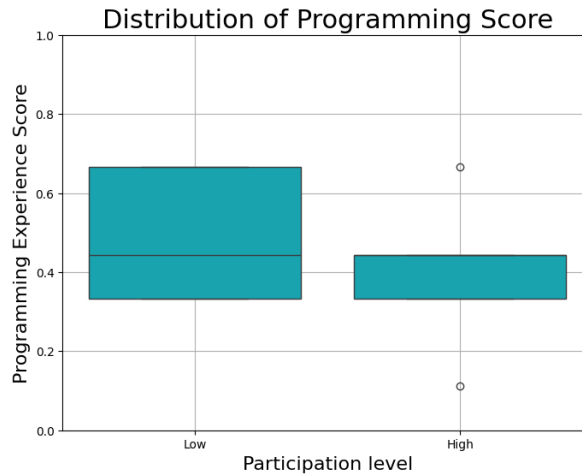


Figure 5.3: The distribution of the programming experience score in participation level groups.

5.3.2. Qualitative results

No groups explicitly discussed students' programming experience. However, multiple groups showed a dynamic that could be attributed to the individual members' experience. The high-participation students assigned tasks to low-participation students when they thought it necessary, as illustrated by these quotes by high-participation students about low-participation students: "For this issue, our experience will not help us. So maybe you could help us" and "You don't have enough issues and we have to give you some. Do you think you can learn the framework quickly enough?". The high-participation students were quick to notice the task they were assigning was 'easy' or 'should not take much time', as illustrated by these quotes: "I feel like it's not going to be as hard as you think" and "I think that's a perfect issue to do for them. It should be easy to do". Interestingly, the low-participation students did not take up tasks out of their own initiative, but waited for other students to assign them tasks. The high-participation students took up the role of assigning the tasks, sometimes before the low-participation students even had the chance to offer to take up the issue.

Differences between the participation levels become especially clear in when problems arise in their assigned tasks. High-participation students might run into problems, but discuss possible solutions with their group or ask for input so they can continue their task. Low-participation students announce they have run into a problem, which sometimes does not happen until other students ask them about problems as demonstrated by the conversation snippet below, where LP stands for 'low-participation'. The issue in this conversation is trivial and can quickly be resolved, but this student has been stuck on it for multiple days without notifying their team members or asking for help.

High-participation student: "LP, Did you have any problems that you want to bring up now?"

Low-participation student: "I think my machine is banned again, I'm having a lot of problems with accessing and pushing code"

High-participation student: "Did you try using the VPN?"

Low-participation student: "I didn't know that was a thing until yesterday".

Moreover, the difference in initiative-taking for solving problems becomes clear in the following three quotes by a high-participation student and two low-participation students, respectively. "I ran into some trouble and I didn't know how to fix it, but I found a very old repository on the internet and I think this will work", illustrating there was a problem but this student put in effort to solve it. "I don't know how to do this", followed by silence, indicating the student does not take the initiative to solve the problem or ask for help. Finally, "I have a problem: My computer is being stupid. Can't get it to work. And if I try this, I get an error. So I don't know how to fix that." is another trivial issue, for which the student could have at least tried to look up the error code on the internet.

5.4. RQ2: Software Metrics

A separate statistical test has been performed to find the correlation between each metric and the participation level. Since most metrics are likely correlated, it would be better to test all metrics simultaneously to avoid accumulating errors, but this research does not have enough samples to do so. As can be seen in Table 5.4, only LOC, commits, and MR comments are correlated with the participation level. This can also be seen in the boxplots in Figure 5.4. Therefore, we reject H_{20} and we accept H_{23} , H_{24} , H_{25} , H_{26} , and H_{27} .

Metric	Normal distr.	Equal var.	T-stat / U-stat	P-Value	Cohen's d
LOC	No	N/A	48.0	0.00033	1.60
Commits	Yes	Yes	3.13865	0.00428	1.82
Pipelines triggered	Yes	Yes	0.08733	0.19981	0.51
Pipeline failure ratio	Yes	Yes	0.27455	0.39416	0.16
CCC	No	N/A	26.0	0.42591	0.48
Maximum CC	No	N/A	15.0	0.89331	0.23
Issue comments	No	N/A	34.0	0.10874	0.18
MR comments	Yes	Yes	2.08263	0.02967	1.21

Table 5.4: P-values of the t-tests and Mann-Whitney U tests and Cohen's d for the metrics

Effect size

Effect sizes, measured using Cohen's d as shown in table 5.4, were calculated to assess the relationship between the metrics and participation levels along with the significance. The largest effect sizes were observed for the number of commits and LOC, both indicating large effects, suggesting that students who made more commits and wrote more lines of code tended to participate significantly more. Similarly, merge request comments also showed a large effect. A medium effect size was found for the number of pipelines triggered, while total cyclomatic complexity showed a small-to-medium effect. The other metrics showed small effect sizes. These effect sizes are similar to the significance found, meaning the more significant correlations also have a bigger effect.

5.4.1. Qualitative Results

Two groups explicitly addressed the metrics and tactics for balancing them, as illustrated by the quote "We have to make a plan of attack for the next sprint, with the main goal to all have enough code contribution". This happened during four separate meetings, two for each group. However, their reasons for doing so might not be what the course coordinators expect or prefer as illustrated by these two quotes by a high-participation student and a low-participation student, respectively: "It's only to please the ones grading us", and "The main point is to convince the TA that we actually did it". One group mainly considered the number of commits and merges as the main metric. The other group considered the LOC as the most important. To measure this outside of TA meetings, they tried to run some of the tooling themselves, as shown by "I'll try to figure out how to run GitInspector". Campbell's law became relevant here: Some students contributed to the metrics in non-meaningful ways, as illustrated by: "I'll fix some typos to create more merge requests for myself".

Generally, both groups employed the same strategy to manage imbalances: larger tasks, whether measured by LOC or commits, were assigned to members with lower contributions, while those with higher contributions worked on the report, as shown by "I will do this issue because I want more commits". Even though Campbell's law applies here, it still has the desired effect of students participating more in the project.

