TU WIEN Informatics

Continuous Analysis, Monitoring, and Comparison of Student Project Portfolios in Software Engineering Courses

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Manuel Stöger, BSc
Matrikelnummer 11775194

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Thomas Grechenig

Wien, 23. Jänner 2023

Unterschrift Verfasser	Unterschrift Betreuung

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Continuous Analysis, Monitoring, and Comparison of Student Project Portfolios in Software Engineering Courses

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Manuel Stöger, BSc
Registration Number 11775194

to the Faculty of Informatics

at the TU Wien

Advisor: Thomas Grechenig

Vienna, 23rd January, 2023

Signature Author

Signature Advisor

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Continuous Analysis, Monitoring, and Comparison of Student Project Portfolios in Software Engineering Courses

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Manuel Stöger, BSc
Matrikelnummer 11775194

ausgeführt am
Institut für Information Systems Engineering
Forschungsbereich Business Informatics
Forschungsgruppe Industrielle Software
der Fakultät für Informatik der Technischen Universität Wien

Betreuung: Thomas Grechenig

Wien, 23. Jänner 2023

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Manuel Stöger, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 23. Jänner 2023

Manuel Stöger

Danksagung

Ich möchte mich bei meiner Familie und meinen Freunden für die Unterstützung während meines Studiums bedanken. Mein Dank gilt auch meinen Freunden, die diese umfangreiche Arbeit korrekturgelesen und mir wertvolle Ratschläge für meine Arbeit gegeben haben.

Außerdem möchte ich mich bei dem Administrator des SEPM-Kurses bedanken, der mir die Daten zur Verfügung gestellt hat und mir regelmäßig Informationen zu allen organisatorischen und technischen Fragen des Kurses gab.

Zu guter Letzt möchte ich die Unterstützung meines Betreuers bedanken, der mir das Thema zur Verfügung stellte und mir auch wertvolle Anregungen gab.

Acknowledgements

I would like to thank my family and friends for their support throughout my studies. I would also like to thank my friends who proofread this extensive work and gave valuable advice on my writing.

In addition, I would like to thank the admin of the SEPM class who provided me with the data and regularly gave me information on all organizational and technical issues of the course.

Last but not least, I would like to thank the support of my supervisor who made the topic available to me and also provided me with valuable input.

Kurzfassung

Die Nutzung von Daten aus Software Repositories hat in den letzten Jahren an Bedeutung gewonnen. Eine gute Gelegenheit, eine große Menge an Software Repositories zu analysieren, bieten die Projekte von Studierenden einer Softwareengineering-Lehrveranstaltung an der TU Wien. Da sich in einem Portfolio (einem Semester) ähnliche Projekte befinden, kann der Einsatz von Data Mining in einem akademischen Umfeld wertvolle Erkenntnisse für Lehrende und Betreuende liefern. Werkzeuge, die im universitären Kontext typischerweise zur Analyse von Software Repositories eingesetzt werden, haben meist einen individuellen Fokus: Einerseits unterstützen sie zum Beispiel die Überprüfung und Bewertung von Studierenden, andererseits steht die automatische Benotung der Studierenden im Mittelpunkt. Es besteht jedoch noch eine Nische für eine Lösung, um eine fundierte Entscheidungsgrundlage bieten zu können, die grundlegende Visualisierungs- sowie fortschrittliche, kontinuierliche Analysefunktionen kombiniert.

In dieser Arbeit wird der Einsatz von Data Mining für Software Repositories in Kombination mit Datenvisualisierung in einem universitären Umfeld vorgestellt, um die derzeitige Lücke in diesem Bereich zu schließen. Das Ziel der Arbeit ist es, mithilfe eines Prototyps für die kontinuierliche Analyse, Überwachung und den Vergleich von Software Repositories, ein datengestütztes Werkzeug zu schaffen, um eine Verbesserung der Qualität der Lehre im Bereich des Softwareengineerings zu ermöglichen.

Hypothesen und Anforderungen für die Unterstützung der Lehre wurden aus den Limitierungen bestehender Forschungsprojekte an der TU Wien abgeleitet. Basierend auf den Anforderungen wurden zunächst Mockups erstellt und auf deren Nutzen evaluiert, parallel dazu wurden die Hypothesen mit dem Informationsbedarf der Experten abgeglichen. Die Ergebnisse und Rückmeldungen aus der ersten Interviewrunde flossen weiters in die finale Umsetzung ein. Abschließend bestätigten fünf Fachleute in der Evaluierung des Prototyps die Gültigkeit der Hypothesen.

Das System ist daher geeignet, datenbasierte Einblicke zu gewähren. Es ermöglicht Lehrenden und Betreuenden, die Leistungen ihrer Studierenden objektiver zu bewerten und zu verstehen. Damit liefert es wertvolle Erkenntnisse für die Lehre im Bereich des Softwareengineerings.

Keywords: *Softwareengineering-Lehre, Datenvisualization, Software Repository Mining, Education Intelligence, Data-Mining*

Abstract

The use of data from software repositories has gained significant attention in recent years. Student projects of a software engineering class at TU Wien represent a unique opportunity to analyze a large amount of software repositories. Having various similar projects within one portfolio (one term), the use of data mining in an educational setting can provide valuable insights for lecturers and supervisors. Tools commonly used in the academic context for analyzing software repositories tend to have a specialized focus. On the one hand, for example, they support inspection and assessment of students. On the other hand, tools also center on grading students automatically. However, the niche for a solution which combines basic visualization and advanced continuous analysis capabilities to assist in making informed decisions, has not yet been filled.

This thesis suggests the use of software repository data mining in combination with data visualization in a university setting to address the current gap in this area. The primary objective is to prototype a data-driven tool for continuous analysis, monitoring, and comparison of software repositories, with the goal of enhancing the quality of teaching in the field of software engineering.

Existing research projects at the TU Wien were analyzed to derive hypotheses and requirements for support in the context of a university class. Based on the requirements, in a first step, mock-ups for the prototype were created and their practicality evaluated. At the same time the hypotheses were verified by the experts to determine their information needs. The results and feedback from the first round of interviews were incorporated into the prototype and its visualizations. Finally evaluating the prototype, five experts confirmed the validity of the hypotheses.

The findings demonstrate that the system is appropriate for facilitating data-driven insights. It enables lecturers and supervisors to assess and comprehend the performance of their students more objectively. As a result, the system provides valuable intelligence in a software engineering education setting.

Keywords: *Software Engineering Education, Data Visualization, Software Repository Mining, Education Intelligence, Data Mining*

Contents

CHAPTER 1

Introduction

Collecting and analyzing data from source code repositories of software projects has gained importance in empirical software engineering research during the past decade [116, 121]. The scientific field of Mining Software Repositories (MSR) enables researchers to examine information which originates during the software development process, such as source code, Version Control Systems' (VCSs) metadata, and issue reports [25, 63, 108, 109, 121, 126]. Hence, the focus of the MSR area is on extraction and analysis of heterogeneous data, based on existing repositories [104]. On the one hand, MSR aims to find interesting, practical, and helpful information to assist developers in making informed decisions. This helps to improve the overall software development process [16, 94, 104]. On the other hand, MSR permits the empirical investigation of numerous aspects which include software evolution, developer networks and characterization, bug prediction, and effort estimation [73]. The aforementioned characteristics allow to answer tailored (research) questions. For this purpose, MSR-based studies typically select repositories which fit specific criteria [43].

Previous research on MSR has mainly focused on open-source software repositories [55, 121]. However, the application of MSR in the context of software engineering education to assist in making informed decisions has received less attention. This thesis aims to bridge this gap by examining the potential of using MSR-based data mining in combination with data visualization techniques. Throughout this work this combination is referred to as an "Education Intelligence (EI) system".

The primary objective of the present thesis is to prototype and demonstrate a web-based EI system to continuously analyze, monitor, and compare student project portfolios in a software engineering course at TU Wien. In this context data-driven insights offer instructors and supervisors the opportunity to more objectively evaluate, understand, and gain valuable insights from their students' work.

Inspired by the term Business Intelligence (BI), the concept of Education Intelligence (EI) — as used in this thesis — is outlined in Definition 1.1. Business Intelligence provides principles and approaches for employing fact-based support systems to enhance corporate decision-making [64] and so does EI in the context of a university course. An "Education Intelligence system" can be used to support lecturers and supervisors in evaluating students' work based on available collected information.

Definition 1.1 *Education Intelligence (EI) refers to the use of data-driven techniques to gather, analyze, and interpret information from various sources in a software engineering education context.*

In the following subsections, the thesis' topic is introduced in more detail: Section 1.1 illustrates the problem and Section 1.2 outlines the motivation behind the study. In Section 1.3 the research questions are introduced, while Section 1.4 presents the expected results. The scientific contributions come next in Section 1.5 and finally, Section 1.6 provides an overview of this thesis' structure.

1.1 Problem Description

At the TU Wien, in a software engineering course like Software Engineering and Projectmanagement students start by working on an individual assignment, where each student gets the same assignment specification, which once passed, is followed by a group assignment. The groups then individually work on their projects, are graded by a teaching assistant, and produce independent artifacts. Each group has its individual focus on, for example User Interface (UI), Continuous Integration (CI) or testing, which results in different valid solutions. Although each group uses the same programming language and frameworks (Java, Spring Boot and Angular), these individual aspects made a direct comparison impossible in the past.

Until now no viable solution was found for continuous portfolio-wide metric monitoring to extract a basis of comparison for all groups' project state regarding for example, their work-breakdown, project development or software quality. Students also need to track their time spent on the project during the group phase (110 hours are estimated), however, the real effort of students and the effort distribution within and over groups is not visualized yet. Another unknown aspect of the class is the groups' constellation and its influence. There was never any evaluation done on how a student's performance evolves from the Individual Phase to Group Phase, or how software-relevant metrics differ between groups.

After passing the individual phase, each student joins a group by either forming a group or by being assigned to a group by the course staff. A group is made up of five to six students and may, thus, be self-arranged, randomly-arranged or both if a self-arranged

group consists of fewer than five students. These groups are then distributed over two research divisions:

Div. A All groups receive the same assignment specification, which consists (among other elements) of seven mandatory User Stories (USs) and three extended User Stories of which two need to be chosen. So, every group is limited to these nine User Stories but is free to choose how to structure their work.

Div. B Groups are working on their own project idea and are only restricted by time: The project's effort must not exceed 110 hours per person. In contrast to Research Division A, these groups must also deliver a project proposal and contract, which is used as assignment specification.

The current situation does not allow the course staff team to get an overview of the group's current status without manually collecting the required information for each group separately. In the past external tools were not integrated into the repository instance for keeping an overview of the quality of the produced artifacts or of the project progression due to administrative overhead. Another difficulty about integrating common measurement tools for software projects is their typical focus on code quality[1] or code contribution[2,3]. However, in an educational context, code quality as well as code contribution are only parts of the grading scheme. Typical aspects of the grading scheme also consider: workload-distribution, time effort distribution, code contribution, quality of the outcome, consideration of feedback, fulfilling non-functional requirements, etc. Nevertheless, the individuality of groups does not allow a direct comparison based on the same set of criteria.

1.2 Motivation

A novelty of the examined course at the TU Wien is that all student groups of research division — as previously described in Section 1.1 — are working on the same assignment specification, are limited by time and therefore the produced data is very practical for comparing and also ranking groups based on that.

Creating a tool to allow lecturers to investigate the student projects in rather little time (compared to the current state) can provide meaningful benefits for both, lecturers and students. When lecturers are able to analyze the development of a project, individual controlling and steering of a group is easier and problems or workload imbalances might come up earlier than they do now.

To deliver an appropriate visualization solution, however, the specific information requirements must first be examined, followed by the design, implementation, and evaluation of a data mining and visualization prototype.

[1] https://www.sonarqube.org/, Accessed: *23.01.2023*
[2] https://github.com/ejwa/gitinspector, Accessed: *23.01.2023*
[3] https://github.com/Woutrrr/metricminer2, Accessed: *23.01.2023*

1.3 Research Questions

Within the scope of the presented diploma thesis, the following questions will be answered:

RQ1 a) What is a suitable continuous mining process and architecture for continuous analysis, monitoring, and comparison of software portfolios in software engineering education?

b) How can this continuous process be implemented using GitLab?

c) What are the expert's information needs?

RQ2 a) What is a suitable intelligence system for software engineering education?

b) How do experts rate the proposed prototype?

RQ3 a) What is the difference between student projects inside a project portfolio?

b) What is the difference between project portfolios of different terms?

In the context of this work, the term *project portfolio*, as will be defined later in Definition 2.1 of Section 2.1, is, among others, characterized by the collection of several similar student software projects within one term.

1.4 Expected Results

This thesis provides a viable tool-solution for course staff teams to keep an overview of group projects using various metrics without any additional work and to compare terms with each other easily. Thus, the proposed solution will adapt and build upon the idea of the existing research project "Portfoliotrix"[4] [53] by Genfer et al. and "Binocular" [56] by Grabner et al.

The expected result of this diploma thesis is the development of a mining process architecture to collect available data, which is then combined with an existing data visualization (or Business Intelligence) system. The prototype is suitable to be used in a software engineering course, assisting the staff team. As a result of the data mining and visualization process, knowledge about the groups itself, its students and generally terms regarding

- planned effort and real effort (Individual Phase and Group Phase),
- distribution of points during the Individual Phase, and
- distribution of grades (Individual Phase, Group Phase, and final grade)

becomes derivable. Using the final prototype, an easier monitoring and comparison of the project's progresses, a comparison of the projects based on metrics and comparing

[4] https://github.com/INSO-TUWien/portfoliotrix, Accessed: *23.01.2023*

different terms based on these metrics should be possible. This should also allow to draw conclusions about changes within the course, for example workload related changes in the assignment specification over past terms. All these metrics are then integrated in a visualization dashboard application to provide useful visual representations for approximately 25-30 groups per term.

1.5 Contributions

This thesis establishes a data-driven approach in the software engineering education field, allowing a semi-live monitoring of students and terms based on Git information. Secondly, fundamental study of available data and visualizations show how data can be used in the scientific education field to support course staff teams, link data and help to assure educational quality. Overall, this thesis intends to indicate the necessity of more research to be conducted in the software engineering education field. *Education Intelligence* (*EI*) enhances data-driven evaluation of both student's performances and class design and is able to support staff in designing classes, exams, projects, etc. in the future.

1.6 Structure

The structure of this thesis follows the "Data Science Road Map" of Cady [17] describing the steps required to solve a data science problem. Figure 1.1 illustrates the necessary steps. Consequently, this thesis is structured as follows:

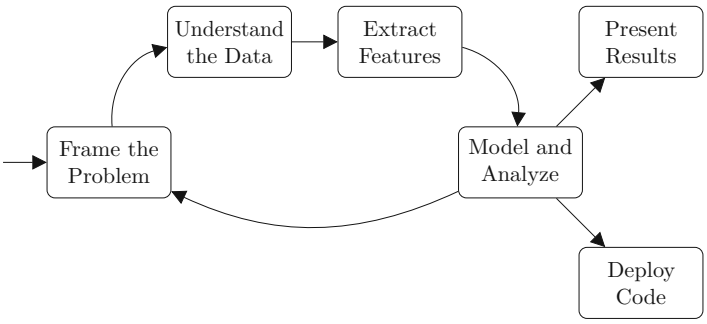


Figure 1.1: Data Science Road Map [17]

- *Chapter 1*, the current chapter, describes the problem examined within this thesis, followed by the motivation, research questions and expected results. Moreover, the scientific contribution provided in the research area of data-driven software engineering education is outlined. The pendant in Figure 1.1 is the first element *Frame the Problem*.

- *Chapter 2* explains all fundamental concepts and terms of this study. First of all, the required domain related concepts and terms are explained, followed by an introduction of data visualization and relevant statistics.

- *Chapter 3* then explains the current research state and introduces the most relevant papers this thesis is based on. The first subsection is divided into the three main research aspects of this thesis: Mining Software Repositories, Data Visualization and Software Engineering Education. Section 3.2 is about already existing tools, divided into Education-related and Data Visualization, describing their purpose and advantages and disadvantages. Finally, the distinction to the current state of the art is made in Section 3.3.

- The methodology used for this study, is described in *Chapter 4*. Firstly, the approach separated per research question is outlined in Section 4.1, followed by the methodological literature and technology review in sections 4.2 and 4.3. In addition, the development process and the proof of concept are explained in sections 4.4 and 4.5, concluding with the evaluation approach in Section 4.6.

- *Chapter 5* describes the interview process to collect the needs of domain experts working in the study's field. Firstly, the created basic visualization concepts are presented in Section 5.1, followed by the study design in Section 5.2 and eventually preparing the interview results in Section 5.3.

- The core of this thesis are *Chapter 6 and 7*. First of all, the organization of the data is explained (Figure 1.1, *Understand the Data*) followed by the implementation details of the Extract, Transform, Load (ETL) transformer (Figure 1.1, *Extract Features*) and the data storage method (Figure 1.1, *Model and Analyze*). In *Chapter 7* follows the visualizations of the underlying data using Apache Superset (Figure 1.1, *Present Results*).

- *Chapter 8* presents the findings of this thesis on a technical basis in achieved during the development process in Section 8.1. Section 8.2 describes the data baseline (that is what data to expect) and Section 8.3 illustrates the analysis of the underlying data. Followed by Section 8.4 presenting an expert-based evaluation of the visualizations of Chapter 7. Section 8.5 concludes this chapter with a discussion of the results and the limitations of this work.

- Finally, *Chapter 9* summarizes the findings and insights gathered during the research process and provides an outlook on potential future study in this area.

CHAPTER 2

Foundations

On the pages that follow, an overview of the foundations is provided and crucial definitions for understanding this thesis' concept and purpose are presented.

2.1 Domain Concepts

(Software) Portfolio. According to Turner [117], projects which form a group and being managed in a coordinated way for added benefit can be described as portfolio. Elonen and Artto [39] and Simon et al. [105] extended that concept and defined a software portfolio as a collection of individual projects competing for resources. In 2021 Genfer et al. [53] extended that definition to include a quality management aspect. That is that software portfolios are also meant to achieve a certain quality standard and process. In accordance with the definitions outlined above, the term *student project portfolio* is used in this thesis as defined in Definition 2.1.

Definition 2.1 *A student project portfolio describes a group of software projects of a software engineering course within one term. All projects within the portfolio may be individual or group projects and are subject to a common quality standard, which ensures that each project meets a certain level of excellence and meets the course's objectives.*

The classic objectives of portfolio management are, according to Cooper et al. [31]: value maximization, tying to strategy, and balancing. These objectives can also be mapped to teaching strategies:

- *Value maximization* can be turned into: maximizing the number of successful students
- *Tying to strategy* can be compared to: improve the teaching quality
- *Balancing* can be mapped to: balance the expected effort of the class with the student's real effort

Version Control System (VCS). Version control, commonly referred to as source control, is the process of tracking and managing modifications to software code. Version Control Systems (or also called revision control systems) are software tools which assist software development teams in managing source code modifications over time [127]. It is common practice for software engineers to make ongoing modifications to files and code, including the addition and deletion of features, during the software development process. Before creating the final edition, it is anticipated that there will be a number of adjustments. Due to larger and more complex systems, there are increasingly more revisions, making it challenging to manage and organize the codes and files. Consequently, the VCSs presence is helpful to speed up and streamline the software development process [127].

Software Repository. The central place to keep (software development) resources that users can pull, track, watch and change when necessary. A software repository is the core element of a VCS [11, 85]. In the context of this thesis, a software repository refers to the storage location of a software project including all relevant data like ITS or VCS.

Git. Spinellis [110] describes Git as a distributed revision control system that is accessible via a free software license on all popular development platforms. Git elevates the software's changes to the status of first-class citizens, which is a key distinction between it and its ancestors. Software revisions are extremely important to developers, which is why Git provides each developer with a full private copy of the software repository and a variety of techniques to handle changes.

Mining Software Repository (MSR). The field of Mining Software Repositories (MSR) targets the analysis of VCS's repositories underlying data. The data examination allows researchers to empirically investigate and uncover insights from software engineering [109].

Issue Tracking System (ITS). An Issue Tracking System is used to manage and keep track of bug reports and feature requests — the *issues*. They are typically utilized in collaborative situations, particularly in large or distributed collaborations. These systems frequently integrate resource allocation, time accounting, priority management, and supervisory processes while also being integrated into a VCS [8, 9].

Extract, Transform, Load (ETL). Extract, Transform, Load (ETL) stands for the process of extracting, transforming and loading data from multiple, disparate sources into a single, unified data repository [38, 119].

Business Intelligence (BI). Negash and Gray [88] describe Business Intelligence (BI) as a data-driven system that integrates data collection, storage, and knowledge management with analysis to offer input to decision-making. The term was first introduced by Luhn [81] in 1958 in his article *A Business Intelligence System*. Business intelligence

is concerned with the analysis of huge quantities of data, often about a company and its operations, in order to enable knowledge workers such as executives, managers, and analysts to make smart decisions [19, 26, 88]. Business intelligence in computer-based systems employs a big database as its source of information and the foundation for advanced analysis. Analyses span from simple reporting to slice-and-dice, drill down, ad hoc query response, real-time analysis, and forecasting [88].

Pseudonymization. The European Union defined the term of *Pseudonymization* in 2018's General Data Protection Regulation (GDPR), Article 4(5) as the process of administering personal data in such a way that it cannot be linked to a specific person without additional information. In addition, it is important to keep this extra information separate [52]. With the help of this technology, businesses may analyze personal data, while at the same time respecting each individual's right to privacy [66].

Pseudonymization vs. Anonymization. In contrast to Pseudonymization, according to Hintze and El Emam [66], Anonymization is an even stronger form of de-identification. When data is anonymized, the sensitive data elements are replaced with unrelated ones. Which then results in data which cannot be used to re-identify the original data, or can only be attributed to an identified or identifiable natural person with a considerable effort in terms of time, cost and personnel [98]. In summary, pseudonymous data is where identifying information has been replaced with a consistent, reversible value, whereas anonymous data cannot be directly linked to an individual.

SQL. Initially introduced by Chamberlin and Boyce [21] at IBM[5], the Structured Query Language (SQL) is a domain-specific programming language developed to be used in relational database management systems (RDBMSs) for data management. The foundation of SQL is the relational model whose development dates back to Edgar F. Codd and his paper "A Relational Model of Data for Large Shared Data Banks" [29] published in 1970. In 2012 Chamberlin [20] reflected about the early history of SQL, saying that initially the language was supposed to be just a simple language to "walk up and use [...]" [20]. It became, however, the most widely used database language today [22, 47]. As of October 2022, RDBMSs also dominate the ranking of all database management systems, according to the db-engines.com[6] website.

REST. Fielding described the term Representational state transfer (REST) in his dissertation as a coordinated set of architectural styles that tries to reduce latency and network communication while making sure that component implementations are as independent as possible and can scale as needed. In his work he discusses the design of the Web in which each element (a so-called resource) can be named by a unique identifier, the resource identifier or a Uniform Resource Identifier (URI) [46]. The following example

[5] https://ibm.com, Accessed: *23.01.2023*
[6] https://db-engines.com/de/ranking, Accessed: *23.01.2023*

by Richards [99] which illustrates the behavior of REST assists in understanding the concept: When a browser requests a resource, a server sends the browser the Web page's current state to the browser. The representation can then be rendered by the browser at that point. Simply put, the server renders the data after sending it along with the requested page's current state of the data. A state transition occurs when a user clicks on one of the links on the displayed page because the browser has to render the next page. Which could be thought of as another state of the application [99].

CRUD. Create, Read, Update, and Delete (CRUD) describes four basic HTTP (Hypertext Transfer Protocol) operations used in a software application. Users must be able to create data, (visually) access it in the UI, update or edit it, and delete it. The three components that make up CRUD programs are an Application Programming Interface (API) (or server), a database, and an UI[7].

Course Concepts

As this thesis is about a course at the TU Wien, its concept, structure and terminology are described in more detail.

European Credit Transfer and Accumulation System (ECTS). The European Credit Transfer and Accumulation System (ECTS) is used in the European Higher Education Area to make studies and courses more transparent. The idea is to define a uniform representation of learning based on established learning outcomes and the workload associated with those outcomes. One ECTS credit represents 25 hours of work by a student[8] [36].

The course is classified as a six ECTS class, which is equivalent to an effort of 150 hours for students. During a term, the class is split into two main phases: a) Individual Phase (IP) and b) Group Phase (GP).

Individual Phase. During the Individual Phase (IP) students work on a programming assignment on their own for about three to four weeks, with an estimated effort of 40 hours. The assignment is a scenario of a CRUD application, having a SQL database layer with three tables, a REST layer for communication and a frontend. These components follow the two-tier architecture, which involves a backend system to handle data storage and processing, and a frontend interface for users to interact with the backend functionality.

