Master's thesis

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

Mathea Godø Hildre

Challenges and Possibilities When Process Mining GitLab for a Software Engineering Course

Master's thesis in Computer Science
Supervisor: Torgeir Dingsøyr
July 2022

NTNU
Norwegian University of
Science and Technology

Mathea Godø Hildre

Challenges and Possibilities When Process Mining GitLab for a Software Engineering Course

Master's thesis in Computer Science
Supervisor: Torgeir Dingsøyr
July 2022

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

NTNU
Norwegian University of
Science and Technology

Abstract

It can be difficult and time-consuming to give elaborate feedback in large project-based software engineering courses. Detailed and regular feedback is a necessity to facilitate effective collaborative learning.

This master's thesis presents an exploratory case study that examines how software repository mining can be implemented in TDT4140 Software Engineering, a project-based software engineering course with 500 attending students from different programs. The focus has been on discovering challenges when implementing software repository mining in this setting, as well as uncovering whether it would be helpful to students and supervisors.

To determine this, quantitative and qualitative analysis of a questionnaire answered by students, interviews with teaching assistants, and data gathered from the students' GitLab repositories have been carried out to try to identify six different dysfunctions in the teams: Unfair or unevenly distributed workload, too little time is spent working on the project, the participants are not committed to the project, lack of a plan or strategy for the project, poor leadership, and too specialized tasks.

The results show that even without designing the course around software repository mining, some metrics derived from the students' GitLab data have a statistically significant correlation to some of the six dysfunctions. This indicates that analysis of the students' software repositories can be used to aid in identifying group dysfunctions. Some challenges were also revealed in the study. Pair programming and incorrect assignment of GitLab issues make it difficult to determine work distribution within a group. Other work in the course is not represented in GitLab and would lead to a false picture of reality, especially in groups where some members focus on programming and others more on the other required work. The students have a difference in experience with programming and with the use of Git, which can lead to user error and weaknesses in the analyses.

The thesis discusses these findings and concludes with advice to course coordinators who wish to use software repository mining in similar courses.

Sammendrag

Det kan være vanskelig og tidkrevende å gi grundige tilbakemeldinger i store prosjektbaserte emner i programvareutvikling. Detaljerte og regelmessige tilbakemeldinger er nødvendig for å tilrettelegge for effektiv samarbeidslæring.

I dette masterprosjektet er det utført en utforskende casestudie for å undersøke hvordan software repository mining kan implementeres i TDT4140 Programvareutvikling, et prosjektbasert programmeringsemne med 500 studenter fra forskjellige studieretninger. Målet med studien har vært å oppdage utfordringer ved implementering av software repository mining i denne sammenhengen, samt å avdekke om det ville være nyttig for studenter og veiledere.

For å fastslå dette ble det utført kvantitativ og kvalitativ analyse av et spørreskjema besvart av studenter, intervjuer med lærerassistenter og data samlet inn fra studentenes GitLab-repositories for å forsøke å identifisere seks forskjellige dysfunksjoner i teamene: Urettferdig eller ujevnt fordelte arbeidsoppgaver, det blir prioritert for lite tid til å gjennomføre prosjektet, deltagerne er ikke forpliktet til prosjektet, mangel på plan og strategi, dårlig lederskap og for spesialiserte arbeidsoppgaver.

Resultatene viser at selv uten å designe kurset rundt software repository mining, kan man finne en statistisk signifikant korrelasjon mellom metrikker utledet fra GitLab-data og noen av de seks dysfunksjonene. Dette indikerer at analysen av studentenes software repositories kan brukes som hjelp til å identifisere gruppedysfunksjoner. Noen utfordringer ble også avdekket i studien. Parprogrammering og feil i tilordning av GitLab issues gjør det vanskelig å si noe om arbeidsfordeling i en gruppe. Annet arbeid i emnet er ikke representert i GitLab og vil føre til et uriktig bilde av virkeligheten, spesielt i grupper der noen medlemmer fokuserer på programmering og andre mer på annet nødvendig arbeid. Studentene har forskjellig erfaring med programmering og med bruk av Git, noe som kan føre til brukerfeil og svakheter i analysene.

Oppgaven diskuterer disse funnene og avsluttes med råd til emnekoordinatorer som ønsker å bruke software repository mining i lignende emner.

Preface

This thesis has been written for my masters degree at the Norwegian University of Science and Technology. The focus of the thesis is on software repository mining for education.

I would like to thank my supervisor, Torgeir Dingsøyr, for his support and guidance through this process. I am grateful for the time and effort he has invested in me. The weekly meetings we have had, where I and others could ask questions and get advice have been invaluable. And for taking the time to help me with data collection and anonymization I am especially thankful.

I would also like to extend a thanks to my supervisor on my specialization project, Hallvard Trætteberg, who helped me decide what I wanted to focus my research on.

To all the teaching assistants I got to interview, and all the students who responded to my questionnaire, thank you!

To my dad, for motivation, support, and for all your time, I could not have done this without you.

Mathea Godø Hildre

Trondheim, 11th July 2022

Contents

List of Figures

List of Tables

1. Introduction

This chapter starts by stating the motivation for the choice of research topic. It then goes on to set the goal of the research, as well as two research questions. Lastly an overview of the structure of the rest of the thesis is shown.

1.1. Motivation

According to Oakley et al [5], learning together in teams has been a proven way to get a deeper level of learning. It also creates a setting for learning teamwork skills, such as communication, organization, and leadership. These are skills that are invaluable when students enter their industries as professionals. Other soft skills, that are promoted by using project-based learning, including being able to share knowledge and listen to others [6]. Modern software systems are complex, and working in teams is essential [7].

Teamwork does not come without challenges, and in a dysfunctional team, we can expect negative feelings, which lead to poor performance [8]. A variety of different dysfunctions can be an additional challenge to a student software engineering team. A balance of contributions is amongst the factors that correlate to team performance and satisfaction [9]. Other problems that may occur are problems with planning and organization, a lack of leadership, or problems with commitment to a project.

Research has shown that dysfunctional teams are unlikely to resolve their problems without intervention [10]. It is therefore important to follow up with the student teams and give them helpful feedback. Students prefer elaborate feedback, but it can be time-consuming and demanding to expect an instructor to provide that, especially in

large courses with as much variety in submissions as tend to be the case for project-based courses [11]. With the software engineering courses relying on distributed version control systems for development, this opens the possibility of software repository mining to generate knowledge about the students' work processes, and help alleviate some of the work done by supervisors while still providing students with elaborate feedback.

Following the course TDT4140 Software Engineering Spring 2022, this study aims to explore how software repository mining can be used to give students better feedback and support in a project-based learning course.

1.2. Goals and Research Questions

Goal *The main objective is to explore the challenges and possibilities of using analyses of software repository data to give helpful information to supervisors for providing elaborate feedback in a project-based software engineering course.*

To meet the objective of the study, this project aims to identify possible challenges with process mining software repositories for software engineering courses, as well as look into the benefits of using software repository mining.

Research question 1 *How can metrics derived from data from software repositories be analyzed and presented in a way that gives value to students and supervisors?*

By exploring correlations between software repository metrics and dysfunctions within student teams, the aim is to see what possibilities exist for early identification of problems within student teams. Since this study focuses on supervision and feedback rather than assessment, it is also interesting to see how supervisors would like the data presented for it to be helpful to them.

Research question 2 *What do students and supervisors think about the idea of using metrics from software repositories to identify problems within the group?*

This question aims to identify both technical challenges and preferences amongst students and supervisors when it comes to gathering and analyzing their data for feedback purposes.

1.3. Thesis Structure

Table 1.1 shows the structure of this thesis.

Table 1.1.: Structure of the thesis.

	Section	Description
1	Introduction	This section presents the motivation for the project. Next, it presents the goal of the project as well as the research questions the thesis aims to answer. Lastly, the structure of the thesis is described.
2	Theory	This section of the thesis gives the necessary introduction to the field to be able to follow the research done in the project. It presents information about project-based learning, version control systems, process mining, and metrics that can be derived by mining data from software repositories.
3	Methodology	This section presents the methodologies that are used throughout the study, and why they were chosen.
4	Results	This section presents the result of the research that is done in this study.
5	Discussion	In this section the results of the study are discussed and compared to previous knowledge in the field in order to answer the posed research questions.
6	Conclusion and Future Work	This section concludes the thesis. It presents the contribution of the study and suggests themes for further research.

2. Theory

This chapter presents relevant background information. It aims to give the reader the necessary introduction to theory to understand the choices made in the study. The chapter conveys literature on collaborative learning, version control systems with a particular focus on Git and GitLab, process mining for software engineering courses, and teamwork dynamics.

2.1. Collaborative Learning

Collaborative learning is when a group attempts to learn something together. Collaboration is more than cooperation. Participants that cooperate can divide and conquer, each member being responsible for a part of a bigger project. A group that collaborates puts effort towards creating a joint problem space [12]. With a common understanding and while working on a problem together, the members can share knowledge, and gain the many benefits collaborative learning can give to the learners.

Collaborative group work is often considered an activity that accommodates deep learning [13] [14], as well as engaging and effective learning [15] [16]. Team members will help each other learn, and thereby ensure learning by all [16]. The shared responsibility for the tasks can also help reduce anxiety and build self-esteem [16] [14]. Teamwork skills, communication, and project management are also developed in collaborative learning [17] [14]. The group members can also retain more individual knowledge when working collaboratively [13].

Problems also occur when people work together in teams. Some common problems are

the "free-rider" problem and the "sucker effect" problem [14]. A free-rider is someone who does not do their part in a project. And the sucker effect happens when a student who would normally do their part suspects their team members to be free-riders, and to avoid being the sucker who does all the work, they also become a free-rider. More problems in group work will be discussed in Section 2.4

2.1.1. Project-based Learning

Project-Based Learning (PBL) is described by Fincher and Knox as: "a teaching approach that engages students in sustained, collaborative focus on a specific project, often organized around a driving question chosen by students themselves or provided by an outside organization or company" [18]. Even though PBL can be organized as individual assignments, they are more often structured to be carried out in teams [19]. Common concepts in PBL are collaboration, community, design, and technology [19]. Soft skills are defined by Matsouka and Mihail [20] as "personal attributes that enhance an individual's interactions, job performance and career prospects." Such skills have become increasingly important qualities for new industry hires. PBL is a fitting educational approach to provide students with experience in these skills while also teaching a technical curriculum.

2.1.2. Project-based Learning in Software Engineering Education

Software Engineering is a field where PBL can quite easily be fitted into a course. Students have access to standard industry tools, and through the inclusion of a capstone project, students can get realistic experience in most aspects of software development [21]. A project-based approach can, as stated in Section 2.1.1, strengthen the learning of soft skills. Amongst those, teamwork is a particularly important skill that students in software engineering should master before starting their professional careers [22], and it is a skill that lecture-based learning does not facilitate. When introducing PBL in software engineering education, it opens the possibility of cooperating with real industrial clients on actual problems. Real industry experience is considered a great benefit in preparing students for entering the software engineering industry [23] [24].

2.1.3. Supervision in Project-based Learning

According to Hinsz, Tindale, and Vollrath [25] it is likely that groups can process feedback better than singular members. With all the benefits that students can gain from PBL, it is important to facilitate the best possible implementation of the method. How to give feedback and what to focus on is, therefore, something of importance when designing a PBL course.

In student teams that turn dysfunctional, early intervention is important to help the teams improve their team dynamics. Dysfunctional teams that are left to resolve their own issues tend to only get worse with time [10] [26]. Having multiple meeting points with supervisors throughout the semester or having other ways of getting regular feedback is also important [27] [23]. Regular feedback throughout the semester ensures that the students get formative assessment, and not just summative assessment.

Another important factor is that students can be critical of feedback that is considered vague or ambiguous [28]. More feedback also seems to correspond to better learning. Elaborate feedback should be provided to the students, that can correct misconceptions and fill the students' gaps of knowledge [29]. In addition to all of this, it is important to create an environment where students can go to their supervisors or instructors with their questions without worrying that it might impact their grades [26].

Peer feedback is something that is often considered in project-based courses, and including a peer assessment as part of the evaluation in courses that include teamwork has been previously advised [27]. However, in courses that use it in their evaluation, there tends to be a portion of the students who disagree with the peer assessment [27] [26].

Project-based courses in universities can have a high number of students attending, and it should still be possible to ensure a good quality of supervision [23]. One way to make this more manageable is by having some of the feedback be automatic feedback [29].

With the introduction of a course designed to use PBL, there is also the option of looking to the industry. When it comes to agile development, an important element

of many methodologies is the review. A review in agile development is a meeting to show what has been done in the latest sprint (work iteration), and get feedback from the stakeholders and team members present. Feedback in project-based courses, especially those that have the students act as agile teams, can, with benefits, be done as reviews [30]. Short, frequent review meetings can offer students quality feedback with limited resources and introduce formative assessments to the course.

2.2. Version Control Systems

A version control system (VCS) is a system that manages changes in your software product over time. It stores the current version of your files as well as a history of the changes made to these. The first VCS was released in 1972, and was called Source Code Control System (SCCS) [31]. SCCS made it possible to save changes of a file, and thereby have different versions of the file. But it did not let multiple authors change the same file simultaneously. After SCCS, VCS have gradually improved in functionality and ease of use. Among the many alternatives today, some of the popular ones are Git, Concurrent Versions System (CVS), Apache Subversion (SVN), and Mercurial.

In their book, "Pro Git", Scott Chacon and Ben Straub write about three types of version control systems [32]. The early systems were local version control systems. That means that the files and changes were stored locally. These systems were typically intended for use by a single person to keep versions of their own work. Examples of local VCS are SCCS and Revision Control System (RCS).

The second type of version control system is called a centralized version control system. In these systems, the files and changes are stored on a server. This allows for easier collaboration, and an administrator can control access and permissions. However, there is a single point of failure. If something were to happen to the server, the team can lose everything they had stored there with the exception of the local copies that team members have on their machines. Examples of centralized version control systems are Azure DevOps Server, CVS, and SVN.

The last type of version control system is distributed version control system. In these systems, each client retains the whole repository with its full history. This makes for simple recovery of data if the server should be corrupted, as long as at least one client has checked out a version recently. Examples of distributed version control systems are Git and Mercurial.

Version control systems are an integral part of the software engineering industry in our time. It allows for collaboration on the same code simultaneously. Files are stored in versions, which makes it possible to revert changes or compare the state of your files at different times. The redundancy that comes with distributed version control systems also ensures the existence of backup copies of the files in the different clients that have checked out a version.

2.2.1. Git

Git is a distributed version control system (DVCS) that was initially released in April 2005. It was created by Linus Torvald and was made for source control management for the development of the Linux kernel [31]. Google Trends offers comparisons between different search terms and thereby a good estimate for the popularity of the different VCS. Comparing Git, Mercurial, SVN, and CVS for 2021, shows that Git is by far the most searched of the four competitors (See Appendix A).

Where most other version control systems store data as files and changes to these, Git stores snapshots of the full state of your project. If a file has not been changed, instead of storing a new copy it stores a reference to the file from a previous snapshot [32].

Another difference between Git and most other VCS is how lightweight branching is implemented. Branching in Git is to make a new pointer to a snapshot. Development can then continue in both branches, leading to different versions of the product being worked on in parallel. Later the branches can be merged together. This allows for team members to work on different features without their work in progress (WIP) breaking the code. This way, all members can work on a functioning program, and fully implement and test features in a contained manner.

The Three Sections of a Git Project

A git project has three sections: The working directory, the staging area, and the git repository [32]. The working directory is where you modify the code. When you want to store this, you move the modified files to the staging area. Next time you want to commit your changes, your modified files are committed from the staging area and stored in your repository. A git repository is where all the snapshots of the project are saved. Locally, this is in a .git folder.

Git Commands

When using Git there is a series of commands that will likely be used. These will be presented now:

git init	After having installed Git locally, you can initialize it in a folder by writing git init. This creates a new repository as a .git folder in the current location.
git clone	Gets a local copy of a Git repository from a URL.
git add	Adds files from your working directory to the staging area. Files can either be mentioned individually as arguments or the "." argument stages all modified files.
git commit	Stores a snapshot in the repository. The modified files that have been staged will have new versions in this new snapshot, while unmodified files will be stored as links to previous versions. Commits include an author and a comment, which can be very useful when cooperating on a shared code base.
git status	Shows modified files, and whether they are staged or not.
git branch	Lists local branches or makes a new branch if provided with a name as an argument.

git checkout	Switches to the branch whose name was provided as an argument.
git merge	Tries to merge the mentioned branch into the current branch. This might lead to a merge conflict if changes have been made in the same files in both branches.
git push	Sends local commits to a remote repository.
git pull	Gets the latest versions of remote branches and merges the remote version of the current branch with the local one.
git stash	Temporarily stores modified files from the working directory and restore the files there to the state of the last commit.

2.2.2. GitLab

GitLab is a DevOps Platform that was founded by Dmitriy Zaporozhets and Valeriy Sizov in October 2011. It follows an open core model. As of May 2022, it has an estimate of over 30 million registered users [33].

GitLab is a repository manager, which keeps the remote repository for your GitLab projects. In addition to this, it also offers features for team planning, code reviews, wikis, Continuous Integration(CI), and much more. We will look at how GitLab handles two of these features, Source Code Management and Issue Tracking, now:

Source Code Management

Figure 2.1 shows the main page of a GitLab project. Here you can see the file structure of the project, and when the last changes were made in the different files. Looking at the menu to the left under Repository, different views can be accessed. The Commits View can be seen in Figure 2.2. From the GitLab repository, most Git commands can be executed through the user interface and directly to the remote repository. Files can be added or modified. Branches can be created and merged. And you have a simple overview of previous commits and branches, which can be compared to see changes between them.

Issue Tracking

Tracking a project through issues is a common way of doing project management in many of the different software development methodologies. Figure 2.3 shows how an issue looks in GitLab. An issue describes something to be done in order to improve the software product. In GitLab issues can be assigned to one or more developers. They can be assigned a milestone, given time estimates, due dates, and labels. And GitLab offers a space to discuss the issues. In agile methodologies such as Kanban and often in Scrum, you find issue boards where you categorize issues and place them in separate columns. Figure 2.4 shows GitLab's version of an issue board. This can be used for a team to keep track of which tasks should be done when and by who. This makes it easier to collaborate in agile teams.

GitLab APIs

GitLab also makes project data accessible through its APIs. Public information can easily be gathered from their endpoints, while private data can be accessed if the requester is authenticated either through an OAuth2 token, a personal access token, a project access token, session cookies, or, for some endpoints, through a GitLab CI/CD job token. Access to the data from the API makes it easy for developers to use their data for different purposes, one being for process mining.

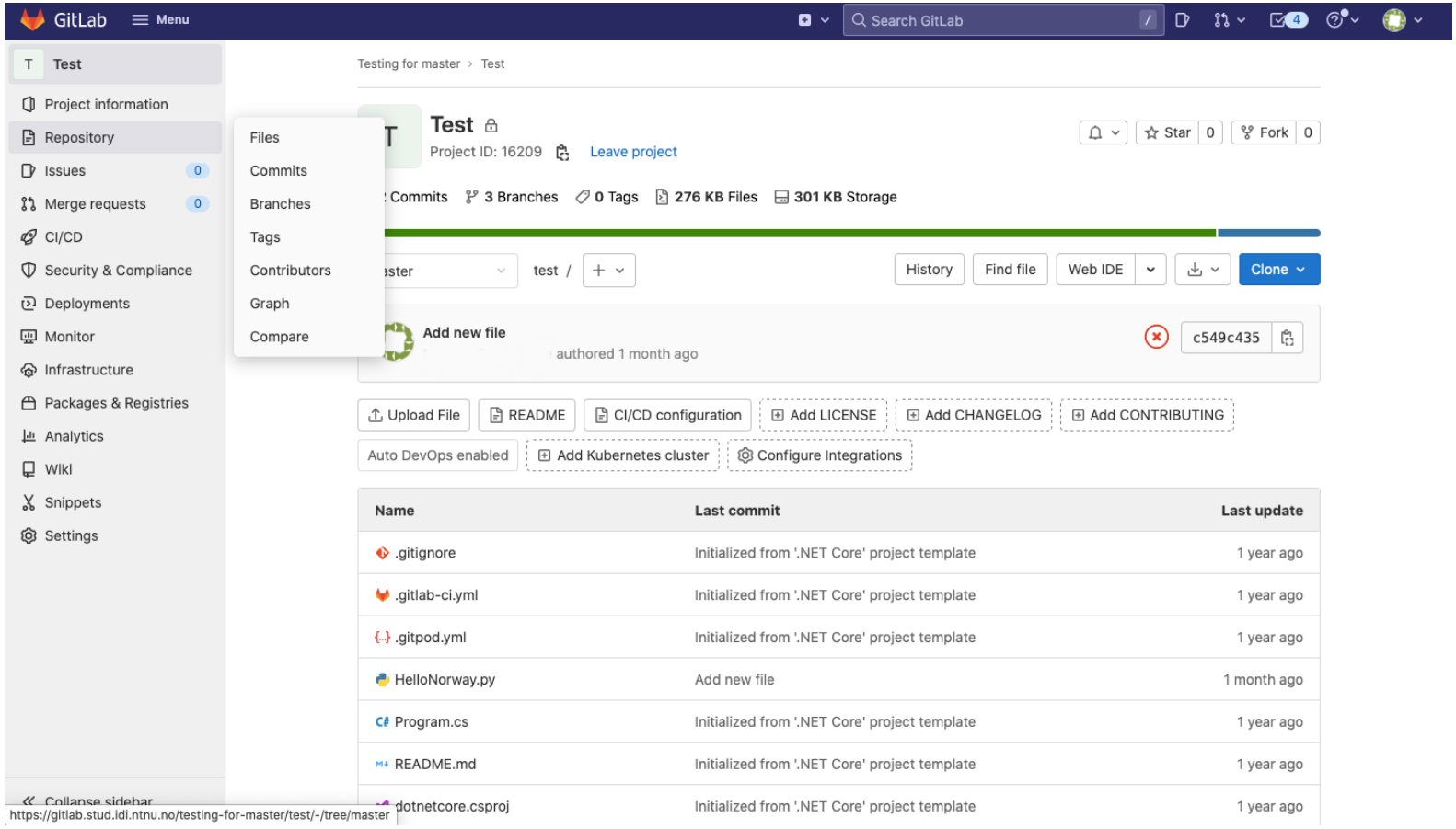


Figure 2.1.: Repository management in GitLab.