Both groups were advised by their TAs to balance their contributions. They did take the advice to balance the metrics, but kept realistic expectations: "I don't want anyone to fail because of low contribution, but it is technically everyone's own responsibility to pick up enough tasks". Interestingly, the TA of another group also discussed contribution imbalance with their group, which did not affect the task assignments during meetings in that group.

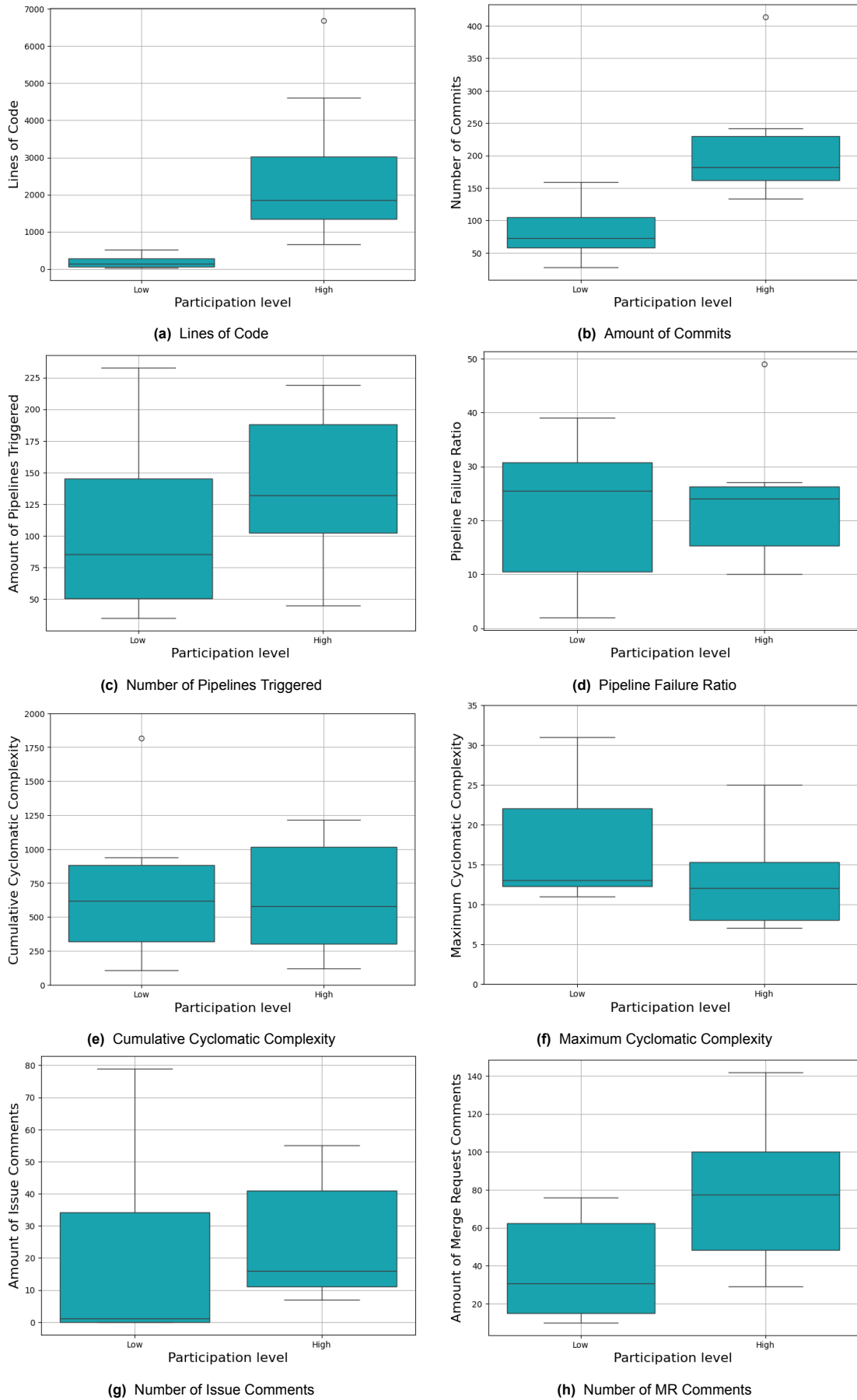


Figure 5.4: The distribution of software metrics in participation level groups.

5.5. Exploratory Findings

In addition to the primary results, the qualitative data revealed several patterns and behaviours that, while not directly measurable in this study, provide valuable insights into the group dynamics. These findings, though exploratory, suggest areas for further investigation and may provide additional context to this research. In this section, these observations are discussed, so they can be used as a basis for recommendations for future work.

5.5.1. Member Familiarity

Although the data is not substantial enough to support this theory, it showed that groups with high member familiarity already have an established dynamic. Therefore, accurately describing the participation levels of these students is more complex. Possible reasons and implications for this are discussed in the next Chapter.

5.5.2. Dominance in Meetings

The loudest students in the meetings are not always classified as high-participation, nor are high-participation students always the most dominant in the meetings. Some actions were identified as behaviour exhibited by the most dominant students in the meetings. These have been listed below.

- Talking really fast and therefore skipping over details, not allowing others to ask questions by covering up with conversation or questions of their own.
- Go into too much detail about every solved issue of the past sprint.
- Getting caught up in details that are not relevant to the success of the project. For example the spelling of a word in the meeting agenda or the interpunction in a merge request.
- Underestimating how much time an issue will take, of both their own and other people's issues. Quotes like 'this will literally take five minutes' are uttered often. Claims to have issues finished by infeasible deadlines are made.
- Claims the team should be better at communicating are made, without going into what is meant by effective communication.
- Use of very strong language such as 'We will just die', and 'It's the end of the world'.
- (Over)use of sarcasm, sometimes used as an excuse to be mean to team members.

When considering this behaviour in the context of the participative stances by Strijbos and de Laat [19], the most dominant student in the meetings was usually either considered an over-rider or, interestingly, a free-rider. Despite this, the remarks are often not related to the discussion or even to the project by any means. If they were, the student might not have been classified as low participation.