[7]Source: https://www.freecodecamp.org/news/crud-operations-explained/, Accessed: *23.01.2023*
[8]Source: https://www.bmbwf.gv.at/Themen/HS-Uni/Studium/Anerkennung/ECTS-System.html, Accessed: *23.01.2023*

1. **Backend**: The backend code for the assignment must be written in Java[9] using Spring Boot[10] and H2[11], an in-memory database.
2. **Frontend**: The frontend, on the other hand, has to be written exclusively using Angular[12]. The use of other external libraries is strictly forbidden.

To progress to the next stage, the so-called Group Phase, students must achieve a minimum score of 40 out of 80 points on the Individual Phase.

Group Phase. During the Group Phase (GP), students form a group of typically five to six people and work on a group assignment. This phase lasts about ten to eleven weeks (that is, until the end of the semester) with an estimated effort of 110 hours per student. Compared to the IP, the technology stack does not change, however, groups are more free to switch certain elements, as, for instance, using a different database system. During the Group Phase, each group works on their project, which is more complex and, thus, more time-consuming than the project of the Individual Phase.

Grading. To better understand the evaluations in Chapter 8, the way the final grade is determined, is described in more detail.

Firstly, at the very beginning of the class, students need to take an online Moodle exam (a so-called *entry test*), where they can achieve a maximum of ten points.

Secondly, there are also some restrictions for the Individual Phase. Students need to achieve at least 40 out of 80 points and additionally, at least 45 points in combination with the entry test.

After passing the IP in the Group Phase students are graded with standard Austrian grades ranging from 1 (Excellent) to 5 (Not sufficient). The points of the IP are converted into Austrian grades using the *standard grade distribution*, as shown in Table 2.1. At the end, the final grade is determined by weighting the grades of the IP by 0.25 and the GP by 0.75. A more detailed description is later provided in Chapter 8.

Grade	Verbal	Definition
1	Excellent	if $x \geq 87.5\%$
2	Good	if $75.0\% \leq x < 87.5\%$
3	Satisfactory	if $62.5\% \leq x < 75.0\%$
4	Pass	if $50.0\% \leq x < 62.5\%$
5	Fail/Not Sufficient	if $0\% \leq x < 50\%$

Table 2.1: Standard grading scheme

[9] https://www.java.com/, Accessed: *23.01.2023*
[10] https://spring.io/projects/spring-boot, Accessed: *23.01.2023*
[11] https://www.h2database.com/, Accessed: *23.01.2023*
[12] https://angular.io/, Accessed: *23.01.2023*

2.2 Data Visualization

In 1999 Card et al. [18] noted that data visualization is used to raise cognition of abstract data. The task of data visualization, the practice of visual data presentation to assist reader comprehension, has gained in popularity for decades. The visualization of data comes in where useful information of often enormous datasets is extracted. The result then may be shown graphically as an image, chart, or graph. This method increases the usefulness of common data sources, such as Excel spreadsheets, by enabling users to identify trends and patterns that are not easily apparent in a column of numbers [34]. As Figure 2.1 illustrates, crucial insights can be gained by asking various types of questions related to the visualization. Telea [115] defined that the term *insight* (of data) is used to define two types of information:

a) **Answers to problem-specific questions**: Problem-specific questions regard certain phenomena, procedures or datasets. The objective of visualization is to efficiently and effectively answer these questions. A problem-specific question can, for example, be about the minimum, maximum, or outliers of values or about the distribution of values within a dataset.

b) **Unknown facts concerning a problem**: The idea of this type is to look at and examine the data and, for example, compare it to similar data or datasets of the past to gain knowledge about the dataset.

Figure 2.1: Questions that are targeted by the visualization procedure [115]

Furthermore, the field of data visualization can be divided into two areas: scientific visualization (scivis) and information visualization (infovis) [18, 115]. While scivis is concerned with realistic representations of the world, infovis focuses on the representation of notions of an often abstract character [87].

In the 1980s the term of scientific visualization emerged, as a response to the growing amount of computer generated data [18, 115]. Visualizations can lie within that field if their primary concern is to realistically visualize some kind of phenomena. In addition, a time component might also be part of such a visualization [50].

In contrast to scivis, infovis's aim is to enable data analysts developing internal mental models of the information contained in datasets. This may then be utilized for characterization, prediction and/or decision-making [87]. The practice of visualizing data aims to accomplish the following three primary objectives [75, 87]:

a) **Exploratory Analysis**: At the beginning, one has no hypothesis about the viewed data. In that case, an exploratory analysis aims to potentially find useful information.

b) **Confirmatory Analysis**: The starting point of this analysis are one or more hypotheses. The visualization then either confirms or rejects these hypotheses.

c) **Presentation**: The presented facts are fixed beforehand, while the most suitable presenting method primarily depends on the user.

In 2020 Midway [84] proposed the following ten principles for creating data visualizations:

1. Diagram First
2. Use the Right Software
3. Use an Effective Geometry and Show Data
4. Colors Always Mean Something
5. Include Uncertainty
6. Panel, when Possible (Small Multiples)
7. Data and Models Are Different Things
8. Simple Visuals, Detailed Captions
9. Consider an Infographic
10. Get an opinion

2.3 Statistics

The *Model and Analyze* stage of the Data Science Road Map in Figure 1.1 requires fundamental understanding of statistical methods. In the following, the required concepts are explained in more detail.

Population versus Sample. Härdle et al. [68] describe the difference between a sample and a population in the following way: A *population* is the set of all elements that are of relevance for statistical research. The population size, N, simply is the number of items in the population. A population can be finite or infinite in size and can also be artificial. A *sample* is any finite subset of observations gathered from the population. The sample size, represented by n, is the number of elements in a sample [68]. Figure 2.2 illustrates the relation between these two terms.

[13] Source: https://www.omniconvert.com/what-is/sample-size/, Accessed: *23.01.2023*

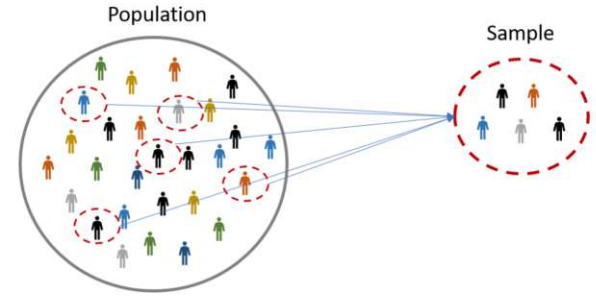


Figure 2.2: Population vs. Sample[13]

Normal Distribution. According to Bethea [10], the normal distribution — also known as the bell-shaped error curve — is the most frequent continuous distribution. The underlying assumption of the normal distribution is that contained errors in experimental observations are a product of variation and numerous independent causes, each only creating a small disturbance in the overall distribution. Based on the Central Limit Theorem (CLT), the distribution of a sample mean $\bar{x}$ approaches the normal distribution with variance $\frac{\sigma^2}{n}$ and population mean μ as the sample size of n grows [10]. The probability density function $f(x)$ for the normal distribution is shown by Equation 2.1

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \quad \text{for } -\infty < x < \infty \tag{2.1}$$

In addition, when drawn as a plot, function $f(x)$ looks as made apparent in Figure 2.3. The bell curve is drawn with a mean $\mu = 0$, within $\pm 1\sigma$ lie 68.2% of all data points, within $\pm 2\sigma$ are 95% and within $\pm 3\sigma$ are 99.7%. This is the so-called *68–95–99.7 rule*.

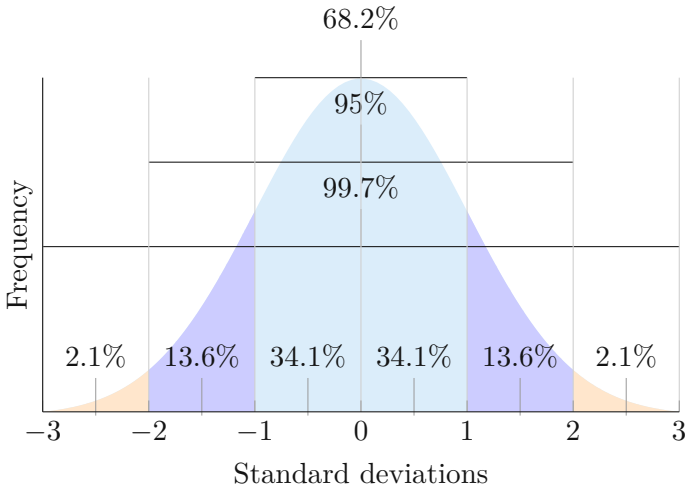


Figure 2.3: Normal Distribution[14]

2.3.1 Measures of Location

According to Chen [28], a basic step in data exploration is to get an estimate of a typical value of a variable. However, when measured or counted, variables can have thousands of different values. The idea of location estimates is to get an idea of where most of the data is placed [28].

Mean. The mean — also known as average value, simple mean or the arithmetic mean — is the sum of all values divided by the number of values. The symbol $\bar{x}$ represents the mean of a sample from a population, whereas the mean from a population is denoted by μ. To calculate it for a set of n-values the formula of Equation 2.2 is used:

$$\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.2}$$

It is important to distinct between a lowercase n and uppercase N. If capitalized, it refers to a population, whereas the lowercase version regards only a sample from a population [28]. For the population mean μ, the following formula of Equation 2.3 is used:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.3}$$

Median. The median — also known as 50% percentile or 2^{nd} quantile — is the value where half of the data is above and half of the data is below. Equation 2.4 illustrates how that value can be found:

$$\tilde{x} = \begin{cases} x_{\frac{n+1}{2}}, & \text{if } n \text{ is } \textit{odd} \\ \frac{1}{2}(x_{\frac{n}{2}} + x_{(\frac{n}{2}+1)}), & \text{if } n \text{ is } \textit{even} \end{cases} \tag{2.4}$$

The advantage of using the median instead of the mean is that the former is not skewed by a small number of very big or very small values. Thus, it gives a better picture of a *typical* value [17, 28]. As there is no standard way of symbolizing the median, in this thesis the symbol are used as defined in Definition 2.2.

Definition 2.2 *The symbol of $\tilde{x}$ refers to the sample median and $\tilde{\mu}$ refers to the population median.*

Mode. The mode, denoted by $\bar{m}$ and defined as the most often occurring value in a sample dataset, is a third measure of location [10, 68]. If the variable is discrete, the mode simply is the highest frequency value. However, according to Härdle et al. [68], with continuous data recorded with appropriate precision, most observations are likely to be distinct. Hence, the concept of the *most frequent value* is rendered worthless.

[14] Source: https://johncanning.net/wp/?p=1202, Accessed: *23.01.2023*

2.3.2 Measures of Position

The relative position of a value within an ordered dataset is called the measurement of position. In this thesis, only the concept of quantiles is considered, in the following it is explained in more detail.

Quantile. Quantiles — also known as percentiles [28] — are a proportional representations of the entire number of observations. Typically, quantiles are named according to the number of intervals into which the range is divided [90]. Common quantiles have particular names, such as quartiles (four groups), deciles (ten groups) and percentiles (100 groups). To calculate any kind of quantile, it is important to sort the data in ascending order.

Interquartile range (IQR). The middle 50% range of the data is called the Interquartile range (IQR). It is calculated by subtracting the third quartile (Q_3) from the first quartile (Q_1) of a set of data. The formulas for calculating the positions of Q_1 and Q_3 within an ascending ordered dataset are presented in Equation 2.5 and 2.6. For example, if Q_1 is 5, then the fifth data point within the dataset is the value of Q_1.

$$Q_1 = \frac{n+1}{4} \tag{2.5}$$

$$Q_3 = \frac{3(n+1)}{4} \tag{2.6}$$

2.3.3 Measures of Variability

After determining the location of the data using statistics such as the mean, median and mode, the next important aspect of the data is the spread (or variability) around these values [10].

Variance. According to Bethea [10], the *sample variance* s^2 is the most popular method to report variability. It is also known as mean-squared-error [28, 68]. Equation 2.7 shows the formula to calculate the sample variance s^2 for n values, where $\bar{x}$ is the previously defined sample mean.

$$s_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}_n)^2 \tag{2.7}$$

As the sample variance s^2 is merely an estimation of the true population variance σ^2, as to Bethea [10], a similar formula can be used when the variance of a population should be derived. Similar to the mean, different symbols are used, as shown in Equation 2.8, where μ is the population mean.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{2.8}$$

Standard deviation. A (sample's) standard deviation is defined as the positive square root of the variance [10]. For the sample variance the formula of Equation 2.9 is used, for the population variance Equation 2.10. Since it is on the same scale as the original data, the standard deviation is considerably easier to read than the variance [28].

$$s = \sqrt{s^2} \tag{2.9}$$

$$\sigma = \sqrt{\sigma^2} \tag{2.10}$$

2.3.4 Measures of Shape

Characterizing the location and variability of a data collection is a fundamental task in many statistical analyses. Skewness and kurtosis are two basic characteristics of the data [112].

Skewness. Skewness is a measure of symmetry, or more precisely, the absence thereof. A symmetric distribution is when both sides of the mean are mirror images [90, 112]. Three kinds of skewness can apply to a distribution: Right (or positive), left (or negative), or zero skewness. Figure 2.4 illustrates a normal distribution when skewed in any of the three ways. The right side of a right-skewed distribution is longer than its left side, and vice versa for a left-skewed distribution. Equation 2.11 shows the formula to calculate the skewness of a given *sample*.

$$g_1 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s}\right)^3 \tag{2.11}$$

A simpler calculation can be used, according to Gurker [59], as in Equation 2.12.

$$g_1^{(2)} = \frac{Q_3 - 2Q_2 + Q_1}{Q_3 - Q_1} \tag{2.12}$$

The following listing explains how to interpret the resulting value of g_1.

1. $g_1 > 0$: right-skewed (positive skewness)
2. $g_1 \approx 0$: symmetric
3. $g_1 < 0$: left-skewed (negative skewness)

However, Gurker [59] mentions that there exist more formulas and measurements to determine the skewness. Equation 2.11 is also more difficult to interpret when used with a small sample size compared to Equation 2.12.

[15] Source: https://codeburst.io/2-important-statistics-terms-you-need-to-know-in-data-science-skewness-\and-kurtosis-388fef94eeaa. Accessed: *23.01.2023*

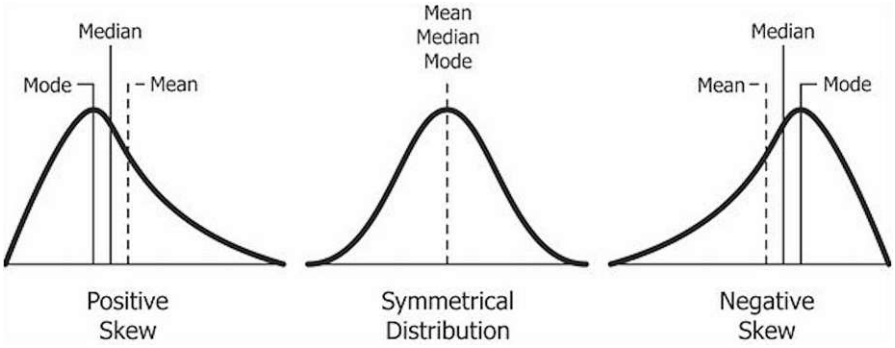


Figure 2.4: Example of Skewness[15]

Kurtosis. Kurtosis determines whether the data is heavy-tailed or light-tailed in comparison to a normal distribution. That is, datasets with a high kurtosis have a large fraction of outliers, with a low kurtosis lack outliers [112]. Nisbet et al. [90] describe kurtosis as the degree to which the data for a variable is grouped closely around the mean. Equation 2.13 shows the formula to calculate the skewness of a given *sample*.

$$g_2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s}\right)^4 \tag{2.13}$$

The resulting coefficient g_2 can then be categorized as follows:

1. $g_2 < 3$: Platykurtic (flat arched)
2. $g_2 \approx 3$: Mesokurtic (medium arched)
3. $g_2 > 3$: Leptokurtic (steep peaked)

Figure 2.5 illustrates how kurtosis influences the trace of a bell curve.

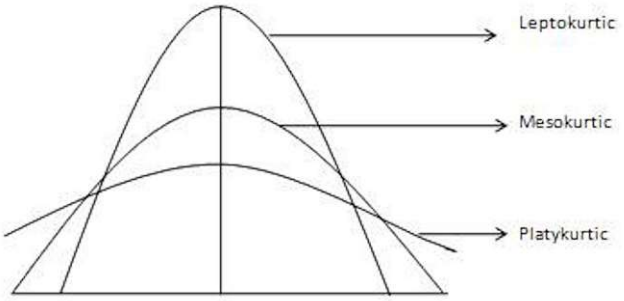


Figure 2.5: Example of Kurtosis[15]

CHAPTER 3

State of the Art

This chapter presents the academic state of research in the areas of Mining Software Repositories (MSR), Data Visualization and Software Engineering Education. After outlining the current state of research and discussing how the thesis is distinguished from existing research, this chapter concludes with a description of currently available and appropriate tools and their capabilities for visualizing datasets.

3.1 Current State of Research

The present thesis' topic essentially overlaps with the following three research fields of software engineering: (a) Mining Software Repositories (MSR) (b) Data Visualization (c) Software Engineering Education. In the following, the current research state in each of these fields is outlined.

3.1.1 Mining Software Repositories

Data mining, Mining Software Repositories (MSR) and knowledge discovery have already been studied extensively. Computing power, however, has advanced for the last 20 years and so rose the interest in these fields for mining large datasets [65]. In 1996 Fayyad et al. [42] noted the difficulty of finding patterns in genuine, innovative or at least potentially helpful data. More precisely, according to Spadini et al. [109], extracting information from Git repositories is a non-trivial task. As demonstrated in previous work, this requires the analysis of huge amounts of data originating from many sources, including source code, version control systems, and issue tracking systems [25, 108, 126]. Adapting these tools can be a genuine and valuable aid to researchers throughout their development operations [89].

MSR is a new field which focuses on gaining basic and useful information about characteristics from different, already existing repositories [76]. The extraction of data from

different contributors to a software project which aims to detect unknown facts gained popularity in recent years [62, 125]. More so since the application of MSR techniques include the prediction of bugs, as already studied by Vandecruys et al. [118] in 2008. In this area, Zaidman et al. [126], for example, analyzed the co-evolution of production and test code side by side. Hassan and Xie [64], on the other hand, proposed the concept of Software Intelligence (SI). For software professionals, it provides relevant information to help them make decisions every day [64].

Kalliamvakou et al. [72] mentioned, source code repositories are today typically hosted on platforms such as GitHub and GitLab. These platforms offer numerous other features and integrations and, thus, a great deal of additional information may be extracted through the process of data mining. However, there are a number of new possible problems associated with examining this newly accessible data [72]. For example, Bachmann et al. [5] found that a set of bugs, tracked in a bug tracking system, can easily be biased since not all kind of bugs might be collected. The aforementioned biases may undermine the validity and generalizability of research that use source code repository datasets [72].

According to Nguyen et al. [89], although there exist free tools to mine software repositories, as for example SonarCloud[16], BlackDuck[17] or CodeSense[18], these platforms delete analyses in short periods in order to save disk space. For this reason, Nguyen et al. proposed an analytics and visualization tool, based on Apache Superset, called PANDORA[19]. This tool is capable of automatically and continually mining data from different tools and platforms and of visualizing its analysis results afterwards [89].

3.1.2 Data Visualization

Visualization is not only important for providing information, but also helps analysts understand their data in greater detail [57].

The study of Kuipers and Visser [77] revealed that software portfolios can contain a great number of projects with millions of lines of source code. The large number of source code lines makes manual collection and analysis of portfolio data impractical due to the large amount of time required. In order to collect sufficient portfolio metrics for further analysis and visualization, data mining of software portfolios requires the creation of a highly automated procedure.

As also identified by Kuipers and Visser [77], data visualization is an important part of software portfolio management and, thus, has gotten some attention in research. Grant [57] and Dumbach et al. [37] provide a broad overview of several data visualization strategies. The identified fundamental concept of visually exploring data is to offer insights into data, to draw conclusions and to allow direct interaction with data [74]. The

[16] https://www.sonarsource.com/products/sonarcloud/, Accessed: *23.01.2023*
[17] https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html, Accessed: *23.01.2023*
[18] https://codescene.com/, Accessed: *23.01.2023*
[19] http://sqa.rd.tuni.fi/superset/dashboard/1/, Accessed: *23.01.2023*

idea is that the flexibility, creativity, and general knowledge combined with the storage capacity of computers are more likely to lead to success when people are involved in the process of data exploration. According to Keim [74], an advantage of visually exploring data are better results, in particular where algorithms fail. This process can also be understood as hypothesis development, enabling the user to acquire insights into the data and generate new ideas. The verification afterwards can be done applying automated approaches from statistics or machine learning.

3.1.3 Software Engineering Education

In an educational context, information mining based on student's Git repositories was examined by Buffardi [16]. The setting used was similar to the class examined in this thesis, as students used Git and User Stories for their group projects. Based on the group project's findings Buffardi concludes that a more objective contribution of evidence can be derived from software artifacts. However, it is also explicitly stated that this allows students to manipulate metrics. Hence, traditional assessment may be supported by such information.

Parizi et al. [93] called the use of Git in software development a "gold mine of resources" not yet used to its full potential. Particularly with regard to analyses and findings for educational purposes. Chen et al. [27] and Parizi et al. [93] pointed out, teamwork is one of the most crucial parts of software engineering. However, collaboration is still some sort of black-box in the software engineering education field. Similarly to Buffardi, Parizi et al. [93] pointed out that the most difficult task is to identify and demarcate the individual work to a single member within a team. Besides that, Parizi et al. also emphasized the aspects of manipulation and bias when using tools. In particular, due to the lack of suitable performance tools which can freely be used [51, 61, 93, 106]. However, since only quantitative metrics are considered in [93], the proposed "performance evaluation metrics" are not capable of capturing the difficulty of the task solved by a team member. Hence, to accurately assess student's performance, it is essential to treat the tool-provided numbers as an extended information base [32, 93]. Otherwise, in 2006 Patton and McGill [94] added educational metrics to student portfolios and software quality metrics. They added that such portfolios provide a quantitative rather than qualitative comparative assessment of submission based on these metrics.

Teamwork skills in software engineering, often for international and interdisciplinary teams, call for more and new requirements. Specifically for the development of software applications as the field of software engineering continues to expand [27, 54, 107]. Challenges, as noted byChen et al. [27], are manifold. On the on hand, for example, students need to learn how to collaborate as a team while being in different roles. On the other hand, they need to learn how to collaborate as a team in order to address challenges in software development. The development of skills in software engineering education is crucial, as the industrial sector also demands the training of specialists in basic and vital professional skills [14, 27, 102].

For a four-year period Wills [124] studied a course on how to introduce a group based project activity in an undergraduate computer science setting at the Worcester Polytechnic Institute (Worcester, Massachusetts, United States). In Wills's setting, group programming projects were introduced since he argues that the difficulty of single student projects does not cover the needs in a real software engineering project. He found that, in a large introductory class, it is beneficial for everyone involved — students as well as lecturers and others — to conduct group programming exercises as compared to individual ones.

Throughout their education, computer science students generally engage in several collaborative projects. The group assignment design is motivated by a number of factors. These include that forming groups helps courses scale to the increasing number of students enrolled in computer science. While, at the same time, students are provided the opportunity to work on software projects that are larger in scope than individual course projects [2]. In terms of group performance, learning, cooperation patterns, and member responsibility, the structure of group projects has a significant impact on the students' success [70]. The influence of grading on students' group work experience was shown in the past [30]. In particular, task-grade inequality has a negative impact on student's attitudes toward group work [24]. Conversely, it has been shown that communication and collaborative abilities are areas where computer science graduates commonly fail to meet corporate standards and expectations [2, 33, 96].

The work of Le et al. [78] targets the problem that individual assessment of student achievement might be difficult, specifically if not all members are contributing to the project. Moreover, in an academic setting a supervisor may be responsible for a large number of software engineering teams and also has to monitor the development and performance of each individual student throughout the semester. Thus, Le et al. suggest to provide a "weekly formative feedback" for both, teams as well as supervisors. The idea of this generated feedback to assist in evaluating and supporting the team throughout the entire term. Based on data mining techniques, the feedback is created by analyzing the text content of the student's work [78].