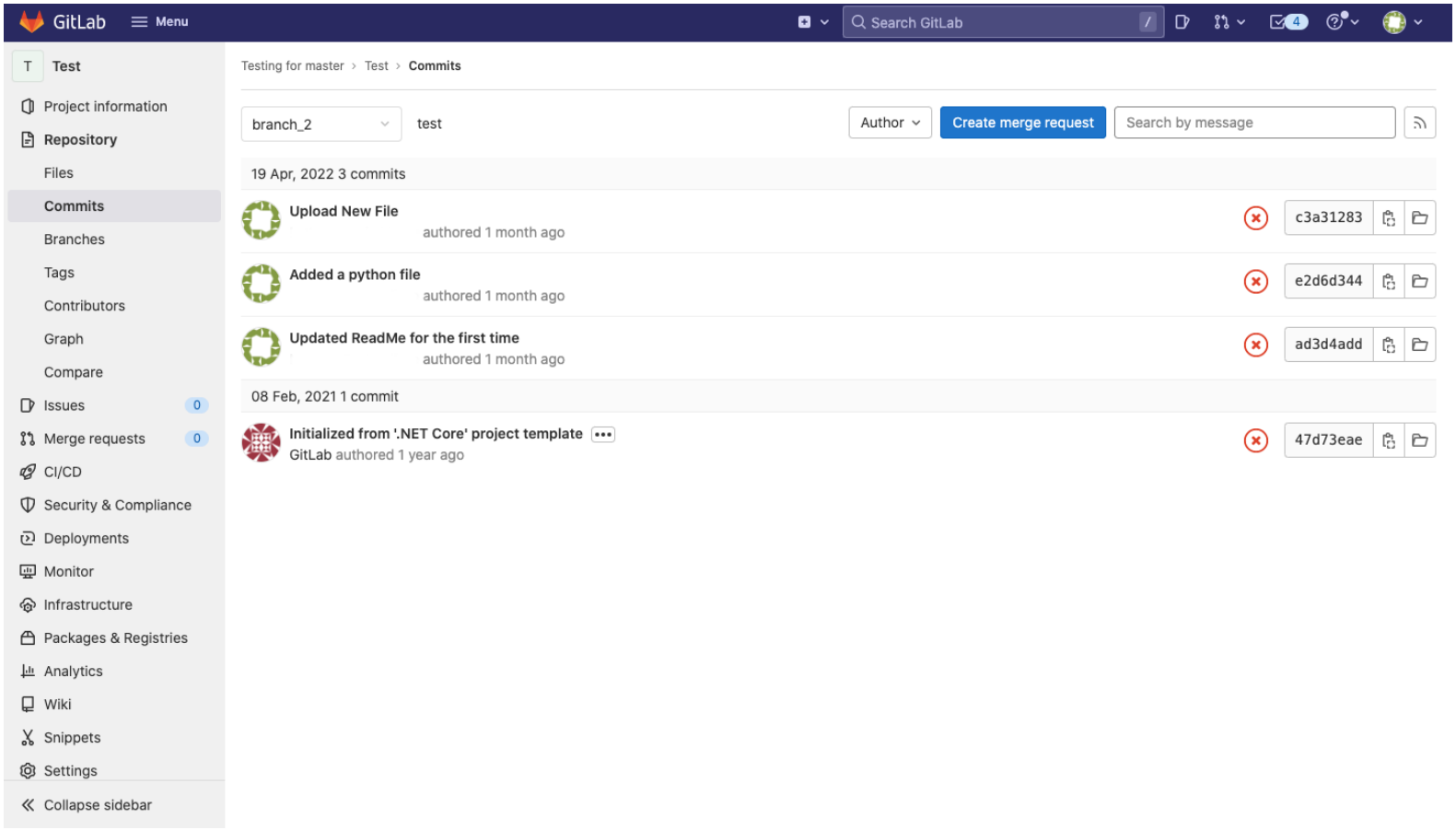


Figure 2.2.: Overview of commits in a branch in GitLab.

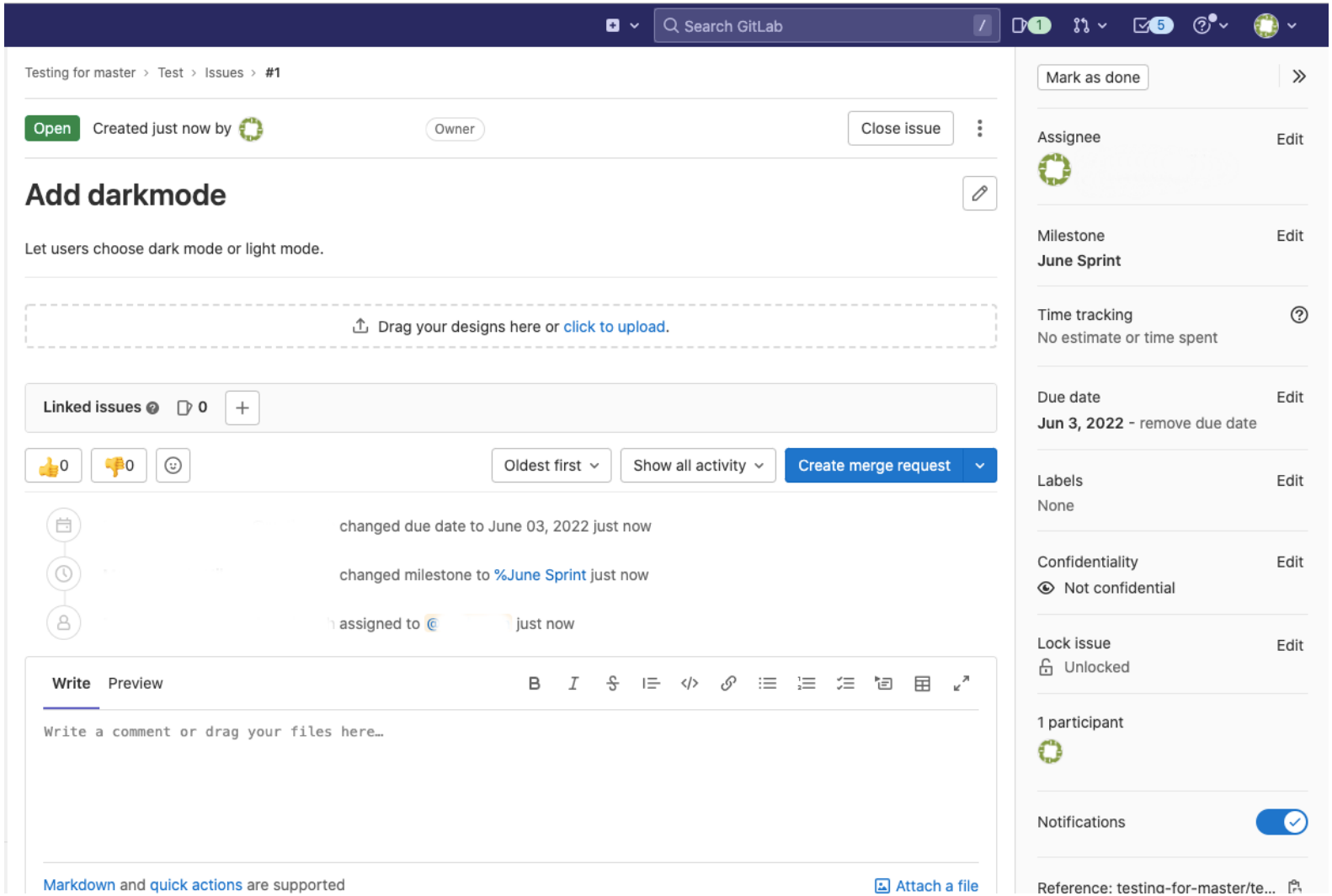


Figure 2.3.: A single issue page in GitLab.

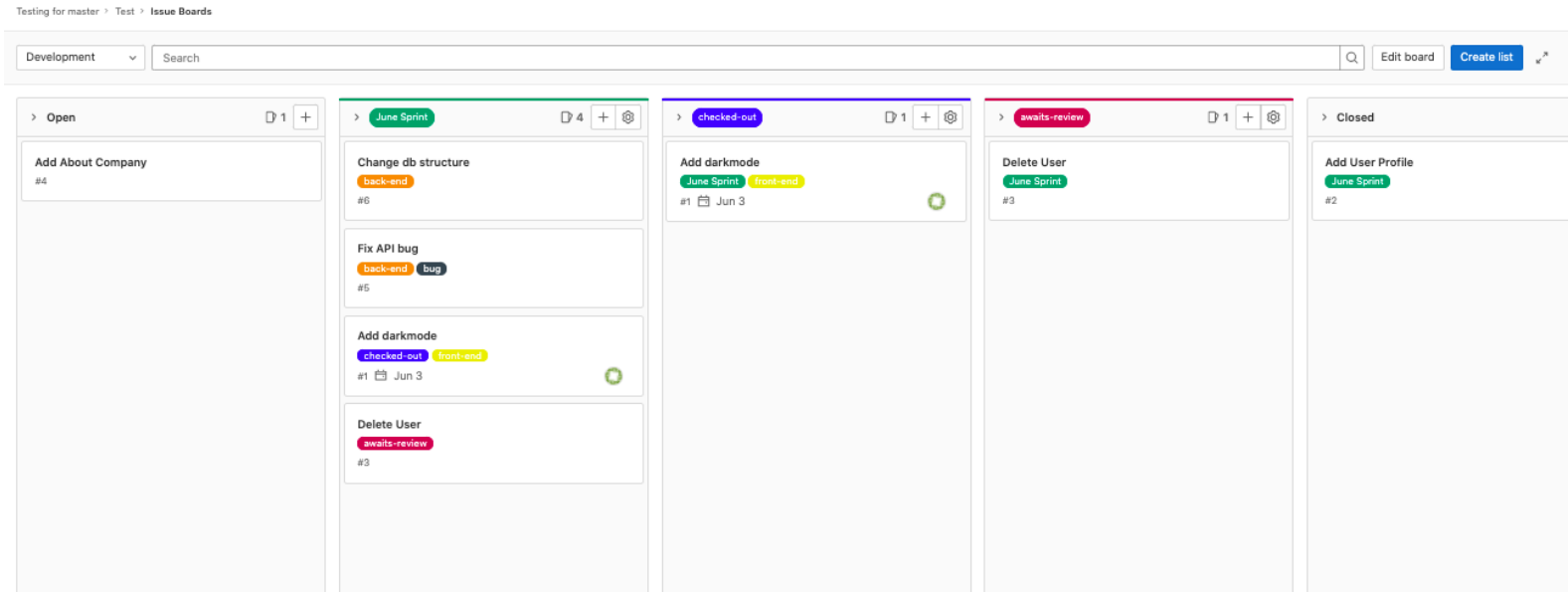


Figure 2.4.: Issue board in GitLab.

2.3. Process Mining for Software Engineering Education

Process mining is a discipline that incorporates data mining, process modeling, and analysis. According to van der Aalst [34], "the idea of process mining is to discover, monitor, and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's systems." Project-based software engineering courses that use distributed version control systems lend themselves well to process mining of VCS.

Mining Software Repository (MSR) is a field that focuses on data mining of software repositories, such as version control systems. Farias et al [35] did a systematic mapping study on recent MSR studies to determine the main purposes, focus, and object of analysis in these studies. The main purposes of recent MSR studies are comprehension, prediction, and identification. The focus tended to be on defects, with the second most prevalent focus being that of contribution and behavior of developers. The main objects of analysis are code, commit data, and bug reports.

In research in software engineering education, MSR has been used to try to automate some of the assessments in courses where this has been particularly resource-demanding. Studies have had success in finding relevant information to be considered when assessing students' work [36], but this did not lead to a fully automated assessment procedure.

Determining the contribution of individual members to a team is a complex endeavor, that has been tried in many studies [37] [38]. There is no single way of measuring member contribution. In Parizi, Spoletini, and Singh [37] they used five metrics: number of commits, number of merge pull requests, number of files, total lines of code, and a calculated time spent on the project each day.

Another focus has been on identifying problems within the students' processes. Some studies have generated reports for instructors, so they could intervene if any dysfunctions were identified [39]. Mittal and Sureka [40] explored different metrics with the aim of revealing the student teams' process quality. Process aspects they inspected were: "workload distribution between team members, regularity in contributions from start till

the deadline, quality of commit messages, component and developer entropy, quality of efficiency of bug fixing process." They concluded that the metrics they explored could be useful for instructors to gain better insight into the students' processes, but were skeptical about using the data for grading since the aspects they had considered could easily be misrepresented in an attempt to give an impression of higher quality work.

2.4. Teamwork Dynamics

To achieve good collaborative learning, there are some aspects of teamwork dynamics that need to be working. This section presents some typical teamwork dysfunctions found in literature and presents a teamwork dysfunction model. After that my specialization project, which is used as a foundation for deciding on dysfunctions to focus on for this thesis, is presented.

2.4.1. Dysfunctions in Teamwork

A major problem within teams can be an uneven distribution of work. If one person in a team does most of the work, he keeps the others from learning, and he may reduce his own learning opportunities [41]. More often, the problem is social loafing. Social loafing is defined by Karau et al [42] as "... the tendency for individuals to expend less effort when working collectively than when working individually". As mentioned in Section 2.1, a person who does not contribute their part in a project is called a "free-rider". According to Pfaff, [43], the lack of a free-rider in a group was a significant predictor of members' good attitude to teamwork.

A teamwork dysfunction that is specific to learning processes is that of having too specialized tasks. If the work is divided there is a chance that the students only learn the part they do, and miss out on many of the learning opportunities [44].

Another dysfunction in a team project can be that of time management. Kuhrmann and Münch [45] made a setting where student teams could test different stress factors. In the short-term project of the study, time constraints and missing strategies were the

factors the students felt impacted them the most.

Lack of commitment is an issue that can be dysfunctional in teamwork [46]. This lack of commitment from a team member can weaken team cohesion [47]. There are some ways to mitigate this problem. If it is possible in the work situation, teams could be shuffled periodically. Another option is to increase individual accountability [48]. In an educational setting, if a problem student is identified, an instructor may have to confront the student about their behavior [43].

Communication is a teamwork quality that is often mentioned in literature [9] [49]. A lack of good communication and coordination can lead to a dysfunctional team. Hansen [50] did a literature review on the benefits and problems with using student teams. One suggestion for improving team projects was to have members have assigned roles. He found that assigning students with specific responsibilities helped reduce the free-riding problem. An option that may give better cooperation and performance is to have rotating roles and authority [51].

2.4.2. The Lencioni Model of Dysfunctions in Teamwork

The Lencioni model [46] seen in Figure 2.5 was developed by Patrick Lencioni. In 2012 he published a book with a detailed example of how his model could be used in a fictional business setting. The model consists of five important dysfunctions that hinder good teamwork: Absence of trust, fear of conflict, lack of commitment, avoidance of accountability, and inattention to results.

2.4.3. Specialization Project

During my Specialization Project, Fall 2021 [3], literature from Team Theory and literature about Process Mining of student VCS was explored with the aim of discovering which dysfunctions would be relevant for further studies in a master's thesis. Nineteen common dysfunctions were identified. These can be seen in Table 2.1. Eight of these dysfunctions were considered particularly difficult to identify from GitLab data. These were discarded. Some of the dysfunction had enough similarities to be grouped as one. Two of the

Figure 2.5.: Lencioni model of dysfunctions in teamwork.

remaining dysfunctions, “Negative affective tone” and “Fatigue” were discarded because they were considered more difficult to identify from GitLab data than the remaining ones.

Table 2.1.: Dysfunctions identified during specialization project [3].

Dysfunction	**Identifiable**
Inattention to results	✓
Avoidance of accountability	X
Lack of commitment / contribution	✓
Fear of conflict	X
Absence of trust	X
Negative affective tone	X *
Negative behavior and attitude	X
Social loafing	✓
Fatigue	X *
Lack of openness	X
Time constraints	✓
Poor communication / lack of communication	✓
Unequal distribution of labor	✓
Personality clashes	X
Performance drops	X **
Missing strategy / No plan of attack	✓
Poor leadership	✓
Lack of management support	X
Too specialized (Students do not learn all they should)	✓

3. Methodology

This chapter will give a detailed description of the research strategy, data collection methods and analysis done in the study. Oates [1] is used as the theoretical foundation for the choices of strategies and data collection methods.

3.1. Research Strategy

According to Oates: "A strategy is your overall approach to answering your research question." The book goes on to introduce six common strategies: surveys, design and creation, experiments, case studies, action research, and ethnographies. For this project, a case study was chosen as the best fitting research strategy.

3.1.1. Case Study

A case study is a strategy where we focus on one instance of something and delve deep into that case with the aim of generalizing new knowledge about the topic being explored. There are a few reasons why a case study is a fitting strategy for this project. First, the course chosen for this study has a large number of students and student groups, and it follows a structure that is common in many software engineering courses. It is therefore possible to make generalizations from the findings in this study that can be of value to coordinators of other similar courses. The case will be further expanded upon in Section 3.1.2. Another reason why a case study is a good choice for this study is because identifications of metrics that can predict problems with group dynamics is complex. There can be no isolation of problems to be studied, the way you would in an experiment.

It is necessary to look at the group and their work from different perspectives, and in a true environment.

In the design of a case study there are a few decisions that need to be made. These will be explored now.

Types of Case Studies

Oates describes three types of case studies: A descriptive study, an explanatory study, and an exploratory study.

This study is an exploratory study that seeks to find challenges and opportunities for the use of software repository mining in a project-based software engineering course. The aim is to get an understanding of how to use mining of GitLab in this setting, and the challenges that should be considered when implementing the technique in a similar case.

Approach to Time

In Oates [1], three approaches to time are presented: A historical study, a short-term, contemporary study, and a longitudinal study.

This study is a short-term, contemporary study. That means it examines a case that is occurring at the time of the research. The course that is chosen as the case is held during the same semester as the data collection happens. This is the best suited approach to time for this study. We are interested in seeing if dysfunctions in a team can be predicted during the semester. A historical study could have been an option if the chosen case was a course that had been held in a previous semester, but since it introduces the uncertainty of the participants' memories, and the case chosen is an appropriate one for this study, that was not an option that was considered.

Selection of Cases

Being a student writing my thesis at the Norwegian University of Science and Technology, and wanting to study dysfunctional student teams in software engineering, it was natural to seek a collaboration with a course at the university. TDT4140 Software Engineering

was chosen because it is a project course with a large number of students participating. It is also a course that is similar to many software engineering courses offered worldwide. It is what is considered a typical instance. As mentioned above, this is a case that can be generalized to similar courses and with the large number of students attending, it allows for gathering sufficient data.

Generalizations

Oates [1] presents the four main types of generalizations suggested in Walsham's article from 1995 [52]. They are: A concept, a theory, implications, and rich insights.

This study does not aim to contribute new concepts or theories, but will explore what implications the findings might have for similar cases. Implications are defined in Oates as: "suggestions about what might happen in other similar instances, possibly with specific recommendations for action."

Data Generation Methods

The data generation methods chosen for this study are a questionnaire, interviews, and data collected from GitLab. These methods will be discussed further in Sections 3.2.1, 3.2.2, and 3.2.3.

Research Process

Figure 3.1 shows the process followed in this study. At first, during a specialization project fall 2021 **literature was explored** to find metrics for the focus of this study, as well as to find inspiration for **research questions** that make up the aim of this study. **A case study** was chosen to be the best fitting research strategy. Data would be gathered through **interviews**, **questionnaires**, and **documents** in the form of GitLab data. The data collected would be both **qualitative** and **quantitative** in nature, and require both kinds of analysis.

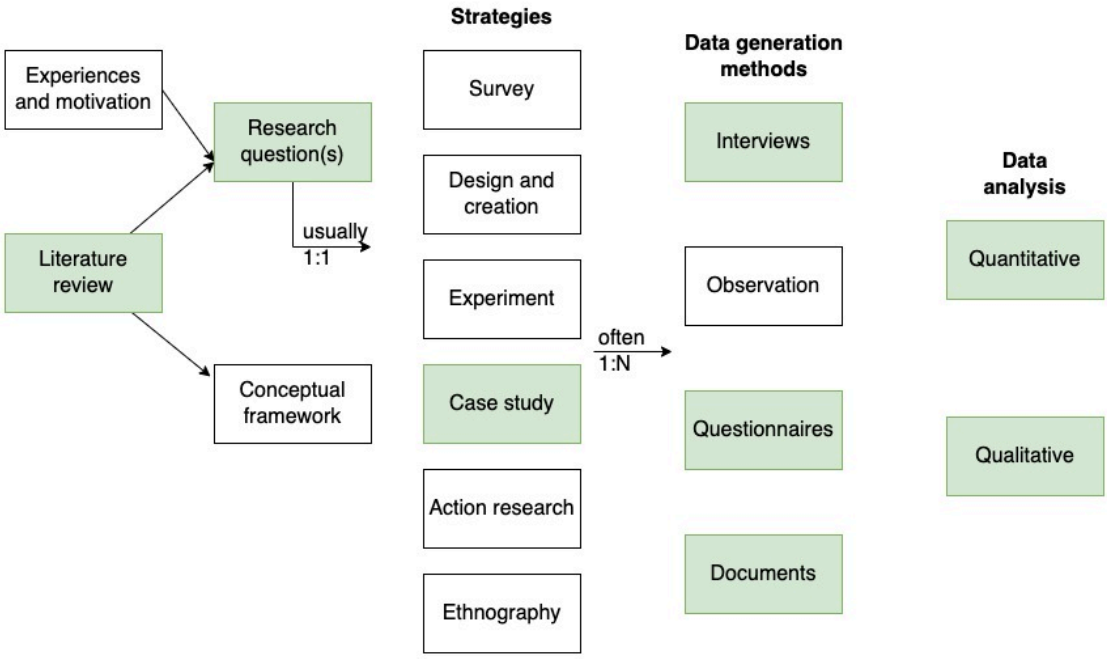


Figure 3.1.: Model of the research process [1, p. 33].