Previously, the correlation between gender and dominance in the group was mentioned. The original paper this information was based on, used dominance in the project meetings as a basis [34]. Therefore, another FET has been performed, testing the correlation between dominance in the group meetings, solely based on the observations, and the participation level. As can be seen in the contingency table in Table 5.5, more students have been included in this test. That is because the dominance in group meetings is only based on the observations, so no students have been filtered out due to disagreements with the TA and all students' genders are known. The features are not correlated ($p = 0.628$).

Dominant gender in the group	Dominant in group meetings	
	No	Yes
No	2	4
Yes	8	6

Table 5.5: Contingency Table: The most dominant gender in the group and dominance in the group meetings.

5.5.3. Interpersonal Dynamics

The groups participating in the research were all diverse. Some groups were composed of highly extraverted, high-achieving students with high GPAs and substantial programming experience, often taking on a dominant role in the meetings. Students who may typically fall into a medium participation category were overshadowed and pushed into a low-participation role. This dynamic suggests that group formation affects individual participation levels.

6

Discussion

This chapter discussed the implications of these results and the recommendations stemming from them. It dives into the limitations of the study and provides recommendations for future research.

6.1. Implications and Recommendations

No correlation was found between the programming experience and the participation level in this student software development project. However, the LOC, number of commits, and number of MR comments were found to be significant and with a large effect size. The implications for these findings in practice are mainly related to monitoring, since research on recommendations for group formation is abundant [31, 42, 43].

The course the participants were taking consisted of roughly ninety groups, all monitored by a TA. Most TAs monitor up to four groups, meaning there are at least 23 TAs, all monitoring and judging their groups slightly differently. The consensus among TAs is that LOC is not a valid metric and does not reflect the contributions accurately. However, this study shows that LOC, when measured using GitReporter [9], is meaningful for showing participation. My recommendation is to use the LOC, obtained through GitReporter, combined with the amount of commits to get an estimation of the participation level. However, these metrics should not be used as a stand-alone method, since human behaviour cannot be reduced to two single integers [52]. These two metrics can provide insight into the code contribution and the number of MR comments can provide insight into the involvement in others' code. However, paying close attention to subtle behaviour in the meetings is still the most valuable for determining participation. This advice can be applied to similar situations with student software development groups, and possibly even professional software development groups.

Furthermore, there was no evidence of a correlation between programming experience and participation, meaning that not having (enough) programming experience is not a valid excuse for not participating in the project. If students use this as an excuse, it would be useful to look at any underlying reasons for not contributing. These findings align with the conclusion that programming experience usually equalises after the first introductory computer science course [6, 7]. This also implies that taking programming experience into account for group formation in group projects after these introductory courses would not be meaningful. In general, CS curricula could benefit from a little more focus on teaching collaborative strategies, instead of the focus on teaching hard skills such as programming, especially when students get closer to working in the industry.

6.2. Threats to Validity

This study has some limitations, which will be discussed below. Most of these will form a basis for recommendations for future work.

6.2.1. Subjectivity of Participation Level

The disagreement between the observations and the TA interviews regarding the assignment of participation levels indicates that this measure is subjective. One explanation for this discrepancy is the difference in measuring participation level. The observing researcher was present at meetings, assigned the participation levels, and then looked at the software metrics. The TA had been involved in monitoring the meetings as well as the metrics from the start of the project. Code contributions might have influenced the TA decision, but only discussions about code contributions could have affected the judgement of the researcher.

Furthermore, the interpersonal dynamics in the group might have affected these disagreements. Active participation and confidence in meetings by some students could have dominated the others, misguiding the observer or TA. As discussed in Section 5.5.3, this does not always indicate a high participation level. Seeing dominant behaviour in meetings from over-riders is not surprising. However, free-riders performing some similar actions is interesting. It might mean that these free-riders are masking their lack of contributions by seeming involved in the meetings.

The opposite might also be possible: Medium participation students stepping up to a leadership role in lower participation groups. Quantifying these results is infeasible due to the limited amount of data gathered in this research. To measure and confirm these results, further research using a different approach would be required.

6.2.2. Participant Profile

The project that the participants from this study were recruited from is relatively late in the bachelor's programme, meaning differences in programming experience might have been diminished already. The course this research took place in does not teach programming, so any specific programming skills the students acquired in the first five weeks would have been through their own initiative, possibly inflating the effect of participation level on programming experience.

Furthermore, the experiment started in week 6 out of 10 of the project, meaning students already had had some time to get to know each other and establish the group dynamics. This means we have not been able to see when the dynamics are established and the effect of member familiarity might be smaller.

Moreover, the participants were not very diverse. They are all from the same university and taking the same course simultaneously. This introduced fewer confounding variables, but also ensured the results are not generalisable to students in other study years, courses or universities, let alone software development groups in a professional setting.

The group formation in this project is based on project preference, meaning some groups of students consist of friends while others first chose the project and therefore got assigned to a random group of students. This introduced the confounding variable of member familiarity which only existed in some groups. Groups with high member familiarity might have more implicit and non-verbal communication than other groups and, as a result, may appear less engaged on the surface. In contrast, groups with less familiarity may exhibit more explicit forms of participation, such as verbal contributions.

Another reason groups with high member familiarity might be more difficult to classify is the fact friend groups will have contact outside of project meetings, where the project might be discussed outside of the context of research. Furthermore, the project meetings can also include discussions unrelated to the project, distracting from the task at hand.

6.2.3. Sample Size

The study had 20 participating students, which may not be representative of the population of CS students. Of those 20 students, four students were filtered out due to disagreement and two more were filtered out due to being medium-participation students. This left 15 students to analyse the relationship between metrics and participation level. One of these 15 students had not filled in the questionnaire, meaning there was one data point less to analyse the relation between programming experience and the participation level.