Pérez and Rubio [95] examined a software engineering class with a setup similar to the one studied in this thesis. Each group is formed randomly by a teacher and consists of nine students. These groups then work on a software development project made up of different phases. The result of each phase in turn serves as input for the next phase. During one of these phases groups either work in a conventional way or in a Project Based Learning (PBL) way. The strategy of PBL is to actively incorporate students' experiences into the class to improve their academic performance as well as their educational skills and learning experiences. Their study showed that groups which followed the PBL approach improved their academic performance and, in addition, the experience of learning and improving skills was positively evaluated.

3.2 Available Tools

There already exist numerous commercial and non-commercial tools which can be used to analyze the quality of any codebase. In the following, the first three tools (MetricMiner, gitinspector and TEAMSCOPE) are more academic and thus, appropriate in a university context, whereas Section 3.2.2 only covers tools which specifically target companies. The latter tools particularly target the visualization needs in an economic environment for software and Business Intelligence (BI).

3.2.1 Education-related Tools

Education-related tools are specifically designed for the needs of software engineering education classes. The following tools aim to support the lecturers in inspecting and grading students. However, Bogarín et al. [13] underlined that, generally speaking, there do not exist many tools which are capable of supporting specialists in an educational context. In software engineering courses, in particular, is not a single specialized framework which can be used without special knowledge [37].

MetricMiner

MetricMiner[20] [108], developed by Sokol et al., is a Java framework for mining software repositories which includes functionality for automated cloning and metric extraction from Git and SVN repositories. MetricMiner also presents data in a dashboard and facilitates doing statistical analysis by the use of R scripts that are automatically developed and performed [89, 108].

Gitinspector

Gitinspector[21] is a tool for inspecting Git repositories which supports lecturers in grading their students based on Git statistics. Along with a timeline showing the effort and activity of each author, it visualizes the data of each contributor. Gitinspector was initially created at Chalmers University of Technology and Gothenburg University to gather data for student project statistics. The list of source file extensions can be customized, however, by default only source files are included in the statistics. Multiple languages are available, the results can be viewed as HTML (Hypertext Markup Language), JSON, XML (Extensible Markup Language), or plain text. Figure 3.1 shows a screenshot of the HTML output of gitinspector.

[20] https://github.com/Woutrrr/metricminer2, Accessed: *23.01.2023*
[21] https://github.com/ejwa/gitinspector, Accessed: *23.01.2023*
[22] Source: https://github.com/ejwa/gitinspector, Accessed: *23.01.2023*

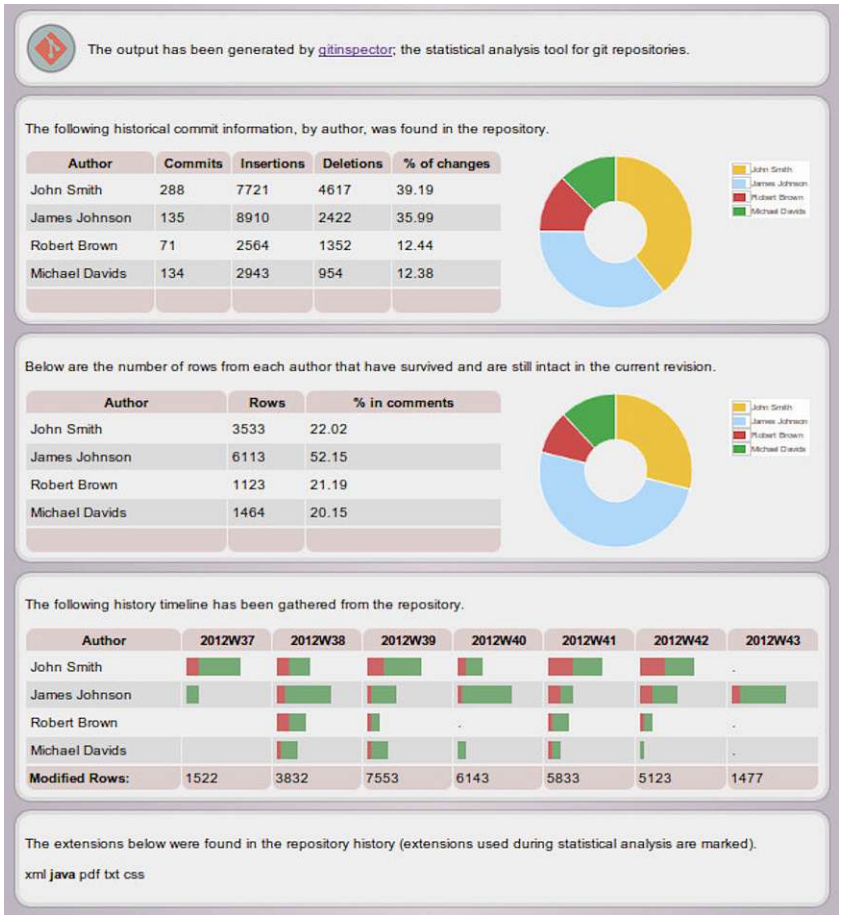


Figure 3.1: gitinspector Screenshot[22]

TEAMSCOPE

TEAMSCOPE is a prototype developed by Ju and Fox [71] helping lecturers in a project-based learning course. The system is able to monitor how successfully the team performs certain procedures and results. Using the platform specific API of a team collaboration technology, such as GitHub, teamwork telemetry can be collected from an arbitrary number of team-based student projects. The developed prototype then computes and visually provides numerous measures of interest based on collaborative telemetry.

3.2.2 Data Visualization Tools

This section focuses only on tools which are designed to be used for data visualization or BI use cases in the industry. The capability, handling, and cost range of these tools vary widely depending on their specific use case. In the following, four different tools — which were carefully examined for suitability — are described in more detail: Apache Superset, OpenSearch, Microsoft Power BI and Plotly.

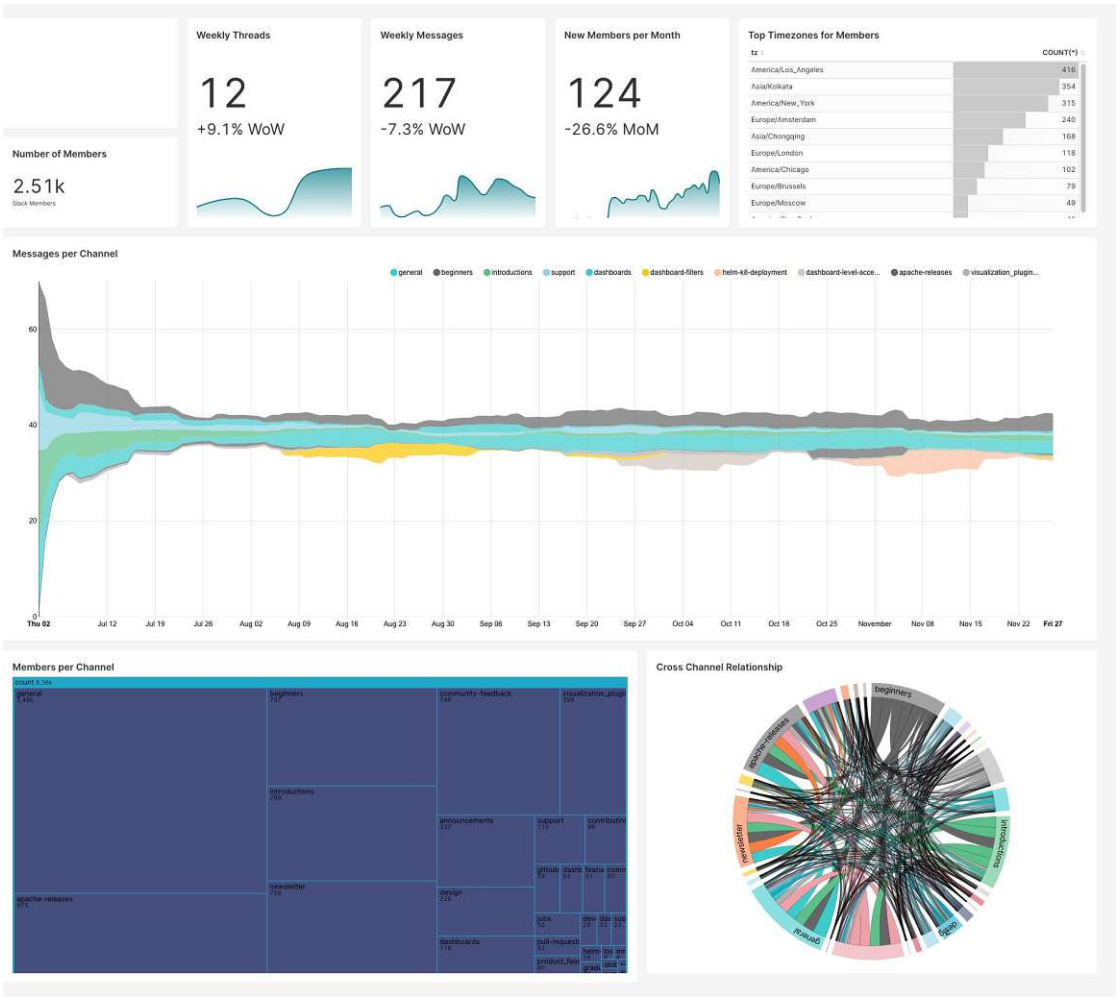


Figure 3.2: Apache Superset Slack example dashboard

Apache Superset

Apache Superset[23] is a powerful, code-free business intelligence tool for the visualization of datasets and the creation of interactive dashboards. Founded by *Maxime Beauchemin* at Airbnb[24] it is now hosted by the Apache Software Foundation's GitHub[25]. The tool is written using Flask[26], a Python[27]-based micro web framework. Figure 3.2 shows the example dashboard, which is part of the example data provided by its Docker setup[28].

[23] https://superset.apache.org/, Accessed: *23.01.2023*
[24] https://airbnb.io/projects/superset/, Accessed: *23.01.2023*
[25] https://github.com/apache/superset, Accessed: *23.01.2023*
[26] https://flask.palletsprojects.com/, Accessed: *23.01.2023*
[27] https://www.python.org/, Accessed: *23.01.2023*
[28] https://superset.apache.org/docs/installation/installing-superset-using-docker-compose, Accessed: *23.01.2023*

Interactive graphs (as in Figure 3.2) are made in superset with NVD3[29] (a JavaScript library built on D3.js[30]), deck.gl[31] (for geospatial charts) and Apache ECharts[32]. The latter one will be the default implementation for future chart visualizations[33].

A dashboard within Apache Superset is made of several graphs, called slices, which can be moved, resized and rendered maximized (full-screen). Moreover, data displayed by one graph can be downloaded as an image (jpg-format) or exported as a Comma-separated values (CSV) file. The dashboard as a whole can also be exported as an image [83].

Superset consists of two main interfaces: A SQL-Integrated Development Environment (IDE) called *SQL Lab* and the main interface for preparing and exploring data.

Semantic Layer. An essential concept of Apache Superset is the *semantic layer* which is an abstraction to turn the underlying data to a simplified representation. It is often housed in some form of SQL-speaking database or a data engine. A thin semantic layer's primary objective is to enable data modification for the specific purpose of visualization. Apache Superset provides two kinds of datasets based on this semantic layer: physical and virtual datasets. A physical dataset represents a table or View in a database whose information can automatically be retrieved since a physical dataset reflects a true, physical table (such as schema and column types). A virtual dataset, on the other hand, is a conversion from any free-form SQL query against an existing database. The result set is called a *Virtual Dataset*, behaves exactly like a physical dataset and can be used in Apache Superset to create charts. The concept of a semantic layer and its virtual dataset will later be relevant in Section 6.3.

OpenSearch

OpenSearch[34] is a fork of Elasticsearch 7.10.2 and Kibana 7.10.2, licensed under Apache 2.0. Consisting of a search engine daemon, OpenSearch — similar to the well-known ELK Stack — allows users to create visualizations based on ingested data. The OpenSearch Dashboard provides a UI to search, aggregate, view and analyze data.

The OpenSearch project can be divided into two main components: The *OpenSearch* and the *OpenSearch Dashboards*.

[29] https://nvd3.org/, Accessed: *23.01.2023*
[30] https://d3js.org/, Accessed: *23.01.2023*
[31] https://deck.gl, Accessed: *23.01.2023*
[32] https://echarts.apache.org, Accessed: *23.01.2023*
[33] https://blogs.apache.org/foundation/entry/the-apache-software-foundation-announces71, Accessed: *23.01.2023*
[34] https://opensearch.org/, Accessed: *23.01.2023*

OpenSearch. Based on Apache Lucene[35], the OpenSearch component is a distributed search and analytics engine. It provides full-text searches on added data with features such as searching by field, by multiple indices, boost fields, ranked results (by score) and result aggregation.

Data must be indexed before it can be searched, the resulting structure is appropriately called an index. A JavaScript Object Notation (JSON) document is the fundamental data unit in OpenSearch and uses a unique ID to identify each document within an index.

OpenSearch Dashboards. The second main part is the OpenSearch Dashboards component which is the default visualization tool for all provided data. The dashboard also serves as a general interface for interacting with the OpenSearch service. Compared to Apache Superset, the default available charts are more limited: 16 visualization types are available in OpenSearch whereas in Apache Superset 60 are available. Figure 3.3 shows an example dashboard built by aiven.io.

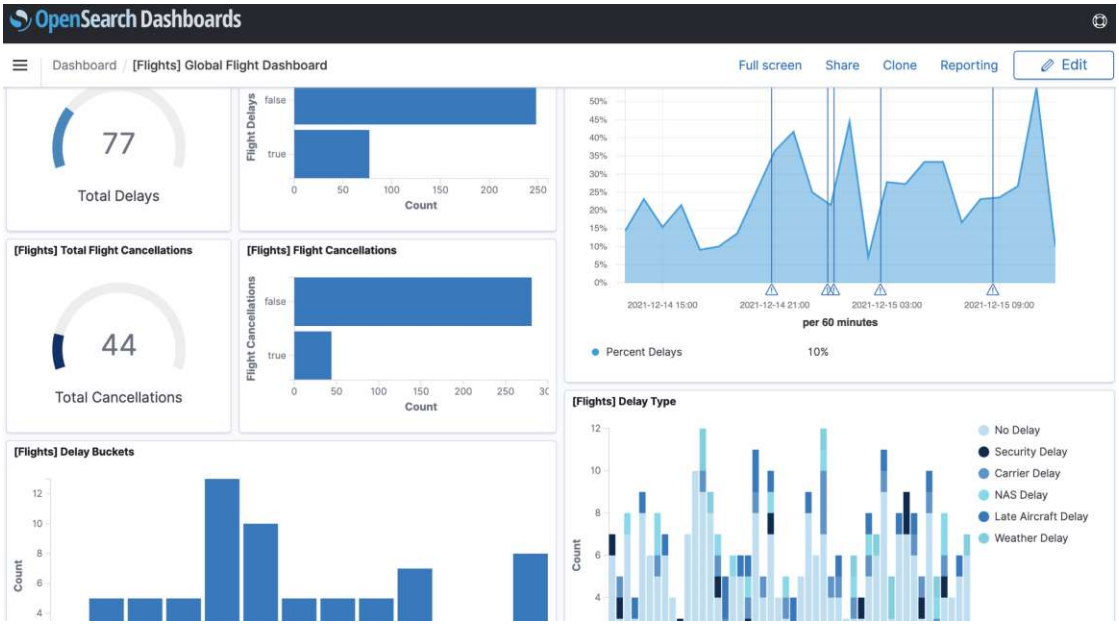


Figure 3.3: OpenSearch example dashboard[36]

Microsoft Power BI

Power BI emerged from the Microsoft Excel add-ins Power Pivot, Power Query, and Power View and may be used in combination with or without Excel and various other datasources [45]. This tool targets businesses transforming their raw data into meaningful

[35] https://lucene.apache.org/, Accessed: *23.01.2023*

[36] Source: https://developer.aiven.io/docs/products/opensearch/dashboards/, Accessed: *23.01.2023*

information that provides deeper insights and supports decision-making. The platform connects to a number of Microsoft native and third-party sources. This allows enterprises to quickly display and interpret their data through interactive, configurable dashboards and reports[37]. Figure 3.4 demonstrates an example of a dashboard created with Microsoft Power BI.

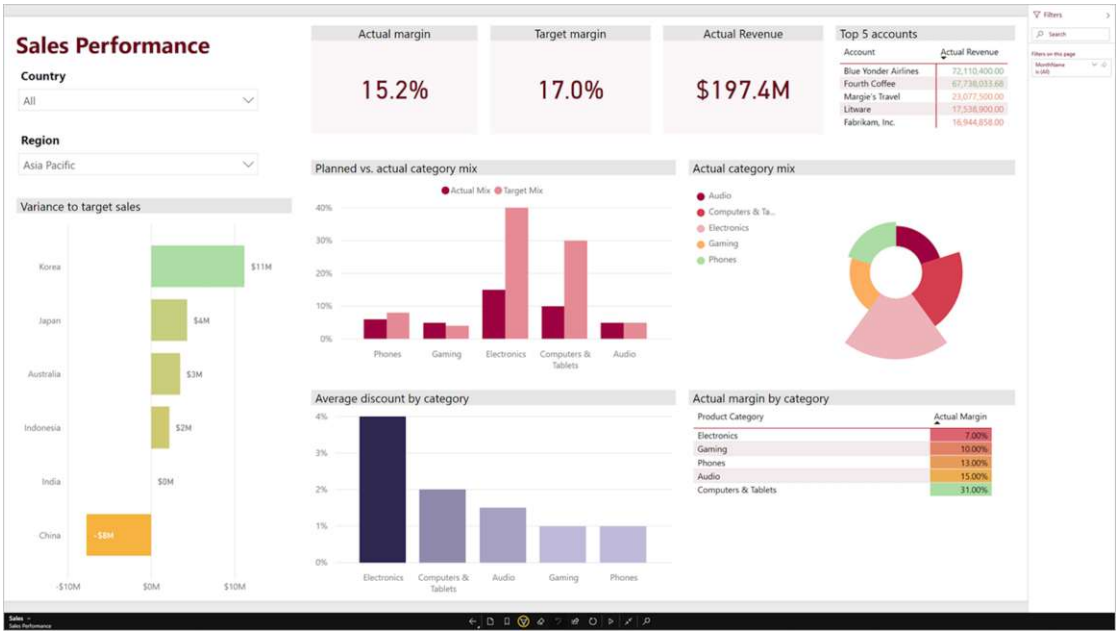


Figure 3.4: Microsoft Power BI example dashboard[38]

Plotly

Plotly — also known as plot.ly — was created using the Python and the Django framework. Its capabilities include data analysis and visualization. It is free for users, but only with restricted functionality; to access all functions, a professional membership is required. As can be derived from Figure 3.5, it generates charts and dashboards online but may also be used offline within Jupyter notebook[39] and pandas[40]. There are several types of charts accessible, such as statistical charts, scientific charts, 3D charts, multiple axes, dashboards and so on. Plotly on premises is a service that, like plot.ly cloud, allows hosting data on one's own private cloud behind your own firewall to ensure full protection of personal information. APIs for Python, R, MATLAB, and Julia are accessible.

[37] Source: https://powerbi.microsoft.com/en-us/why-power-bi/, Accessed: *23.01.2023*
[38] Source: https://docs.microsoft.com/en-us/power-bi/consumer/mobile/mobile-windows-10-app-presentation-mode, Accessed: *23.01.2023*
[39] https://jupyter.org/, Accessed: *23.01.2023*
[40] https://pandas.pydata.org/, Accessed: *23.01.2023*
[41] Source: https://dash.gallery/Portal/, Accessed: *23.01.2023*

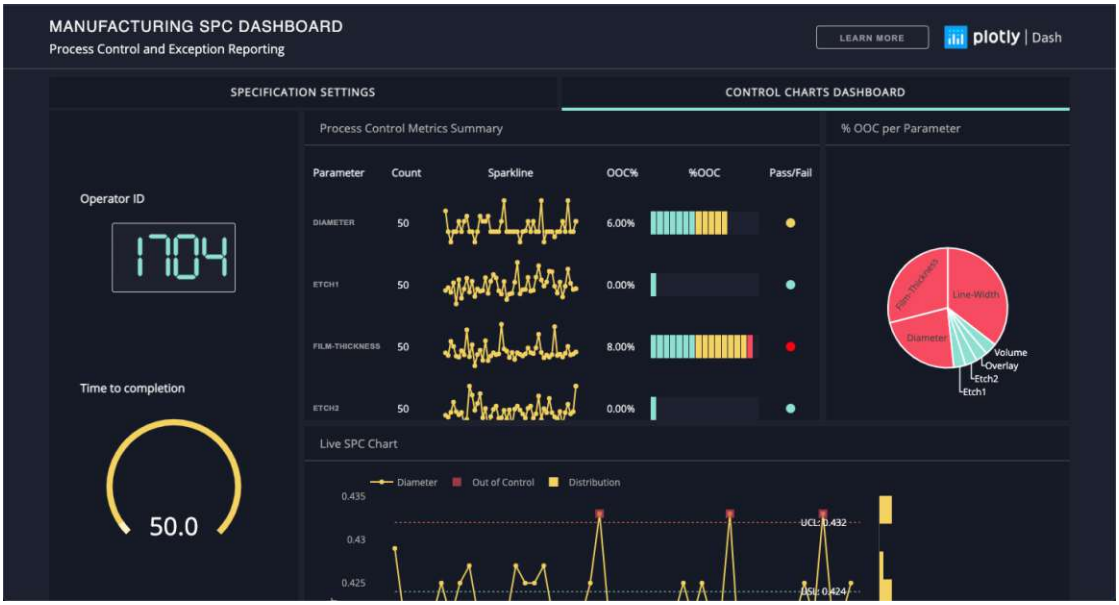


Figure 3.5: Plotly example dashboard[41]

3.3 Distinction from Current Research

Based on the literature review conducted within the scope of this thesis, there is no directly comparable solution to the proposed idea in a software engineering education context. The primary focus of typical tools for investigating student projects is more on how to generate a quality overview of a vast amount of student projects and on how to, consequently, grade them [69, 79].

There are solutions such as Codeboard.io[42] which is designed as an online exercise grading platform to be used in classrooms for the teaching of programming. Codeboard.io serves as an exercise repository and provides an online IDE interface that allows students to address the assignment directly from the web browser [23]. The instructor can upload a problem and test cases to evaluate the student's solution. This also implies that the instructor has to specify a concrete expected result, as students will be assessed automatically on the basis of (unit) tests. However, as the data source of this thesis is not a programming class, students have already acquired at least basic programming skills in preceding lectures. The proposed prototype in this thesis focuses on hard facts of student projects. Whereas, as described in Section 1.1, the scope of the examined class is also to teach soft skills necessary in the software engineering industry. As Devedzic et al. [35] stated, the problem about measuring soft skills is the difficulty of measuring and defining a certain metric Thus, the focus of the prototype will exclusively be the support of lecturers during their grading process with an analysis tool. There are also some critical studies about grading students automatically, as Baniassad et al. [7] show

[42] https://codeboard.io/, Accessed: *23.01.2023*

in *STOP THE (AUTOGRADER) INSANITY: Regression Penalties to Deter Autograder Overreliance*. They refer to the problem that, when students are aware of the autograding mechanism and its expected outcome, their reliance on the tool's feedback increases and their personal, careful reflection decreases.

In the context of a software portfolio there already exists a research-based solution which visualizes numerous software projects side by side [53]. Another solution widely used in the industry is SonarQube[43] which is highly configurable and extensible but does not fulfil the low barrier-to-entry requirement since it is an additional service which needs to be integrated and configured.

A recently published proof of concept by Weiß in his master thesis *A Lightweight and Integrated Software Repository Mining and Visualisation Approach for Software Engineering Education* [122] tries to extract information based on Git in a software engineering education context. However, Weiß's focus is on which information and how it can be extracted, and less on how that information can also be combined with existing project management data of GitLab and its visualization.

[43] https://www.sonarqube.org/, Accessed: *23.01.2023*

CHAPTER 4

Methodology

The methodological focus of this thesis is on the design and implementation of a prototype for a visualization system as well as on the consultation of experts about their information needs and a final evaluation. The prototype is divided into two parts: (a) The creation of a visualization concept for several student groups' software projects and (b) the implementation of these visualizations and its corresponding architecture for mining all necessary information. In a first step, the visualization conception was created. In a second step, the implementation was added, which is also reflected in the thesis's structure. This chapter describes all extra methodological processes and activities which accompanied the prototype's visualization, implementation and assessment.

4.1 Research Questions

Referring to the research questions, as defined in Section 1.3, the following steps are performed to answer the defined research questions. These steps follow the recommended Design Science approach by Wieringa [123].