3.1.2. TDT4140 Software Engineering Spring 2022

As mentioned in Section 3.1.1, TDT4140 was chosen as an appropriate case for this study. TDT4140 Software Engineering is a course offered at NTNU. According to the course website [53], it focuses on: "Practical and theoretical understanding of software engineering for small, co-located development teams, with special emphasis on development processes, requirements engineering, software quality, and technological choices."

Structure of the Course

TDT4140 is a project-based course where the students are divided into teams to complete a semester project. The groups are free to make most of their choices during development, including the technology stack. But, the course has a focus on agile development, and the groups are required to use the Scrum methodology and document their process in different deliverables throughout the semester.

During spring 2022 the course was divided into five phases, as can be seen in Figure

3.2. The first phase was a theory module that lasted for three weeks. During these weeks the students had six four-hour lectures about general software development and themes in agile development. For this phase, they had one submission: S2 - Theory test, where the students were individually tested on the knowledge they had gained during the phase.

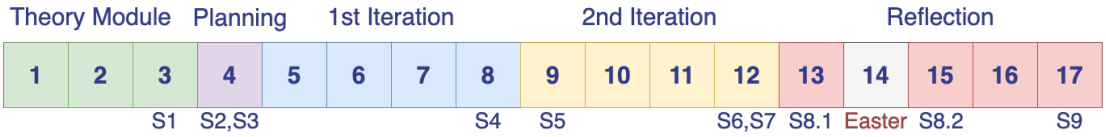


Figure 3.2.: Plan for TDT4140 Software Engineering spring 2022. [2]

The next phase was the planning phase. The student groups met their supervisor, who also acted as the product owner for the project. The supervisor described the product they wanted the students to make, and during the planning process, the team started the preparation for the project by completing two submissions: S1 - Group Contract. This was a document the student teams created together that set the rules and expectations for the project work; and S3 - Pre-study. In this delivery, the students made a report documenting and arguing their choices for the project.

During the first iteration, the student groups started the development. At the end of the first iteration, they had two assessments: S4 - Review. Here the groups demonstrated their work to the product owner and another teaching assistant who acted as an evaluator; and S5 - Retrospective. The groups had a retrospective meeting and delivered a report, where they evaluated and reflected upon their process, teamwork, and the state of their product. The second iteration was carried out in the same way as the first one, ending in two assessments, S6 - Review 2, and S7 - Retrospective 2.

The last phase had a focus on reflection. During this time, the students had three submissions. First, in S8.1 each individual student wrote a reflection essay, where they could show what they had learned in the course by connecting theory and experience about a given theme in the course curriculum. The second deliverable, S8.2, was a peer review of two other students' essays. Lastly, the group wrote a group reflection report in S9. The report should discuss their project work and process work concerning the course

curriculum and account for 30% of the students' grades.

The course was divided into four villages. Each village had seven teaching assistants, and one of them was the village leader. The remaining six teaching assistants each supervised three student teams and acted as evaluators for three other student groups. For the groups they supervised, they also acted as product owners for the project, and they had weekly meetings with the groups.

Students

Approximately 500 students attended the course during spring 2022, during the students' fourth semester. These students came from eight different study programs. Some of the students had participated in a project-based software engineering course before, while others had minimal experience with software development.

3.1.3. Setting the Scope for the Project

In Section 2.4.3 the work done in my Specialization project was presented. From the dysfunctions found in Table 2.1 six final dysfunctions were formed and focused on in the study. These were chosen as the ones that were most likely to be identifiable from GitLab data, and they are problems that might occur in the student groups that make up the participants of the study.

D1: Unfair or unevenly distributed workload.

This is defined as: Some in the group put more time and effort into the project than others.

D2: Too little time is spent working on the project.

This is defined as: The group members do not spend sufficient time working on the project compared to what is planned or expected.

D3: The participants are not committed to the project.

This is defined as: A lack of effort and willingness to work for good results.

D4: Lack of a plan or strategy for the project.

This is defined as: The group does not communicate about the approach and strategy for the project.

D5: Poor leadership.

This is defined as: The group does not have anybody who takes responsibility for the project, its progress, and implementation.

D6: Too specialized tasks.

This is defined as: The group has distributed work tasks to a degree where the members do not learn all they are supposed to in the course.

3.2. Data Generation Methods

From Oats, a data generation method is defined as: "The means by which you produce empirical (field) data or evidence." We look at the four data generation methods covered in the book:

Interviews: Interviews are conversations between a researcher and a participant who gives the researcher information for their study. Interviews can be structured, semi-structured, or unstructured.

Observations: Observations are when a researcher studies the participants of their study through direct observation to see what they do, not just what they report to do.

Questionnaires: Questionnaires are a set of questions the participants of a study answer. The researcher then analyzes the responses and looks for patterns.

Documents: Documents can be found documents, where the information is produced outside of the study and gathered and analyzed by the researcher. They can also be researcher-generated documents, where the researcher documents things that would not exist if not for the study.

In this case study, the data generation methods are interviews with teaching assistants, a questionnaire sent to the students of the course, and documents in the form of data extracted from GitLab.

3.2.1. Online Questionnaire

In this study, different metrics derived from GitLab data are compared to the students' experiences of group dynamics in their teams. To collect data about the students' experiences, a self-administered questionnaire was chosen as a fitting data generation method.

Questionnaire Design

For each of the six dysfunctions presented in Section 3.1.3, three questions were made to find the student's experience with that problem in their team. The majority of the questions were formatted as Likert scale questions with the options "Strongly Disagree", "Disagree", "Neutral", "Agree", and "Strongly Agree". To make sure the students read the questions and considered them before answering, some questions had a "positive" answer being "Strongly Agree", while others had "negative" answers being "Strongly Agree". The questions were not sorted by theme for the same reason. The idea was that if the questions were too predictable, in that the questions belonging to the same dysfunction followed each other, they would be read less carefully by the students. After answering the Likert questions, a single open question followed, which asked the students about their opinion about their GitLab data being gathered and analyzed to give them better feedback from their supervisors.

The full questionnaire can be found in Appendix B.

Data Collection

The questionnaire was made and hosted in nettskjema.no [54]. The website has a focus on security which was a requirement for a questionnaire that gathered anonymous data from the participants. The students were given a five-day window to answer the questionnaire. The majority of the responses came within the first two days. Since a high response rate was considered important, a gift card was promised to a randomly selected student who answered the questionnaire. Registration for the chance of winning the gift card happened voluntarily in a Google Form so that the students' emails could not be connected to their responses, and any student who wanted to participate in the study and be entirely anonymous had that option. Information about the questionnaire was posted to Blackboard, the Learning Management System(LMS) used by NTNU. It was also sent to the students per email. In the end, 52 students answered the questionnaire. With a total of approximately 500 students taking the course, this gives a response rate of 10%.

3.2.2. Interviews

This study is motivated by the prospect of using GitLab data to better guide student teams. An important perspective of the case is therefore that of the supervisors of the groups. As explained in Section 3.1.2, the teaching assistants had dual responsibilities. They acted as supervisors and product owners for some groups, and they acted as evaluators for other groups. Teaching assistants were therefore invited to partake in interviews regarding their roles as supervisors, and what they had observed from the student groups they were guiding.

Interview Guide

The aim of the interviews had two focuses:

1) To find out which of the six dysfunctions the supervisors identified in the student teams, and how they concluded that it was a problem in the group.

2) To find out how supervisors preferred to have findings from GitLab data presented to them, to be effective in helping the students. And to see if the supervisors could see any possible problems in using GitLab data when helping the student teams.

To best answer these questions an interview guide was prepared. The first part of the interview revolved around the first focus. For each group, the teaching assistant supervised they were first asked which problems they had identified in the group. After that, they were given the definitions of the six dysfunctions introduced in Section 3.1.3. For each of them, the supervisor was asked whether they could see the problem in the group and how it was observable to them. Then they were asked to rate the problem in the group following the traffic light model.

The traffic light model is a version of Likert scale questions, and was introduced to the teaching assistants thus: Green means that the group does not face this challenge at all, or that their problems from this challenge are insignificant. Yellow means that the group faces the challenge and that there can be a risk of them developing greater problems from it. Red means that the challenge gives the group significant problems.

The second part of the interview concerned the second focus. Here, the supervisors were asked how they preferred to have findings from the GitLab data presented, what should be considered when presenting this data to supervisors, and if they could see any problems using this data to help guide students.

The full interview guide can be found in Appendix C.

Data Collection

When finding participants for the interviews, I was set in contact with two villages. While recruiting, I was allowed to take part in a weekly meeting to present my research. From the first village, only two were willing to participate. For the second village, I offered free lunch sponsored by the department as compensation. This led to an increase in interest, but eventually, three teaching assistants from this village partook in interviews. In total, five teaching assistants were interviewed. As mentioned in Section 3.1.2, each village

had six teaching assistants acting as supervisors. Out of the teaching assistants invited to interviews, 42% accepted. Out of all the teaching assistants in the course, 21% were interviewed.

Two of the interviews were performed online using a video conferencing tool. The remaining three interviews were performed in person. There is no reason to suspect that the format of the interview impacted the responses of the participants.

3.2.3. GitLab Data

Section 2.2.2 shows details about the tool. The purpose of this study is to see if data from GitLab can be analyzed to predict problems in student teams. The dataset collected from GitLab should include data fields that make it possible to construct metrics that might reveal the six dysfunctions presented in Section 3.1.3.

Data Collection

To collect data from GitLab a Python script was made to fetch the following fields through the GitLab API:

1) Commits:

- author
- title
- message
- authored date
- committed date
- additions, deletions, and total of the two
- diff: old path, new path, changes in file (There were a few other fields that were not used within the diff)

2) Issues:

- title

- description
- date created
- date updated
- date closed
- issue state (opened/closed)
- author
- issue type
- time statistics
- status of tasks completed
- milestone due date
- milestone start date
- assignees

Anonymization

According to Oates [1], a typical disadvantage of case studies is that it can be difficult and time-consuming to gain the right access to necessary data sources. For me, getting access to student repositories was a hurdle in this study. After negotiation with the Norwegian center for research data (NSD) the options were:

1) *Informed consent from all participants to use their data.* The problem with this option is that to study the group dynamics of a group, all members would have to opt into the study. The risk here would be that few or no groups would agree to this.

2) *Extracting an anonymized dataset from GitLab and using the anonymized data in the study.* The problem with this approach is mainly that it can be difficult to do quality control of the dataset. This will be discussed in Section 5.3.3.

The risks of 1), that there would be given consent to extract too little data to support the research, were deemed to be too great. Therefore, work was done to create an anonymized dataset for the analysis. The anonymization happened as the dataset was

extracted. The author field in both commits and issues was anonymized and given incremental IDs. The anonymization process introduced some weaknesses and possible uncertainties to the dataset. These will also be discussed in Section 5.3.3.

Measures of the Groups' Experiences with the Dysfunctions

For the GitLab Analysis, different metrics were constructed and compared to measures of the groups' experiences with the previously introduced dysfunctions (See Section 3.1.3). In preparation for this comparison, the data from the questionnaire was processed. In the process the different questions were given numbers from 1-5, taking into account that some of the questions were negated. The questions were grouped into six categories, one for each dysfunction. For each response and each dysfunction, the mean value of the questions was calculated. Then, for groups with multiple respondents, the mean of the dysfunction values was calculated. This gave a dataset with 39 groups, that all had six dysfunction values. A few groups had to be excluded because they structured their projects into subprojects in a way that was difficult to process. Because of this, the final number of groups that were analyzed was 36.

Omitted Commits

At the beginning of the GitLab analysis, it became clear that a few of the commits were outliers, and they influenced the results of the analyses. These commits were omitted from the analyses. In total, for all groups, 5 commits were omitted. The condition for the omission was the date of the commits. A start date was set as the date of the first commit that could reasonably be expected to have been part of the project. One group used a template that had a commit by GitLab in 2019. The remaining four commits were all made after the end of the project, and it did not seem from the commit messages as if any additional work was done.

3.3. Analysis

The methods of analysis for the different data collected will be presented in this section. For the different data sources, it is necessary to do both qualitative and quantitative analysis.

3.3.1. Questionnaire

Analysis of the questionnaire was done in two ways. Quantitative analysis was done for the Likert scale questions. Here each question was explored to see what portion of the students experienced the problem. The different answers were coded to a value from 1-5. From this, the mean was calculated for each question. The mean represents how prevalent the problem is amongst the teams. For the three questions aiming at discovering the same dysfunction the average of the means were calculated to understand how much of a problem the dysfunction is for the class. The answers to the Likert scale questions were further used in the GitLab analysis. This was explained in Section 3.2.3.

The last question of the questionnaire was analyzed qualitatively. Each response was read carefully and the problems were categorized and will be presented in Section 4.1.3.

Test of Quality of Questions

After designing the questionnaire, it was pre-tested by my supervisor and two fellow master's students. Due to time constraints, a pilot test was not performed. An assumption was made that the extra time it would take to conduct a pilot test, would lead to a lower response rate, as this would mean the data collection would happen closer to the students' exams. A high response rate was considered more important than an extra step of quality control before data collection. To compensate for the lack of a pilot test, quality control of the questions in the questionnaire was performed after the data was collected. To test whether students answer consistently on questions that aim at revealing the same dysfunction we look at each student's responses to the three questions. For each question, the responses are given a value between 1-5. If the response indicates that the

group faces the problem ("Strongly Agree" to a question that asks if they face a problem or "Strongly Disagree" to negated questions) the corresponding value is 1, and if the response indicates that there is no problem the corresponding value is 5. A high number is an optimal situation in a group, while a low number means that they experience a problem. The distance from the lowest score to the highest score for a student reveals the consistency in their answer for that dysfunction. Averaging the distance for each student gives a measure of the consistency in answers for a dysfunction. This is done for all the dysfunctions and the results can be found in Section 4.1.1.

3.3.2. Interviews

The interviews were first transcribed, then analyzed mostly qualitatively. For each dysfunction, it is explained how prevalent it is in the groups and how the teaching assistants identified the problem. The groups that the teaching assistants were supervisors for were all classified as somewhere between green and red. These classifications were compared to the students' answers to the questionnaire. In Section 5.1.11 the comparison between the two will be presented.

After focusing on the dysfunctions, the teaching assistants' opinions on how they would like the GitLab analyses presented to them are presented. Thereafter, we look at possible problems with using GitLab data for analysis of the group dynamics, as identified by the teaching assistants.

3.3.3. GitLab Data

For the analysis of the GitLab data that was collected in this study, a list of possible metrics that could reveal problems within the groups was created and edited throughout the process. As new information was learned the priority of the different metrics changed. The metrics that were explored are not comprehensive. The final metrics will be described in this section, and the results of the analysis will be presented in Section 4.3.

Tops in Work Done

For each of the days in the project duration, the work done that day was measured. Work done was first measured in commits. The number of commits in total was calculated. Then, for each day the group committed, the number of commits was counted. This lets us calculate the portion of the total work for each day. Tops are the days where the commit portion is over a certain percentage. After checking different options, the ones that had the closest to a linear relationship for the portion of commits and group experience were 1% and 15%. Counting up the number of days exceeding the set percentages, we have the following metrics:

- Number of Days With More Than 1% of Total Commits
- Number of Days With More Than 15% of Total Commits

A commit can be anything from a few lines to multiple features being implemented. Therefore, another method of calculating work was also used, that is lines of code. However, looking at the total code lines for each commit, the work done by the students would drown in the imported Libraries, and automatically generated files. Therefore, some processing had to be done to get a realistic and useful measure for code lines. First, commits that were merges were disregarded. This might mean that some work could be overlooked, but the member doing the merge should not necessarily get credited with the code lines written in the branch that is merged. Also, if this was not done, most work would be counted at least twice, because the groups generally followed the practice of not pushing directly to the master branch.

The groups were allowed to choose a technology stack at will. This meant that the structure of files and file endings differed for each group. After printing each file ending, to see which would contain student work, and examining which locations did not contain work done by the students, the following file endings were included, and paths that included the following strings were excluded:

> **The following file endings were included:** ".py", ".js", ".css", ".ts", ".jsx", ".java", ".vue", ".tsx", ".md", ".html", ".scss", ".kt", ".sql"

Paths including the following strings were excluded: “/migrations/”, “/.bin/”, “/bin/”, “/.env/”, “/Lib/”, “settings.py”, “.config”, “cypress”, “node_modules”, “package.json”, “package-lock.json”, “/index.html”, “LICENSE”

After doing the same process for code lines as was done for commits, we get the following metric:

- Number of Days With More Than 7% of Total Code Lines

Last Minute Work

Something else that might reveal problems in group dynamics is how much of the total work is done right before a deadline. To measure this, the portion of the total code lines for the project done within the last 5 days before the final deadline was calculated. Since metrics that can reveal the problem before the end of the semester are preferred, the same metric is calculated for the last 5 days before the Midterm Deadline. This gives us the following metrics:

- Percentage of Total Work Done the Last 5 Days Before Midterm Deadline
- Percentage of Total Work Done the Last 5 Days Before End of Project Deadline

Time of Commits

To reveal the students’ work habits, and their relationships especially with their planning, we explore when commits are made. To do this, the committed date field is used to find the day of the week or the time of the day to get the following metrics:

- Percentage of Total Lines of Code Committed During Weekends
- Percentage of Total Lines of Code Committed Between 08:00 - 16:00
- Percentage of Total Lines of Code Committed Between 16:00 - 00:00
- Percentage of Total Lines of Code Committed Between 00:00 - 08:00

Workload Distribution Within the Groups

It was also interesting to see how the work was distributed amongst the members. The attempt to look into this had some weaknesses for this project. Each commit has an author field that is used to link it to a member of the student group. This name, however, can be different from the name of the student. Typically, it can be the email address the student uses in their integrated development environment(IDE) or the email that is connected to their GitHub account. But it can be something else as well, and a single student may have different author names throughout the project. To try to see the distribution of commits amongst the members of the group the following was done in the anonymization process: Every time a commit was processed that had a new author name it was given an anonymized ID. This ID replaced the original name in the data set. A dictionary kept a connection between the first three letters of the author name and the anonymized ID. Then, the next time an author name starting with those three letters was processed it was given the same anonymized ID. The problem with this is that a single member could have different author names that did not necessarily start with the same three letters, and therefore be split into two members in the analysis. Another possible problem is that a group might have two members starting with the same three letters. They would then be merged into one member in the analysis. This is not ideal, and because of the anonymization process, it was not possible to check the frequency of these problems. However, the same way of connecting author name and team member was done in a master's project earlier with success [55]. The two measures for work that was introduced earlier were used here as well. For each member, their portion of the total commits and lines of code within the project was calculated. The standard deviation of the different members' portion of the work was then used as a metric that might reveal problems in group dynamics. We call the metrics:

- Standard Deviation - Portion of Commits per Member
- Standard Deviation - Portion of Lines of Code per Member

Comparison to Other Groups

From the data set, it is possible to look at a group compared to the other groups. It is interesting to see if how the group compares to the average at Midterm can reveal problems in their group dynamic. This is done here by calculating the average number of code lines for all the groups and comparing the group's code lines to the average. This gives the following metric:

- Group's Lines of Code Compared to the Average at Midterm

Issues

Information about issues in GitLab was also extracted in the data collection process. Using issue assignment as an additional measure of workload distribution was an option, but it suffers from the same weaknesses as commit distribution does. Looking into issues might still reveal useful relationships. Three metrics that were explored are these:

- Number of Unique Dates of Issue Creation
- Ratio of Edited to Non-Edited Issues
- Average Number of Days Between Creation of Issues and when it was Last Updated

Pearson Product-Moment Correlation

On statistics.laerd.com [4] the Pearson product-moment correlation is described:

> "The Pearson product-moment correlation is used to determine the strength and direction of a linear relationship between two continuous variables."

In the GitLab analysis in this study, the metrics described above are compared to the experience of dysfunctions within the student teams. To determine if any relationship exists, the Pearson product-moment correlation is calculated. This is done using SPSS [56], which is a statistical analysis program. The Pearson correlation, as it will be called in this section, is a number between -1 and 1. 1 is a perfect positive correlation between

the two data sets examined. -1 is a perfect negative correlation. 0 means there is no correlation.

For a Pearson correlation to be determined accurately, three conditions have to be met [4]:

1. There must be a linear relationship between the two variables.
2. There must be no significant outliers.
3. There must be bivariate normality in order to assess statistical significance.

For the last one, it is difficult to assess whether there is bivariate normality. A step taken instead to get to an acceptable level of assurance is to test both variables for normality. Normality is tested for in SPSS by doing the Shapiro-Wilk test.