Statistical testing on a small number of samples is less reliable, since the effect of coincidence is larger. Small sample sizes increase the likelihood of encountering Type II errors, or false negatives. This has been tried to mitigate by using the effect size in the analysis, in addition to the p-value of the statistical tests. However, due to the small sample size, it was not possible to perform ANOVA on the software metrics to find correlations between the metrics themselves. Instead, t-tests were performed on each separate metric. This could lead to aggregated errors.

Because of the small sample size, it was infeasible to test for diversity properly. The nationality was now considered as 'Dutch' or 'International', meaning the number of nationalities per group was not taken into account for statistical testing.

6.2.4. Accuracy of Metrics

The results obtained from the LOC metric might not be representative of the quantity of the code written by students. The literature contradicts itself on the meaningfulness of using LOC as a metric for developer productivity [9, 52]. However, the research agrees that it is not detrimental to filter out white lines and lines with only brackets. Nor does it hurt to assign lines changed by small refactors to the original author. However, the literature that mentions these measures also mentions the boilerplate code as something that should be filtered out. GitReporter does not do this. This is mainly due to the fact that it works on multiple programming languages and detecting boilerplate across different languages is quite tricky. Clarity on this topic would strengthen LOC as a metric for measuring participation.

Furthermore, one TA mentioned the merge request comments as a valuable metric for determining participation and contribution to the project, since involved students are aware of the code others wrote. A correlation was found between the amount of MR comments and the participation level. However, this metric only takes into account the number of comments per contributor and not the quality of those comments. From experience, the TA implied that this metric can be cheated by commenting often, but without much useful contribution. Comments like 'LGTM' or 'Looks good to me' or 'Approved' are counted as contributions in this study but do not say anything about the involvement of the commenter.

6.3. Recommendations for Future Work

Based on the exploratory findings and the limitations mentioned before, some recommendations for future work can be made.

6.3.1. Participant profile

As mentioned before, this group project is quite late in the BSc and the study was performed quite late into the project duration. It would be interesting to see whether the effects of programming experience are any different from earlier in the programme. Also, for monitoring groups, it is important to know from which moment the differences in participation level become apparent so they can be identified and mitigated.

The literature showed that member familiarity has a positive effect on group performance in projects [41]. However, it has not been researched whether this effect is possibly enhanced by skewed judgements by TAs, teachers, or other tutors. Friend groups already have an established pecking order and friends choose their friendship over a fair grade. This might mislead teachers and TAs to not intervene when it is necessary.

Furthermore, to obtain a more homogenous participant population, it would be better to perform this research on a project with teacher-assigned groups. This reduces the effect of member familiarity on the results and leads to a more accurate classification by researchers and TAs.

6.3.2. Sample Size

To ensure the feasibility of this thesis, one dimension has been removed from the categorization into group roles by Strijbos and de Laat [19]. Other research could opt to remove the effort dimension and try to look at orientation only. This is usually not the feature that is the most valued in group projects, but it is often in the learning goals of projects at university, since it relates more to collaboration than code contribution.

Since there were not enough samples, the research was limited with regard to statistical testing. This means that we could only identify two categories of participation level. Having more participants opens up possibilities for statistical testing such as testing the data of the medium participation students or identifying the orientation along with effort.

6.3.3. Accuracy of Metrics

Research into the effect of boilerplate code on contributor statistics could strengthen the use of LOC as a metric to measure developer participation. It would have to be investigated if low-participation students commit more boilerplate code to cheat the metrics. This does introduce the difficulty of having projects in multiple programming languages, so it is recommended to conduct this research on a project with in a given programming language.

This research has shown a correlation between the amount of MR comments and participation. However, it would be interesting to see research regarding the correlation between the participation level and the quality of the MR comments. TAs mentioned these comments often included non-useful remarks like 'LGTM'. Due to the time limit in this thesis, it was not feasible to perform content analysis on the merge request comments. Performing content analysis is a time-intensive task, but could be valuable and possibly add an objective measure for determining the participation level.

The TAs used GitLab tools to gather the LOC which work similarly to the GitLab API or use the GitLab API in the background. That number might differ from the number that GitReporter calculates since GitReporter assigns lines affected by small refactors to the original author instead of the refactoring author. Trying to cheat the metrics is a known tactic for social loafers to make it seem their contributions are more substantial than they are. Cheating the LOC metric would increase the number of LOC assigned to the social loafer when calculated using the GitLab API, but not when calculated with GitReporter. It would be interesting to research whether cheating the metrics can be determined this way.

6.4. The Use of LLMs in this Thesis

Problematic use of Large Language Models (LLMs) such as ChatGPT in research is becoming more frequent¹. In this thesis, ChatGPT has been used as a writing coach. This means that it was not asked to write sections of this thesis, but rather provide feedback and synonyms and occasionally search queries as well. This is similar to how other students might use tools like Grammarly². A list of prompts asked to ChatGPT is included below.

¹<https://pubpeer.com/publications/CAABBF887348FB2D1C0329E0A27BE6>

²<https://app.grammarly.com/>

Prompt	Answer
The final grade is of 10% presentation, 40% product and 50% process. What fits in the blank?	Composed
Give me an academic synonym for solo	Individual or single
Give me a less wordy way to say "make more difficult"	Complicate
Should it be "to create" or "in creating" in this sentence?	Reply explaining the difference in nuance between the two.
Please recommend some tooling for creating transcripts	Recommendations of various tools.
Help me brainstorm titles for my thesis, these are the aspects I want to incorporate. Give me ten options.	Ten options for titles.
Can you give me some alternatives for the word 'However'?	Nevertheless, nonetheless, yet, despite that, even so.
Please provide some synonyms for "better". I need to use them in my thesis	Superior, enhanced, improved, advanced, and optimal.
How do I translate "In tegendeel tot" to English? I can't find the word that captures the context accurately.	Contrary to, in contrast to, on the contrary to, opposite to, unlike.
I want to search for papers about group work in computer science, specifically on the effect of students who do all the work, not letting the other learn. Give me some search queries.	Some search queries.
Can you convert this RIS citation to Bibtex for me?	The reference in BibTeX format.
Please convert this table from Excel to LaTeX	The formatted table.
What word is a bit less strong than 'contradicting'?	Disagreeing or contrasting or deviating.
Please add a column to this LaTeX table	The table with an added column.
Are the commas and grammar in this sentence correct?	The corrected grammar.