RQ1. Prior to developing a tailored program to extract the available repository information, an exploratory analysis of archived the data must be done. To identify the needs of experts and to gain a better understanding expert interviews are conducted (Section 5.2 and 5.3). The interviews are followed by the Treatment Design and Implementation (Chapter 6 and 7) to store the data in a persistent datastore.

RQ2. To answer the second research question, a conception (mock-up) for visualizing the data about the group projects is created (Section 5.1). After an initial evaluation of existing tools, the proposed visualization concepts are implemented as far as possible in the context of this thesis (Chapter 7). This prototype will be evaluated by qualitative expert interviews (Section 8.4) to verify the compatibility of the visualization with the

expert's needs. In addition, all mock-ups which the tool is not capable of showing are validated in terms of usability.

RQ3. The prototype intelligence system will be used to derive knowledge about the groups and terms applying visual and statistical analytics (Section 8.3).

4.2 Literature Review

Based on the research questions, a systematic literature research was conducted, with the following two objectives in mind:

- To find out whether there are any studies which already focus on data-driven software engineering education
- To identify papers which serve as a solid basis for this study

The identified fields of research of the present thesis are: *Software Engineering Education*, *Information Needs*, *Data Visualization* and *Mining Software Repositories*. For a broad overview, the theses of Weiß [122] and Genfer et al. [53] served as a starting point for literature research. The search engines Google Scholar[44], IEEE Xplore[45] and the ACM Digital Library[46] were used most often for keyword searches. Siddaway's [103] recommendations for finding the most representative results, synonyms were used for individual terms. These search services have sophisticated capabilities, allowing a refined research, such as for those publications which have several keywords in their abstract.

The *Binocular* publication of Grabner et al. [56] already contained useful references to start with MSR. In addition, simple GitHub search functionality was used to identify possible newer tools for MSR.

4.3 Technology Review

Based on the selected tools in Section 3.2.2, an analysis was conducted to find out how capable they are for the prototype's use case. The functionality was evaluated by simple, small use cases to answer the following questions:

1. Is it possible to use the technology for free?
2. Is the use of the tool restricted by any license?
3. Is the tool sill actively under development?
4. Can the tool be extended/customized? Is source code documentation available?

[44] https://scholar.google.at/, Accessed: *23.01.2023*
[45] https://ieeexplore.ieee.org/Xplore/home.jsp, Accessed: *23.01.2023*
[46] https://dl.acm.org/, Accessed: *23.01.2023*

Due to its small overhead, when compared to the other tools, to quickly create dashboards, the final decision was to use Apache Superset. In addition, the choice was influenced by its Open Source Community and its flexibility in implementation and extensibility.

4.4 Development Process

In the following section the development process and the tools applied the implementation of the Python crawler are described in more detail.

4.4.1 Project Organization

The Python crawler was implemented applying an iterative workflow which was organized using GitLab Issues and its Board – as made apparent in Figure 4.1. All tasks were organized using these three stages, starting with the one most to the left. By default, each new task was considered to be an *Open* issue. If a task was under active development, it was placed in the *In Progress* column. When a task was completed, it was moved to the *Closed* column.

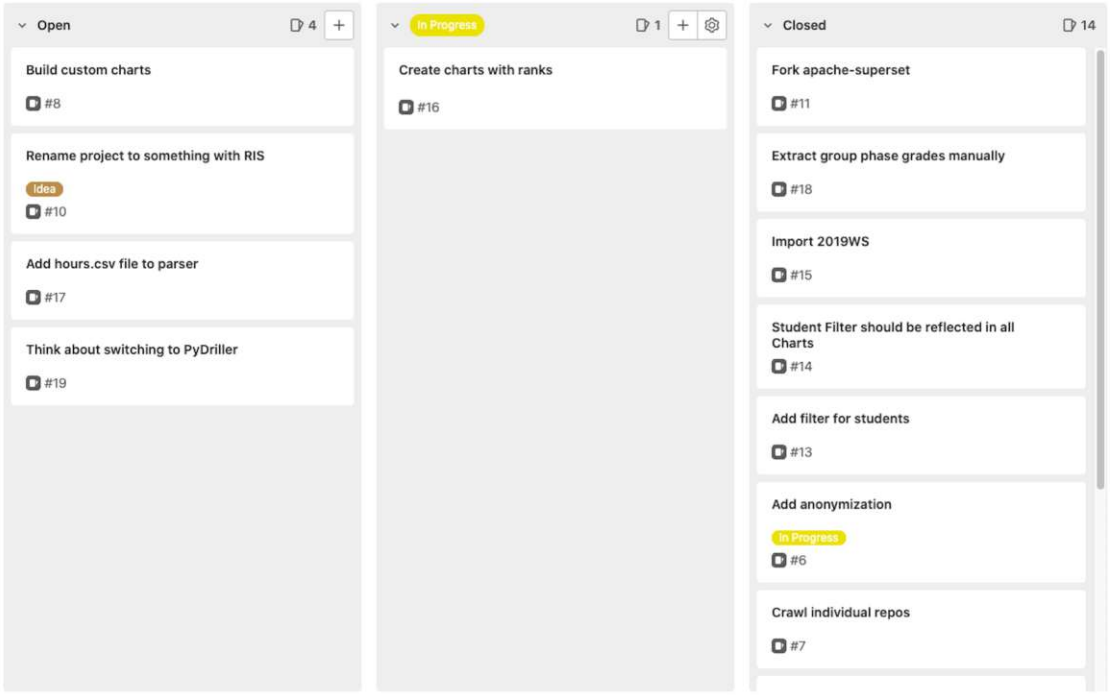


Figure 4.1: GitLab's development board

4.4.2 Development Tools

The tools used during the development process are briefly described in the following.

Interface Design Tool. For creating the initial drafts of visualizations (as shown in Section 5.1) and generating ideas, an Interface Design Tool named Figma[47] was used. Figma can be used to create high fidelity prototypes including interactions. The idea of using a high fidelity prototype was, due to the limited time of this thesis, to create a working demo within a relatively short time, as this kind of prototyping is highly suitable for exploration and testing [101].

Source Code Versioning System. Git was used as a central place for all sources and documentation during the development process. As this study was a one-man-project, most Git strategies, as for example branching, only sometimes were useful since there was no collaborative work to be done.

Online Repository Hoster. As a repository hoster for Git the private infrastructure of GitLab[48] was used. Compared to the public GitLab instance, this allowed for easier handling of all critical student data and information, especially in terms of data documentation in the GitLab Wiki.

Integrated Development Environment. For all coding activities of this thesis, Visual Studio Code[49] was used as an IDE.

4.4.3 Implementation Process

Prior to starting the implementation process, the available data sources had to be explored in terms of knowledge discovery and finding connections between different kinds of datasources [91]. To describe the system architecture, the following diagrams were used:

- **Domain Model**: Based on the explored data, an exploratory domain model [41] was created to identify connections and relations of the data.
- **Database Diagram**: The identified elements of the domain model were then transformed into a database diagram. This prototype uses, as required by Apache Superset, a relational database. The diagrams are later presented in Figure 6.2 and 6.3 in Section 6.2.2.

Implemented features were verified only by manual acceptance tests since the implementation of automated tests would have been too complex considering the time constraints

[47] https://www.figma.com/, Accessed: *23.01.2023*
[48] https://reset.inso.tuwien.ac.at/repo/, Accessed: *23.01.2023*
[49] https://code.visualstudio.com, Accessed: *23.01.2023*

and the scope of this thesis. As later shown in Section 8.2, the expected outcomes were familiar, which is why the program output had to be compared to these outcomes only.

4.4.4 Data Visualization Process

As Apache Superset allows direct database access, the Materialized Views (MVs) and Views of the PostgreSQL database were visualized using its own capabilities. The ten principles of creating data visualizations (recall from Section 2.2) were followed during the visualization process wherever and whenever appropriate and suitable.

4.5 Proof of Concept

Based on the initial investigation of the available data, the visualization concepts were created, and the Python transformer was developed to build the data storage foundation for the later Apache Superset prototype. The proof of concept was realized using a local Docker infrastructure having the potential to be easily transformed into a Kubernetes-based infrastructure.

4.6 Evaluation

The evaluation was done in two separated expert interview rounds which is a well established method in research [120]. The first evaluation solely considered the visualization concept, the second evaluation only beheld the prototype realization. The participating experts were the core staff members of 188.909 Software Engineering and Projectmanagement[50] who are expected to work with the final version of this prototype in the future.

Expert Interviews

An online tool[51] was used to design the survey, allowing an iterative workflow compared to an analogue survey. The questionnaire contains questions based on a Likert-scale [40] as well as open questions, as suggested by A. and Pfleeger in "Personal Opinion Surveys" [1]. Although open-ended questions might cause misinterpretation [1], possible misunderstandings could be clarified during the interview. The manual evaluation effort remained reasonable compared to closed-ended questions due to the modest number of survey participants.

Expert Evaluation

Based on the idea of *Technical Action Research*, the proposed prototype was finally evaluated by experts in a separate session [123]. The main aim of this survey was a)

[50] https://tiss.tuwien.ac.at/course/educationDetails.xhtml?dswid=6316&dsrid=388&courseNr=188909&semester=2022S&locale=en, Accessed: *23.01.2023*

[51] https://www.google.com/intl/de_at/forms/about/, Accessed: *23.01.2023*

to answer the research questions of Section 1.3 and b) to evaluate the usability of the tool. For the evaluation itself, certain scenarios were presented to the experts, based on the hypotheses of the first survey. Although the prototype's usability was not the primary objective of the survey — and cannot be modified easily — a system usability scale questionnaire was used to evaluate the level of the experts' satisfaction. [15].

CHAPTER 5

Information Needs in Software Engineering Education

In this section the insights gained from the expert interviews, which were used to rate and rank the proposed concepts, are provided. Firstly, the created concepts are explained (Section 5.1), followed by the study design (Section 5.2) and, finally, the results of the conducted interviews are presented (Section 5.3).

5.1 Concepts

Prior to creating the survey of the upcoming Section 5.2, the created mock-ups are shown and discussed in more detail. The concepts for visualizing the available data were designed using the Figma tool. The foundation of the design is an open source design system named Carbon Design System[52] developed by IBM[53].

5.1.1 Single Views

This section introduces the single views which are later embedded into the full-screen views in Section 5.1.2.

Group Overview Card

Figure 5.1 presents a summary of a single group's data, divided into two main rows.

The first row consists of an overview of the time booking history on the left-hand side and the distribution of the commits per student on the right-hand side. The former is

[52] https://carbondesignsystem.com/, Accessed: *23.01.2023*
[53] https://www.ibm.com, Accessed: *23.01.2023*

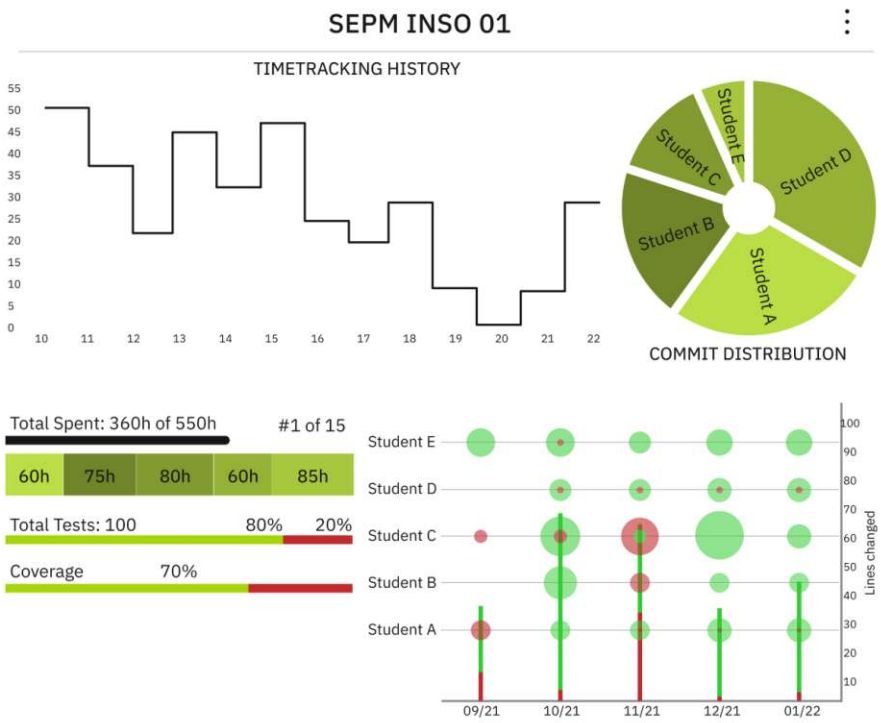


Figure 5.1: Mock-up: Group status overview card

the sum of the hours booked by all the students, whereas the latter is the cumulated number of commits per students represented as a pie chart.

The second row is again divided into two columns. The left one shows stats about the project progress and artifact quality.

The first row displays the total time spent on the project in textual form as well as a progress bar underneath. The maximum number of hours which should be spent (represented as 550h in the respective case) reflects the maximum of 110 hours per student (for six students it would be 660). In addition, their rank, compared to all other groups regarding their time spent on the project, is textually displayed as `#1 of 15`. In this example, the group is at the first position of 15 groups.

The second row shows tiles with the amount of time spent on the project per student: the broader the tile, the more time was spent. The color of the tile represents the student, as in the first row for the commit distribution.

The third row displays the total number of tests, including the number of successful (green) and failed (red) tests. The ratio is also demonstrated in percentages above.

The last row illustrates the current code coverage of the existing tests; the wider the green bar at the bottom, the more code is covered by tests.

The right-hand side of the second row contains a more complex visualization of commits and line changes, inspired by GitLens[54]. The x-axis represents a month — for example 09/21 meaning September 2021 — the left y-axis shows the names of each student in the group. The right y-axis is a linear scale representing the sum of changed lines. For each month, the number of lines changed are displayed as vertical bars. The green bar represents insertions, the red bar deletions. For each student, for each month, a green and/or red circle is shown, which means that the bigger the circle, the more changes were performed by this particular student.

Code Contribution Distribution

Figure 5.2 visualizes the code contribution for each group using a box plot. The vertical axis displays the groups and the y-axis the normalized values of each student's code contribution. The y-axis value for each student is simply normalized by considering the percental ownership of the codebase. The smaller the gap between minimum and maximum value, the better it is from a course's point of view since every student should contribute about the same amount to the project.

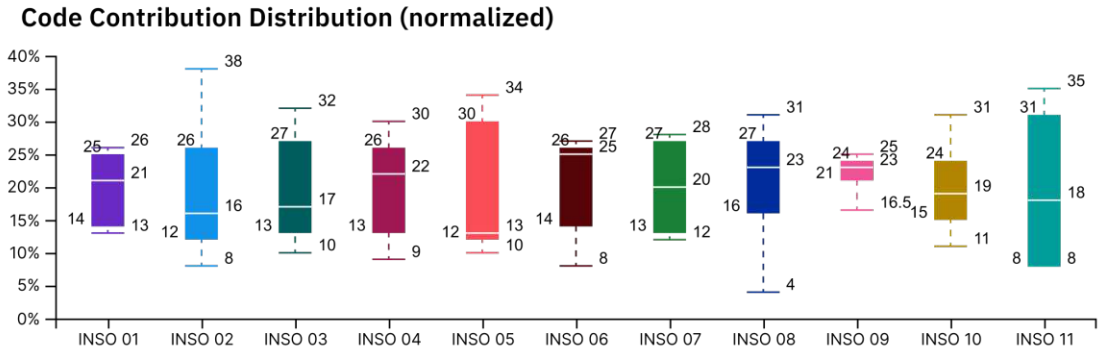


Figure 5.2: Mock-up: Code contribution distribution per group normalized in percent

Commit Distribution

Figure 5.3 shows the distribution of commit timestamps over 24 hours for all groups. The x-axis corresponds to the daytime and the y-axis to the commit frequency per daytime.

Time Tracking History

Figure 5.4 demonstrates the history of time tracked for each group per week. This perspective allows experts to identify peaks and bottoms over the time of several weeks. Ideally, each student contributes ten hours per week to the project.

[54] https://gitlens.amod.io/, Accessed: *23.01.2023*

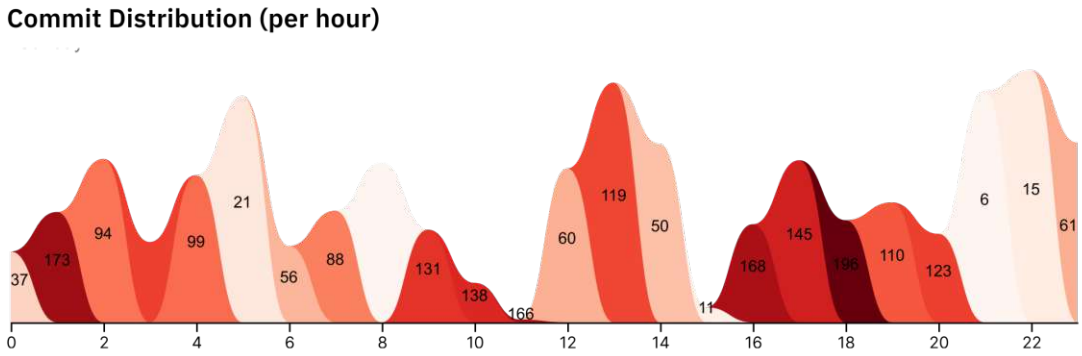


Figure 5.3: Mock-up: Commit distribution per hour (24-hour format)

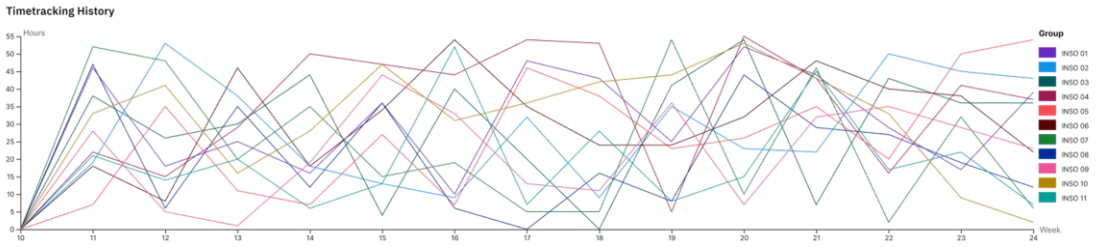


Figure 5.4: Mock-up: Time tracking history per group over time (weekly basis)

Time Distribution

Figure 5.5 shows a radar chart, each data point representing a group and the sum of its booked hours. The scale ranges from zero to 100 percent and normalizes the time spent on the project to the expected time so that each group can be represented the same regardless of its size.

Quantile Coherence

Figure 5.6a categorizes groups, as displayed on the left-hand side, into 25, 50, 75 and 90% quantiles regarding their worked hours, Lines of Code (LoC) and coverage. This allows tracing groups over different categories. For example, groups with low effort producing a high amount of LoC, or groups with a low number of LoC but a high coverage can be identified.

Ranking of Spent Hours

Several concepts were created for ranking groups based on their time spent on the project.

Figure 5.7b shows an extended variant of Figure 5.7a with a box plot on top, which allows an easier grouping.

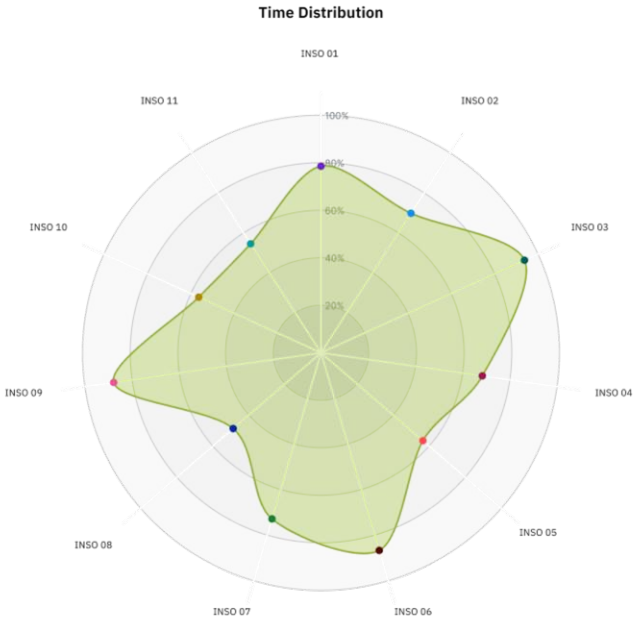


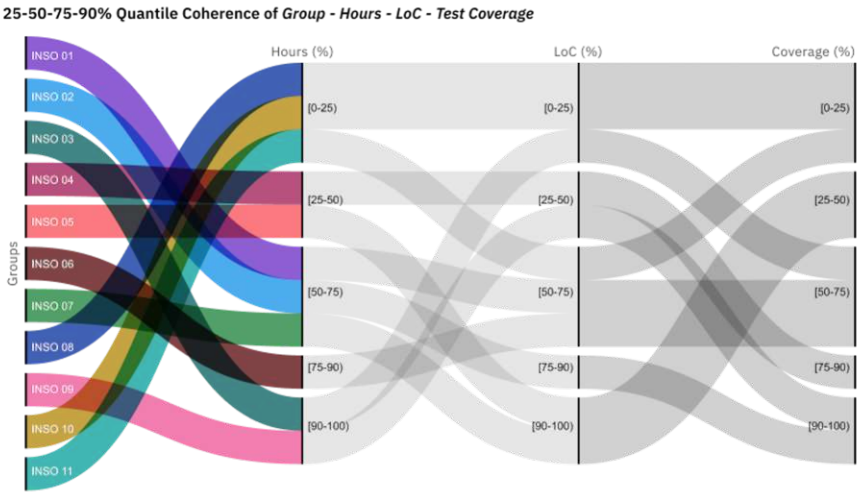
Figure 5.5: Mock-up: Time distribution radar chart for all groups

Figure 5.7c is another iteration of how to visualize and rank groups based on their working hours.

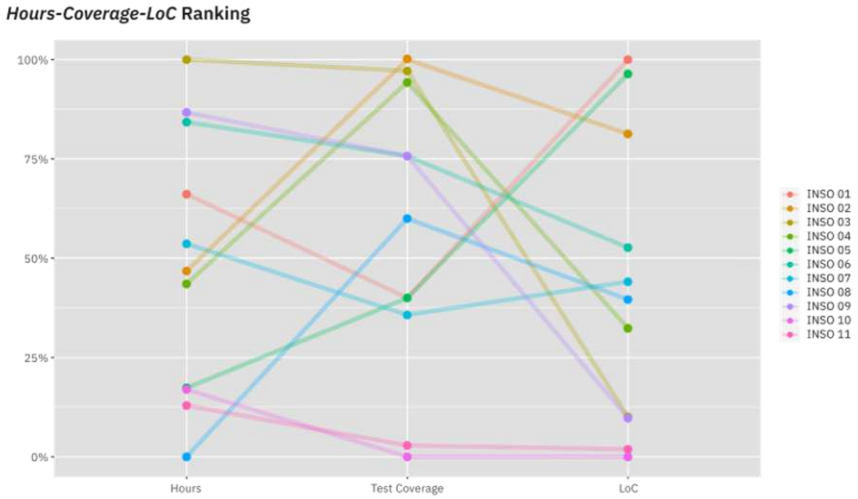
Figure 5.7d is a modification of Figure 5.7c. In the former, the bars are arranged in descending order according to the standard deviation of the time spent by the students within a group. Compared to Figure 5.7c, in Figure 5.7d the standard deviation is also displayed in text form in the diagram.

Ranking of Tests and Coverage

Figure 5.8 shows different variations of how to rank groups based on their number of total number of tests, successful tests, failed tests and coverage. Figure 5.9 is a modified variation of Figure 5.8 by an adding an x-axis at the top which displays the quantiles of the number of tests. To illustrate, if, for example, a group is situated on the left-hand side of the 25% mark, at least 75% wrote more tests than these groups.