For the group experience of dysfunctions, the results of the test of normality can be seen in Table 3.1. For there to be normality, the Shapiro-Wilk significance value must be greater than .05. As can be seen in the table, only three of the six dysfunctions fulfill the requirements of normality. D6 is particularly far from normality, and because of this, the Pearson correlations will not be calculated for metrics against D6. For D2 and D3 correlations will still be explored, since, according to laerd: "The test is somewhat robust to deviations from normality." In the cases where there are findings for D2 or D3, the lack of normality will be reiterated.

Table 3.1.: Shapiro-Wilk significance value for the six dysfunctions.

Dysfunction	Shapiro-Wilk Significance Value
D1	.256
D2	.040
D3	.020
D4	.114
D5	.065
D6	<.001

The normality of the data sets for the metrics will be described along with the results of the correlation analyses in Section 4.3. According to Oates [1], any correlation coefficient between 0.3 and 0.7 demonstrates a reasonable correlation. According to Cohen [57] the strength of the correlation is as seen in Table 3.2.

Table 3.2.: Strength of associations. Table found on statistics.laerd.com [4]

Coefficient value	Strength of association
0.1 <| r | <0.3	Small correlation
0.3 <| r | <0.5	Medium/moderate correlation
| r | >0.5	Large/strong correlation

4. Results

This chapter presents the results found in the study. Qualitative and quantitative findings from the questionnaire, qualitative findings from interviews with teaching assistants, and quantitative findings from Pearson correlation analysis of GitLab metric data sets are all presented here.

4.1. Questionnaire

This section will present the results of the questionnaire that was described in Section 3.2.1. First, the Likert scale questions will be presented grouped by the dysfunction they attempt to identify. Next, we will look at the similarity in answers between members of teams that had multiple members respond to the questionnaire. After that, the students' opinions about GitLab data being used to analyze their group dynamics will be explored.

4.1.1. Dysfunctions

For each dysfunction defined in Section 3.1.3 three questions were asked to the students. In this section, we will look at how the students answered these questions, and what that says about the dysfunctions in the student teams. The questions here are translated from Norwegian. As explained in Section 3.3.1 each option in the Likert scale part of the questionnaire is given a value between 1-5. These are used to calculate the mean response for each question and each dysfunction in this section.

D1: Unfair or unevenly distributed workload

In an attempt to learn the students' experience of this dysfunction in their team, three questions were asked:

1. Someone in our group has spent more time on the project than others.
2. The effort put into the project was unevenly distributed in our group.
3. In our group, every one put in an approximately equal amount of work to the project.

The distribution of answers can be found in Table 4.1. The average score of the three questions is 2.4. There is a difference of 1.0 between question 1 and the two other questions.

To test whether students answer consistently on questions that aim at revealing the same dysfunction we use the method described in Section 3.3.1. This gives a consistency score of 1.35 for D1.

Table 4.1.: Questionnaire responses to D1 - Unfair or unevenly distributed workload.

Response	Q1	Q2	Q3
Strongly Disagree	1.9% (N=1)	9.6% (N=5)	13.5% (N=7)
Disagree	3.8% (N=2)	23.1% (N=12)	40.4% (N=21)
Neutral	3.8% (N=2)	15.4% (N=8)	11.5% (N=6)
Agree	44.2% (N=23)	34.6% (N=18)	30.8% (N=16)
Strongly Agree	46.2% (N=24)	17.3% (N=9)	3.8% (N=2)

D2: Too little time is spent working on the project

To find out how the students experienced this dysfunction in their team, they answered the following three questions.

1. Our group worked approximately the number of hours expected in the course.

2. In our group, some members worked less on the project than we had agreed upon.

3. We experienced a shortage of time on the project because we did not spend enough hours compared to what was planned.

The distribution of answers can be found in Table 4.2. The average score of the three questions is 3.4. There is a difference of 0.6 between question 2 and question 3. The consistency score for D2 is 1.81.

Table 4.2.: Questionnaire responses to D2 - Too little time is spent working on the project.

Response	Q1	Q2	Q3
Strongly Disagree	5.8% (N=3)	15.4% (N=8)	23.1% (N=12)
Disagree	28.8% (N=15)	36.5% (N=19)	48.1% (N=25)
Neutral	9.6% (N=5)	11.5% (N=6)	15.4% (N=8)
Agree	42.3% (N=22)	21.2% (N=11)	7.7% (N=4)
Strongly Agree	13.5% (N=7)	15.4% (N=8)	5.8% (N=3)

D3: The participants are not committed to the project

To figure out the students' experience with this dysfunction in their team, these three questions were asked.

1. In our group, some showed a lack of effort.

2. Everyone in our group worked to get a good grade.

3. We experienced that not everyone in the group was committed to the work.

The distribution of answers can be found in Table 4.3. The average score of the three questions is 3.7. There is a difference of 1.1 between question 1 and question 2. The consistency score for D3 is 1.4.

Table 4.3.: Questionnaire responses to D3 - The participants are not committed to the project.

Response	Q1	Q2	Q3
Strongly Disagree	17.3% (N=9)	1.9% (N=1)	30.8% (N=16)
Disagree	26.9% (N=14)	3.8% (N=2)	26.9% (N=14)
Neutral	26.9% (N=14)	5.8% (N=3)	13.5% (N=7)
Agree	15.4% (N=8)	44.2% (N=23)	17.3% (N=9)
Strongly Agree	13.5% (N=7)	44.2% (N=23)	11.5% (N=6)

D4: Lack of a plan or strategy for the project

In an attempt to learn how the students experience this dysfunction in their team, they were asked three questions.

1. Our group had a plan for what to do.
2. Not everyone in the group understood how the project was to be carried out.
3. We had good communication in the group about the strategy for the project.

The distribution of answers can be found in Table 4.4. The average score of the three questions is 3.5. There is a difference of 1.0 between question 1 and question 2. The consistency score for D4 is 1.46.

Table 4.4.: Questionnaire responses to D4 - Lack of a plan or strategy for the project.

Response	Q1	Q2	Q3
Strongly Disagree	1.9% (N=1)	17.3% (N=9)	1.9% (N=1)
Disagree	5.8% (N=3)	21.1% (N=11)	5.8% (N=3)
Neutral	13.5% (N=7)	11.5% (N=6)	15.4% (N=8)
Agree	57.7% (N=30)	36.5% (N=19)	59.6% (N=31)
Strongly Agree	21.2% (N=11)	13.5% (N=7)	17.3% (N=9)

D5: Poor leadership

Three questions were asked to learn about the students' experience of this dysfunction in their team.

1. In our group we had someone who made sure we had good progress.
2. No one in our group took clear responsibility for the organization of the project.
3. In our group, someone was given the responsibility of following up on all or parts of the project.

The distribution of answers can be found in Table 4.5. The average score of the three questions is 3.4. There is a difference of 0.8 between question 2 and question 3. The consistency score for D5 is 1.44.

Table 4.5.: Questionnaire responses to D5 - Poor leadership.

Response	Q1	Q2	Q3
Strongly Disagree	3.8% (N=2)	19.2% (N=10)	5.8% (N=3)
Disagree	5.8% (N=3)	48.1% (N=25)	36.5% (N=19)
Neutral	23.1% (N=12)	13.5% (N=7)	23.1% (N=12)
Agree	57.7% (N=30)	17.3% (N=9)	28.8% (N=15)
Strongly Agree	9.6% (N=5)	1.9% (N=1)	5.8% (N=3)

D6: Too specialized tasks

To understand the students' experience of this dysfunction in their team, they were asked the following three questions.

1. In our group, there was little variation in the work tasks for each member.
2. Each group member did not get to cover the entire curriculum through the project work.

3. We made sure that everyone got to try different tasks during the project.

The distribution of answers can be found in Table 4.6. The average score of the three questions is 3.3. There is a difference of 0.7 between question 2 and question 3. The consistency score for D6 is 1.48.

Table 4.6.: Questionnaire responses to D6 - Too specialized tasks.

Response	Q1	Q2	Q3
Strongly Disagree	11.5% (N=6)	11.5% (N=6)	3.8% (N=2)
Disagree	50% (N=26)	17.3% (N=9)	15.4% (N=8)
Neutral	15.4% (N=8)	32.7% (N=17)	13.5% (N=7)
Agree	17.3% (N=9)	26.9% (N=14)	55.8% (N=29)
Strongly Agree	5.8% (N=3)	11.5% (N=6)	11.5% (N=6)

4.1.2. Agreement within the Groups

The experience of a group with the different dysfunctions was used in the GitLab analysis in the study. Few of the groups had multiple members answer the questionnaire. It is interesting to see whether the groups that had multiple students answer the questionnaire agreed on which problems their group faced. A total of 52 students answered the questionnaire. 9 groups had 2 members answer, 2 groups had 3 members answer. For each of the groups who had multiple members answer the questionnaire the following procedure was done to see whether they agreed when identifying problems: For each of the three questions belonging to a dysfunction the difference from the highest score and the lowest score amongst the members was calculated. Then, for the dysfunction, the differences for the three questions were averaged. This was used as a measure of agreement within the group for that dysfunction. To see the general agreement in groups with multiple respondents, the mean of the agreement scores was calculated, and was as follows for the six dysfunctions:

D1 - Unfair or unevenly distributed workload: 0.90

D2 - Too little time is spent working on the project: 0.92

D3 - The participants are not committed to the project: 0.67

D4 - Lack of a plan or strategy for the project: 0.90

D5 - Poor leadership: 0.77

D6 - Too specialized tasks: 0.87

4.1.3. Students' Opinions about Using GitLab Data

At the end of the questionnaire, the students were asked a single open question. Translated from Norwegian, this is what they were asked:

> " **What do you think about letting teaching assistants analyze your GitLab data in order to provide better guidance?** GitLab stores data such as issues and commits. If you analyze this data and look at e.g. when things are pushed and how often, as well as who is assigned to various issues, etc., you can say something about collaboration and group dynamics. Knowledge of this can give teaching assistants a tool to help groups solve problems they encounter during their projects. Do you see any possible challenges with this?"

The respondents had different opinions about this. Some were positive to the idea. One respondent answered: "*I see this as a great advantage, and as a tool to increase the learning outcome by correcting the work along the way*". Another student said: "*I think this can be a good solution. This will make it possible for teaching assistants to comment if someone in the group does not contribute much, so that the burden [of handling that] does not remain on the group. At the same time, the assistant can then make suggestions for improving collaboration based on the analysis.*"

Others were more skeptical, and mentioned different challenges that would have to be handled in order for an analysis to be helpful. These challenges can be seen in Table 4.7 along with how many of the respondents brought them up in their answers.

In the next sections, findings from the students' responses will be presented in order from most frequently to least frequently mentioned. All quotes are translated from Norwegian to English.

Table 4.7.: Challenges mentioned by students in the questionnaire.

Challenge	**# Mentions**
Pair programming	17
GitLab does not Represent Reality	17
Other Parts of the Course Work is not Represented	7
Improper Use of Git	7
Problems with Issue Assignment	5
Students Misrepresenting Their Work	4
Different Skill Levels Amongst Students	3
The Feeling of Being Surveilled	3

Pair Programming

From the responses from the students, it is clear that they consider pair programming an obstacle to getting accurate data on the distribution of workload. A respondent answered the question: "*Also, we were encouraged to work using the pair programming method, where often only one person in the pair could be seen directly in GitLab. For example, in our group, there was one person who had 0 commits on GitLab, but there is another person who I feel contributed less.*" An option some of the students mentioned they had used to solve this problem was adding a co-author in the commit message. But a few of the respondents mentioned how they either had problems adding a co-author or didn't pay much attention to it: "*It was a bit tricky to get co-authors right when we pair program, so it can be wise to keep in mind that it may be missing in some places.*" To get a more correct representation of who did the work, some students, therefore, recommended getting a proper introduction to how to add someone as a co-author. However, even if

all commits that had multiple students working on it had them added as co-authors, that does not solve all problems. Another student pointed out: "*'Co-authored' is easily thrown in if someone sits and watches for a while while others do the whole job.*"

GitLab does not Represent Reality

Another fact made clear in the students' responses is that not all trust the data gathered from GitLab to accurately represent the reality of their group's work. A variation of the following statement was one of the most common comments: "*Using git as a tool to analyze group dynamics would probably give a very incorrect picture of my group.*" Some comment that the work done can be complex and that data from GitLab cannot represent the complexity in a good way. However, many of the students who felt that GitLab data could not accurately represent their group's work were still positive about using an analysis as an additional tool: "*The GitLab data should not be used as a basis for describing the collaboration and group dynamics, but perhaps [it could be used] as a small supplement.*"

Other Parts of the Course Work are not Represented

One reason why students did not think GitLab could accurately represent the work done in the groups, is because a large part of the course was focused on writing reports. One of the students phrased their concerns thus: "*In addition to delivering a product, PU[Software Engineering] is about delivering various assignments that are often time-consuming. In our group, it was therefore common for someone to start on a report, make a presentation for review, etc. while others continued programming. This made it possible to reach the finish line with all the deliveries and was a dynamic the group agreed on. As a consequence of this, if you only look at GitLab, you will make assumptions that some in the group contributed less than others, even if this is not the case when you look at the course as a whole. I'm afraid that the teaching assistant would misinterpret the data and choose to focus on the wrong things.*"

Improper Use of Git

Another concern amongst the students was that since git is not part of the curriculum of the course, some students would not have the foundational understanding of the tool to use it in a good way. One student shared these thoughts: "*The idea behind using GitLab statistics and history is good, and can probably help provide valuable insight into work style, the quantity of work, and distribution. However, this presupposes that everyone involved uses the git tool for all its worth, with frequent commits with good descriptions, links to issues, and MRs[Merge Request]. Our group experienced that some did well, while others had never cloned a repo without using a step-by-step guide. PU[Software Engineering] has students with very different backgrounds and experiences and it will probably be difficult to do valuable research with GitLab as the only source as the tool is used very differently between students and groups.*" Other students shared their difficulties using Git in their team. "*It could have worked well, but then we would have needed an introduction to Git. In our group, a minority had any knowledge of it, and instead of everyone pushing and pulling and using Git, we sent code to one member who then pushed from his [computer].*"

Problems with Issue Assignment

A problem when using issues in GitLab as a measure of who worked on what was revealed by a few students: "*At [the department's instance of GitLab], it is not possible to assign two or more people to an issue, which means that it may seem as if someone has been less involved in the project because they have worked with someone else.*" Others commented that assignments to issues and pull requests were often forgotten.

Students Misrepresenting Their Work

Another thing that was mentioned a few times was that students knowing their data would be analyzed would likely adapt their Git behavior accordingly. One student phrased that like this: "*I do not think it will help much, as it will primarily increase the workload in the course, which is already too large. And because of this, people will only make more*

and more complicated workarounds to avoid the problems." Another concern was that the knowledge of their GitLab statistics being analyzed could lead to members prioritizing quantity of commits over quality, as explained by a student here: "*A possible challenge is that there will be a great focus on how many pushes / commits each person makes, which could then take the focus away from good pushes/commits (where good code is written and a certain amount of work is done) to pushing/committing often to "showcase" that they work a lot.*"

Different Skill Levels Amongst Students

A few students had concerns about how different skill levels would impact the analyses. They pointed out that students who already knew the technology stack they were going to use could work an equal amount to someone new to the stack, but would likely end up with more code lines, commits, and issues assigned to them.

The Feeling of Being Surveilled

The last problem revealed in the questionnaire was that a few students found the idea of their GitLab data being analyzed to be surveilling. One of the students expressed his opinion: "*I think that seems very monitoring. My perception of the point of the project work is that we should learn Scrum, not be monitored to see who pushes the most.*" Others were okay with the group getting the results of an analysis, but found it uncomfortable for the teaching assistants to get the information.

4.2. Interviews with Teaching Assistants

Interviews with teaching assistants were done in the time frame February 28 to March 29 2022. A total of five interviews were conducted, with each teaching assistant being supervisor for three student groups. In the following sections the findings from the interviews will be presented.

4.2.1. Identification of Problems in Groups

This section will focus on the six dysfunctions defined in Section 3.1.3. For each dysfunction we will see how often it seems to be a problem for the groups, and for the cases where the teaching assistant identified a problem, we will see how they observed it in the group. In the presentation of the findings, the results will be presented after the traffic light model defined in Section 3.2.2. An overview of how the teaching assistants classified the 15 groups for each dysfunction can be seen in Table 4.8

Table 4.8.: Problem identification by teaching assistants.

Group	D1	D2	D3	D4	D5	D6
A	Green	Green	Green	Green	Green	Green
B	Yellow	Green	Green	Yellow	Yellow	Yellow
C	Green	Green	Green	Green	Green	Green
D	Yellow	Green	Yellow	Green	Green	Yellow
E	Green	Green	Green	Green	Green	Yellow
F	Yellow	Green	Green	Green	Green	Red
G	Green	Green	Green	Green	Green	Green
H	Yellow	Green	Green	Green	Green	Yellow
I	Yellow	Red	Yellow	Green	Yellow	Green
J	Yellow	Green/Yellow	Green	Green	Green	Yellow
K	Yellow/Red	Yellow	Yellow/Red	Yellow	Yellow	Green
L	Green	Green	Green	Green	Green	Green
M	Green	Green	Green	Green	Green	Green
N	Yellow	Green	Yellow	Red	Yellow	Yellow
O	Yellow	Green	Yellow	Yellow	Yellow	Yellow

D1 - Unfair or unevenly distributed workload

With 6 / 15 groups being classified as green, "Unfair or unevenly distributed workload" was the dysfunction most often identified in the groups by the teaching assistants. Eight of the groups were considered yellow, and one was considered between yellow and red. When prompted to tell about the challenge in the group that fell between yellow and red, the teaching assistant explained that the group had complained about a large workload and that some members seemed to have a lack of motivation. The teaching assistant pointed out that even though the students spend most of their time in the course on programming, that part does not matter much in the final evaluation.

Two of the teaching assistants identified an uneven distribution of the workload by uneven participation in discussions. One of them said: "*When you are asking a question it is very clear who answers. And that is whether it is something technical or process-related or about evaluation.*" Another teaching assistant experienced varying attendance. A couple of the members would sleep past the meetings, and the members expressed that that also happened when they were supposed to work. That group also had one particularly motivated member and a few members that did not know how to proceed with the project.

Lastly, there were a couple of groups where the students expressed no issues, but the teaching assistant still classified them as yellow. For one of them, the teaching assistant saw that one of the members was a strong resource for the group that they relied on heavily. The group utilized pair programming, but the teaching assistant would have liked to see them switch the pairs more often so that more could learn from the most resourceful.

Another teaching assistant said that the group never mentioned anything about it when asked about uneven distribution of work during their meetings, but continues: "*But I do get the impression that some of them make more of an effort and have more will than others. Or are more engaged.*" The assistant goes on to say that the group works together in 4-hour work sessions and that when looking at their registered hours of work they all have the same number of hours. Still, the teaching assistant decided that they

were yellow, saying: "*Because some may have a better ability to gain knowledge or are willing to spend more time [on the project].*"

D2 - Too little time is spent working on the project

This dysfunction is the one that had most groups classified green with 12/15. One group was classified as red, one yellow, and one between green and yellow. For most of the groups, the teaching assistants said they spent more time than what they were expected to use on the course. A couple of the groups had problems with time estimates within the project but still spent as much time as could be expected working. One teaching assistant said this: "*I would say that they used enough time, but that they might have miscalculated how long it would take to do certain things like setting up the foundation [of the project.]*"

For the group that was considered red, the teaching assistant said: "*I have experienced it is a problem that they spend too little time. For example, they started working on the presentation the day before a review.*" This teaching assistant also said that he had brought it up in their meetings.

For the group that was classified as yellow, the teaching assistant identified the problem through their mid-term review. After having looked at their registered hours he saw that the difference between the member who spent the least time and the most time working on the project was well over 10 hours, which the teaching assistant considered a significant difference in a course where they were expected to work 12 hours a week.

For the group that was placed between green and yellow, the teaching assistant said that they had not spent less time than what was expected. Still, he placed them between green and yellow. No further explanation was given.

D3 - The participants are not committed to the project

For D3 4/15 groups were classified as yellow, one was classified as between yellow and red, and the rest were green.

For two of the groups that were classified as yellow, one of the reasons they were

considered less committed to the project was because they did not know or care about the assessment criteria in the course. One teaching assistant said: "*They have not familiarized themselves with the assessment criteria, so they had to ask me about information that is readily available on Blackboard.*" Another teaching assistant said that the members of a group seemed to not care about getting an A. They were putting in the effort it would take to give them a B or a C.

For another of the groups that were classified as yellow, the teaching assistant talked about when he took part in a user test for them. Whenever he pointed out something that could be changed they were eager to improve. He contrasted this to their meetings where only two of the members participated in discussions. Because he could not gauge the commitment of the other members the teaching assistant chose to put the group yellow.