7

Conclusion

This study identified the participation level of students working on a group project in the second year of their Computer Science bachelor's degree. This was done by observing the groups during multiple sessions, administering a questionnaire, and conducting an interview with the teaching assistants who monitored the groups. It tried to analyze the association of programming experience, which was determined based on the questionnaire, with this participation level but found no correlation, meaning that not having programming experience is not an excuse to not participate in group projects.

Furthermore, the correlation between this participation level and the software metrics of the code repository of the group project was determined. It was found that the number of lines of code students wrote, the number of commits students made, and the number of comments that students posted on merge requests are all correlated with the participation level. In the end, some recommendations for monitoring groups during similar group projects were made.

This study gives an insight into the causes and effects of participation in group projects. It raises some important questions for further research in his field. For example, investigating if the participation level can be determined from the quality of the comments on merge requests. Especially if this quality could be determined automatically, this could be a valuable addition for monitoring groups working on projects that is not directly related to the code contribution per student.

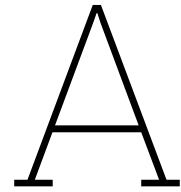
References

- [1] Tyson Henry. “The Changing Role of Computing Education: Fostering Collaboration”. In: *Issues In Information Systems 7.1* (2006). DOI: 10.48009/1_iis_2006_67-71.
- [2] Carol L. Colbeck, Susan E. Campbell, and Stefani A. Bjorklund. “Grouping in the Dark: What College Students Learn from Group Projects”. In: *The Journal of Higher Education 71.1* (2000), pp. 60–83. ISSN: 00221546, 15384640. URL: <http://www.jstor.org/stable/2649282>.
- [3] Anne-Maarit Majanoja. and Timo Vasankari. “Reflections on Teaching Software Engineering Capstone Course”. In: *Proceedings of the 10th International Conference on Computer Supported Education - Volume 1: CSEDU*. INSTICC. SciTePress, 2018, pp. 68–77. ISBN: 978-989-758-291-2. DOI: 10.5220/0006665600680077.
- [4] James D. Lang et al. “Industry Expectations of New Engineers: A Survey to Assist Curriculum Designers”. In: *Journal of Engineering Education 88.1* (1999), pp. 43–51. DOI: <https://doi.org/10.1002/j.2168-9830.1999.tb00410.x>.
- [5] Christopher Scaffidi. “Employers’ Needs for Computer Science, Information Technology and Software Engineering Skills Among New Graduates”. In: *International Journal of Computer Science, Engineering and Information Technology 8* (Feb. 2018), pp. 01–12. DOI: 10.5121/ijcseit.2018.8101.
- [6] Edward Holden and Elissa Weeden. “The experience factor in early programming education”. In: *Proceedings of the 5th Conference on Information Technology Education*. CITC5 ’04. Salt Lake City, UT, USA: Association for Computing Machinery, 2004, pp. 211–218. ISBN: 1581139365. DOI: 10.1145/1029533.1029585.
- [7] Chris Wilcox and Albert Lionelle. “Quantifying the benefits of prior programming experience in an introductory computer science course”. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. Vol. 2018-January. SIGCSE ’18. Baltimore, Maryland, USA: Association for Computing Machinery, Inc, Feb. 2018, pp. 80–85. ISBN: 9781450351034. DOI: 10.1145/3159450.3159480.
- [8] Diba Mirza et al. “Undergraduate Teaching Assistants in Computer Science: A Systematic Literature Review”. In: *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ICER ’19. Toronto ON, Canada: Association for Computing Machinery, 2019, pp. 31–40. ISBN: 9781450361859. DOI: 10.1145/3291279.3339422.
- [9] Michael Guttmann, Aleksandar Karaka, and Denis Helic. “Attribution of Work in Programming Teams with Git Reporter”. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education*. Vol. 1. SIGCSE 2024. Portland, OR, USA: Association for Computing Machinery, Mar. 2024, pp. 436–442. DOI: 10.1145/3626252.3630785.
- [10] Reza M. Parizi, Paola Spoletini, and Amritraj Singh. “Measuring Team Members’ Contributions in Software Engineering Projects using Git-driven Technology”. In: *Proceedings of the 2018 IEEE Frontiers in Education Conference (FIE)*. San Jose, CA, USA: IEEE Press, 2018, pp. 1–5. DOI: 10.1109/FIE.2018.8658983.
- [11] Niki Gitinabard et al. “Student Teamwork on Programming Projects: What can GitHub logs show us?” In: *Proceedings of the 13th International Conference on Educational Data Mining (EDM)*. 2020, pp. 409–416. ISBN: 978-1-7336736-1-7.
- [12] Jan Jaap Sandee and Efthimia Aivaloglou. “GitCanary: A Tool for Analyzing Student Contributions in Group Programming Assignments”. In: *Proceedings of the 20th Koli Calling International Conference on Computing Education Research*. Koli Calling ’20. Koli, Finland: Association for Computing Machinery, 2020. ISBN: 9781450389211. DOI: 10.1145/3428029.3428563.