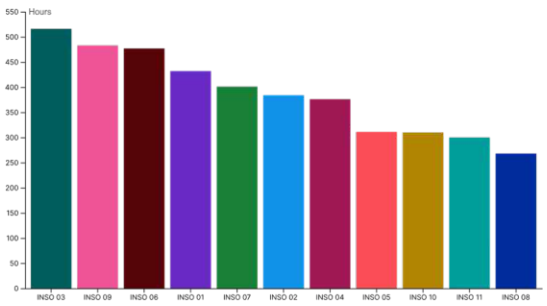


(a) Mock-up: Hours-LoC-Coverage quantiles for each group

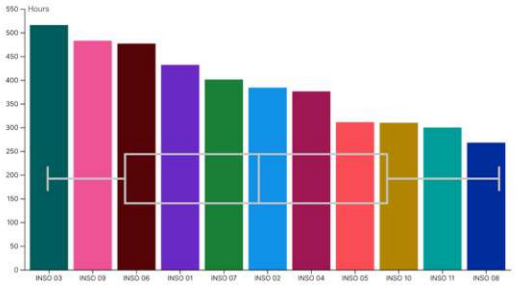


(b) Mock-up: Variation of Figure 5.6a

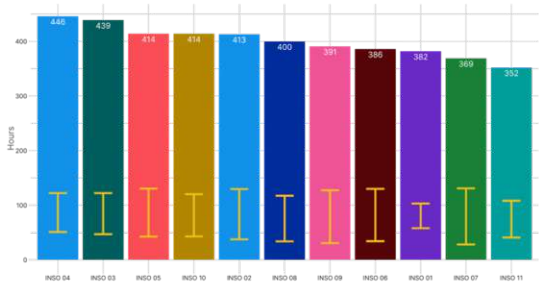
Figure 5.6: Mock-up: Flow ranking



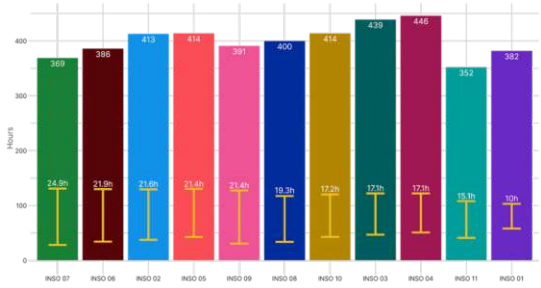
(a) Mock-up: Simple ranking of groups based on their time spent on the project



(b) Mock-up: Modification of Figure 5.7a, extending with a box plot



(c) Mock-up: Groups ranked descending by working time



(d) Mock-up: Variation of Figure 5.7c, ranked by standard deviation

Figure 5.7: Mock-up: Variations of ranking groups by their time spent on the project

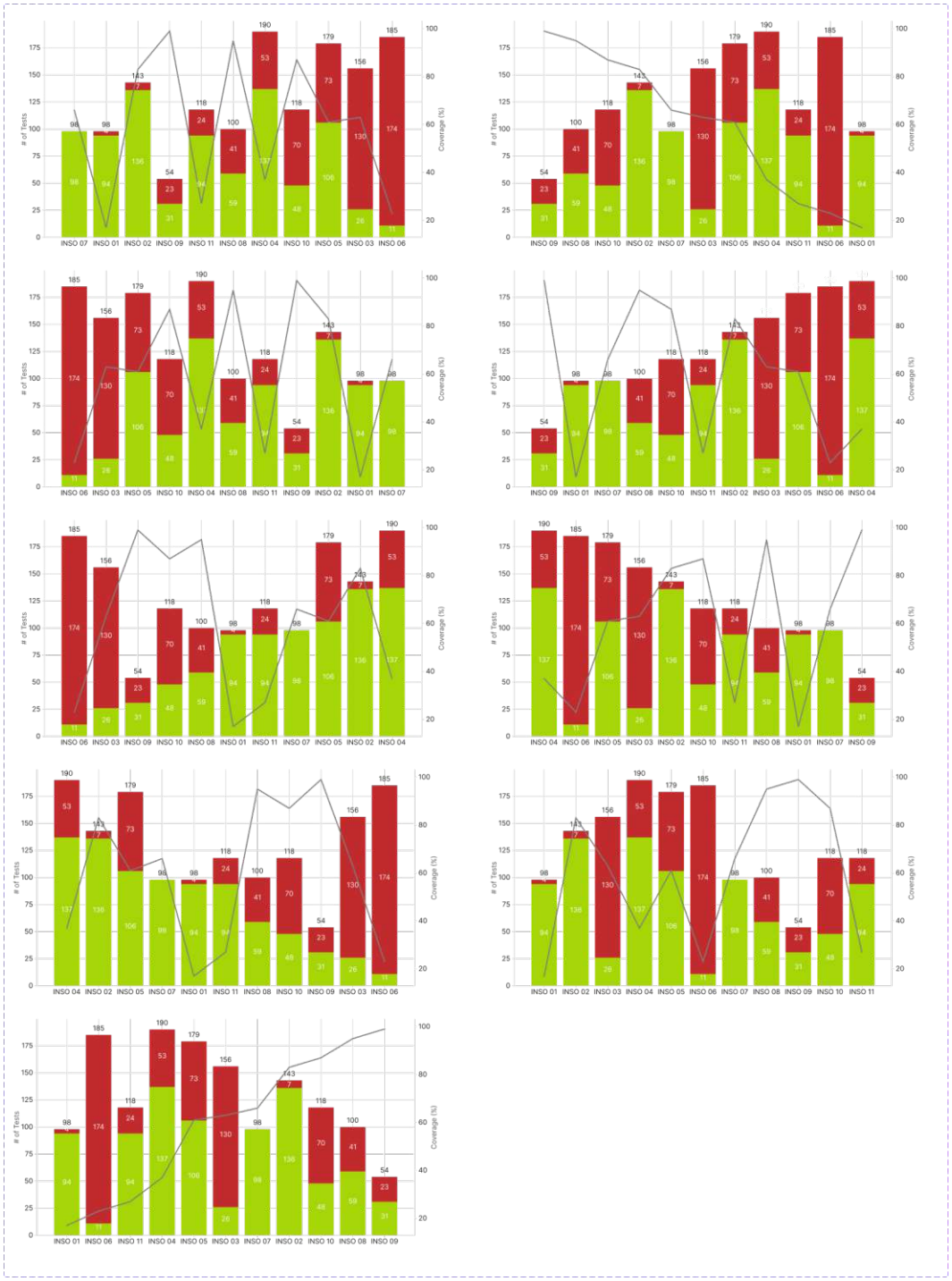


Figure 5.8: Mock-up: Variants of group rankings by their number of tests and coverage

Figure 5.9: Mock-up: Variation of Figure 5.8, added quantiles

5.1.2 Full-screen Views

This subsection visualizes the single views of the previous Section 5.1.1 in an integrated full-screen way.

Each of the following figures were designed using a 16:9 ratio with 1920px in width and 1028px in height. At the top a navigation bar indicates the current active page. On the left-hand side of each of the figures a control panel is shown to select single groups which should be displayed. Hence, all other groups are not displayed. In addition, on the top there are two date pickers placed to select a certain range of time.

Group Overview

The overview concept of Figure 5.10 is a collocation of multiple figures of Figure 5.1. These figures were tailored so that at least eight cards are shown on one and the same page.

Figure 5.10: Mock-up: Group overview of the current status

Group Comparison

The Group Comparison view of Figure 5.12 consists of several single views. In addition, the top left corner of this view shows general information about the selected groups. The mock-up is a collocation of the following figures: Figure 5.2, 5.3, 5.4, 5.5 and 5.6a.

The general information given in Figure 5.11 shows

- the average commits per day, including the current trend,
- the average test coverage per group,
- a box plot representing the current success rate of all tests of the groups.

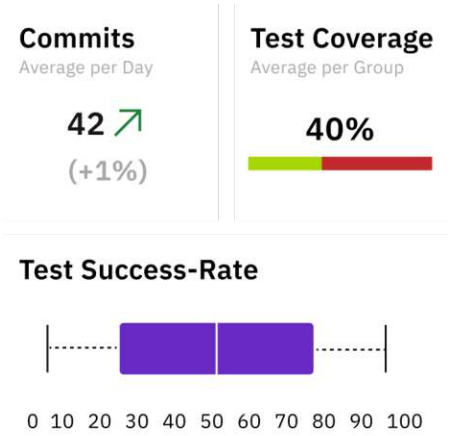


Figure 5.11: Mock-up: General group statistics

Figure 5.12: Mock-up: Group Comparison view

Group Ranking

Figure 5.13 illustrates the idea of ranking different groups of a single term based on various metrics extracted from available data or from deeper analysis of the current state of their project. For this purpose, different variations of charts, as already mentioned in the subsection *Ranking of Tests and Coverage*, are combined into a single view. This allows the course staff to identify groups which are out of the expected values of a metric — for example, by spending much more or less time than expected at a specific time during the term.

Figure 5.13: Mock-up: Ranking view of all groups

5.2 Study Design

The survey (see Appendix C) was designed based on "Personal Opinion Surveys" [1] and grouped into ten categories. These blocks contained both open and closed questions, covering specific aspects of information needs relevant for comparing and ranking student groups. The ten categories of the questionnaire were:

1. **Demographic**
   The purpose of this block was to collect information about the survey participant in order to obtain additional information for the evaluation of the survey.

2. **General Questions**
   To better identify the expert's role within the class, this block concentrated on the participant's activities and experience.

3. **Grading and Comparison**
   This section's questions aimed to collect general data about the required information in order to be able to grade students individually and groups as a whole and to compare students as well as groups and terms with each other.

4. **Group Comparison and Ranking**
   Similar to block three, this block centered on the information required for comparing and ranking groups. In contrast to the third block, block four's questions were intended to serve as an information basis to derive the current status of groups.

5. **Current situation**
   The questions of this block allowed the participant to fill in information about the current information gathering situation.

6. **Hypotheses**
   Based on the literature research, this block was used to validate hypotheses which built the foundation of the mock-ups and prototype created. The questions in this block were inspired by a preceding master thesis of Weiß [122].

7. **Available Views**
   Based on the charts available in Apache Superset, the mock-ups, which were realized, were presented in this block and rated using a Likert-scale [40].

8. **Future Views**
   All mock-ups which could not be created in Apache Superset were grouped in this block and rated by means of a Likert-scale [40]. The idea undermining this block is to evaluate the usefulness and importance of certain new concepts and to come up with a prioritization for future work.

9. **Similar Views Comparison**
   All figures that visualize the same data source in different ways were compared in this section, which allowed an easier prioritization for the future.

10. **Full screen Views**
    Finally, the mock-ups also include visualizations of several charts combined in a dashboard. The questions of this block are designed to identify possible missing elements.

In the pilot evaluation a first draft of the survey was presented to an expert who was asked to validate the overall quality, outline misunderstandings and provide feedback. The first evaluation revealed that some questions were misleading and, thus, easily misinterpreted. Regarding the visualizations (as shown in Chapter 7) beneficial feedback was provided. Some questions were edited by including more textual definitions, to the disadvantage of a longer reading time and the advantage of avoiding ambiguous meanings. This was especially necessary for the understanding of a) the proposed mock-up visualizations and b) the Apache Superset tool and its functionality. To prevent any personal bias or influence on the participants, the questions were formulated as neutrally as possible [1].

All interviews were conducted remotely via Zoom[55]. Based on the pilot, it was estimated that each interview will last for about an hour. The interview was introduced providing a short overview of the interview topic, if no further questions arose, the interview started. The interviewer's screen was shared so that the participant could read the questions independently and ask for clarifications if something was unclear. Otherwise, the answer was provided by the interviewee and filled in by the interviewer.

5.3 Results

In the following the results of the interviews conducted with five domain experts are presented in more detail.

Demographic

The first two questions asked about basic demographic information so that the participants could be classified. The domain experts interviewed were between 26 and 42 years old and exclusively male.

General Questions

The second section about general questions contained six questions however, four of these were depended on previous answers and were skipped if the participant's role did not match the prerequisite.

The participants' experience, role and activities within the class were more varied. When being asked about their years of experience, one participant answered with *> 20 years*, one with *10-15 years*, one with *5-10 years* and two with *2-5 years*. The participants' answers to the question about their role (they were allowed to select multiple roles) can be summarized as illustrated by Figure 5.14.

[55] https://zoom.us/, Accessed: *23.01.2023*

Figure 5.14: Roles of interview participants

The last question of this block was interested in the number of groups a participant is working with. Based on the selected role(s), the following were the answers:

Role *Lecturer*: All groups (2x)
Role *Assistant*: 1, 3, All groups
Role *Admin*: All groups (2x)
Role *Tutor*: 1 group (2x)

Grading and Comparison

The third section contained four general questions about grading and comparing students and groups. Starting with the information needs for grading student groups as a whole, the result can be seen in Figure 5.15. It becomes clear that all of the participants consider the code quality[56] and the Git-Contribution for grading. Four out of five also consider the quality of the implemented features. In addition, 60% also consider the time efforts and the theoretical knowledge.

The tenth question asked about which additional information — based on the answer possibilities of Figure 5.15 — is required to grade students individually. The answer was essentially the same as for grading groups as a whole.

The final two questions of this block focused on group comparisons, which was answered by two participants with *Yes* and by three with *No*. The following question about the reasons why they did not compare groups in the past was answered by the three experts with:

- objectivity (2x)
- individual project; unfair (1x)

[56] Code Quality = for example Test Coverage, Successful/Failed Tests, etc.

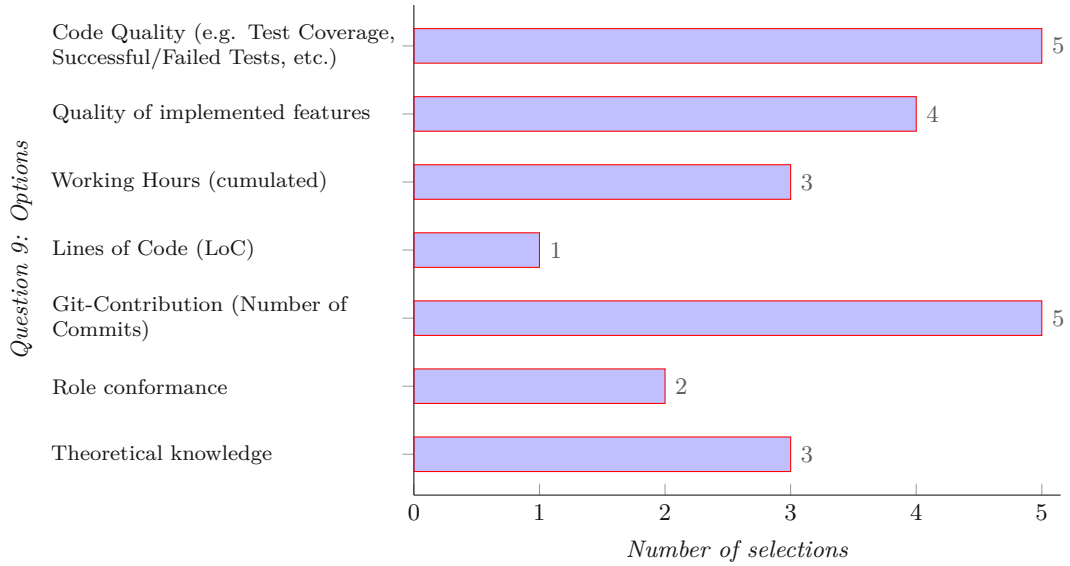


Figure 5.15: Information needs student group grading

Group Comparison and Ranking

Section 4 contained two questions about the experts' needs to compare and rank groups. As opposed to the previous section, the concept of group comparison and ranking was aimed to serve as an information basis to derive the current status of groups.

The participants answered the first question about the number of relevant groups in two ways:

- To see and compare/rank all groups
- To see and compare/rank only the relevant groups

The second question was again about which metrics are relevant; the answers can be seen in Figure 5.16. It shows a notable difference for all options, when compared to Figure 5.15. In this context the working hours are more important (since selected by all participants) and generally four options were selected by at least 80% (compared to three options in Figure 5.15). However, the quality of the implemented features or the student's theoretical knowledge, for example, is less important for ranking and comparing groups. An additional answer was given by one participant who mentioned that *Teamdynamics* should also be considered when comparing and/or ranking groups.

Current situation

The fifth section asked about the current situation of information gathering. Three open-ended questions followed in this section. The first question, interested in how the time spent is currently collected, was solely answered with *GitLab Wiki*. Followed by the question of how the workload distribution is determined. It was answered with *Issue*

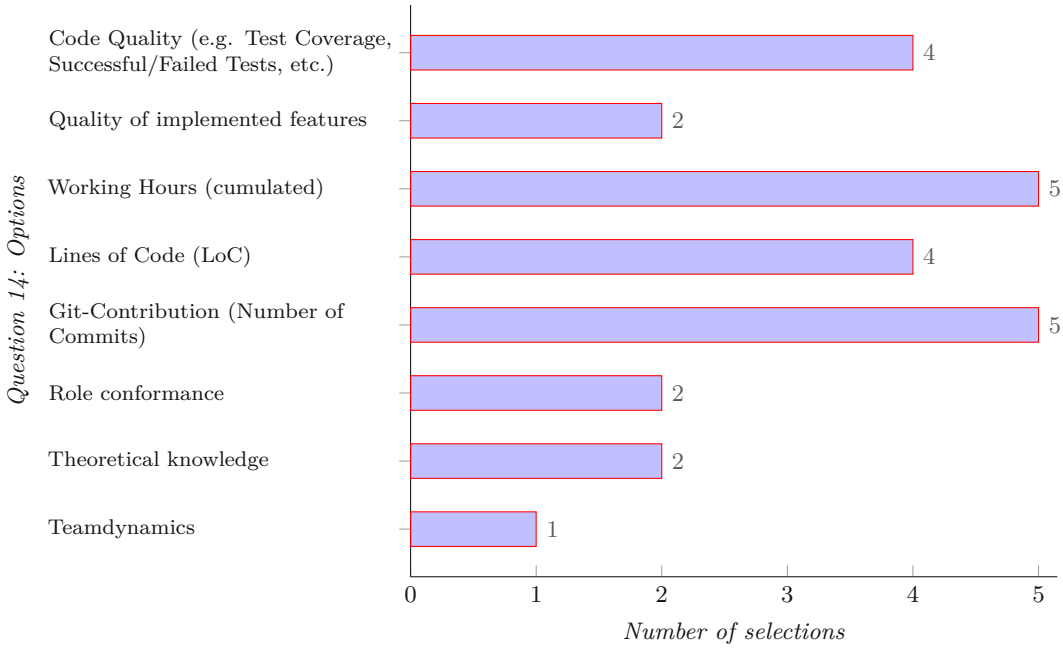


Figure 5.16: Information needs student group comparing and ranking

distribution three times, Meetings (respectively jour fixe) twice and gitinspector once. The third question asked was how Git information is collected. There was again only one answer, namely *gitinspector*. Some participants added that they were using plain Git and GitLab for this task in the past.

Hypotheses

The fifth block was about verifying the initial hypotheses of this thesis. Since all following 15 questions were based on a five-point Likert-scale [40], the resulting box plots are listed in Figure 5.17. The scale was defined as *Strongly Disagree* (1) to *Strongly Agree* (5)[57], the diamond shape in each plot indicates the average value. Seven out of the 15 hypotheses achieved a minimum score of 3, three were voted with at least *Agree* (value 4), only one — question 25 — used the entire range and only question 23 got a rating of maximum 3. In addition, six questions got a rating between 2 and 5, which shows a wide spread regarding these questions and highly reflects personal opinions on these particular hypotheses.

[57] 1 Strongly Disagree; 2 Disagree; 3 Neutral; 4 Agree; 5 Strongly Agree

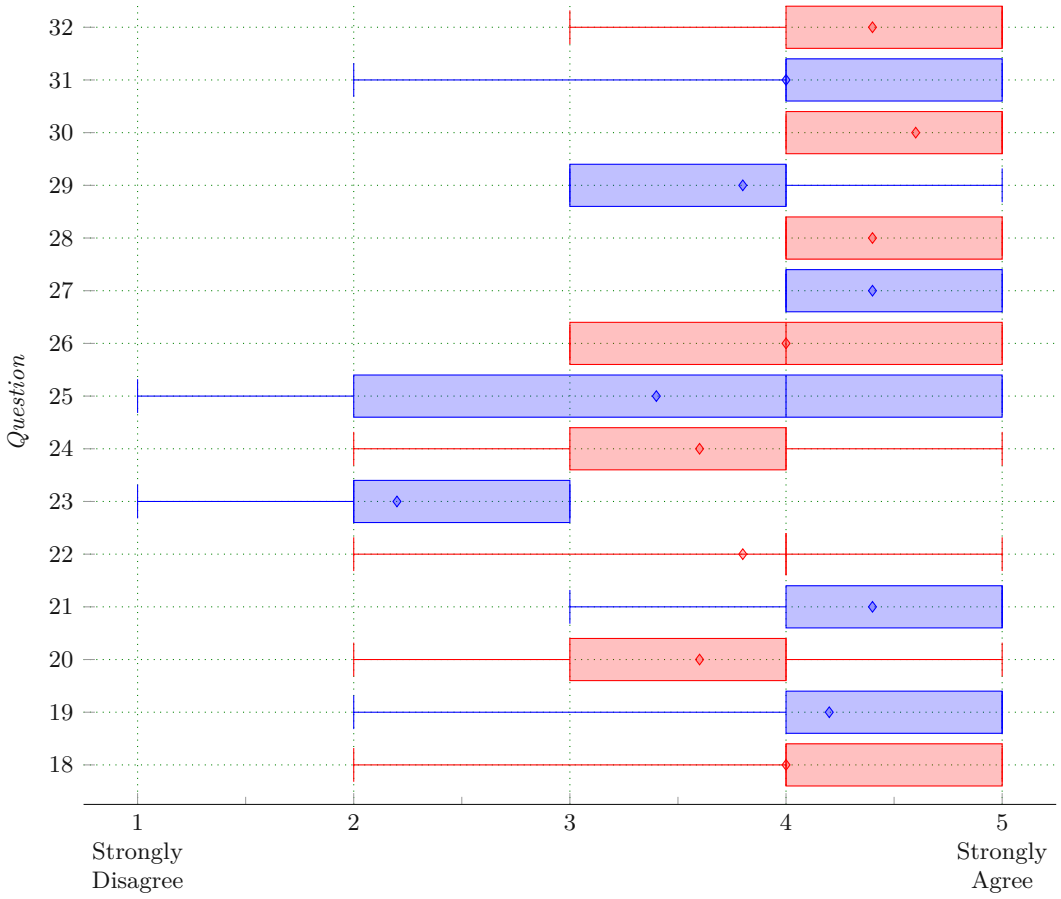


Figure 5.17: Hypotheses results

Available Views

The next block was about evaluating the already existing views and their purposefulness. The scale ranged from *Not useful at all* (1) to *Extremely useful* (5)[58]. The block started with a *General visualizations* subblock. Figure 5.18a shows the results as a box plot. It demonstrates that only two of these five visualizations were rated better than 3, one better than 2 and one between 2 and 4. The visualization of question 34 was rated as irrelevant, with a maximum score of 3.

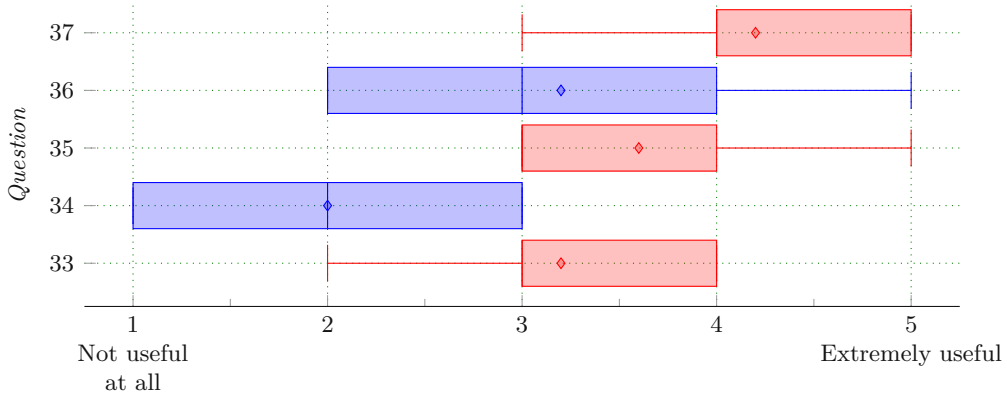
The last question of this subblock was an open one, aiming to find the kind of information which should be visualized in a similar way, resulting in the following answers:

- Amount of time spent on the project
- Average time logged additional to the amount of time logs for a better overview (lots of short times spent vs. lots of long times spent, ...)
- All what I want to know about the group
- Time spent in hours; LoC (diff, insertions, deletions)

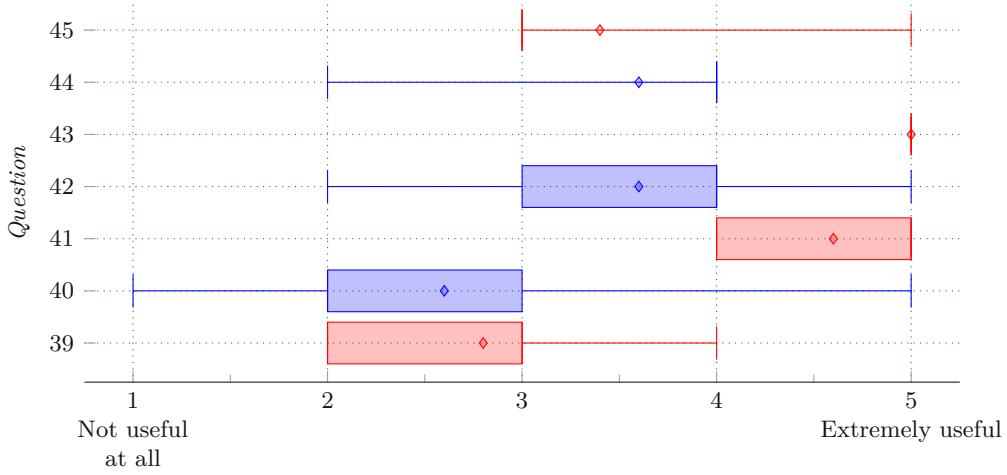
The next subblock was about the *Repository-related visualizations*. The results are shown in Figure 5.18b: the first two repository related visualizations were rated rather at the lower end to the middle, whereas the subsequent were rated better. The visualizations presented in questions 41 and 43 achieved an excellent rating: the former question got twice a 4 and three times 5, the latter was solely rated with *Extremely useful*.