For the group that was said to be between yellow and red, the teaching assistant focused his answer on one member and said: "*We had a meeting yesterday, and there was one person. You could see from their body language that it seemed like they did not want to be there if you understand? You can hear when they talk about what they have done or ask questions. It just showed. Yes, they seemed tired of it. It's just how they are.*"

D4 - Lack of a plan or strategy for the project

For this dysfunction, 3/15 groups were classified as yellow, one as red, and the rest were green. For the groups that were considered green, they were described as well structured, thinking far ahead, being good at communicating, and using their resources by asking questions as soon as they had any. One of the teaching assistants said this about one of the green groups: "*On this [problem] they are actually green. Even if they are not accomplishing it, they have a pretty clear plan. They are good at that.*"

For the teams that were considered yellow, one of them did not have problems with communication within the team. But the teaching assistant felt like he could have helped them make a clearer plan, and more actively worked towards using more of the curriculum. Another teaching assistant had this to say about his yellow group: "*Yes, that would be*

yellow. There is some control, but they may not have a goal they are working so clearly towards. There may be more individuals who take some initiative on their own, and it is not a streamlined production" About the one team that was considered red, the teaching assistant said this when asked to explain how he could see the problem in the group: "*The meetings. There is a lot of back and forth. It is quite clear that they have made technical choices that not everyone has been included in making.*"

D5 - Poor leadership

For this dysfunction, no group was observed to have a problem that warranted them being classified as a red team. 5/15 groups were considered yellow. The remaining were considered green. An interesting finding here is that amongst the teams that were classified as green there were a variety of leadership compositions. There were groups where there was a clear leader, one group had two clear leaders who had taken responsibility for different parts of the project, and one group was described as having a flat structure where all members were pushing them in the right direction. There was no mention of any team that had explicitly chosen a leader, but that may still have happened for some. One group that had a single leader that was likely implicitly chosen was described by their teaching assistant: "*I do not know if they have a leader. But I have an impression of who the leader is. Because it is the person who talks the most. And if I ask questions, it is often the case that the others in the group look back at that person or mention the person's name to see if they have any input.*"

For the groups that were considered yellow, it seemed like they had not discussed leadership. One group was described like this: "*They are having a hard time structuring things. There is no single person that steps forward as a leader. They throw out suggestions.*" Another teaching assistant mentioned the group meeting unprepared, and the members not being aware of where they were at in the project.

D6 - Too specialized tasks

For the final dysfunction, 7/15 groups were classified as green. One was red, and the rest were yellow. For one of the green groups, the teaching assistant said they had divided the work to some degree, with some students having more of their focus on programming, and others doing more of the writing. However, the teaching assistant said it did not seem to be a problem in the group because they worked so closely as a team.

For the red group, the teaching assistant described the problem like this: "*Yes, it is a problem. One of the first times I met them, the person I assume knew a lot was not present. So the others in the group, even though they showed up for their work session, could not work properly, because they could not extract data from the database. So it was a clear problem, where the only thing they could do was work on frontend and design. And I know there are many in the group that says they know either backend or frontend, but not both. It might be connected to how they do pair programming. Compared to other groups it was very clear that this problem existed here. Also because they have commented themselves that they know there are differences and that some might not have the knowledge of all parts of the project that they should have had.*"

For the yellow groups, there were a couple of reasons mentioned why they were classified that way. For a few of the teams, the teaching assistants said that based on how they answered questions they assumed some did more of the programming and others more of the writing.

One of the groups classified as yellow had a pretty clear division of work within the project, where one person was skilled at backend development and another took the responsibility of designing, and they had problems communicating and making trade-offs. The group was described thus: "*It definitely felt like some were more in control of design, at least one in the group was skilled at backend and took care of that. And then I think they divided it so that the remaining do most frontend work.*" This teaching assistant also said that the ideal situation would be one where everyone was equally involved in all processes.

Other Problems Identified

Before going through the six defined dysfunctions the teaching assistants were asked to talk about the problems each group faced. Three challenges were mentioned that fall outside the scopes of the defined dysfunctions previously presented. Some groups had problems with the technical aspects of development. One group had differing expectations that lead to some internal conflicts. The teaching assistant explained what they had observed: "*So a computer science student had the expectation that an industrial economy student would fully participate in the programming, while the industrial economy student thought he could write [reports] and the computer science student could write code.*" The last challenge that was mentioned a few times for different groups was members having an uneven level of knowledge. One group was described this way: "*Some are very competent and do things very quickly and without including the others.*"

4.2.2. Teaching Assistant's Perspective

At the end of the interview, a few questions were asked regarding how the teaching assistants would prefer to have an analysis of GitLab data presented to them, and if they could think of any problems using such an analysis as a tool for supervision. The findings are presented in this section.

How to Present the Data

The teaching assistants were presented with a few options for how data from GitLab could be presented to them. The options were: using the traffic light model, having a metric be presented as a number on a line between significant values, or as numbers in a table. All but one of the teaching assistants would prefer the data being presented by the traffic light model. They thought the traffic light model was intuitive, easy to understand, and required less effort. One teaching assistant said this: "*Yes, I do think that as a teaching assistant, color code actually works quite well. It is easy to get acquainted with and it is quick, so you also have time to spend on the students.*"

When asked if they had any other ideas for presentations that would fit better than any of the mentioned, one teaching assistant would prefer something that could cluster and categorize the problems.

One thing the teaching assistants considered important to think about when presenting the data to them was that it should be presented in an organized way and be easy to interpret. The data should focus on what the groups can improve. It was also mentioned that teaching assistants need to use it as a tool to further understand the groups and not as an exclusive way of telling what the group is struggling with.

Possible Problems

The teaching assistants mentioned many of the same possible problems as the students did in the questionnaire. Some mentioned how pair programming could make the data less representative. Another thing mentioned was that if they knew that their GitLab data was analyzed, they would change their use of GitLab. An assistant said this: "*But when [I had the subject last year] we distributed the issues and such so that it would look nice in GitLab without us actually doing it like that in practice.*"

Someone else mentioned that the results of the analyses might be impacted by user error in GitLab since most of the students do not have experience using the tool.

Lastly, there was a concern about privacy and the students feeling surveilled. "*It's important that they do not feel monitored, I think. Because then they will just add nonsense on GitLab that is not really true.*" The teaching assistant that said this felt that the student should be informed about the process beforehand with it being made clear that it was the supervisor who would use the data to help them and not the evaluator. They should be informed that their data from GitLab would not be used to evaluate them.

4.3. Analysis of GitLab Data

In this section, the results of the Pearson correlation analysis of the data sets made for each of the metrics introduced in Section 3.3.3 will be presented. The statistically significant results can be seen in Table 4.10, and will also be explained in Sections 4.3.1 and 4.3.2. The dysfunctions will only be mentioned by their ID, but the names are repeated in Table 4.9, and the definitions can be found in Section 3.1.3.

Table 4.9.: ID's and names of the six dysfunctions focused upon in the analyses.

ID	Name of Dysfunction
D1	Unfair or unevenly distributed workload
D2	Too little time is spent working on the project
D3	The participants are not committed to the project
D4	Lack of a plan or strategy for the project
D5	Poor leadership
D6	Too specialized tasks

4.3.1. Commits

Amongst the metrics created and described in Section 3.3.3, and that were created from commit data, four of them gave statistically significant results for Pearson's correlation analyses. None of the statistically significant results had any outliers removed.

For the metric "Portion of Work Done Between 8:00 and 16:00", statistically significant results were found for D1, D2, and D3. The data set for the metric is normally distributed, with a Shapiro-Wilk significance value 0.448 (>.05). The scatter plots for the metric and dysfunctions can be seen in Figures 4.1, 4.2 and 4.3. For D1 there is a moderate positive correlation, $r(36) = 0.419, p < 0.05$. D1 is normally distributed as determined by Shapiro-Wilk's test ($p > .05$). D2 is *not* normally distributed. With that in mind, there is a moderate positive correlation, $r(36) = 0.402, p < 0.05$. D3 is also *not* normally distributed. There is a strong positive correlation between the metric and D3, $r(36) =$

Table 4.10.: Pearson correlation coefficients and significance value for each statistically significant finding.

Metric	N = 36	**D1**	**D2**	**D3**	**D4**
Unique dates for Issue Creation	Pearson Correlation	-.437	-	-.448	-.395
	Sig. (2-tailed)	.008	-	.006	.017
Portion of work during day	Pearson Correlation	.419	.402	.543	-
	Sig. (2-tailed)	.011	.015	<.001	-
Portion of work during evening	Pearson Correlation	-.381	-.355	-.541	-
	Sig. (2-tailed)	.022	.034	<.001	-
Tops >7% (Code lines)	Pearson Correlation	.562	.524	.571	.519
	Sig. (2-tailed)	<.001	.001	<.001	.001
Tops >1% (Commits)	Pearson Correlation	-	-	-	-.352
	Sig. (2-tailed)	-	-	-	.035

$0.543, p < 0.01$

For the metric "Portion of Work Done Between 16:00 and 00:00" there were statistically significant findings for three of the dysfunction. Also for this metric, that was D1-D3. The data set for the metric was normally distributed, with a Shapiro-Wilk significance value 0.327. The scatter plots for the metric and dysfunctions can be seen in Figures 4.4, 4.5 and 4.6. D1 has a moderate negative correlation with the metric, $r(36) = -0.381, p < 0.05$. For D2 there is also a moderate negative correlation, $r(36) = -0.355, p < 0.05$. D3 has a strong negative correlation to the metric with $r(36) = -0.541, p < 0.01$.

Looking at "Number of Days Where Over 7% of Total Code Lines were Committed", the metric is normally distributed with a Shapiro-Wilk significance value 0.12. The scatter plots for the metric and dysfunctions can be seen in Figures 4.7, 4.8, 4.9 and 4.10. For this metric, there were findings for the dysfunctions D1-D4. All of the dysfunctions have strong positive correlations. D1 has the following correlation for the metric, $r(36) = 0.562, p < 0.01$. For D2 the correlation is $r(36) = 0.524, p < 0.01$. D3 has a

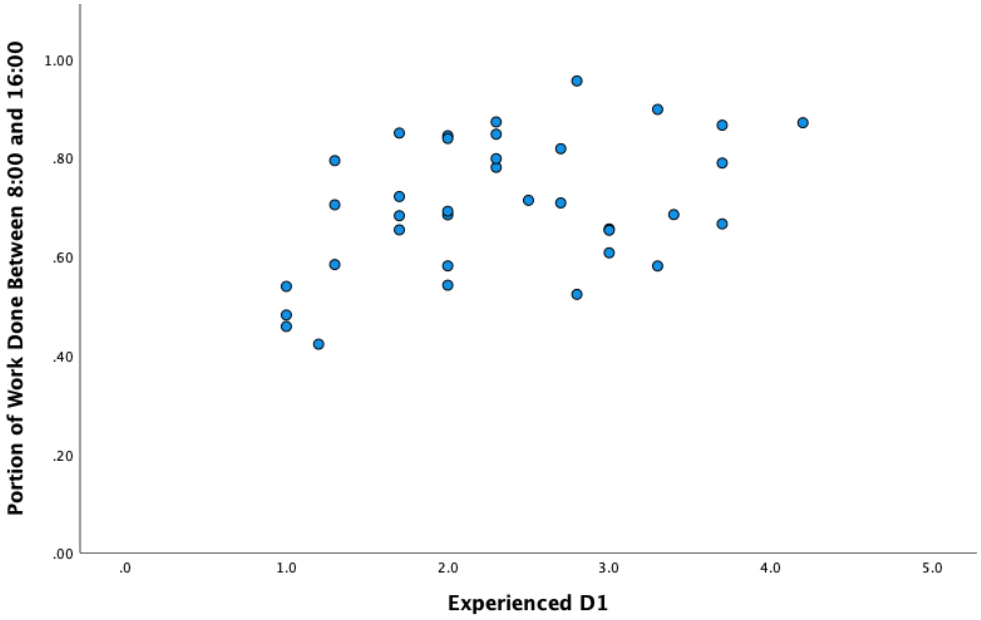


Figure 4.1.: Scatter plot: Portion of work done between 8:00 and 16:00 (D1).

correlation of $r(36) = 0.571, p < 0.01$. And D4 has a correlation of $r(36) = 0.519, p < 0.01$. D4 is normally distributed, as reported in Section 3.3.3.

The last metric with results for commit data is ´´Number of Days With Over 1% of Total Commits", which has a moderate negative correlation of $r(36) = -0.352, p < 0.05$ with D4. The metric is normally distributed with a Shapiro-Wilk significance value 0.448. The scatterplot for the metric and dysfunction can be seen in Figure 4.11.

Four of the metrics had data sets that were not normally distributed. These were: "Portion of Work Done Between 00:00 and 8:00", "Portion of Work Done During Weekends", "Number of Days With Over 15% of Total Commits", and "Percentage of Work Done (Code Lines) the last 5 Days Before End of Project". Because of the lack of normality, these metrics were not further examined. For the remainder of the metrics made from commit data, there were no significant findings.

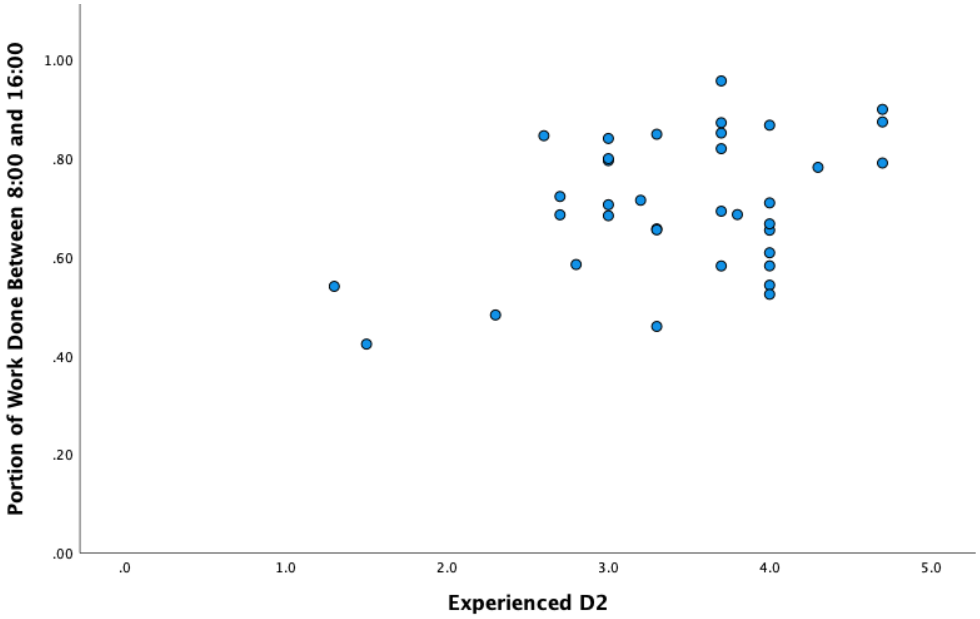


Figure 4.2.: Scatter plot: Portion of work done between 8:00 and 16:00 (D2).

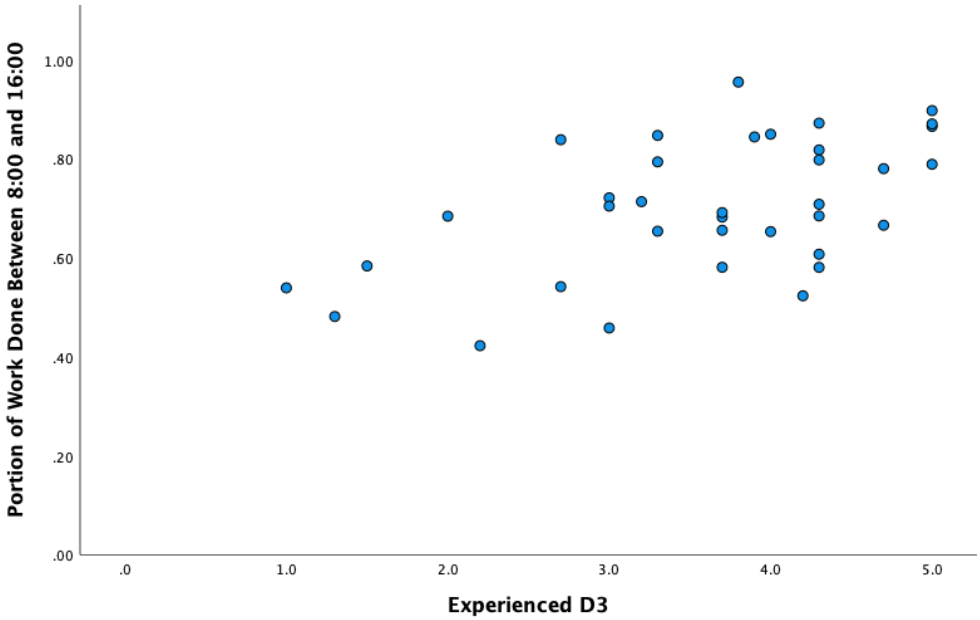


Figure 4.3.: Scatter plot: Portion of work done between 8:00 and 16:00 (D3).

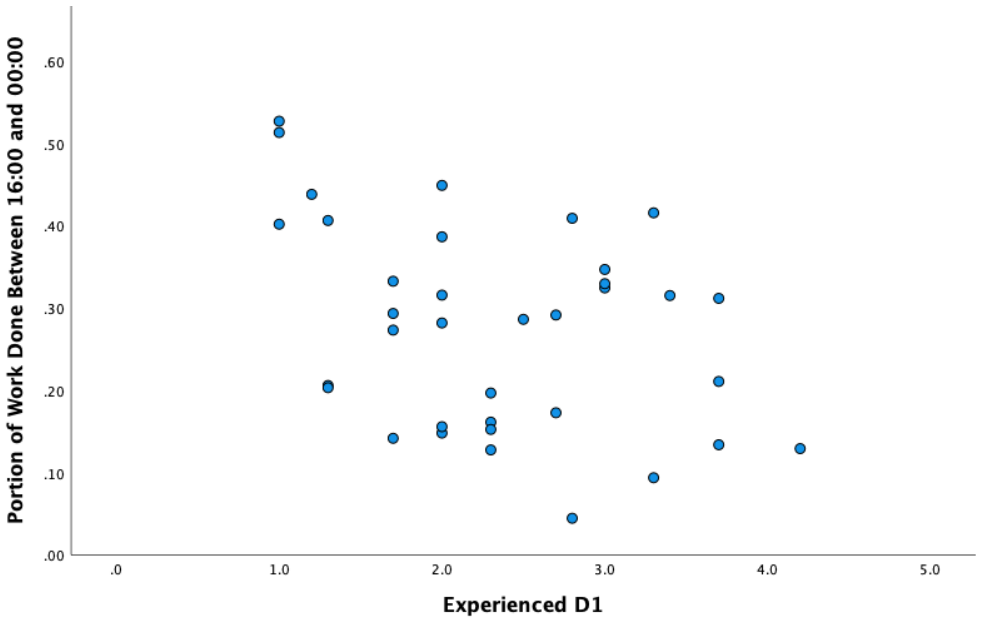


Figure 4.4.: Scatter plot: Portion of work done between 16:00 and 00:00 (D1).

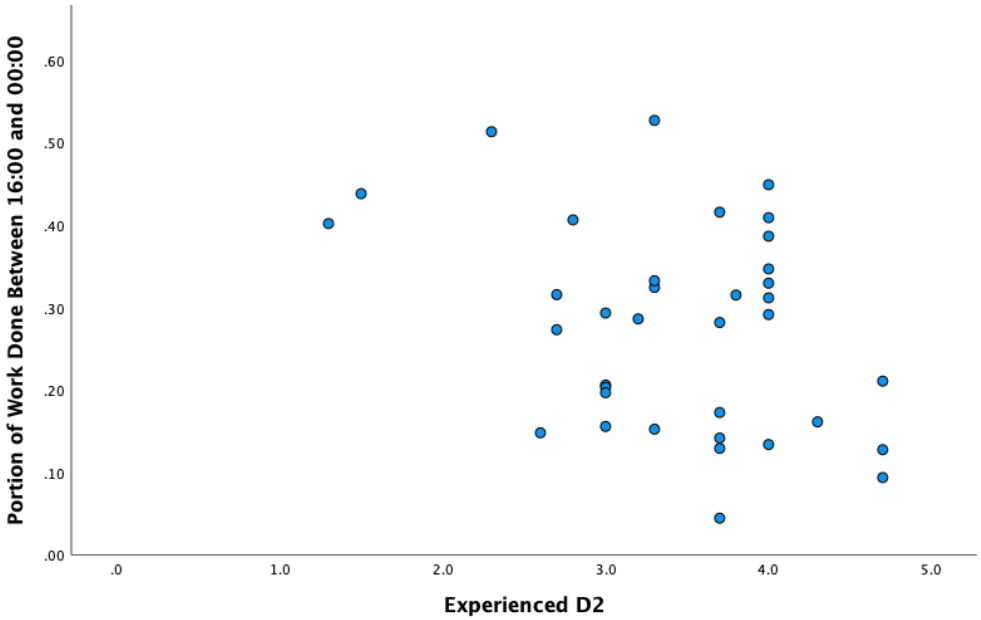


Figure 4.5.: Scatter plot: Portion of work done between 16:00 and 00:00 (D2).

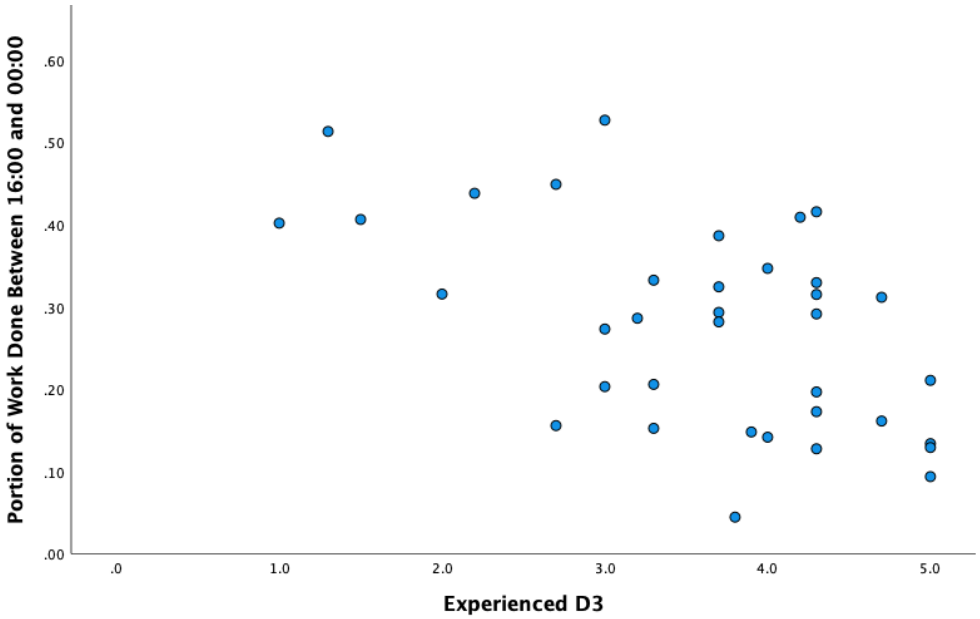


Figure 4.6.: Scatter plot: Portion of work done between 16:00 and 00:00 (D3).