- [13] Roberta Evans Sabin and Edward P. Sabin. "Collaborative learning in an introductory computer science course". In: *Proceedings of the Twenty-Fifth SIGCSE Symposium on Computer Science Education*. SIGCSE '94. Phoenix, Arizona, USA: Association for Computing Machinery, 1994, pp. 304–308. ISBN: 0897916468. DOI: 10.1145/191029.191156.
- [14] Sherlock A. Licorish et al. "Understanding students' software development projects: Effort, performance, satisfaction, skills and their relation to the adequacy of outcomes developed". In: *Journal of Systems and Software* 186 (2022), p. 111156. ISSN: 0164-1212. DOI: <https://doi.org/10.1016/j.jss.2021.111156>.
- [15] Elise Deitrick, Michelle Hoda Wilkerson, and Eric Simoneau. "Understanding Student Collaboration in Interdisciplinary Computing Activities". In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER '17. Tacoma, Washington, USA: Association for Computing Machinery, 2017, pp. 118–126. ISBN: 9781450349680. DOI: 10.1145/3105726.3106193.
- [16] Noreen M. Webb. "Peer interaction and learning with computers in small groups". In: *Computers in Human Behavior* 3.3 (1987), pp. 193–209. ISSN: 0747-5632. DOI: [https://doi.org/10.1016/0747-5632\(87\)90023-9](https://doi.org/10.1016/0747-5632(87)90023-9).
- [17] Matthew Berland, Don Davis, and Carmen Petrick Smith. "AMOEBAs: Designing for collaboration in computer science classrooms through live learning analytics". In: *International Journal of Computer-Supported Collaborative Learning* 10.4 (2015), pp. 425–447. ISSN: 1556-1615. DOI: 10.1007/s11412-015-9217-z.
- [18] Bram de Wever and Jan-Willem Strijbos. "Roles for structuring groups for collaboration". In: *International Handbook of Computer-Supported Collaborative Learning*. Ed. by Ulrike Cress et al. Computer-Supported Collaborative Learning Series. Springer Nature, 2021, pp. 315–331. ISBN: 978-3-030-65290-6. DOI: 10.1007/978-3-030-65291-3_17.
- [19] Jan Willem Strijbos and Maarten F. De Laat. "Developing the role concept for computer-supported collaborative learning: An explorative synthesis". In: *Computers in Human Behavior* 26.4 (July 2010), pp. 495–505. ISSN: 07475632. DOI: 10.1016/j.chb.2009.08.014.
- [20] Robert Mcquade et al. "Students' strategies for managing social loafers in PBL: Interactional means of dealing with unequal participation in group work". In: *Interactional Research into Problem-Based Learning*. Purdue University Press, Aug. 2020, pp. 275–297. ISBN: 9781557538048. DOI: 10.2307/j.ctvs1g9g4.14.
- [21] David Hall and Simone Buzwell. "The problem of free-riding in group projects: Looking beyond social loafing as reason for non-contribution". In: *Active Learning in Higher Education* 14.1 (Mar. 2013), pp. 37–49. ISSN: 14697874. DOI: 10.1177/1469787412467123.
- [22] C. Kevin Synnott. "Guides to Reducing Social Loafing in Group Projects: Faculty Development". In: *Journal of Higher Education Management* 31.1 (2016), pp. 211–221. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2890604.
- [23] Praveen Aggarwal and Connie L. O'Brien. "Social loafing on group projects: Structural antecedents and effect on student satisfaction". In: *Journal of Marketing Education* 30 (3 May 2008), pp. 255–264. ISSN: 02734753. DOI: 10.1177/0273475308322283.
- [24] Burhanuddin Yasin et al. "'It's Unfair' The Effect of Free Riding And Social Loafing of Group Discussion In Cooperative Learning". In: (July 2022), pp. 222–228.
- [25] Charles M. Brooks and Janice L. Ammons. "Free Riding in Group Projects and the Effects of Timing, Frequency, and Specificity of Criteria in Peer Assessments". In: *Journal of Education for Business* 78.5 (2003), pp. 268–272. DOI: 10.1080/08832320309598613.
- [26] Luis Miguel Serrano-Cámara et al. "An evaluation of students' motivation in computer-supported collaborative learning of programming concepts". In: *Comput. Hum. Behav.* 31.C (Feb. 2014), pp. 499–508. ISSN: 0747-5632. DOI: 10.1016/j.chb.2013.04.030.
- [27] Ilenia Fronza and Xiaofeng Wang. "Towards an Approach to Prevent Social Loafing in Software Development Teams". In: *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2017, pp. 241–246. DOI: 10.1109/ESEM.2017.37.