The last subblock was concerned with the *Project Management related Visualizations*. The results are presented in Figure 5.18c, where all visualizations were rated at least to be *Slightly useful*. However, the majority was rated with 3 being at least *Somewhat useful*, the median of each question is at least located at 4, *Moderately useful*.

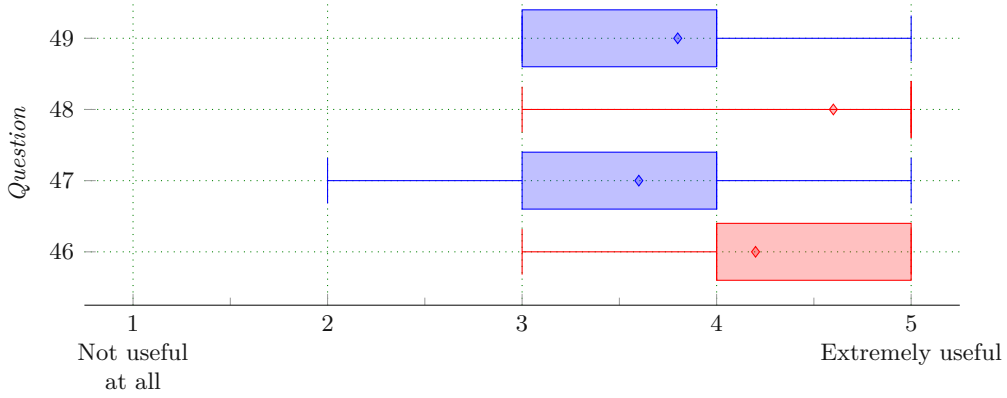
[58] 1 Not useful at all; 2 Slightly useful; 3 Somewhat useful; 4 Moderately useful; 5 Extremely useful



(a) General visualizations results



(b) Repository-related visualizations results



(c) Project Management related visualizations results

Figure 5.18: Available views evaluation result

Future Views

The eight section focused on the evaluation of the importance of certain new concepts, previously described in Section 5.1. The scale was defined ranging from *Not important at all* (1) to *Very important* (5)[59]. The first question evaluated the visualization as shown in Figure 5.1. Figure 5.19 shows the results in the form of a box plot using the full range. Only one participant voted with *Not important at all*, the median conversely is located at 5, *Very important*.

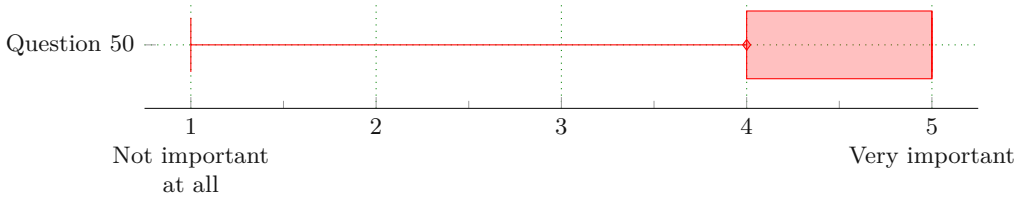


Figure 5.19: Evaluation of Figure 5.1

Since this type of visualization is a new concept, an open-ended question asked what type of information was missing or should be changed for a comprehensive overview. The following answers were given:

1. Test/Coverage and Total Spent should be differently visualized
2. Maybe display the relation between loc in project and number of commits (students committing one liners or whole files)
3. Split timetracking into students (one graph each), timetracking summary at the bottom
4. Hours per student
5. Survived lines of code (similar to gitinspector)

The third question regarded Figure 5.8 and 5.9, the resulting box plot can be seen in Figure 5.20. The participants could evaluate all different modeled options, resulting in five rankings for *Tests (Green)*, *Tests (Red)*[60], *Coverage*, *Tests (sum)*, *Group*. Surprisingly, the option for *Tests (Red)* got better results than the one for *Tests (Green)*. The option for coverage got the best result overall, except for the *Group* option. However, it should be kept in mind that ranking groups by their name is just a listing, no ranking.

[59] 1 Not important at all; 2 Slightly important; 3 Neutral; 4 Moderately important; 5 Very important
[60] Tests (Red) is equivalent to failed tests

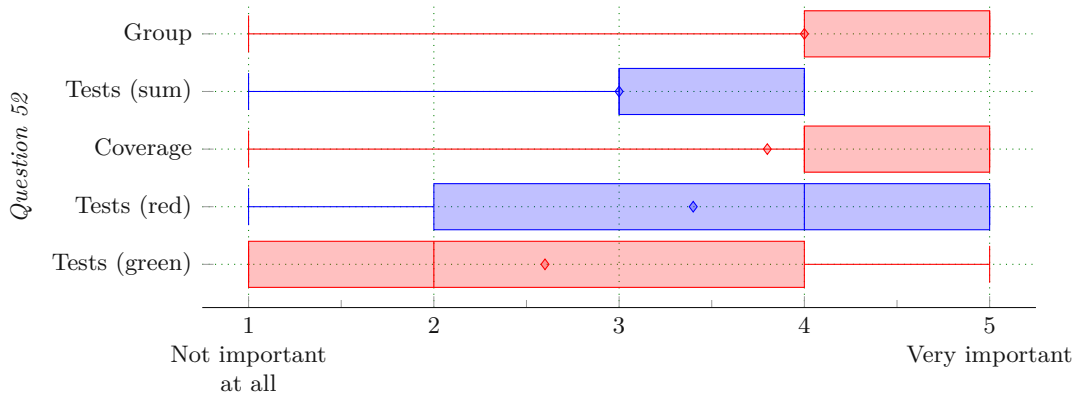


Figure 5.20: Evaluation of Figure 5.8 and 5.9

The remaining seven questions of this block were structured as follows:

> Question 53 and 54 evaluated the concept of Figure 5.7c respectively Figure 5.7d. Question 55 and 56 focused on Figure 5.6a and Figure 5.6b, followed by the radar chart of Figure 5.5 (Question 57). Question 58 addressed the pie chart at the top left of Figure 5.1 and the last question of this block asked about the visualization of the Code Contribution as shown in Figure 5.2.

As can be derived from Figure 5.21, with the exception of questions 54 and 59, all concepts achieved good results. Question 53, 55 and 58 were solely rated with *Neutral* or better and more than 75% of the participants responded to question 57 with *Neutral* or better.

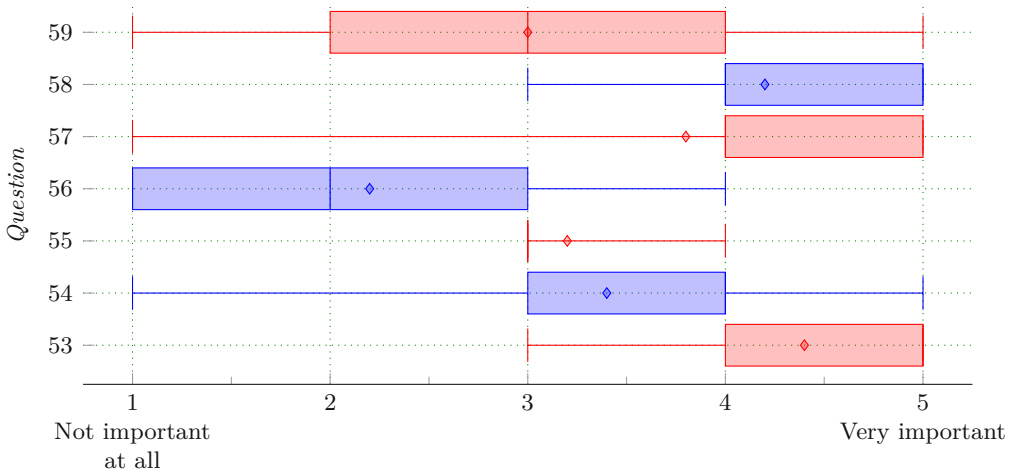


Figure 5.21: Evaluation of Figure 5.8 and 5.9

Similar Views Comparison

The second to last block was about the evaluation of visualizations of the same data in different ways. Starting with question 60, regarding different time tracking variations, participants should select the top three (or less) out of four versions. The results are that Figure 5.7d and Figure 5.5 were selected four times, Figure 5.7a three times and Figure 5.7c twice.

The next comparison concerned the flow rankings where Figure 5.6b got four votes and Figure 5.6b only one. This question was followed by the comparison of the test and coverage rankings where Figure 5.8 was selected three times and Figure 5.9 twice. For the already existing visualization of the time spent on the project by students, the weekly sum, as shown in Figure 7.4, was selected four times and its cumulative variation once. The votes for the commit timeline visualization were more varied as the weekly sum of Figure 7.3 got three votes and the cumulative variation two.

Full screen Views

The focus of the last block the evaluation of the full screen concepts shown in Figure 5.10, 5.12 and 5.13. Since these three questions were open-ended, the answers were essentially no different than those to the individual concepts of the previous blocks. The full screen views also seemed a bit overwhelming for the participants. So, no additional value could be derived from the participant's answers.

CHAPTER 6

Extract, Transform, Load Implementation

The following chapter describes the implementation of the Extract, Transform, Load (ETL) process. Recall that ETL outlines the three processes of transforming raw data from multiple locations into a unified schema suitable for analysis. The first chapter, 6.1 presents the overall format of the available data, followed by the description of the transformation and loading implementation in 6.2. This chapter concludes with the realization of data loading from the PostgreSQL database in 6.3.

6.1 Data Organization

Starting with the 19.2 GB data provided by the course admin, the first stage of the ETL process is about extracting available data. A separate extraction process is not required to be performed since it was already done in the past. Therefore, this section focuses on exploring and describing the data and its structure.

When exporting a GitLab project to be archived, a `export.tar.gz` file is created, containing all relevant data of that respective project. All available repository- and wiki-data is exported at the end of each term. Figure 6.1 shows the structure of the exported files. However, it only shows the relevant files needed for this prototype to be built and run. There are two important folders: At first the `export/tree/project` folder, which contains all project management related data stored in GitLab directly, and the `repo` folder which contains the codebase including the Git folder.

Besides the GitLab and Git related data, for each term, there are two additional Excel/CSV files. These contain information about all students attending the Individual Phase (IP) in the respective term. The first file consists of the student's forename, surname, matriculation number and their achieved points. As explained in Section 1.1,

Figure 6.1: GitLab's export structure

students need to track their time when working, which also applies for the IP, however, in a simpler way: It is sufficient to write down the working hours in a plain text file and add up the sum (which is done by the student). The second file contains the values for the time spent on the project having the format: first name, last name, login (= matriculation number), name (of the course), course year, term, hours.

GitLab's Export Structure

Exporting a GitLab project results in many files having different information. All files are in a `ndjson` format, which is some sort of extended JSON format. Instead of using one root array at the top-level, `ndjson` just stores the objects separated by newlines so that each individual line is a valid JSON object. In the following, all listings only show the minimal properties necessary needed to create the database. All the other fields are omitted due to their irrelevance for the data aggregation.

issues.ndjson and merge_requests.ndjson. The first two listed files in Figure 6.1, contain the entire information regarding all the issues of a project and Merge requests (MRs). They also include all time tracking entries of the project, which are the only places where GitLab allows tracking time[61]. As can be derived from Listing 6.1, there is an object called *timelogs* which stores time tracking information (so-called Timelogs) belonging to that issue.

[61] https://docs.gitlab.com/15.0/ee/user/project/time_tracking.html, Accessed: *23.01.2023*

```
{
    "id":9876,
    "title":"Kickoff-Meeting",
    "project_id":1234,
    "created_at":"2021-04-14T14:17:26.887Z",
    "updated_at":"2021-04-14T14:29:04.922Z",
    "iid":1,
    "closed_at":"2021-04-14T14:29:04.903Z",
    ...
    "timelogs":[
       {
           "id":887766,
           "time_spent":3000,
           "user_id":234,
           "created_at":"2021-04-14T14:17:26.915Z",
           "updated_at":"2021-04-14T14:17:26.915Z",
           "spent_at":"2021-04-14T00:00:00.000Z",
           "project_id":1234
       },
       ...
    ],
    ...
}
```

Listing 6.1: Example of an JSON object in `issues.ndjson`

In addition to issue-based time tracking, Timelogs can also be assigned to MRs. Hence, the `merge_requests.ndjson` must also be processed to extract all `timelogs` entries of a project. As made apparent in Listing 6.1 and 6.2, the entries for `timelogs` objects are identical in its structure.

```
{
  ...
    "timelogs":[
       {
           "id":223344,
           "time_spent":3600,
           "user_id":234,
           "created_at":"2021-06-14T10:32:27.093Z",
           "updated_at":"2021-06-14T10:32:27.093Z",
           "spent_at":"2021-06-14T10:32:27.063Z",
           "project_id":1234,
      ...
        },
    ...
    ],
  ...
}
```

Listing 6.2: Example of a `timelogs` object in `merge_requests.ndjson`

Most properties of Listing 6.1 and 6.2 are rather self-explanatory, except for the following two:

- `iid`: In addition to an ID of a MR or issue, GitLab stores an internal ID which is displayed by the web UI. That ID is unique for a single project[62] only.
- `time_spent`: This property stores the time spent on the project in seconds (time added with by `/spent` command).
- `spent_at`: A user can potentially also set the date when they spent that amount of time, which is stored in `spent_at`.

Although students are technically able to track time on a day different to the date the entry is created (which can be identified if `created_at` and `spent_at` differ from each other), the crawler always takes the value of the `created_at` field. Thereby, the course staff team wants to *force* students to track their time when they really work and do not add entries as they like. This facilitates identifying students which are lazy regarding their time tracking.

project_feature.ndjson. The `project_feature.ndjson` solely contained one JSON object for all exported projects. As Listing 6.3 shows, the only interesting property is `project_id` since all other JSON objects within an export always refer to that ID. Although one may assume that an ID should only be assigned once, the SQL model later had to be designed to use its own IDs. IDs were reused by GitLab over time, which lead to wrong matching of users to projects.

```
1 {
2     ...
3     "project_id":1234,
4     ...
5 }
```

Listing 6.3: Excerpt of `project_feature.ndjson`

project_members.ndjson. The last file listed in Figure 6.1 is the `project_members.ndjson` which consists of all members of a project. Listing 6.4 displays the structure. The important part of these JSON objects are their `access_level` properties. There are different roles in GitLab which can be assigned to any user in a project. The crawler only needs the student information and can ignore all other users, including, for example, admin, tutors or course assistants. Student user entries are identified by their `access_level` value of 30. According to GitLab, this value represents the *Developer* role[63].

[62] Source: https://docs.gitlab.com/15.0/ee/api/#id-vs-iid, Accessed: *23.01.2023*
[63] Source: https://docs.gitlab.com/15.0/ee/user/permissions.html, Accessed: *23.01.2023*

```
{
    "id":4444,
    "access_level":30,
    "user_id":234,
    ...
    "user":{
        "id":234,
        "email":"<student_email_address>",
        "username":"<student_username>"
    }
}
```

Listing 6.4: Example of an JSON object in `project_members.ndjson`

6.2 Transformation and Loading Implementation

The following section discusses some of the implementation's noteworthy features of the data transformation and loading process. The implementation was performed in an iterative process based on the available data. The tasks which needed to be implemented were organized in GitLab's Issue Board[64]. The preliminary task was to gather information about the available data and its structure. Consequently, some manual data collection, an initial architectural design and the fundamental database structure had to be completed upfront. Although the data of Section 6.1 was organized to be easily useable, no data cleansing nor cleaning was done. The implementation was held as flexible as possible. Hence, handling missing information was part of the implementation and default values are used.

The following section provides an overview of the state of the crawler's implementation, the technologies used and the architecture chosen for the program's database. This section closes with a discussion of the implementation of the pseudonymization methodology for the protection of sensitive student data.

6.2.1 Technology Stack

Before starting to implement the data mining program (also called crawler) into practice, a decision about the concrete technology stack needed to be made. Due to the nature of the data, Python was finally chosen as the programming language for the implementation. Python is the most preferred language for — including but not limited to — data science through enhancing both performance and productivity. This is achieved by the use of low-level libraries and clean high-level APIs[65] [97, 111]. It provides a large ecosystem with a plethora of external libraries for a wide range of problems while using a dynamic typing and multi-paradigm approach. For this reason, it is an excellent choice for rapid

[64] https://about.gitlab.com/blog/2018/08/02/4-ways-to-use-gitlab-issue-boards/, Accessed: *23.01.2023*

[65] Source: https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html, Accessed: *23.01.2023*

prototype development. In addition, multiple libraries were used by the program to extract all information relevant for this thesis.

The `ndjson`[66] library was used to process the `ndjson` files. It allows iterating over a file containing multiple JSON objects without manually opening and loading the file for each object.

`GitPython`[67] was used to iterate over Git repositories and read the available data. Although there would have also been other libraries which build on top of GitPython such as, for instance, PyDriller[68] [109], GitPython was chosen due to pre-existing knowledge about the library.

To access the data storage (as described later in this section) and to use an Object–relational mapping (ORM) toolkit for creating and manipulating the database, SQLAlchemy[69] was used.

For the pseudonymization part, the `pandas`[70] and `Faker`[71] libraries were used.

Data storing was the second crucial part where considerations had to be made since the database had to be compatible with the Apache Superset visualization tool. The eventually chosen database was PostgreSQL since Apache Superset is shipped with the PostgreSQL connector library psycopg2[72] if used with the `docker-compose`[73] file[74] (which was done for this prototype) and pre-existing knowledge about PostgreSQL. However, any other database which has a SQLAlchemy dialect implementation in Python could have been used.

6.2.2 Implementation Details

The current prototype is specially designed to work with GitLab exports and to be executed as a command line program using the Python interpreter. In the following, the configuration options and requirements on how to use the program to extract and store available data are presented.

[66] `http://ndjson.org/`, Accessed: *23.01.2023*
[67] `https://gitpython.readthedocs.io/en/stable/`, Accessed: *23.01.2023*
[68] `https://github.com/ishepard/pydriller`, Accessed: *23.01.2023*
[69] `https://www.sqlalchemy.org/`, Accessed: *23.01.2023*
[70] `https://pandas.pydata.org/`, Accessed: *23.01.2023*
[71] `https://faker.readthedocs.io`, Accessed: *23.01.2023*
[72] `https://www.psycopg.org/docs/`, Accessed: *23.01.2023*
[73] `https://docs.docker.com/compose/`, Accessed: *23.01.2023*
[74] `https://superset.apache.org/docs/databases/postgres`, Accessed: *23.01.2023*

Program Configuration

The configuration of the command line tool is two-folded: On the one hand, there are arguments which can be passed to the program and which are then evaluated during runtime by the `argparse`[75] library. On the other hand, the program needs a specific JSON configuration file which contains all necessary information to extract and parse the data.

The program arguments determine the extent to which the crawler processes the data during its run. Table 6.1 lists all arguments and their meaning.

As already mentioned before, the crawler needs more information about where to find which folder or file for the term to be processed. The `--file` specifies where to find that particular file. Listing 6.5 shows the structure of the configuration file in question.

Option	Description
`--file FILE`	The JSON file to load all relevant data from
`-p PROCS`	Number of cores to be used during runtime. Default is 1.
`-db DATABASE`	The database to be used for storing all data.
`--project`	Instructs the program to process all project management related data. This processes all files in the `export/tree/project` folder.
`--git`	Instructs the crawler to process the `repo` folder and its Git content. The processing is limited to extracting commit information only (no diff processing)
`--stats`	In addition to the `--git` argument, the stats option then retrieves all information of a commit, including additions, deletions and affected files.
`-h, --help`	Shows a simple help message

Table 6.1: Program arguments

[75] https://docs.python.org/3/library/argparse.html, Accessed: *23.01.2023*

```
{
    "term": "term",
    "individual_phase": {
        "csv_file": "/path_to_csv/file.csv",
        "repos_root": "/path_to_project/project",
        "timetracking_csv": "/path_to_csv/file.csv"
    },
    "group_phase": {
        "grades_csv": "/path_to_csv/file.csv",
        "projects_root_path": "/path_to_root_folder/folder"
    },
    "research_divisions": [
        "Div. A",
        "Div. B"
    ],
    "groups": {
        "Div. A": [
            {"01": "project_01"},
            {"02": "project_02"}
        ],
        "Div. B": [
            {"01": "project_01"},
            {"02": "project_02"}
        ]
    }
}
```

Listing 6.5: Structure of the crawler's JSON config file

In the following the elements of the configuration file are explained in more detail.

- `term`: This element specifies the term to which processed data belongs, for example, 20WS (winter term 2020).
- `individual_phase`: That object defines all relevant locations of IP files.

  `csv_file`: Location of the file which contains all Individual Phase results.

  `repos_root`: Location of the repository folders. Each folder inside must be in the form of Listing 6.1.

  `timetracking_csv`: Location of the file which includes time spent on the project during the Individual Phase by each student.

- `group_phase`: This object specifies all relevant locations of Group Phase (GP) files.

  `grades_csv`: Location of the file which contains all grades after finishing the GP.

  `projects_root_path`: Location of the repository folders. Each folder inside must be in the form of Listing 6.1.

- `research_divisions`: An array which includes the information about the research divisions, as described in Section 1.1. The values used are based on the abbreviations typically used by the course staff team.

- `groups`: A key-value object which is based on the values defined in `research_divisions`. For each value of the `research_divisions` array, the program expects a key in this object. Each value then contains an array of objects, which can have an arbitrary key (typically a consecutive number) and the exact folder name.

The design of the configuration file was chosen to support maximum flexibility of all processed terms.

By convention, the name of a folder containing a student's IP repository is their matriculation number. Therefore, for the IP configuration, the root path for all repositories is only specified once (value of `individual_phase.repos_root`). Based on the values loaded from the CSV file (as specified by `individual_phase.csv_file`), the folders are processed. Similarly, for the GP the `projects_root_path` specifies the root location, the concrete path of a group's repository is then the concatenated value of root path and folder name, for example `/path_to_root_folder/folder/project_01`. The ongoing number of the projects per research division is used by the crawler to create a short name which is unique per term and usually used by the course staff when referring to student groups, for example, resulting in *Div. A 01*.

Execution Flow

Listing 6.6 shows a schematic version of the crawler's entry point. Before processing any data, the program checks if the provided program argument value for the number of processes (`args.p`) to be used later is not greater than the number of available Central processing unit (CPU) cores (`cpu_count()`). The execution of the program always starts with loading the available data of the IP since this is the most exhaustive and complete information available. This is done by calling the `__process_individual_phase_file` method. In a first step, it loads the file containing the matriculation numbers, names of students and achieved points. Afterwards, the file including the information about time spent on the project is loaded, assigned to each student and stored in combination with the current term in the database.

The next step of the crawler is to process each group individually (using the data as described in subsection *GitLab's Export Structure*) by calling the `__process_individual_groups` method. It extracts all available project information, including data related to project management and GitLab user information. However, the data related to project management was empty for all examined projects since students are not enforced to use GitLab capabilities (see Section 8.2).

In a next step, the `__process_individual_phase_groups_async` method processes every student's Git repository, transforming the complete history of the `origin/main` or `origin/master` branches — GitLab switched the default branch naming in 2021[76]. This restriction is necessary because many repositories contain numerous remote references that put additional load on the crawler. Especially since only the main branch is used for grading. The processing of the repositories is done in parallel since each one is independent from the others. See subsection *Parallelization* for more information about the parallelism.