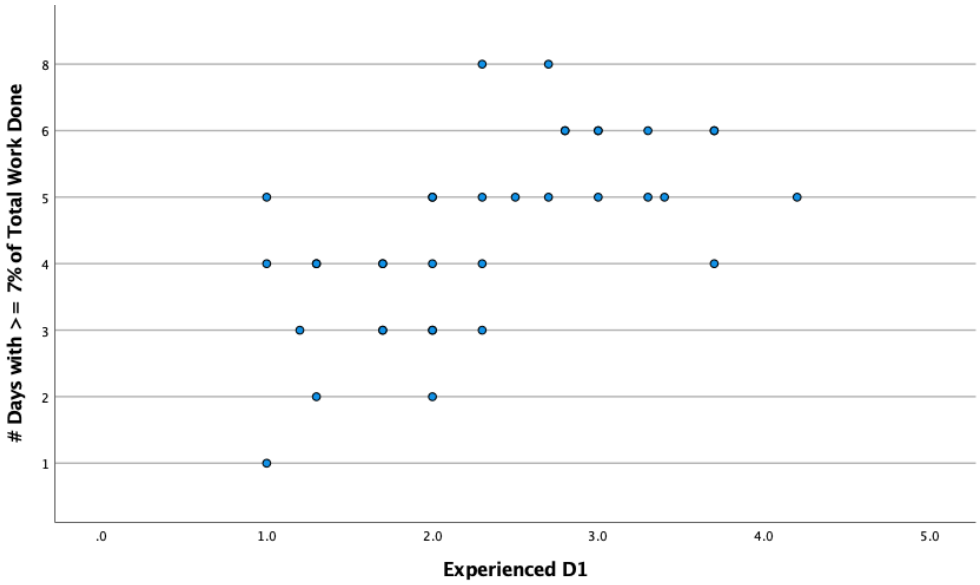


Figure 4.7.: Scatter plot: Number of days where over 7% of total code lines were committed (D1).

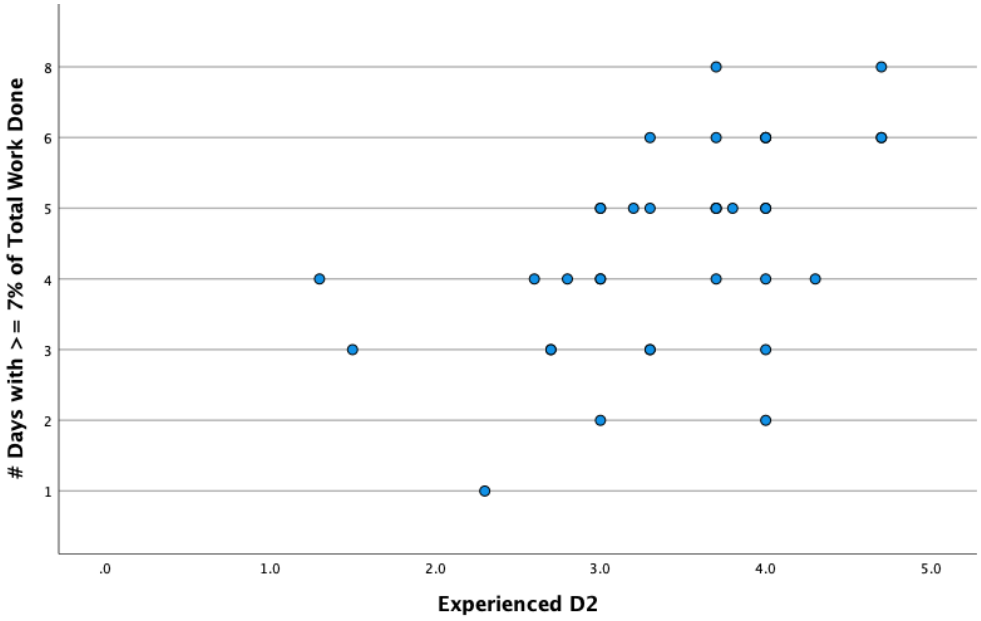


Figure 4.8.: Scatter plot: Number of days where over 7% of total code lines were committed (D2).

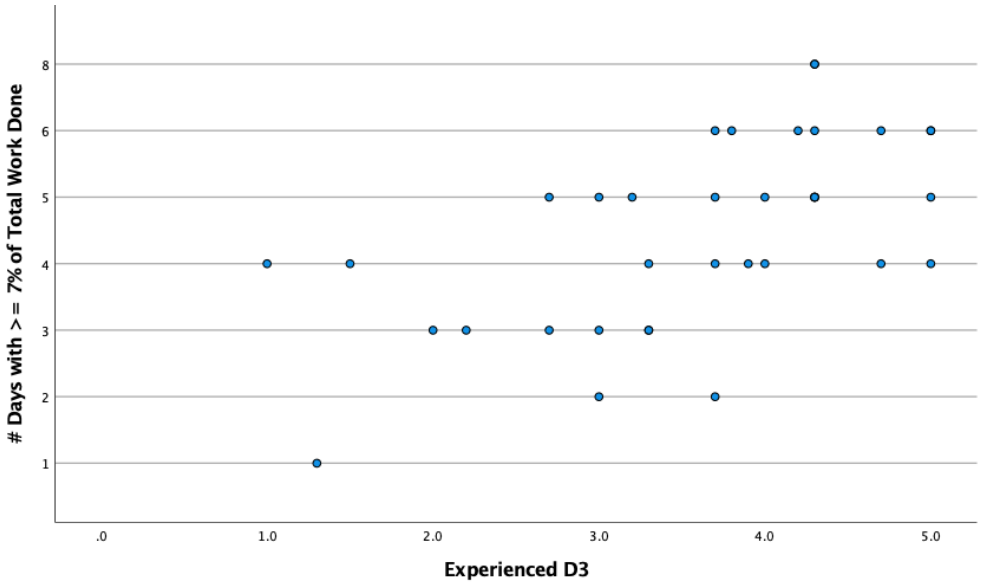


Figure 4.9.: Scatter plot: Number of days where over 7% of total code lines were committed (D3).

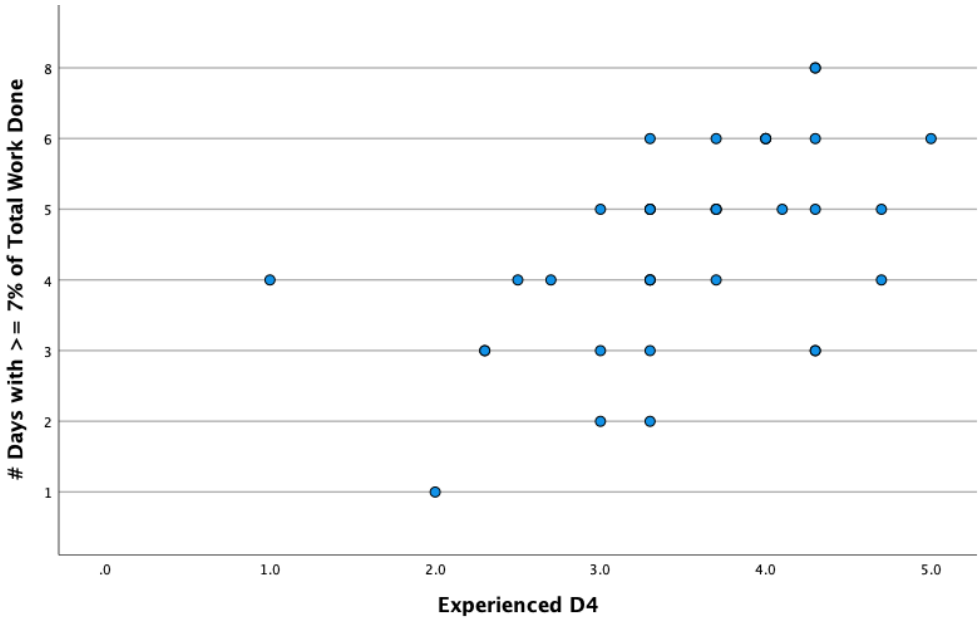


Figure 4.10.: Scatter plot: Number of days where over 7% of total code lines were committed (D4).

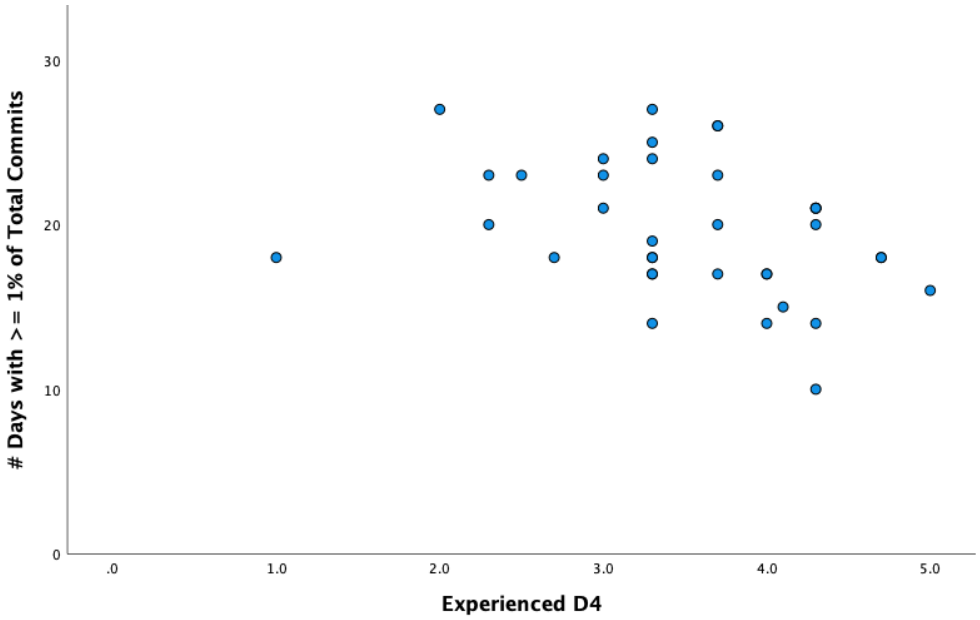


Figure 4.11.: Scatter plot: Number of days with over 1% of total commits (D4).

4.3.2. Issues

For the three metrics that were constructed using issue data, one of them had statistically significant findings, "Number of Unique Days of Issue Creation". The two remaining had no significant findings. For the metric, the data set is normally distributed with a Shapiro-Wilk significance value 0.114. Significant results were found for D1, D3, and D4. For D1 there is a moderate negative correlation, $r(36) = -0.437, p < 0.01$. D3 has a moderate negative correlation to the metric with $r(36) = -0.448, p < 0.01$. Lastly, there is also a moderate negative correlation between D4 and the metric of $r(36) = -0.395, p < 0.05$.

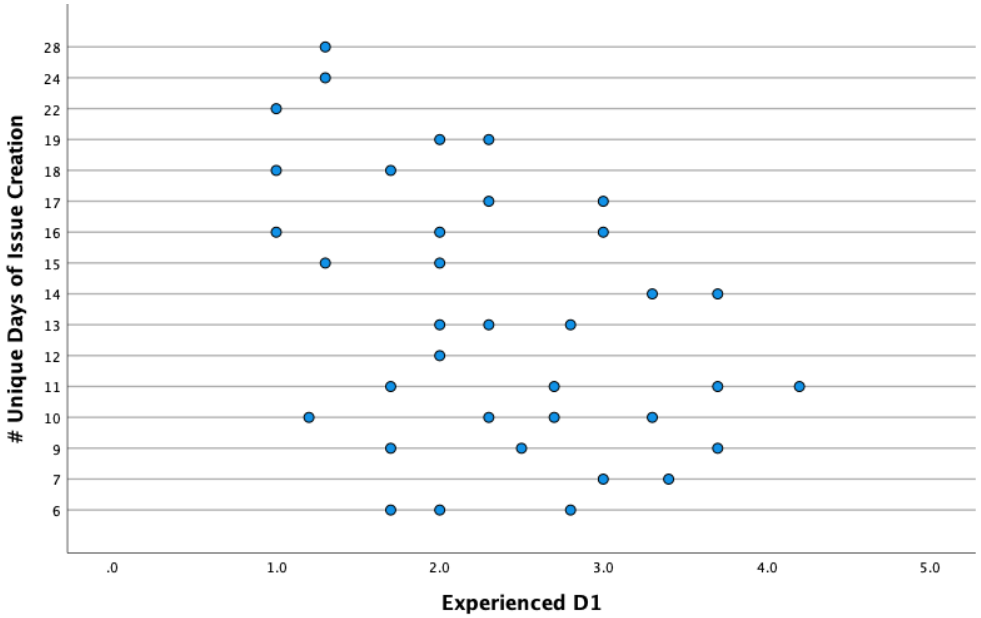


Figure 4.12.: Scatter plot: Number of unique days of issue creation (D1).

There was no significant finding for any metric, neither based on commit data nor issue data, with D5 - Poor leadership.

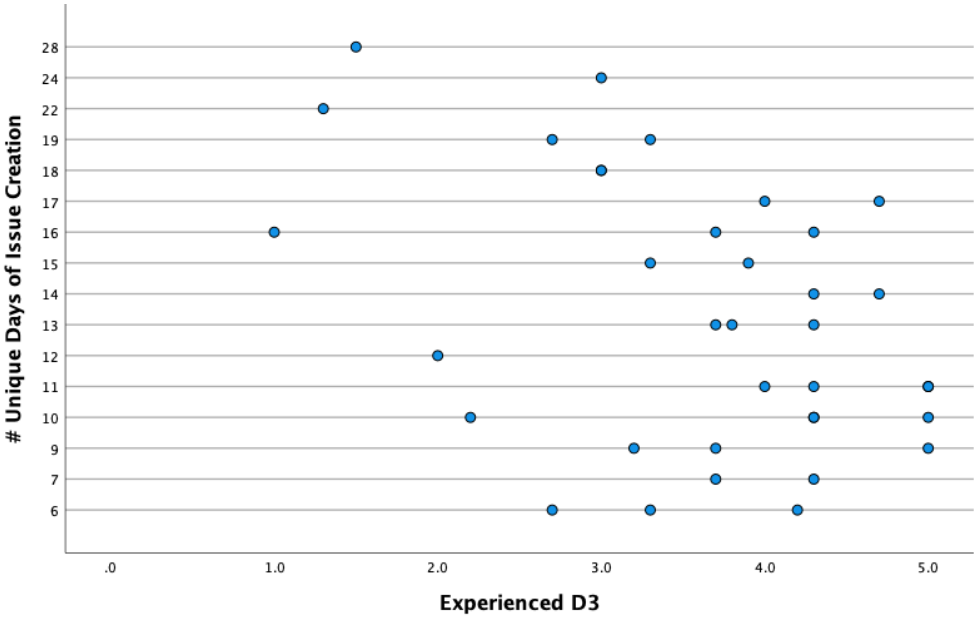


Figure 4.13.: Scatter plot: Number of unique days of issue creation (D3).

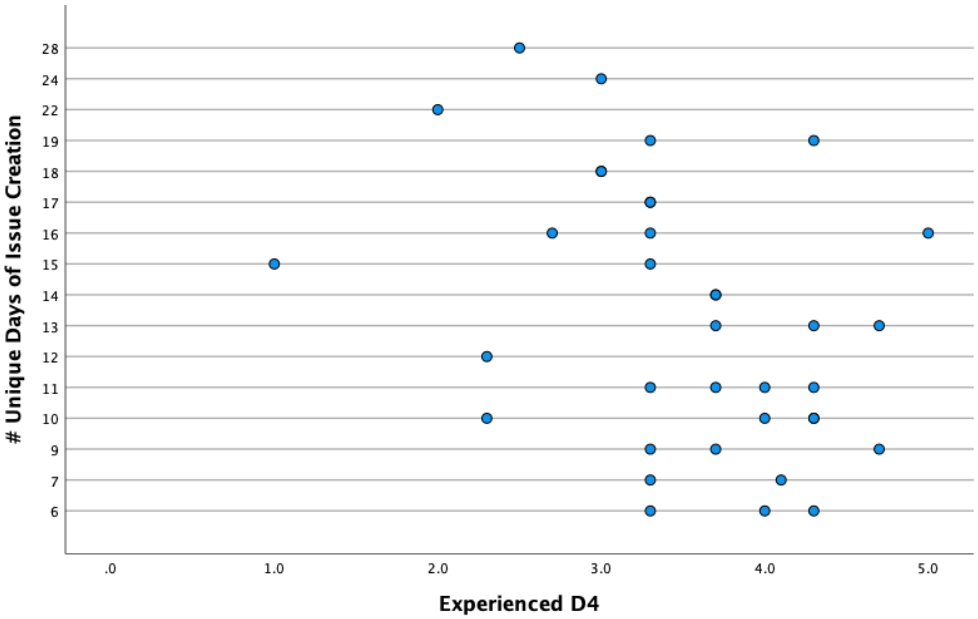


Figure 4.14.: Scatter plot: Number of unique days of issue creation (D4).

5. Discussion

In this chapter, results from the analysis of different data sources will be compared to each other and sources from the literature. The section presents a discussion of the implications of the findings in this study to courses that follow similar structures as the chosen case.

5.1. Challenges

This section will go into different challenges when implementing software repository mining in a project-based software engineering course. It will discuss problems identified by students and teaching assistants, as well as limitations when identifying dysfunctional teams using the chosen tools.

5.1.1. Pair programming

Throughout the study, pair programming has been mentioned as a challenge multiple times. In Section 4.1.3, we see that it tied for the most mentioned challenge by the students in the questionnaire. The teaching assistants also mentioned in Section 4.2.2 how pair programming could make GitLab data less accurate to reality. While constructing GitLab metrics for analysis, the limitation that pair programming posed, became clear. In an attempt to see if both members in a pair could get credited with the work they collaborated on, some groups' commit data sets were examined to see how they noted pair programming in their commit messages. Some groups had routines for documenting this, while others lacked such. The attempt to identify commits that were collaborations and

the members that participated was unsuccessful. The two metrics that took into account the work done by individual members were "Standard Deviation - Portion of Commits per Member" and "Standard Deviation - Portion of Lines of Code per Member". Looking at the results of the Pearson correlation analysis, none of these showed any significant correlation to any of the dysfunctions, even though the metrics were designed in the hope of revealing D1 - Unfair or unevenly distributed workload. The reason why these metrics were unsuccessful in identifying dysfunctions could likely be because pair programming lead to an impression of workload distribution that did not fit well with reality. In any software engineering course attempting to reveal unfair or unevenly distributed workload by examining the number of commits or code lines per member, it is advisable to require a structure for documenting pair programming that lends itself to analysis. Requiring the teams to add "Co-authored by [student]" is an option. However, as mentioned by a student previously, self-reporting pair programming in commit messages can lead to some students mostly watching others work but still getting much of the credit.

Considering the difficulty of determining who did the actual work in a commit, it is also an option not to focus on individual work, but rather focus on the analysis of the group's work as a whole. None of the significant findings in Section 4.3 came from data on individual members' contributions. However, if a metric can be derived that effectively measures contribution, the likelihood of identifying D1 is higher, and, looking at the results of the questionnaire in Section 4.1.1, unfair or uneven distribution of workload is the most common of the six dysfunctions of this study.

5.1.2. GitLab does not Represent Reality

Some students are skeptical of the accuracy of the GitLab data. From the discussion over, that is not an unreasonable concern. One student told about a previous experience he had that made him think that GitLab statistics about contribution give a false picture of reality: "*I experienced this in [a previous course]. The statistics measured the group's most competent and active member to be the least contributing, which was absurd.*" Looking at the metrics that were tested in this study, the vast majority of them found no correlation.

Focusing on the few that gave significant results, it may still be possible to reasonably use predictions from GitLab data as an additional tool for teaching assistants when guiding student teams. But even the strongest correlations identified in this study were far from complete correlations. It is important that supervisors that want to use predictions from GitLab analyses as an aid, remember that they can only be supplements. They should not be the only observations made about the teams. Regular meeting points for feedback are recommended in literature [27] [23] and should still be prioritized.

5.1.3. Other Parts of the Course Work are not Represented

As was pointed out by the students in the questionnaire, the course, that makes up the case for this study, does not revolve solely around software development. Most of the students' grades are determined by other deliverables. A student explained how this shaped the focus of the groups' work: "*Personally, I experienced that many groups did the minimum expected in order to get a good grade in the subject, and that was to place the main emphasis on the reports.*" Looking at the deliverables presented in Section 3.1.2, the only times the actual software product was evaluated was during the sprint reviews, with each review counting 5% of the final grade. A teaching assistant had this to say: "*I find it weird that you are evaluated on the other work when you use most of your time working on the code, and that does not really matter.*" Multiple students explained that they gladly delegated work within the team, so that some students focused more on writing reports and others more on coding. Still, if the members of the group contributed equally, though on different parts, that should not be identified as an uneven or unfair distribution of work. But because only the software product is counted in a GitLab analysis, it would necessarily mean that the students who focused more on reports would be attributed a smaller contribution than they deserved.