- [28] Joanna Wolfe and Elizabeth Powell. "Strategies for dealing with slacker and underperforming teammates in class projects". In: *Proceedings of the IEEE International Professional Communication Conference*. Institute of Electrical and Electronics Engineers Inc., Jan. 2015. ISBN: 9781479937493. DOI: 10.1109/IPCC.2014.7020346.
- [29] Kai Presler-Marshall, Sarah Heckman, and Kathryn T. Stolee. "Identifying Struggling Teams in Software Engineering Courses Through Weekly Surveys". In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*. Vol. 1. SIGCSE 2022. Association for Computing Machinery, Inc, Feb. 2022, pp. 126–132. ISBN: 9781450390705. DOI: 10.1145/3478431.3499367.
- [30] Jay Jr, Bruce Reinig, and Robert Briggs. "Principles for Effective Virtual Teamwork". In: *Commun. ACM* 52 (Apr. 2009), pp. 113–117. DOI: 10.1145/1498765.1498797.
- [31] Sherry Piezon and Robin Donaldson. "Online Groups and Social Loafing: Understanding Student-Group Interactions". In: *Online Journal of Distance Learning Administration* 8 (Jan. 2005).
- [32] Kipling D. Williams and Steven J. Karau. "Social loafing and social compensation: The effects of expectations of co-worker performance." In: *Journal of Personality and Social Psychology* 61.4 (Jan. 1991), pp. 570–581. DOI: 10.1037/0022-3514.61.4.570.
- [33] Andy McKinlay, Rob Procter, and Anne Dunnett. "An investigation of social loafing and social compensation in computer-supported cooperative work". In: *Proceedings of the 1999 ACM International Conference on Supporting Group Work*. GROUP '99. Phoenix, Arizona, USA: Association for Computing Machinery, 1999, pp. 249–257. ISBN: 1581130651. DOI: 10.1145/320297.320327.
- [34] Rosalie J. Ocker. "A Balancing Act: The Interplay of Status Effects on Dominance in Virtual Teams". In: *IEEE Transactions on Professional Communication* 50.3 (2007), pp. 204–218. DOI: 10.1109/TPC.2007.902656.
- [35] R.J. Ocker. "Influences on creativity in asynchronous virtual teams: a qualitative analysis of experimental teams". In: *IEEE Transactions on Professional Communication* 48.1 (2005), pp. 22–39. DOI: 10.1109/TPC.2004.843294.
- [36] Tripp Driskell et al. "Team Roles: A Review and Integration". In: *Small Group Research* 48.4 (June 2017), pp. 482–511. DOI: 10.1177/1046496417711529.
- [37] James E Driskell et al. "What Makes a Good Team Player? Personality and Team Effectiveness". In: *Group Dynamics: Theory, Research, and Practice* 10.4 (2006), pp. 249–271. DOI: 10.1037/1089-2699.10.4.249.
- [38] Ed Lester, Damian Schofield, and Peter Chapman. "Self and Peer Assessment and Dominance During Group Work Using Online Visual Tools". In: *Seminar.net* 6 (1 Nov. 2010). ISSN: 1504-4831. DOI: 10.7577/SEMINAR.2460.
- [39] Chen Chen et al. "'Cowboy' and 'Cowgirl' Programming: The Effects of Precollege Programming Experiences on Success in College Computer Science". In: *International Journal of Computer Science Education in Schools* 2.4 (Jan. 2019), pp. 22–40. DOI: 10.21585/ijcses.v2i4.34.
- [40] Anne Pieterse, Daan Knippenberg, and Dirk van Dierendonck. "Cultural Diversity and Team Performance: The Role of Team Member Goal Orientation". In: *Academy of Management Journal* 56 (July 2012), pp. 782–804. DOI: 10.5465/amj.2010.0992.
- [41] Jeroen Janssen et al. "Influence of group member familiarity on online collaborative learning". In: *Computers in Human Behavior* 25.1 (2009), pp. 161–170. ISSN: 0747-5632. DOI: <https://doi.org/10.1016/j.chb.2008.08.010>.
- [42] Michal Kompan and Maria Bielikova. "Enhancing existing e-learning systems by single and group recommendations". In: *International Journal of Continuing Engineering Education and Life Long Learning* 26.4 (2016), pp. 386–404. DOI: 10.1504/IJCEELL.2016.080980.
- [43] Naseebah Maqtary, Abdulqader Mohsen, and Kamal Bechkoum. "Group Formation Techniques in Computer-Supported Collaborative Learning: A Systematic Literature Review". In: *Technology, Knowledge and Learning* 24.2 (2019), pp. 169–190. ISSN: 2211-1670. DOI: 10.1007/s10758-017-9332-1.

- [44] Lihui Sun, Linlin Hu, and Danhua Zhou. "Programming attitudes predict computational thinking: Analysis of differences in gender and programming experience". In: *Computers & Education* 181 (May 2022), p. 104457. ISSN: 0360-1315. DOI: 10.1016/J.COMPEDU.2022.104457.
- [45] Laura Heels and Marie Devlin. "Investigating the Role Choice of Female Students in a Software Engineering Team Project". In: *Proceedings of the 3rd Conference on Computing Education Practice*. CEP '19 2. Durham, United Kingdom: Association for Computing Machinery, 2019. ISBN: 9781450366311. DOI: 10.1145/3294016.3294028.
- [46] Bjørn Hjorth Westh et al. "Gender Differences in the Group Dynamics of Smaller CS1 Project Groups". In: *IEEE ASEE Frontiers in Education Conference 2023*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2023, pp. 1–9. DOI: 10.1109/FIE58773.2023.10343369.
- [47] Leonardo Silva, António José Mendes, and Anabela Gomes. "Computer-supported Collaborative Learning in Programming Education: A Systematic Literature Review". In: *2020 IEEE Global Engineering Education Conference (EDUCON)*. 2020, pp. 1086–1095. DOI: 10.1109/EDUCON45650.2020.9125237.
- [48] Eirini Kalliamvakou et al. "Measuring Developer Contribution From Software Repository Data." In: *Proceedings - International Conference on Software Engineering*. Jan. 2009, p. 55. DOI: 10.1145/1370750.1370781.
- [49] D. Campbell. *Assessing the Impact of Planned Social Change*. Jan. 1976.
- [50] Keir Mierle et al. "Mining student CVS repositories for performance indicators". In: *SIGSOFT Softw. Eng. Notes* 30.4 (May 2005), pp. 1–5. ISSN: 0163-5948. DOI: 10.1145/1082983.1083150.
- [51] Kaushal Bhatt, Vinit Tarey, and Pushpraj Patel. "Analysis Of Source Lines Of Code (SLOC) Metric". In: *IJETAE* 2 (Apr. 2012).
- [52] J. Rosenberg. "Some misconceptions about lines of code". In: *Proceedings Fourth International Software Metrics Symposium*. 1997, pp. 137–142. DOI: 10.1109/METRIC.1997.637174.
- [53] Thomas McCabe. "A Complexity Measure". In: *IEEE Transactions on Software Engineering* 2.4 (1976), p. 308.
- [54] Arthur H. Watson and Thomas J. McCabe. *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*. NIST, 1996.
- [55] Chris Mair. *Groovy Code Metrics: Cyclomatic Complexity*. 2012. URL: <https://tenpercentnotcrap.wordpress.com/2012/07/08/groovy-code-metrics-cyclomatic-complexity/>.
- [56] Israel Herraiz and Ahmed E. Hassan. "Beyond Lines of Code: Do We Need More Complexity Metrics?" In: *Making Software: What Really Works, and Why We Believe It*. Ed. by Andy Oram and Greg Wilson. O'Reilly, 2011. Chap. 8, pp. 125–144.
- [57] Jalerson Lima et al. "Assessing developer contribution with repository mining-based metrics". In: *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2015, pp. 536–540. DOI: 10.1109/ICSM.2015.7332509.
- [58] Efthimia Aivaloglou and Anna van der Meulen. "An Empirical Study of Students' Perceptions on the Setup and Grading of Group Programming Assignments". In: *ACM Transactions on Computing Education* 21.3 (Mar. 2021), pp. 1–22. DOI: 10.1145/3440994.
- [59] Janet Siegmund et al. "Measuring Programming Experience". In: *Proceedings of the IEEE International Conference on Program Comprehension*. June 2012. DOI: 10.1109/ICPC.2012.6240511.
- [60] Xiangyu Ying et al. "Group laziness: The effect of social loafing on group performance". In: *Social Behavior and Personality* 42.3 (Mar. 2014), pp. 465–472. ISSN: 1179-6391. DOI: 10.2224/SBP.2014.42.3.465.
- [61] Jacob Cohen. *Statistical Power Analysis for the Behavioural Sciences*. Lawrence Erlbaum Associates, 1988. ISBN: 0-8058-0283-5.