After having finished the processing of IP data, the program continues by processing the GP data. For each research division provided in the configuration file, the program loads all group names from the configuration file. Then the project-related information (done by the `ProjectTransformer`) and the user-related information (done by `UserTransformer`) is extracted. The `__process_group_phase_groups_async` is identical to its pendant for the IP (method `__process_individual_phase_groups_async`).

```
1  def run(self):
2      num_cores = min(args.p, cpu_count())
3      __process_individual_phase_file(
4          _config.individual_phase_csv_file,
5          _config.individual_phase_timetracking_csv,
6          _term,
7      )
8      __process_individual_groups(_config.individual_phase_repos_root)
9      __process_individual_phase_groups_async(num_cores,
           _config.individual_phase_repos_root)
10
11     for research_div in _config.research_divisions:
12         _groups = _config.groups.get(research_div)
13         ProjectTransformer(_groups,research_div).run()
14         UserTransformer(_groups).run()
15     __process_group_phase_groups_async(num_cores)
```

Listing 6.6: `MainCrawler` Python schematic class

Parallelization

As previously demonstrated, the crawler can be configured to process available repositories in parallel. This feature was implemented since processing more than 1200 repositories (and more than 100,000 commits) is very time-consuming, even though the branches are limited to one (except for the groups/students which used both `main` and `master` branch).

Before any optimization was done, possible parallelization tasks had to be identified [92]. The processing has to execute some long-running tasks, some sequential tasks as well as to, finally, store the data in the database. The sequential tasks particularly affect loading and processing of CSV files. Moreover, later processing of GP data is depends on this

[76] https://about.gitlab.com/blog/2021/03/10/new-git-default-branch-name/, Accessed: *23.01.2023*

data. However, long-running tasks, such as data related to loading GitLab and Git are easier to parallelize since critical information, as for instance, matriculation numbers, first names or surnames are already present at time of processing. Using these parts, for example by concatenating first name and surname, a commit with any non university email address can be matched with simple email addresses.

Consequently, the following options were identified to be subject for parallelism:

1. Analyzing the commits of a single repository in parallel. Based on research about thread-safety of the GitPython framework, iterating through commits did not turn out to be thread-safe[77]. A workaround for non-thread-safe processing would be to check out each revision individually. This, however, comes with impractical effects of needing much more disk space.
2. Analyzing each student's (for IP) and group's (for GP) repository sequentially and processing multiple students/groups at the same time.

The final decision regarding the prototype crawler fell on the latter variant. In particular, the non-thread-safety of the framework used was not manageable in any way.

Python's *multiprocessing*[78] library — which is part of the language's standard library — provides an API to fully exploit the multiprocessor environment [67]. It implements parallel process programming for shared memory systems. A process is an application's execution entity that can include numerous concurrent threads. As opposed to a process, managing a thread is less costly [82]. To apply the multiprocessing principle to the crawler, its input data must be distributed among different jobs which are then organized in a pool. As a consequence, these workers are running, as aforementioned, simultaneously.

Referring to Listing 6.6, all *async* postfixed methods are designed in the same way to allow parallel processing of the data. Listing 6.7 shows an example of how parallel processing is implemented by using the *multiprocessing* `Pool` in combination with the `apply_async` method. That method does not spawn a number of processes, it evaluates each call within one process by spawning numerous threads. This principle is called task-based parallelism [60]. As shown in line 8, the call to the `async_process_call` method is just schematic. Algorithm 6.1 illustrates how the asynchronous processing and error handling is done in an abstract way. After successful calls to the `apply_async` processing, the main method waits for the arrival of return values. The waiting part is shown by line 13 and onwards, `apply_async` returns an *AsyncResult* object[79] which provides the `get` method to return the result when it arrives, or raises an exception if a timeout occurs. If any exception occurs, it is caught (line 16) and the pool is forced to terminate. This ensures that no data is corrupted, and the error tracing is easier. In the successful case the `join` method is called so that the program waits until all threads are done.

[77] https://github.com/gitpython-developers/GitPython/issues/584, Accessed: *23.01.2023*
[78] https://docs.python.org/3.10/library/multiprocessing.html, Accessed: *23.01.2023*
[79] https://docs.python.org/3.10/library/multiprocessing.html#multiprocessing.pool.AsyncResult, Accessed: *23.01.2023*

```python
def __process_individual_phase_groups_async(self, num_cores: int, directory:
    str):
    p = Path(directory)
    dir_list = [x for x in p.iterdir() if x.is_dir()]
    results = []
    with Pool(num_cores) as pool:
        for group_path in dir_list:
            results.append(
                pool.apply_async(
                    async_process_call,
                    args=(0),
                )
            )
        for result in results:
            try:
                result.get()
            except Exception as e:
                pool.terminate() # kill all processes in the pool
            else:
                # wait for all submitted tasks to complete:
                pool.close()
                pool.join()
```

Listing 6.7: `__process_individual_phase_groups_async` Python schematic Code

As mentioned before, Algorithm 6.1 schematically illustrates how methods are called and errors are handled during parallel processing. This is especially necessary because, for example, some terms have students assigned to multiple groups. The creation of the corresponding GitLab user in the crawler database fails because two threads try to store the same value, but they violate a constraint on unique values. However, a simple retry (which is shown in the first `catch`-block; line 5-9) solved this problem.

Algorithm 6.1: async_process_call

```
   Input : retry
 1 try:
 2 |   Process data with database access
 3 |   return
 4 catch sqlalchemy.Error:
 5 |   if retry ≤ 3 then
 6 |   |   async_process_call(retry + 1);
 7 |   else
 8 |   |   raise exception
 9 |   end
10 catch python.Exception:
11 |   raise exception
12 end
```

The asynchronous implementation is, opposed to Listings 6.6 and 6.7, only provided as an algorithm since the concrete implementation highly depends on the processed elements and would not add additional value.

Database Schema

The database schema was divided into two main schemas: The *csv* and the *gitlab* schema. The idea behind the division is to separate the data into logical groups based on their origin.

Recall that, the base data was originally provided by the course staff and stored in a CSV file. This can be split in basically three main domain objects, the student, their result in the individual phase and the term in which they attended the class. These relations are reflected in the schema, as illustrated in Figure 6.2.

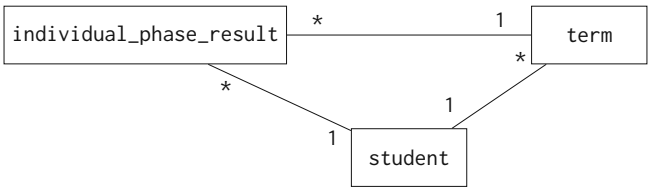


Figure 6.2: *csv* Schema

Compared to the *csv* schema, the *gitlab* schema — as demonstrated in Figure 6.3 — is more complex. Since it also stores more information. As already mentioned in the *Execution Flow* subsection, the data extracted from GitLab is matched against the CSV data. As Figure 6.3 shows, where connections between the two schemas ensure that the stored data is valid. With these constraints, the program checks that no orphan data is stored and that everything belongs to an available term and an available student. These references are marked with the *csv* prefix.

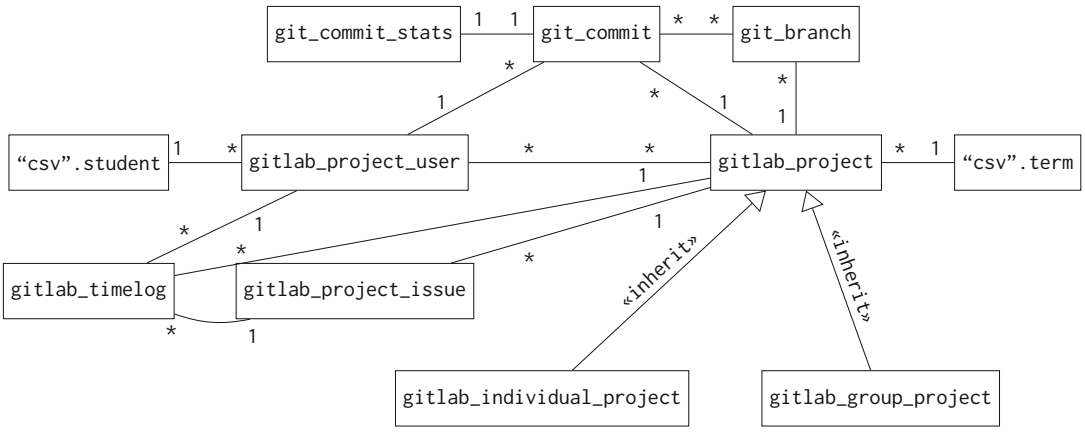


Figure 6.3: *gitlab* Schema

6.2.3 Pseudonymization

As already addressed in Section 6.2.1, for the pseudonymization part Python's pandas- and Faker-library were used. Foremost, the critical data elements had to be identified. In fact they were as follows:

1. A student's
   a) First name
   b) Last name
   c) Matriculation Number
   d) GitLab username
   e) GitLab email
2. A group's
   a) full name
   b) short name
3. Research Division
4. The name of a term

For many types the Faker library provides methods to create meaningful random values, for example, first names, last names, etc. To provide an example, the anonymization of the first name is shown in Listing 6.8.

Firstly, all available data is loaded from the database, using pandas' `read_sql_table` method. For this task to be possible, the method needs to know which table and which connection should be used: The table is dynamically loaded by the `__tablename__` value of the `StudentModel` class, the connection `con` is reused. As at the time of calling that method, there already is an open connection.

In a next step, lines 2 to 4 show that all values of the `firstname` column (denoted by the square brackets) are mapped to an inline function where each value gets a random first name from the Faker library assigned. Afterwards, the new values can be stored in the database. Line 5 onwards shows how the data is finally stored. In addition, the `to_sql` method allows to specify the name of the table, the connection and the schema. In fact, only the schema changes, otherwise the program would override the original values (see Section 6.3 for more information about the different schemas). In addition, there is also the possibility to define a strategy if the table on that particular schema already exists. The `if_exists` parameter specifies the behavior in that case: *replace* means that if that table already exists, it is replaced by the one with the new values.

```
_all_students_df = pd.read_sql_table(f"{StudentModel.__tablename__}", con=engine)
_all_students_df["firstname"] = _all_students_df["firstname"].map(
    {firstname: faker.first_name() for firstname in
        _all_students_df["firstname"]}
)
_all_students_df.to_sql(
    name=StudentModel.__tablename__,
    con=engine,
    schema=INDIVIDUAL_PHASE_DBSCHEMA_ANONYMIZED,
    if_exists="replace",
)
```

Listing 6.8: Firstname anonymization

To transform the student's last name, the same method as shown in Listing 6.8 is used. All other identified properties at the beginning of this subsection are handled slightly different: For example, if there are two students in the same group who share the same last name, they get different last names with Listing 6.8's principle, therefore, cannot be identified anymore. The difference is that every other property listed should remain unique, as it is in the original database. Thus, the inline mapping method is called applying pandas' `unique()` method, as demonstrated in line 7 of Listing 6.9.

```
[...]
_all_students_df["matriculation_number"] = _all_students_df[
        "matriculation_number"
    ].map(
        {
            mat_nr: faker.numerify(text="#########")
            for mat_nr in _all_students_df.get("matriculation_number").unique()
        }
    )
[...]
```

Listing 6.9: Unique property anonymization

It is similar with the research section and the term name. However, it is then still possible to determine which group has taken the course in which term by means of the time series data. For example, since the time tracking data or Git information is based on a timestamp, and thus cannot be modified.

6.3 Data Loading

After completing the steps of the ETL process as outlined in Section 6.2, the data is stored in the database. To make it accessible to be later used by Apache Superset, the data has to be loaded from the database again. To allow maximum flexibility for the future tools, the loading process was mostly implemented using PostgreSQL's Views and Materialized Views (MVs).

PostgreSQL allows creating a view on table(s) either through materialization or view expansion. However, the approach has to be chosen beforehand [114]. In contrast to the classical PostgreSQL View, a Materialized View works in a similar way but persists the result data in a table-like form. This leads to shorter access times, in particular for a great number of database `JOIN` operations [12, 58]. The distinction if either a View or a Materialized View should be used was made applying a simple rule: If there are numerous, independent Views V_1 and V_2 based on the data which is provided by another View V_m, then V_m is implemented as Materialized View. So only the top-most layer, which is accessed by Apache Superset, is implemented as View.

6.3.1 Materialized Views

Listing 6.10 illustrates the PostgreSQL syntax for creating a MV:

- `view_name`: This element defines the name of the MV.
- `query`: The placeholder must be replaced with the concrete SQL query.
- `WITH [NO] DATA`: One can choose between two options when creating a MV:
  - `WITH DATA`: This option populates the data into the MV at time of creation.
  - `WITH NO DATA`: This option marks the MV as incomprehensible, meaning that this MV cannot be accessed until the data is loaded (which must be done at a later moment).

```
1 CREATE MATERIALIZED VIEW view_name AS query WITH [NO] DATA;
```

Listing 6.10: Create Materialized View syntax in PostgreSQL

After creating a MV, it can be accessed like a SQL tables, for example, `SELECT * FROM view_name`. In the following query of Listing 6.11 only the query parameter is illustrated.

```
1  SELECT
2      _project.user_id AS user_id,
3      _project.user_username AS user_username,
4      _project.user_email AS user_email,
5      _project.student_matriculation_number AS student_matriculation_number,
6      _project.student_firstname AS student_firstname,
7      _project.student_lastname AS student_lastname,
8      _project.project_id AS project_id,
9      _project.project_name_filterable AS project_name_filterable,
10     _project.project_full_name AS project_full_name,
11     _project.project_short_name AS project_short_name,
12     _project.term_id AS term_id,
13     _project.term_name AS term_name
14 FROM "gitlab".gitlab_individual_project _ind_project
15 LEFT JOIN "individual".ind_user_project_mview _project ON _project.project_id =
       _ind_project.id
16 LEFT JOIN "gitlab".git_branch _branch ON _project.project_id = _branch.project_id
17 WHERE _branch.id IS NULL
```

Listing 6.11: `ind_empty_repos_mview` Materialized View of empty repositories

6.3.2 Views

As already mentioned, the Materialized Views also serve as a basis for additional SQL Views. Likewise, as shown in Listing 6.10, Listing 6.12 illustrates how to create a View in PostgreSQL.

```
1 CREATE VIEW view_name AS query;
```

Listing 6.12: Create View syntax in PostgreSQL

Based on a MV called `student_term_result_mview` stored in the `public` schema, loading the non-empty repositories, as shown in Listing 6.13, is done by removing the students who are present in the result set of `ind_empty_repos_mview`. The part to remove the respective students is located in the `WHERE` clause starting at line 4.

```
1  SELECT
2      *
3  FROM public.student_term_result_mview _strm
4  WHERE NOT EXISTS (
5      SELECT
6          *
7      FROM "individual".ind_user_empty_repos _empty_repo
8      WHERE _empty_repo.student_id = _strm.student_id AND
9             _empty_repo.student_matriculation_number =
                  _strm.student_matriculation_number AND
10            _empty_repo.term_id = _strm.term_id
11 )
```

Listing 6.13: `ind_results_non_empty_repos` View of all non-empty Individual Phase repositories

Listing 6.14 visualizes the SQL query which extracts the number of attempts per student. Firstly, the `attempts_per_student` subquery of lines 5-9 counts the number of occurrences of a student's matriculation number Secondly, the outer `SELECT` statement counts the number of students per attempt value.

Another notable View is `ind_commits_points_cat`, as displayed in Listing 6.15, which categorizes each IP result by the number of commits and the achieved points. The evaluation of Listing 6.15 is shown in Table 6.3.

For the sake of completeness, all remaining Views and Materialized Views are located in Appendix A and B.

```sql
SELECT
    attempts_per_student.attempts AS attempt,
    COUNT(DISTINCT(student_matriculation_number))
FROM (
    SELECT
        student_matriculation_number,
        COUNT(*) AS attempts
    FROM "public".student_term_result_mview
    GROUP BY student_matriculation_number
) AS attempts_per_student
GROUP BY attempts_per_student.attempts
```

Listing 6.14: `student_attempts_view` View implementation

attempt	count
1	560
2	203
3	43
4	10

Table 6.2: Result of Listing 6.14 for all four terms

```sql
SELECT
    COUNT(DISTINCT(_ind.commit_hash)) AS ind_commits_count,
    CASE
        WHEN COUNT(DISTINCT(_ind.commit_hash)) < 9 THEN '< 9'
        WHEN COUNT(DISTINCT(_ind.commit_hash)) < 16 THEN '10-15'
        ELSE '16+'
    END AS ind_commits_count_cat,
    _ind.committer_student_matriculation_number AS student_matriculation_number,
    _ind.term_id AS term_id,
    _ind.term_name AS term_name,
    _results.result_points AS ind_result_points,
    CASE
        WHEN _results.result_points < 40 THEN '5 (N5)'
        WHEN _results.result_points < 50 THEN '4 (G4)'
        WHEN _results.result_points < 60 THEN '3 (B3)'
        WHEN _results.result_points < 70 THEN '2 (G2)'
        ELSE '1 (S1)'
    END AS grade
FROM "individual".ind_commit_user_mview _ind
JOIN "individual".ind_results_non_empty_repos _results
    ON _results.student_matriculation_number =
        _ind.committer_student_matriculation_number
    AND _results.term_id = _ind.term_id
GROUP BY _ind.committer_student_matriculation_number, _ind.term_name,
    _ind.term_id, _results.result_points
```

Listing 6.15: `ind_commits_points_cat` View of all Individual Phase students with categorized points and commits

6.3.3 Pseudonymization

As discussed in Section 6.2.3, the pseudonymized data is stored in a schema postfixed with `_pseudonymized`, namely `csv_pseudonymized` and `gitlab_pseudonymized`. The structure of the tables is identical since pandas' DataFrame clones and alters the data.

ind_commits_count	ind_commits_count_cat	student_matriculation_number	term_id	term_name	ind_result_points	grade
29	16+	12345678	3	21SS	40	4 (G4)
98	16+	23456789	3	21SS	3	5 (N5)
52	16+	34567890	4	21WS	70	1 (S1)
52	16+	45678901	1	20SS	70	1 (S1)
⋮						

Table 6.3: Result of View `ind_commits_points_cat`

Therefore, any of the following tables can be interchanged without losing information but anonymizing critical data. However, all other constraints are lost, so data of this table(s) cannot be verified for integrity.

Non-pseudonymized table	Pseudonymized table
"csv".student	"csv_pseudonymized".student
"gitlab".gitlab_project	"gitlab_pseudonymized".gitlab_project
"gitlab".gitlab_project_user	"gitlab_pseudonymized".gitlab_project_user

Table 6.4: Pseudonymized SQL tables lineup

These pseudonymized tables, as listed in Table 6.4, force the necessity to create new Views and Materialized Views. All of them are implemented as their non-pseudonymized siblings, but the critical tables are changed. Each View and MV is prefixed with `anon_`. For example, in Listing 6.13 line 7 is swapped with `FROM "individual".anon_ind_user_empty_repos_empty_repo`.

Non-anonymized (Materialized) Views	Pseudonymized (Materialized) Views
"individual".ind_empty_repos_mview	"individual".anon_ind_empty_repos_mview
"individual".ind_user_empty_repos	"individual".anon_ind_user_empty_repos
"individual".ind_results_non_empty_repos	"individual".anon_ind_results_non_empty_repos
"individual".ind_commits_points_cat	"individual".anon_ind_commits_points_cat
"individual".ind_commit_stats_user_mview	"individual".anon_ind_commit_stats_user_mview
"group".grp_commits_cat	"group".anon_grp_commits_cat
"group".grp_user_timelog_view	"group".anon_grp_user_timelog_view
"group".grp_commit_stats_user_mview	"group".anon_grp_commit_stats_user_mview
"public".student_grades_view	"public".student_grades_view
"public".student_term_result_mview	"public".anon_student_term_result_mview

Table 6.5: Pseudonymized SQL (Materialized) Views lineup

Any other SQL-View which is not mentioned in Table 6.5 is, therefore, not affected by any critical data and does not need a sibling View. This clear separation of Views and Materialized Views allows a relatively easy integration into Apache Superset, which was achieved by using a virtual dataset combined with the Jinja templating. Jinja[80] is a web templating library for Python that is designed to be flexible, fast and secure, modeled on Django's[81] templates. It is fast, widely used and secure with the optional sandboxed

[80] https://jinja.palletsprojects.com/en/3.1.x/, Accessed: *23.01.2023*
[81] https://www.djangoproject.com/, Accessed: *23.01.2023*

template execution environment to create HTML, XML or other markup formats that can be sent via HTTP requests [80, 100].

The SQL templating functionality allows getting the ID of any logged-in user before executing a Query. Hence, one can distinguish between different Views based on the user's ID only. Listing 6.16 illustrates how templating is implemented in the prototype. For this prototype only two users were present: the default `superset` and a `public` user. The public user has ID 2 and, therefore, the system always loads the table as specified in line 3 whereas any other user always gets access to the view of line 5.

```
1 SELECT * FROM
2 {% if (current_user_id() == '2') %}
3 "schema".<anon_table>
4 {% else %}
5 "schema".<original_table>
6 {% endif %}
```

Listing 6.16: Jinja templating functionality example

CHAPTER 7

Education Intelligence Visualization

Successful visualization is essential for the comprehension, examination and sharing of analysis results. For this reason, virtually all analytical components have visualization features. However, using additional special visualization components or reporting systems with powerful visualization capabilities are advisable [4].

The subsequent section provides a summary of the prototypical Education Intelligence (EI) visualization system. Based on the concepts of Section 5.1, using Apache Superset and its *plug'n'play* nature of creating charts and dashboards, the built-in capabilities were used to recreate all mock-ups which do not require additional tools and also no coding.

7.1 Data Visualization - Group Phase Data

Based on the idea of the *Group Overview Card* as shown in Figure 5.1 and 5.10 a Group Phase dashboard was created, which is demonstrated in Figure 7.1, 7.2 and 7.3. At the top of Figure 7.1 *general information tiles* can be seen, which simply show numbers for orientation:

- Students: The number of unique students whose data is currently display
- Timelogs: The number of timelogs which represents the basis of the time tracking graph in Figure 7.4
- Issues: Overall number of GitLab Issues
- Open Issues: Number of Issues which do not have a `closed_at` date set
- Timelogs per week: History of added timelogs over time

At the top left-hand side there is a tabular information representation of the selected students. Below these tiles, two tabs are placed: the *Repository Statistics* and *Project Statistics* tab. The first tab shows the entire information which resulted from the MSR-process, as in Figures 7.2 and 7.3. The second tab shows the extracted information related to project management, as in Figures 7.4 and 7.5.

7.1.1 Repository Statistics

The repository statistics shown in Figure 7.1 start with *general information tiles* and two stacked bar charts, summarizing the commits per week and commits per daytime (24-hour format). The general information is followed by Figure 7.2, which shows a table on the left side presenting all commit orphans[82] and a bar chart on the right side which demonstrates the number of commit orphans per group. At the bottom there are box plots showing the number of commits per User for each group, similar to the mock-up in Figure 5.2.

Lastly, for the repository statistics, Figure 7.3 shows simple Git statistics on a timeline (always based on the selected time granularity): The first line chart illustrates the commits per time grain. The second shows on the positive x-axis the insertions per commit and on the negative x-axis the deletions per commit. The last chart represents the number of files affected per commit and time grain.

7.1.2 Project Statistics

The second section of the Group Phase dashboard focuses on the project related visualizations. As shown in Figure 7.4, the top most time series line graph shows the time tracked based on the existing Timelog-entries per group, inspired by the concept of Figure 5.4. An additional tab allows switching between the representation as *Total per week* and *Running Total*. Similarly to the commit Distribution box plot, there is a box plot for the time spent on the project per group, as shown at the bottom.

Secondly, Figure 7.5 shows additional statistics based on the time tracking information. A funnel chart is displayed at the top of the left column, with terms sorted in descending order by total time spent. To allow comparison of a winter term (which has fewer students) with a summer term, the values are presented in percentages. One can see that the order shown is: Term 4 (124%), Term 2 (122%), Term 1 (120%) and Term 3 (115%). Below the Funnel chart there is a simple pie chart indicating the distribution of group sizes, where three different group sizes occurred: groups of five, six or seven students.

The right-hand side of Figure 7.5 has a tab-pane with bar charts, allowing to switch between views. The default tab consists of two bar charts: the top graph shows the absolute value of *time spent on the project per term and group* and the bottom graph shows the value in relation to the 110 hours in percentage. When switching the tab, the

[82] A *commit orphan* is a commit where neither the committer nor the author could be assigned to any student in that group and term.

bar chart is finer, showing the same bar chart per term, group and student. Figure 7.6 demonstrates the difference in detail.