5.1.4. Improper Use of Git

One student summarized a problem with the use of GitLab data for analysis by focusing on the different usage of the tool by students in the course: "*[The course] has students*

with very different backgrounds and experiences attending, and it will probably be difficult to do valuable research with GitLab as the only source since the tool is used very differently between students and groups." The students were not the only ones to see this problem. Teaching assistants were also worried about user errors in GitLab. Inexperience was also here pointed out as a reason for this. A student suggested that learning Git should be an essential part of previous courses. At NTNU, many students that attend the previous IT courses have those as their only experience with programming, and they might not need to learn about Git. The students should however have an understanding of how to use the tool properly during, or at the end of, TDT4140, if we want to reduce the limitations introduced by improper use of Git. One of the students was skeptical about having set routines for GitLab that all students had to follow, saying: "*But then you will in a way "force" the group to follow certain git routines, which is a bit contrary to the intention that the group should find out for themselves what is good.*" Whether to teach the students a set of git routines or whether they should independently figure out which routines they want to implement in their teams and for their project, should be decided according to the learning objectives for the course. The trade-off is between: Independently and by experience learning about best practices of version control systems; or having a set routine taught to the students so that their GitLab data can be more accurately analyzed to describe the group dynamics within the team.

5.1.5. Problems with Issue Assignment

In the responses to the open question in the questionnaire, it was revealed that the department's instance of GitLab does not allow for multiple assignees. This leads to the same problems for the analysis of issues as pair programming does for commit data analysis. During the GitLab analysis, in the process of deriving metrics, the number of issues assigned to each member was explored. Comparing the issue assignments with the member contribution in commits showed an inconsistency that made me choose not to focus on the distribution of work through issue assignments. If someone intends to investigate issue distribution as a way of revealing workload distribution, they should

make sure that members can be assigned to all the issues they work on, even if they work on the same issue as someone else. If it cannot be done through assignments in the VCS, there should be a procedure for it in the issue description. As seen in Section 4.3.2, issue data can reveal dysfunctions even if the metrics used are independent of assigned members.

5.1.6. Different Skill Levels Amongst Students

Different skill levels among members can lead to problems within the teams, as identified by the teaching assistants in this study. Some groups in the study experienced an uneven distribution of work due to some being more knowledgeable than others, or they had some members making decisions for the project without involving other students. In cases such as this course, where students from different programs are put in teams together, the difference in skills and experience should be considered. The difference in skills was not a main focus of this study, but since it was brought up by teaching assistants and students as a team dysfunction, further research should be done on how a difference in competency impacts collaborative learning in project-based software engineering courses, and whether a difference in skills can be identified through the analysis of software repository data.

5.1.7. The Feeling of Being Surveilled

An expected finding in this study would be for students to feel surveilled if they knew their GitLab data was being analyzed. There were not many of the students reporting this as a problem. The ones who did, shared their discomfort at the idea of individuals being singled out and inspected during meetings. It is advisable to consider these students when instructing teaching assistants on how to give group feedback. However, the majority of the students did not mind having their software repository data inspected, and they saw many possible benefits. A teaching assistant pointed to a possible connection between the students knowing how their work is being monitored and students then changing their Git behavior to give the impression of good work habits or well-functioning group dynamics rather than them actually changing their work behavior.

According to Brown [26], it is important to make an environment where students feel free, to be honest about the problems they face in their work. If they think they are always being assessed, they will not ask for help. It is therefore important to explain to the students that analysis would only be used for feedback by their supervisor and that the evaluator that decides their grade will not be given this insight.

With the consequences that may come from the students being aware that their software repository data is being inspected, some may find it a good idea not to inform them. That is, however, not an ethically sound decision. In Gold and Krinke's article from 2021 [58], the ethics of mining VCS is discussed. Informed consent is there considered necessary for the ethical mining of software repositories. As described in Section 3.2.3, I was not permitted to work on a non-anonymized data set in this project for that reason. This leaves educators with a choice between working on an anonymized data set or making sure that the students are aware and agree to their data being used. Because of the difficulties of anonymizing GitLab data, and the need to quality check the data [59], I recommend operating with informed consent where possible.

5.1.8. Manual Work is Required

As mentioned in Sections 5.1.1 and 5.1.5 deciding who does the actual work is a challenge, that would require manually entering co-authors and a procedure for doing so. Other work would have to be done manually to set up software repository mining in a software engineering course. If GitLab is chosen as a tool, it would be necessary to check that students are not merged if they have similar names or split if they use different author names. This should be done manually, or one would need to find another way of ensuring a good matching between author and student.

If the course coordinator wishes to look at different metrics from GitLab data or different dysfunctions, these will have to be derived. However, there should be no problems having a modular system that can be extended upon.

The most time-consuming manual work done in this project was making a quality measure of lines of code. There are different ways of measuring contribution. The number

of commits could be used. As mentioned in Section 2.3, Parizi, Spoletini, and Singh [37] used five different ways of measuring contribution. In this study, I found that commits varied so greatly in size, and the students had such different commit habits, that using the number of commits as a measure of contribution worked poorly. It could be possible to look more into the other three metrics used by Parizi et al: time spent working on the project, the number of merge pull requests, and the number of files. I found these to be either more difficult to derive or less likely to give results than lines of code. The manual work done to end up with the measure used for lines of code in this study was described in Section 3.3.3. In a course with a set technology stack, it would be easier to determine which file endings to include and which paths to exclude when counting code lines. In a course such as TDT4140, it would be necessary to inspect students' files to map which file endings are relevant for that semester. The course coordinator cannot know whether a team will use a new technology stack, which would likely make them an outlier in the analyses, as a portion of their work is not counted.

Another problem with using lines of code in a course with differing technology stacks is that some languages lead to more or fewer code lines because of the language itself. How much this has impacted the difficulty of finding good measures of contribution is difficult to say, and it could be further explored.

5.1.9. Different Leadership Compositions

D5 - Poor leadership did not correlate with any of the GitLab metrics it was tested against. D5 was normally distributed by the Shapiro-Wilk test. A possible reason might be that poor leadership simply cannot easily be identified by looking at GitLab data. Another reason for the lack of findings might be that good leadership can look so different. By examining what the teaching assistants said about leadership in their groups in the interviews, we can observe that green groups, that is groups that did not face this problem, had very different leadership compositions.

Also, whether the groups implicitly or explicitly chose a leader may have affected how they answered questions regarding the dysfunction in the questionnaire. One of

the questions asked whether someone in the group had been given the responsibility of following up on all or parts of the project. Here the groups were split. The largest portion of students, 42.3%, disagreed with the statement. In the calculation of the students' experience, agreeing to this statement would give a positive score. However, because of the different leadership compositions in functional teams, it is not likely that the answer to this question would help identify dysfunctions.

5.1.10. Ideal Distribution of Tasks

Because of the lack of normality in D6: Too specialized tasks, it was not possible to study correlations between the dysfunction and any of the GitLab metrics.

Multiple of the respondents to the open question in the questionnaire admitted to some students doing more coding and others writing reports. These groups should have been identified as having too specialized tasks. However, whether the group would be dysfunctional for such a distribution of tasks is up for debate. In a professional setting, where the aims of a project are mainly focused on the software product, distributing tasks to the most experienced members might be the ideal situation. The student teams in this course were made up of students from different study programs. If the goal of the course is for all students to gain experience with different parts of software engineering, too much division of labor is unfortunate.

The students who reported in the open question that they divided the work, did not consider this problematic. One student described his team: "*In my group, we gradually got an uneven distribution between report writing and coding, but this was not perceived as a problem, as people had different strengths and wanted different things from the course, and that it helped to keep motivation up and ensure that everyone could contribute. Those who were good at writing reports then also communicated the syllabus well to those who were more interested in coding, and they analyzed the software from a more objective point of view. And I think the vast majority learned a lot in the course, even though there was a more skewed distribution of work tasks. Also, a skewed work distribution in this course is necessary, due to the large amount of work [it requires].*"

Looking at what the teaching assistants considered when determining whether a student team faced this challenge, one considered a group green despite being told that they divided work, because of how closely they worked together. Another group was considered yellow, and the teaching assistant explained that by saying that ideally everyone would be involved in all processes. It is clear from this that the teaching assistants have different opinions on what constitutes a problem in the groups with regards to D6 - Too specialized tasks.

5.1.11. Differences in Students' and Teaching Assistants' Perceptions

Not all groups were represented in the questionnaire. Therefore, there is not data from both the teaching assistant and students for more than nine groups. Also, the questionnaire was answered between 7 and 12 weeks after the interview with the teaching assistants, so it is not unlikely that the dynamics in the groups shifted during that time. With this in mind, the differences between the problems identified by the teaching assistants and by the students can be seen in Table 5.1. The cells are color-coded so that a good match (<1 between a teaching assistant's perception and the group's) is green, a difference over 1 but not over 2 gives yellow, and a difference greater than 2 gives red. As can be seen in Table 5.1, the teaching assistants and students do not share the same perception of the groups' dysfunctions, or the dysfunctions shifted from the interviews were conducted to the questionnaire was answered.

In the interviews with the teaching assistants, all of them at some point said that the questions were difficult to answer with the level of insight they had about the groups. One of them said this: "*You must understand that I do not have the best insights.*"

Another reason for the difference in perception might be that the dysfunctions in the groups have changed from the interviews were performed to the questionnaire was answered at the end of the semester. To get a better understanding of the development of the group dynamics during a semester, it would be wise to run a longitudinal collection of students' experiences. Reviews in between the sprints, and weekly meetings set conditions where the students can get frequent feedback and early intervention to help them with

Table 5.1.: Comparison of dysfunctions identified by teaching assistants and students.

Group	D1		D2		D3		D4		D5		D6	
	TA	S	TA	S	TA	S	TA	S	TA	S	TA	S
A	5	2.3	5	3.5	5	4.2	5	4.2	5	3.8	5	2.5
B	3	3.4	5	3.8	5	4.3	3	4.1	3	3.8	3	4.1
D	3	1.0	5	3.3	3	3.0	5	3.0	5	3.7	3	4.0
E	5	2.0	5	2.7	5	2.0	5	2.3	5	3.0	3	2.7
F	3	1.7	5	3.3	5	3.3	5	4.3	5	3.3	1	3.7
G	5	3.3	5	3.7	5	4.3	5	3.7	5	2.7	5	3.7
H	3	1.0	5	2.3	5	1.3	5	2.0	5	3.3	3	1.0
I	3	1.0	1	1.3	3	1.0	5	2.7	3	3.7	5	3.7
J	3	3.7	4	4.0	5	4.7	5	3.7	5	4.3	3	4.3

their problems. According to the theory of supervision in project-based learning, such a feedback culture is ideal for improving dysfunctional teams (See Section 2.1.3).

5.1.12. Student Experience as a Measure of Group Dynamics

In this study, group experience was a measurement constructed from individual team members' responses to a questionnaire. The questionnaire was answered once, at the end of the semester. The experience of dysfunction in a group is subjective. Looking at the scores for agreement within groups found in Section 4.1.2, we see that the difference between the answers within a group on average doesn't exceed one for any of the dysfunctions. On the question of commitment to the projects, the opinions of the members were the most similar. An average of less than one shows that there is some agreement within groups, but there would also likely be situations with disagreement in individual teams. If student experience should be used as a measure of group dynamics it would be beneficial to get a higher response rate than what was achieved in this study, and, as mentioned in the previous section, gathering data on experience more often could

strengthen student experience as a measure.

An alternative to using group experience as a measure of group dynamics could be to use grades. This could then clearly not be done before the students were assessed. But it could be possible to analyze students one year and use correlations discovered then to help identify dysfunctional groups the next time the course is offered. However, Mierle et al [Mierle2005] examined 200 CVS repositories in an attempt to predict students' grades from 166 different features of work habits and found that none of them had significant correlations. Also, if it is possible to predict students' grades from their work habits, it would then not be linked to specific dysfunctions. That could make it more difficult for teaching assistants to use the findings to guide the students. For instance, in Mierle et al's article, the highest correlation between grades and work habits was when looking at the number of times a space followed a comma in the code. This is not something that can easily be used for recommendations to the student teams.

5.1.13. Correlation vs Causation

When finding a relationship between variables through correlation coefficients, we are, as the name suggests, looking at correlation. This means that there is no foundation for claiming causation based on these analyses. If we can see a trend between a metric of the GitLab data and the students' experience of dysfunction in their group, that does not mean that the metric explored is the reason why they experience a problem. As an example of this, we look at a finding in this study: There is a statistically significant correlation between the number of days a group does more than 7% of their total work on the project and the group's experience of D1: Unfair or unevenly distributed workload.

There can be a multitude of different reasons why the group had few days of intensive work. A group might have members with hectic schedules who cannot work together. The members working at different times of the week could lead to more small commits. Without seeing the other members contribute, the group members could assume the others did less work than their share. This could lead to frustration amongst the members, and they identify D1 as a problem. The cause of the problem in the group would then

likely rather be identified thus:

> "D1 is a problem in the group because the members are not aware of the work the others do in the project", than
>
> "D1 is a problem in the group because there was a low number of days where a considerable part of the total work of the project was done."

5.2. Possibilities

Having explored and discussed the challenges of implementing software repository mining in a project based software engineering course, this section discusses the opportunities and benefits that can be gained by the implementation.

5.2.1. Significant Findings from the GitLab Analysis

There were significant findings for five of the GitLab metrics explored in this study. These correlations could be used to help the teaching assistants get extra insight to the group dynamics of the teams they supervise, and to discover and correct dysfunctions early.

The first metric, "Number of Unique Dates of Issue Creation", gave a negative correlation to D1 - Unfair or unevenly distributed workload, D3 - The participants are not committed to the project, and D4 - Lack of a plan or strategy for the project. A high number of days of issue creation means that the student teams likely did not plan far ahead, or lacked an understanding of what they were doing. For D4, it makes sense that the students would experience a lack of plan and strategies, when tasks were added as they were working. It is interesting to see a high correlation also to D1 and D3. If I were presented with a student team that had a high number of unique days of issue creation, and now knowing the relationship between that and D1 and D3, I would ask if all members of their team participated in sprint planning meetings. Further research could try to replicate this and see if they could find the cause of the relationships between the metric and the dysfunctions.

Next, two metrics focused on when code was committed. “Percentage of Total Lines of Code Committed Between 08:00 - 16:00” showed a positive correlation and “Percentage of Total Lines of Code Committed Between 16:00 - 00:00” showed a negative correlation to D1 - Unfair or unevenly distributed workload, D2 - Too little time is spent working on the project, and D3 - The participants are not committed to the project. The finding for D2 is not surprising, but it is interesting, as there was an agreement amongst students and teaching assistants that the teams generally spent more time than expected working on the project. What these findings say is that teams that do large portions of their work within normal working hours seem to distribute work more evenly, not have time pressure, and experience their team as committed. Some teaching assistants mentioned that their groups had joint work sessions. It seems likely that that would happen during the workday and help in distributing work more evenly.

“Number of Days With More Than 7% of Total Code Lines” is the metric that has the strongest correlations, and which has positive correlations to D1, D2, D3, and D4. What this means is that a high number of days where a decent amount of work is done gives the students a general good experience. Opposite, we can say that if the work was spread out more, with more days of work and less work each time, that is a sign of dysfunction. I would expect groups that work together in joint work sessions would likely score higher on this metric, as most of their commits would likely happen during their common work time. An issue with this metric is that it is calculated from the total code lines during the project. This means it can not be used as is to give feedback to students during a semester. But the correlation is worth exploring further. If the gathering of student experiences happened throughout the semester, this metric could possibly be reworked to also be useful during the semester.

“Number of Days With More Than 1% of Total Commits” gives a negative correlation for D4. What this says is that the more days of work the group has, the less happy they are with their planning. This makes sense, and follows the same logic as the number of days of issue creation. In the same way as for the previous metric, this one can also not be used to identify problems in teams during the semester as it is now.

5.2.2. Differences in Identification of Dysfunctions amongst Teaching Assistants

When the teaching assistants identified a possible problem, or a problem, within a team, they were asked: "How can you see this dysfunction in this team?" The responses to this question varied greatly, showing that teaching assistants have individual opinions on what constitutes a problem. Using software repository mining can help even this out, by giving more objective insight into the teamwork.

5.3. Limitations

This section discusses the limitations of this study. Some trade-offs had to be weighed throughout the different processes, and some challenges may have had an impact on the results of the study.

5.3.1. Questionnaire

For each dysfunction a consistency score was calculated as explained in Section 3.3.1. The consistency score looks at the differences between the highest and lowest score for questions regarding the same dysfunction. If each question could reveal the same problem in the group, the consistency score should be low. The consistency scores for the different dysfunctions varied from 1.35 for D1 to 1.81 for D2. These reasons for the variation for the different questions can be that the questions reveal different problems within the group, even if they fall under the same dysfunction. An example of this can be for D3 - The participants are not committed to the project. One question, "In our group, there were some who showed a lack of effort", had 44.2% strongly disagree or disagree. For the question "Everyone in our group worked to get a good grade", a total of 88.4% of the students either strongly agreed or agreed. In this case, both questions show that a majority of the students had a positive experience, but far more students thought their group members were working for good grades.

Something else that might speak of the quality of the data from the questionnaire is

how long the students spent answering it. The student who answered fastest spent only 1 minute 45 seconds. After removing a few outliers, the average response time for the questionnaire was 5 minutes 8 seconds. Some of the respondents spent so little time answering the questions that it would be fair to say they could not have considered them particularly carefully.

5.3.2. Interviews

The main limitation when it comes to the interviews in this study, is, as mentioned previously, that it was so long between the interviews and the questionnaire that the situation in between might have changed. However, since the interviews were not used in the analyses they should not have much of an impact on the results of the study.

5.3.3. Anonymization

During the data collection process of GitLab data, member names were anonymized. As they were, author names with the same three first letters were assembled into one anonymized member. This was explained in Section 3.3.3. The same way of doing this was done in Haugse's master's thesis from 2021 [55]. However he did not work with an anonymized data set, and in his case, a manual check was performed to ensure that members were not split or merged. That may have happened to some degree in this study. Because of the anonymization that had to be done (See Section 3.2.3) it was not possible to do quality control on the commit data, the issues, or the member assembly that should have been done if there was not anonymization.

5.3.4. Statistical Analysis

A Pearson correlation analysis assumes linearity. For simplicity, and because it may still reveal significant correlations, this statistical method was chosen. However, more correlations could have been discovered if we did not require linearity, but simply a monotonic relationship between the values. A monotonic relationship means that as one value grows, so does the other, but not necessarily linearly. Pearson correlations

work with raw data, while an alternative, Spearman correlation, works with ranked data. Linear relationships would also be easier to use for prediction. Therefore, the linear correlations discovered could be more useful than any monotonic relationships discovered.

5.3.5. Choice of Dysfunctions

Comparing the six dysfunctions focused upon in this study, to the five team dysfunctions in the Lencioni model presented in Section 2.4.2, there is a slight overlap. If we look at the questions asked for D3 - The participants are not committed to the project, it corresponds to "Lack of commitment" and somewhat to "inattention to results" in the Lencioni model. The other dysfunctions mentioned in the Lencioni model: "Avoidance of accountability", "Fear of conflict", and "Absence of trust" were considered difficult to determine from GitLab data. The process of determining the six dysfunctions of this study was not a systematic one. A limitation of this study is therefore that the dysfunctions chosen may not have been the most important ones.

5.3.6. Identifying Problems in Well-Functioning Teams

In general, few teams could be classified as dysfunctional in the course. The course is well designed with information easily accessible, and regular opportunities for the students to ask questions and get feedback. Looking at the scatter plots for most of the dysfunctions, we can see that the majority of the points are in the upper half of the Student-Experience-axis. This proves a limitation in this study, as identifying problems that do not exist is not possible.

6. Conclusion

This chapter attempts to answer the research questions of the study. The contributions of the project are presented as a list of implications for similar cases. Lastly, some themes for future work are suggested.

6.1. Conclusion and Contributions

This thesis has tried answering the following research questions:

RQ1: How can metrics derived from data from software repositories be analyzed and presented in a way that gives value to students and supervisors?

> From the qualitative analysis of interviews with teaching assistants and responses to a questionnaire answered by students in the course, I found that some possible benefits of using software repository mining were early intervention in cases of uneven distribution of workload and more detailed feedback. Some metrics tested in this study did have a statistically significant correlation to the following four dysfunctions: D1 - Unfair or unevenly distributed workload, D2 - Too little time is spent working on the project, D3 - The participants are not committed to the project, and D4 - Lack of a plan or strategy for the project. The metrics explored in this study are not comprehensive, and more metrics can likely be derived to better predict these dysfunctions or to reveal new ones.
>
> According to literature, students benefit from elaborate feedback [29]. Early intervention is important to help dysfunctional teams [26]. The correlations

found between GitLab data and students' experiences of dysfunctions in their group dynamic, make it likely that we can automatically identify some group dysfunctions, and thus help the students have more functional teamwork.

Supervisors preferred data to be presented to them in an organized and easily interpreted way so that they could spend more of their time working with the students. A possible way to present the data could be using a traffic light model, with a challenge being green would mean the team has no issues with it, yellow showing potential for problems, and red meaning a problem is likely occurring in the group.

RQ2: What do students and supervisors think about the idea of using metrics from software repositories to identify problems within the group?

Both the students and teaching assistants in this study identified multiple challenges concerning the accuracy of the data from GitLab to reality. From their feedback and from experiences I had while mining the students' GitLab data, the following list of practical implications is the main contribution of this thesis. The implications can be used as advice for anyone who wishes to use software repository mining to help with feedback in their project-based Software Engineering Courses.