Questionnaire

Research towards Group Work in the Software Project

Consent You are invited to participate in a research study titled 'The Participative Stances towards Collaborative Work in Computing Education'. Merel Steenbergen, an MSc student from TU Delft, is doing this study. This research study aims to gain insight into group composition within CS project groups. It will take you approximately 10 minutes to complete. The data will be used for analysis. We will be asking you to answer the questions of this survey.

For our research, it is necessary to collect some personal data. To use this data during our research we need your consent. Personal data that is collected is:

- Email → As an identifier during the research and in case we need to contact you
- Age, gender, nationality, previous experience with the course → To establish a basic background of the participants.
- Your programming experience → To establish a more detailed background of the participants
- Your answers to some statements about group work → To establish the context of the participants.

Your responses to the questionnaire will be kept completely confidential, **the course staff will not see your answers and your answers have no impact on your grade**. Any identifying information will be anonymised before analysing the data. If identifying information will be used for further research you will receive a notification of this, with the possibility to withdraw your consent. Participation in this study is entirely voluntary and you can withdraw anytime until the research has been published. If you change your mind or if you have any questions, you can send an email to [email address]. If you want to withdraw, your data will be permanently deleted from the collected data. Any other information that can be traced back to you will also be permanently deleted. By ticking the box below, you acknowledge that you have read the above and understand your rights.

- I have read the statement above, I understand my rights and I consent to participation in the study as written above.

Q1 What is your gender?

Q2 How old are you?

Q3 Which country/countries are you from?

Q4 What is your GPA?

- < 6.0 (1)
 - 6.1 - 7.0 (2)
 - 7.1 - 8.0 (3)
 - 8.1 - 9.0 (4)
 - > 9.0 (5)
 - I don't know/don't want to answer this question (6)
-

Q5 Is this the first time you are taking this course?

- Yes (1)
- No (2)
-

Q6 Did you know your team members before the project?

- Yes, all of them (1)
- Yes, most of them (2)
- Yes, some of them (3)
- No (4)

Q7 On a scale from 1 to 10, how do you estimate your programming experience?









Very inexperienced Very experienced

1 2 3 4 5 6 7 8 9 10

Where 1 is very inexperienced and 10 is
very experienced







Q8 Answer the following statements on a scale of 1 to 5 where 1 is very inexperienced and 5 is very experienced.

	Very inexperienced				Very experienced
	1	2	3	4	5
How do you estimate your programming experience compared to experts with 20 years of practical experience?					
How do you estimate your programming experience compared to your class mates?					
How experienced are you with Java?					
How experienced are you with C?					
How experienced are you with C++?					
How experienced are you with Python?					
How experienced are you with Javascript?					
How experienced are you with Scala?					

Q9 Which languages, apart from those mentioned above, do you know (medium experience or better)?

Q10 Answer the following statements about programming paradigms

	Very inexperienced			Very experienced	
	1	2	3	4	5
How experienced are you with functional programming?					
How experienced are you with imperative programming?					
How experienced are you with logical programming?					
How experienced are you with object-oriented programming?					









Q11 For how many years have you been programming?

Q12 For how many years have you been programming for larger software projects, e.g. in a company?

Q13 How large were the codebases of the professional/company projects you contributed to typically?

- Not applicable (1)
- <900 lines of code (2)
- 900 - 40000 lines of code (3)
- > 40000 lines of code (4)

Q14 Please indicate how much the following statements apply to you.

	Strongly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Strongly agree
	1	2	3	4	5
In a team, I am indispensable					
In a team, I will try as hard as I can					
In a team, I contribute less than I should					
In a team, I will actively participate in the discussion and contribute ideas					
In a team, it is okay even if I do not do my share					
In a team, it does not matter whether I try my best or not					
In a team, given my abilities, I will do the best I can					
I prefer working alone over working in a group					

End of Survey Thank you for filling in the survey! Please be assured again that your responses will not be shared with anyone except the responsible researchers. The course staff will not have access to individual responses; only anonymized and aggregated data will be included in the final thesis. Your input is invaluable in contributing to our research, and we thank you for your time and effort. If you have any question or want to revoke your answers, you can reach me at [email address].

Please fill in your email address below. This will be the only thing we can identify you by. If you do not want to fill in your email address, please ask the researcher to give you an identifier.

B

Teaching Assistant Interview Protocol

Introduction

5 minutes

- Let the participant sign the consent form.
- Start recording.
- Introduction of research.
- Ask if they have any questions before we start.

Contextual Questions

15 minutes

- Can you tell me something about your experience as a TA, and more specifically as a TA of project courses?
- Did you identify any problems in group X? Specifically, any problems having to do with programming experience or communication.
- How did you identify this problem? And how do you identify problems in general?
- Did you look at the software metrics at all when monitoring this group? If yes, which metrics did you look at and which tools did you use?
- What feedback have you given the group?
- How and how often do you communicate with the group?

Identifying the Roles and closing

10 minutes

- Explain each role, show Table 4.1 and discuss the behaviour.
- Let them explain what roles they think the students have in the group and why.
- Ask if they have anything to add.

Potential Questions

Metrics

- When did you use the metrics during guidance?
- How did you implement the metrics during guidance?
- Did you show the metrics to the students, and if yes: How did students respond to the metrics?

Guidance

- How did you deal with under-/over-performing individuals?
- How did you approach guiding those projects?
- How did you do individual guidance in this group?