Practical Implications

- If the students in the course use the practice of pair programming, that should be taken into account. Either there should be a common Git routine that sets the co-authors of a commit, or commit data will likely not be representative enough to say anything about work distribution by looking at individual members' commits. Unequal work distribution might still be revealed by analyzing the group as a whole if a set Git routine goes against other learning objectives in the course.
- Using code lines as a measure of work done requires filtering of the commit data. Some paths must be excluded and only certain file endings

should be included.

- In the same way as with pair programming, issue assignment should be linked to all members that work on an issue in some way, or it should be disregarded.
- GitLab analysis should not be used as the main way of observing the groups and providing feedback, but it can be a useful supplement.
- If part of the course work is not related to programming, that should be considered during analysis and when giving feedback to the groups.
- Consider making sure all students know how to properly use Git. If the students are already experienced with Git, set routines may make it easier to analyze their data, and can eliminate some uncertainty.

Limitations of the study were discussed in Section 5.3. Anonymization of the data sets during data extraction made it difficult to do quality control. The questionnaire, which was used as the foundation for the measure of student experience of dysfunctions in their teams, was mostly answered by a single member of the team. And there were overall few dysfunctional teams.

6.2. Future Work

This study focused on six dysfunctions and fifteen GitLab metrics. Further work can be done by expanding on both of these. A team dysfunction that was prevalent in the groups in this course was that of different skill levels among members of a team. It would be interesting to see if this could be identified from GitLab data.

Further work could be done to find a better measure of member contribution. Lines of code require manual pre-processing to be used as a measure. In courses with differing technology stacks, lines of code will vary based on the languages used by the groups, and a comparison between groups is not possible.

Two of the GitLab metrics, “Number of Days With More Than 7% of Total Code

Lines" and "Number of Days With More Than 1% of Total Commits", gave statistically significant correlations but could only reveal dysfunctions after the end of the project. More work should be done on the metrics to see if they can also reveal something during the development process.

Bibliography

[1] Briony J. Oates. *Researching Information Systems and Computing.* SAGE Publications, 2006.

[2] TDT4140 Software Engineering. Semester plan. Figure found on the Blackboard (LMS) page for the course.

[3] Mathea Godø Hildre. Specialization project fall 2021. unpublished, 2021.

[4] Laerd Statistics. Pearson's partial correlation using spss statistics. 2017. Statistical tutorials and software guides. Retrieved from https://statistics.laerd.com/.

[5] Barbara Oakley, Richard M Felder, Rebecca Brent, and Imad Elhajj. Turning student groups into effective teams. *Journal of student centered learning*, 2(1):9–34, 2004.

[6] Mario Žagar, Ivana Bosnić, and Marin Orlić. Enhancing software engineering education: A creative approach. In *Proceedings of the 2008 International Workshop on Software Engineering in East and South Europe*, SEESE '08, page 51–58, New York, NY, USA, 2008. Association for Computing Machinery.

[7] Matthew Skelton, Manuel Pais, and Ruth Malan. *Team Topologies: Organizing Business and Technology Teams for Fast Flow.* IT Revolution Press, Portland, OR, 2019.

[8] Michael S Cole, Frank Walter, and Heike Bruch. Affective mechanisms linking dysfunctional behavior to performance in work teams: a moderated mediation study. *Journal of Applied Psychology*, 93(5):945, 2008.

[9] Niki Gitinabard, Ruth Okoilu, Yiqao Xu, Sarah Heckman, Tiffany Barnes, and Collin Lynch. Student teamwork on programming projects: What can github logs show us? *arXiv preprint arXiv:2008.11262*, 2020.

[10] Anil Aggarwal. Dysfunctional groups: An exploratory study. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 455–462, 2016.

[11] Jalerson Lima, Christoph Treude, Fernando Figueira Filho, and Uirá Kulesza. Assessing developer contribution with repository mining-based metrics. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 536–540. IEEE, 2015.

[12] Jeremy Roschelle and Stephanie Teasley. The construction of shared knowledge in collaborative problem solving. *Computer Supported Collaborative Learning*, 01 1995.

[13] Yiping Lou, Philip C. Abrami, and Sylvia d'Apollonia. Small group and individual learning with technology: A meta-analysis. *Review of Educational Research*, 71(3):449–521, 2001.

[14] W. Martin Davies. Groupwork as a form of assessment: common problems and recommended solutions. *Higher Education*, 58(4):563–584, Oct 2009.

[15] Will W. K. Ma. Effective learning through deep learning, what matters: Self, others, way of thinking, and/or design of learning environment? In Will W.K. Ma, Kar-wai Tong, and Wing Bo Anna Tso, editors, *Learning Environment and Design*, pages 3–17, Singapore, 2020. Springer Singapore.

[16] Ozdemir Gol and Andrew Nafalski. *Collaborative learning in engineering education.* PhD thesis, Unesco, Internationa Centre for Engineering Education, 2007.

[17] Patrick T. Terenzini, Alberto F. Cabrera, Carol L. Colbeck, John M. Parente, and Stefani A. Bjorklund. Collaborative learning vs. lecture/discussion: Students' reported learning gains*. *Journal of Engineering Education*, 90(1):123–130, 2001.

[18] Sally Fincher and Daniel Knox. The porous classroom: Professional practices in the computing curriculum. *Computer*, 46(9):44–51, 2013.

[19] Will W. K. Ma. *Digital Communication and Learning: Changes and Challenges*, chapter Effective Learning Through Project-Based Learning: Collaboration, Community, Design, and Technology, pages 317–341. Springer Singapore, Singapore, 2022.

[20] Kyriaki Matsouka and Dimitrios M. Mihail. Graduates' employability: What do graduates and employers think? *Industry and Higher Education*, 30(5):321–326, 2016.

[21] M. Cecilia Bastarrica, Daniel Perovich, and Maira Marques Samary. What can students get from a software engineering capstone course? *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, pages 137–145, 2017.

[22] Dragutin Petkovic, Gary Thompson, and Rainer Todtenhoefer. Teaching practical software engineering and global software engineering: Evaluation and comparison. *SIGCSE Bull.*, 38(3):294–298, jun 2006.

[23] Maria Spichkova. Industry-oriented project-based learning of software engineering. In *2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 51–60, 2019.

[24] Bernd Bruegge, Stephan Krusche, and Lukas Alperowitz. Software engineering project courses with industrial clients. *ACM Transaction on Computing Education*, 08 2015.

[25] Verlin B. Hinsz, R. Scott Tindale, and David A. Vollrath. The emerging conceptualization of groups as information processors. *Psychological Bulletin*, 121(1):43–64, 1997.

[26] Nicola Brown. Assessing individuals within teams in project-based learning courses — strategies, evaluation and lessons learnt. In *2021 IEEE Global Engineering Education Conference(EDUCON)*, pages 846–850, 2021.

[27] Susan Brown Feichtner and Elaine Actis Davis. Why some groups fail: a survey of students' experiences with learning groups. *Organizational Behavior Teaching Review*, 9(4):58–73, 1984.

[28] Margaret Price, Karen Handley, Jill Millar, and Berry O'Donovan. Feedback : all that effort, but what is the effect? *Assessment & Evaluation in Higher Education*, 35(3):277–289, 2010.

[29] Ulrike-Marie Krause, Robin Stark, and Heinz Mandl. The effects of cooperative learning and feedback on e-learning in statistics. *Learning and Instruction*, 19(2):158–170, 2009.

[30] Torgeir Dingsøyr. Agile iteration reviews in a project course: A key to improved feedback and assessment practice. In *2021 Third International Workshop on Software Engineering Education for the Next Generation (SEENG)*, pages 21–25, 2021.

[31] Chris Kemper and Ian Oxley. *Foundation Version Control for Web Developers*, chapter In the Beginning There Were Just Files, pages 25–32. Apress, Berkeley, CA, 2012.

[32] Scott Chacon and Ben Straub. *Pro Git*, chapter Getting Started, pages 1–13. Apress, Berkeley, CA, 2014.

[33] GitLab. About gitlab. Available Online: https://about.gitlab.com/company/, Accessed: 2022-05-24.

[34] Wil van der Aalst. *Process Mining: Data Science in Action*, chapter Process Mining: The Missing Link, pages 25–52. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[35] Mário André de F. Farias, Renato Novais, Methanias Colaço Júnior, Luís Paulo da Silva Carvalho, Manoel Mendonça, and Rodrigo Oliveira Spínola. A systematic

mapping study on mining software repositories. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 1472–1479, 2016.

[36] Wouter Poncin, Alexander Serebrenik, and Mark van den Brand. Mining student capstone projects with frasr and prom. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pages 87–96, 2011.

[37] Reza M Parizi, Paola Spoletini, and Amritraj Singh. Measuring team members' contributions in software engineering projects using git-driven technology. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–5. IEEE, 2018.

[38] Sivana Hamer, Christian Quesada-López, Alexandra Martínez, and Marcelo Jenkins. Measuring students' contributions in software development projects using git metrics. In *2020 XLVI Latin American Computing Conference (CLEI)*, pages 531–540. IEEE, 2020.

[39] Jihie Kim, Erin Shaw, Hao Xu, and GV Adarsh. Assisting instructional assessment of undergraduate collaborative wiki and svn activities. *International Educational Data Mining Society*, 2012.

[40] Megha Mittal and Ashish Sureka. Process mining software repositories from student projects in an undergraduate software engineering course. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, page 344–353, New York, NY, USA, 2014. Association for Computing Machinery.

[41] Chung-Yang Chen, Ya-Chun Hong, and Pei-Chi Chen. Effects of the meetings-flow approach on quality teamwork in the training of software capstone projects. *IEEE Transactions on Education*, 57(3):201–208, 2014.

[42] Steven J Karau and Kipling D Williams. Social loafing: A meta-analytic review and theoretical integration. *Journal of personality and social psychology*, 65(4):681, 1993.

[43] Elizabeth Pfaff and Patricia Huddleston. Does it matter if i hate teamwork? what impacts student attitudes toward teamwork. *Journal of marketing education*, 25(1):37–45, 2003.

[44] Denny E McCorkle, James Reardon, Joe F Alexander, Nathan D Kling, Robert C Harris, and R Vishwanathan Iyer. Undergraduate marketing students, group projects, and teamwork: The good, the bad, and the ugly? *Journal of Marketing Education*, 21(2):106–117, 1999.

[45] Marco Kuhrmann and Jürgen Münch. When teams go crazy: An environment to experience group dynamics in software project management courses. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pages 412–421. IEEE, 2016.

[46] Patrick M Lencioni. *The five dysfunctions of a team: Team assessment*. John Wiley & Sons, 2012.

[47] Yngve Lindsjørn, Dag IK Sjøberg, Torgeir Dingsøyr, Gunnar R Bergersen, and Tore Dybå. Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software*, 122:274–286, 2016.

[48] William B Joyce. On the free-rider problem in cooperative learning. *Journal of Education for Business*, 74(5):271–274, 1999.

[49] Diane Strode, Torgeir Dingsøyr, and Yngve Lindsjorn. A teamwork effectiveness model for agile software development. *Empirical Software Engineering*, 27(2):1–50, 2022.

[50] Randall S Hansen. Benefits and problems with student teams: Suggestions for improving team projects. *Journal of Education for business*, 82(1):11–19, 2006.

[51] Amir Erez, Jeffrey A Lepine, and Heather Elms. Effects of rotated leadership and peer evaluation on the functioning and effectiveness of self-managed teams: a quasi-experiment. *Personnel Psychology*, 55(4):929–948, 2002.

[52] G. Walsham. Interpretive case studies in is research: nature and method. *European Journal of Information Systems*, 4(2):74–81, May 1995.

[53] NTNU. Tdt4140 - software engineering. Available Online: https://www.ntnu.edu/studies/courses/TDT4140#tab=omEmnet, Accessed: 2022-06-15.

[54] UiO: Universitetet i Oslo. Nettskjema. Available Online: https://nettskjema.no/, Accessed: 2022-06-15.

[55] Åsmund Haugse. Git in an educational context. Master's thesis, NTNU - Norwegian University of Science and Technology, Trondheim, Norway, 2021.

[56] IBM. Ibm spss statistics. Available Online: https://www.ibm.com/products/spss-statistics, Accessed: 2022-07-10.

[57] Jacob Cohen. *Statistical power analysis for the behavioral sciences*, chapter The Significance of a Product Moment rs. Routledge, 2 edition, 1988.

[58] Nicolas E Gold and Jens Krinke. Ethics in the mining of software repositories. *Empirical Software Engineering*, 27(1):1–49, 2022.

[59] Daniel Barros, Flavio Horita, Igor Wiese, and Kanan Silva. A mining software repository extended cookbook: Lessons learned from a literature review. In *Brazilian Symposium on Software Engineering*, pages 1–10, 2021.

Appendices

A. Google Trends Version Control Systems

Week	Git	Mercurial	Concurrent Versions System	Apache Subversion
2021-01-03	75	1	<1	4
2021-01-10	82	1	<1	4
2021-01-17	84	1	<1	4
2021-01-24	85	1	<1	4
2021-01-31	87	1	<1	4
2021-02-07	79	1	<1	4
2021-02-14	84	1	<1	4
2021-02-21	85	1	<1	4
2021-02-28	92	1	<1	4
2021-03-07	89	1	<1	4
2021-03-14	86	1	<1	4
2021-03-21	89	1	<1	4
2021-03-28	85	1	<1	4
2021-04-04	84	1	<1	4
2021-04-11	88	1	<1	4
2021-04-18	87	1	<1	4
2021-04-25	86	1	<1	4
2021-05-02	81	1	<1	3
2021-05-09	84	1	<1	4
2021-05-16	89	1	<1	4
2021-05-23	86	1	<1	4
2021-05-30	85	1	<1	4
2021-06-06	91	1	<1	4
2021-06-13	89	1	<1	4

Table 1.: Most searched for VCS on google for 2021 - First half.

Week	Git	Mercurial	Concurrent Versions System	Apache Subversion
2021-06-20	91	1	<1	4
2021-06-27	88	1	<1	4
2021-07-04	90	1	<1	4
2021-07-11	92	1	<1	4
2021-07-18	89	1	<1	4
2021-07-25	88	1	<1	4
2021-08-01	85	1	<1	4
2021-08-08	83	1	<1	3
2021-08-15	88	1	<1	4
2021-08-22	83	1	<1	3
2021-08-29	90	1	<1	4
2021-09-05	88	1	<1	4
2021-09-12	97	1	<1	4
2021-09-19	91	1	<1	3
2021-09-26	96	1	<1	4
2021-10-03	96	1	<1	4
2021-10-10	95	1	<1	4
2021-10-17	94	1	<1	4
2021-10-24	96	1	<1	4
2021-10-31	91	1	<1	4
2021-11-07	95	1	<1	4
2021-11-14	100	1	<1	4
2021-11-21	89	1	<1	4
2021-11-28	96	1	<1	4
2021-12-05	97	1	<1	4
2021-12-12	96	1	<1	4
2021-12-19	82	1	<1	4
2021-12-26	60	1	<1	2

Table 2.: Most searched for VCS on google for 2021 - Second half.

B. Questionnaire

Likert scale questions:

1. Someone in our group has spent more time on the project than others.
2. No one in our group took clear responsibility for the organization of the project.
3. Our group worked approximately the number of hours expected in the course.
4. In our group, there were some who showed a lack of effort.
5. We experienced that not everyone in the group was committed to the work.
6. Our group had a plan for what to do.
7. Not everyone in the group understood how the project was to be carried out.
8. In our group, there were some members who worked less on the project than we had agreed upon.
9. Each group member did not get to cover the entire curriculum through the project work.
10. The effort put into the project was unevenly distributed in our group.
11. Everyone in our group worked to get a good grade.
12. We had good communication in the group about strategy for the project.
13. In our group we had someone who made sure we had good progress.
14. In our group, someone was given the responsibility of following up all or parts of the project.
15. In our group, everyone put in an approximately equal amount of work to the project.
16. In our group, there was little variation in the work tasks for each member.

17. We experienced a shortage of time on the project because we did not spend enough hours compared to what was planned.

18. We made sure that everyone got to try different tasks during the project.

Open question:

What do you think about letting teaching assistants analyze your GitLab data in order to provide better guidance?

> GitLab stores data such as issues and commits. If you analyze this data and look at e.g. when things are pushed and how often, as well as who is assigned to various issues, etc., you can say something about collaboration and group dynamics. Knowledge of this can give teaching assistants a tool to help groups solve problems they encounter during their projects. Do you see any possible challenges with this?

C. Interview Guide

Trying to figure out:

- Which of the challenges considered in my analysis of the GitLab data are their groups facing, by their perception?
- For each challenge, give the group a color: Red, yellow, or green.
- How would you like to have the data presented?
- Do you see any problems with using metrics derived from GitLab as an aid in advising students in the project?

Questions:

1. For group X, what challenges would you say they face as a group?

   "Having analyzed the data gathered from GitLab, I am looking at some specific challenges. I will now present the challenges and ask you a couple of questions about the group in relation the challenges."

2. Tell me about group X's experiences with challenge A.

   "We are now using a traffic light model. Green means the group has no problems or insignificant problems with a challenge. Yellow means that the group is at risk for having problems with the challenge. Red means that the challenge is a problem for the group."

3. For challenge Y, what color would you give to group X?

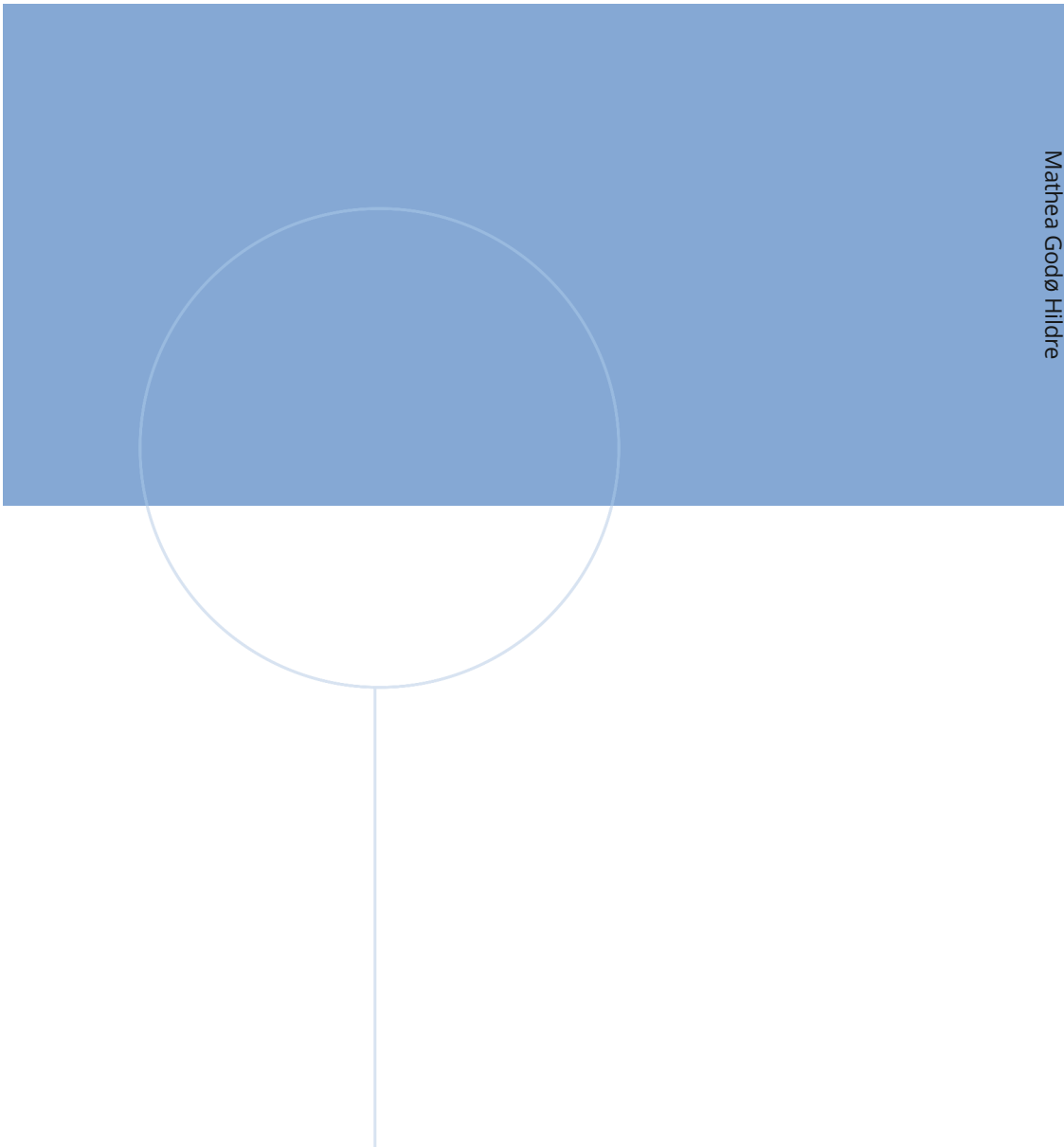
   Repeat for each challenge and each group

4. A few options for presenting the information are: Traffic light, number line, numbers in tables. Would you prefer any of the presentation formats over others? What do you think would be a good representation of the GitLab data for you as a teaching assistant?

5. Are there any presentations you would prefer other than these?

6. What are issues that should be taken into account when presenting this information to teaching assistants and supervisors?

7. Do you see any problems with using metrics derived from GitLab as an aid in advising students in the project?



Mathea Godø Hildre

NTNU
Norwegian University of
Science and Technology