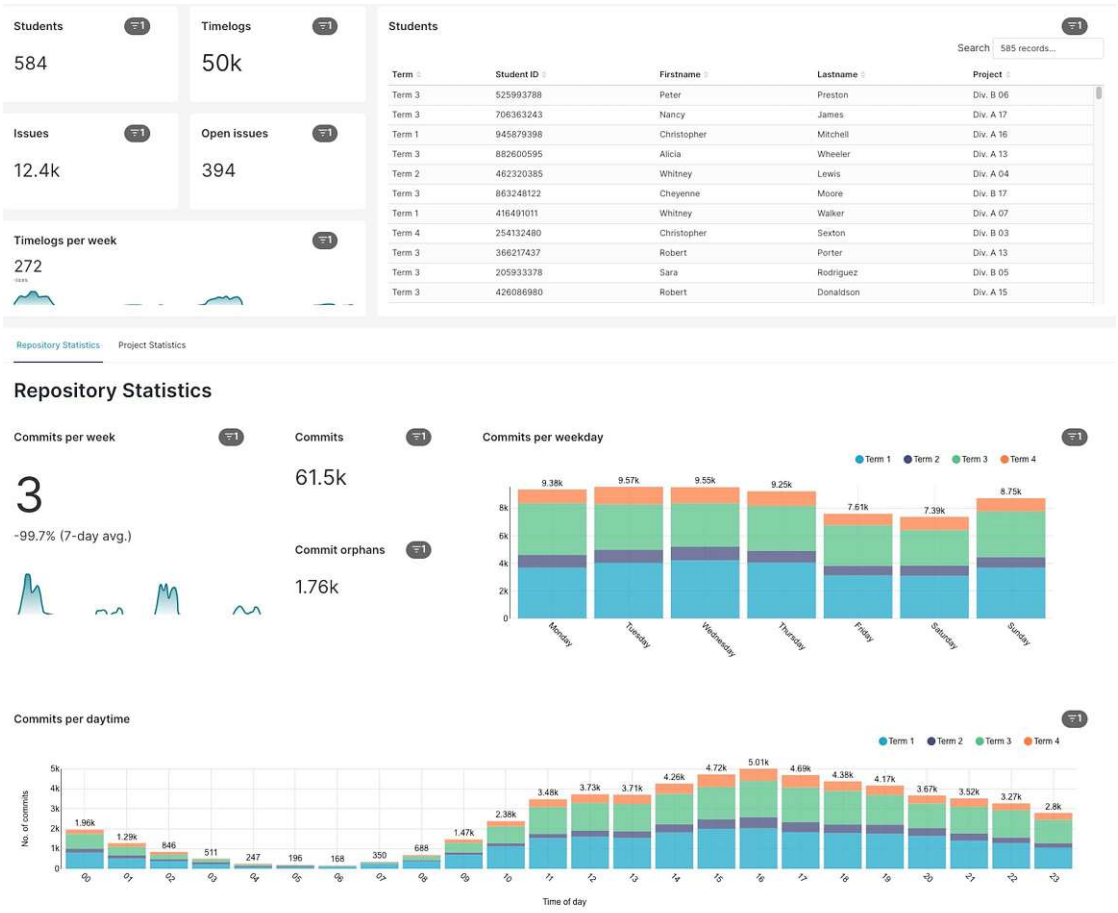


Figure 7.1: Group Phase dashboard (Part 1)

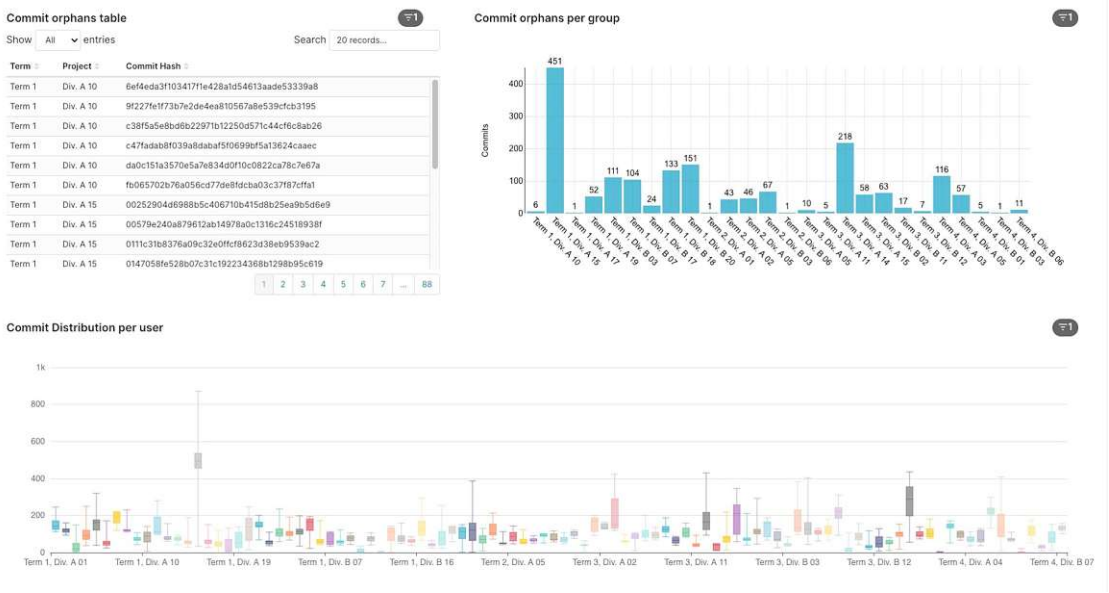


Figure 7.2: Group Phase dashboard (Part 2)

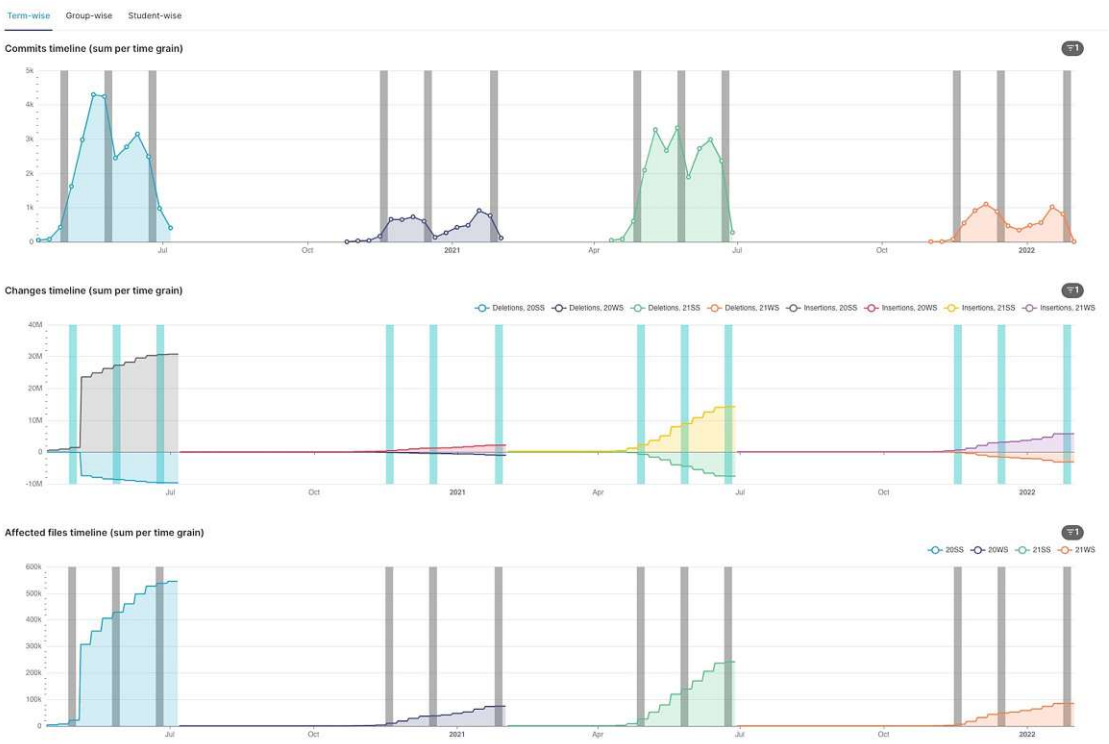


Figure 7.3: Group Phase dashboard (Part 3)

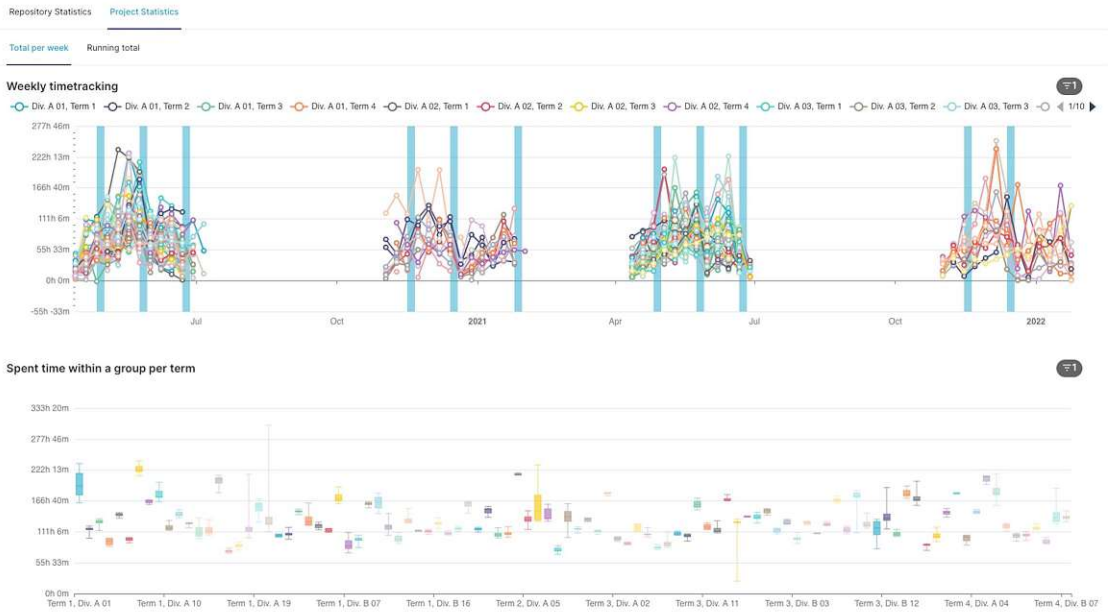


Figure 7.4: Group Phase dashboard (Part 4)

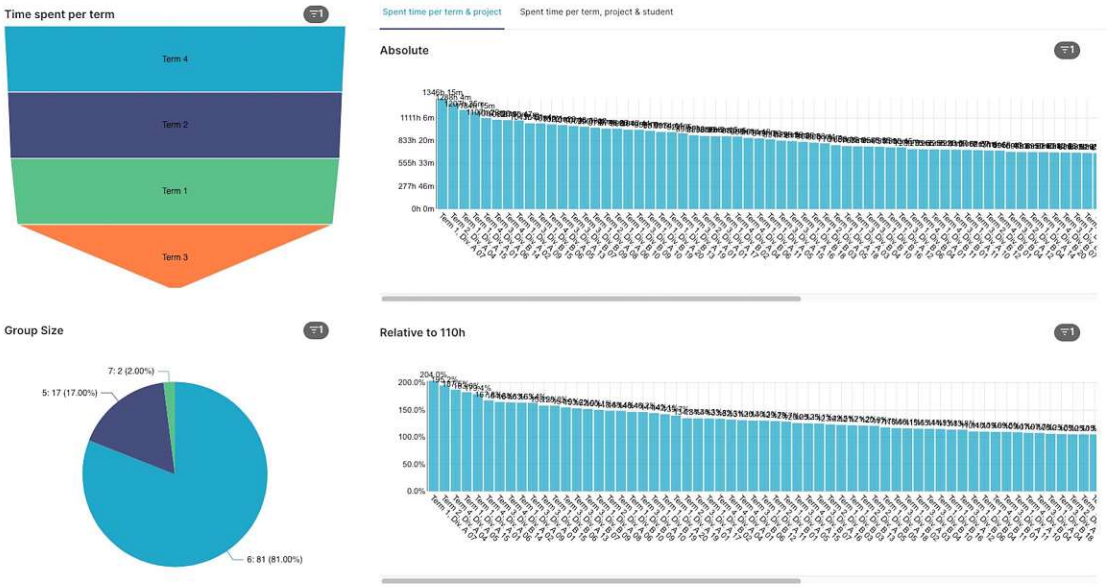
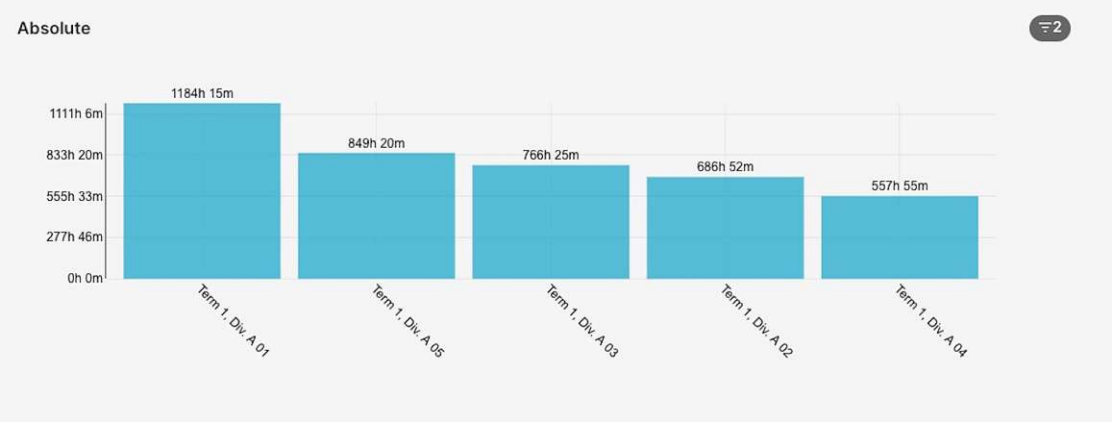
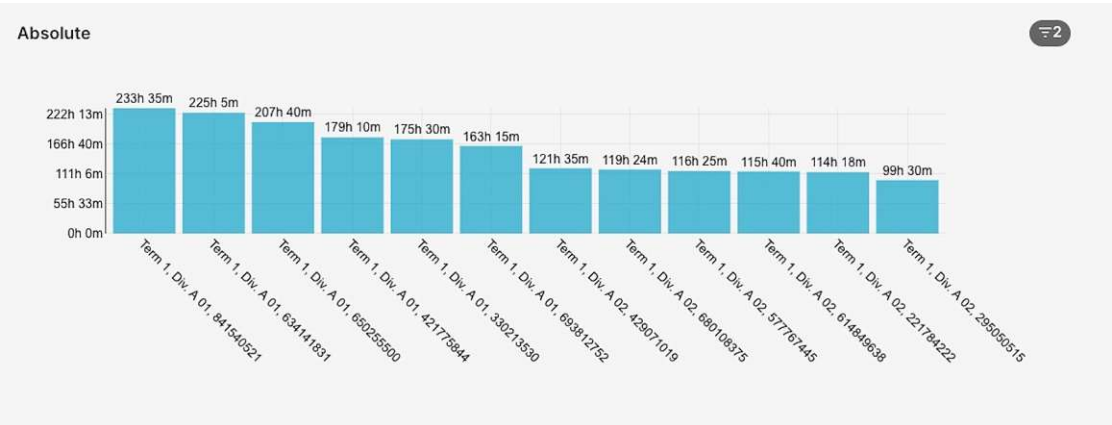


Figure 7.5: Group Phase dashboard (Part 5)



(a) Time spent on the project per term and group



(b) Time spent on the project per term, group and student

Figure 7.6: Time spent on the project during the Group Phase (Bar charts)

7.2 Data Visualization - Individual Phase Data

A similar dashboard, as for the Group Phase data, was also created to visualize the data of the Individual Phase, as shown in Figure 7.7 and 7.8. These visualizations follow the same principles as the ones for the Group Phase. Since students track their time manually, no project related statistics can be extracted. Hence, only repository statistics are considered for the IP.

Starting with Figure 7.7, at the top there are four *general information tiles*, followed by two histograms below on the left. In addition, there is a line chart illustrating the submissions per semester and a box plot graph to the right that visualizes the points achieved per semester during the IP.

Secondly, Figure 7.8 starts with a bar chart at the top left visualizing the *attempts per students* and a pie chart on the right side illustrating the *students per attempt*. The second row shows the distribution of commits per weekday on the left side and per daytime on the right side as stacked bar charts. Lastly, the time series line chart at the bottom visualizes the moments of commits as *sum per day* or as *running total*, depending on the selected tab.

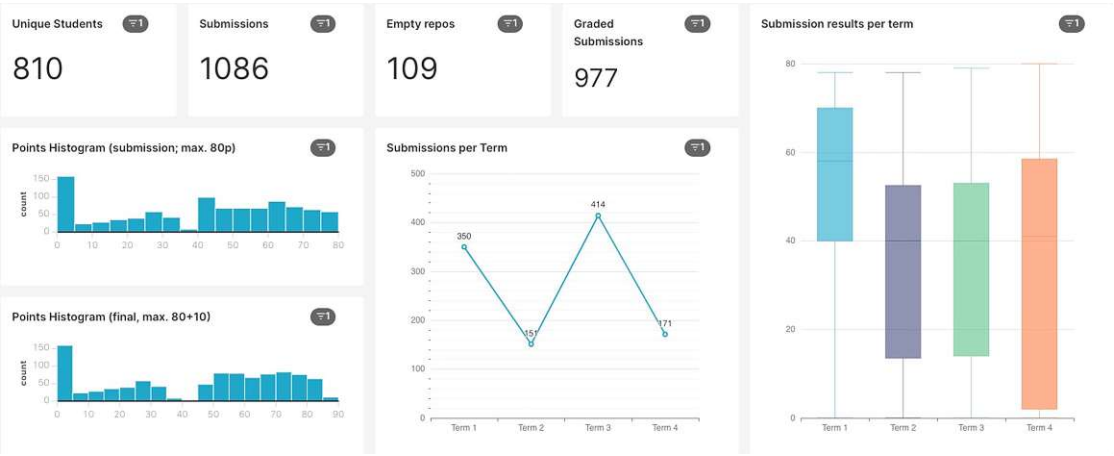


Figure 7.7: Individual Phase dashboard (Part 1)

Figure 7.8: Individual Phase dashboard (Part 2)

CHAPTER 8

Evaluation and Results

This chapter presents the results of the prototype's crawler implementation of Chapter 6 and the data analysis. First, Section 8.1 outlines the result of the execution speed-up of the implementation. In Section 8.2 a data exploration step is presented to draw a baseline about the expectations within the data. The last part of the results demonstrates an analysis of visualized data in Section 8.3. The chapter concludes with the evaluation of the prototype in Section 8.4 and a discussion of the results in Section 8.5.

8.1 Execution Speed-up

The extraction process was executed on an iMac (Retina 5K, 27-inch, Late 2015) using a 3.4 GHz Quad-Core Intel Core i7 processor, 16 GB 2666MHz DDR4 RAM with macOS Catalina 10.15.7 installed.

The program was implemented, as aforementioned in Section 6.2, using the Python interpreter version 3.10.4 of March 24, 2022. The program was executed using Python's `venv`[83] module. The `venv` module allows creating lightweight *virtual environments* which can be isolated from the system. Each virtual environment has its own Python binary and can have its own independent collection of Python packages installed in its directories [86]. As already mentioned in the *Program Configuration* subsection, the crawler can be configured to use several threads to speed up program execution and data processing.

To measure the surplus value of the parallelization as suggested in the *Parallelization* subsection, the following program configurations were executed using the `time` tool on macOS measuring the execution time:

```
1 python3 main.py --file <file> -p <np> --project -db "<db>" --git --stats
```

Listing 8.1: Program execution

[83] https://docs.python.org/3.10/library/venv.html, Accessed: *23.01.2023*

Listing 8.1 shows how to start the program using placeholder for the program arguments. Where `<np>` was replaced with 1 and 8, and `<file>` with the values of the following: `2020SS.json`, `2020WS.json`, `2021SS.json`, `2021WS.json`. These combinations resulted in eight different program executions. The `<db>` option is only a placeholder (and irrelevant in this context) since every run simply got a new database. This configuration is as extensive as possible in terms of time, as the Git processing already takes some time and evaluating stats per commit (meaning insertions, deletions and affected files) adds extra processing time on top. The `time` command line tool provides the following three values as output:

- *real* - duration of the call from start to finish. It refers to the time elapsed between pressing the Enter key and the completion of the command.
- *user* - amount of CPU time spent in user mode.
- *sys* - amount of CPU time spent in kernel mode.

To calculate the speed-up, the value of the *real* output is considered only. Table 8.1 shows the time measured and the number of threads used to process the data. The fourth column is the calculated speed-up which compares the time of one thread t_1 with eight threads t_8: $S_p = \frac{t_1}{t_8}$.

Term	Threads t_i	Avg. elapsed time	S_p
SS20	1	46m 48.268s	3.405
	8	13m 44.634s	
WS20	1	11m 33.315s	3.478
	8	3m 19.296s	
SS21	1	43m 18.307s	3.476
	8	12m 27.425s	
WS21	1	15m 59.821s	3.479
	8	4m 35.849s	

Table 8.1: Measured speed up using parallel data processing

To sum up, the implemented parallelization as suggested in subsection *Parallelization* lead to an average speed up of $\frac{3.405+3.478+3.476+3.479}{4} = 3.4595$ (based on ten runs each).

8.2 Expected Numbers

To verify the correctness of the extracted values and numbers (for example, when counting the database rows), the raw data was analyzed using different tools. For simple CSV files, the `wc`[84] command line tool was used, for json/ndjson files the `jq`[85] command line tool was used.

[84] `https://linuxize.com/post/linux-wc-command/`, Accessed: *23.01.2023*
[85] `https://stedolan.github.io/jq/`, Accessed: *23.01.2023*

The `wc [-clmw] [file ...]` command accepts input files while having various program options available. To verify the exact number of entries in a CSV file, it is sufficient to call it as `wc -l <file>`, which simply prints the number of lines.

For processing `ndjson` files, the lightweight processing tool called `jq` was used to extract certain properties of the JSON. This tool is a small and lightweight open-source program for displaying, filtering, processing and transforming the contents of files. It is very useful for iterating through JSON array items, such as the stored objects in GitLab's exported `ndjson` files. It is also possible to filter the contents of a file to show only certain values or to traverse all objects in an array. As `jq` supports comparisons, conditional and regular expressions as well as various data transformation methods, it is highly suitable for extracting available information for verification purposes.

In the following listings the commands used to extract the numbers, which were used as baseline, are shown. To extract the number of issues per project, the bash commands of Listing 8.2 can be used.

```
1 cat "./<foldername>/*/export/tree/project/issues.ndjson" | wc -l
```

Listing 8.2: Count issues using `wc`

Listing 8.3 counts the number of time tracking entries of Issues and MRs. As already mentioned in Section 6.1, GitLab allows time tracking for both, so values must be summed up at the end. The `jq` program is called with the `-c` option, which results, according to the man-page, in a "compact instead of pretty-printed output." That results in a single line output for each JSON object instead of a more readable one. However, it makes a significant difference since the "pretty-printed output" prints an object with newlines after each property, whereas compact print everything in one line. The `.timelogs[]` then selects the all array elements of the JSON object.

```
1 cat "./<foldername>/*/export/tree/project/issues.ndjson" | jq -c '.timelogs[]' |
      wc -l
2 cat "./<foldername>/*/export/tree/project/merge_requests.ndjson" | jq -c
      '.timelogs[]' | wc -l
```

Listing 8.3: Count timelogs using `jq` and `wc`

Listing 8.4 counts the number of users with the *Developer* role. Counting is done by taking the unchanged input (with the dot argument) and piping the output of the respective object into the `select` function. As a consequence, objects can be filtered based on the `access_level` property.

```
cat "./<foldername>/*/export/tree/project/project_members.ndjson" | jq -c '. |
    select(.access_level==30)' | wc -l
```

Listing 8.4: Count GitLab project user using `jq` and `wc`

The number of commits of a project can easily be extracted using the Git command line tool itself, as in Listing 8.5. As proposed in Section 6.2, only commits of `origin/main` and `origin/master` were considered. Therefore, a distinction was done by specifying the respective remote branch.

```
COMMITS=0;
for d in ./<projects_root>/*/repo ; do
    cd $d
    VALUE_MASTER=$(git rev-list --count origin/master)
    VALUE_MAIN=$(git rev-list --count origin/main)
    echo "$d", "origin/master", $VALUE_MASTER, "origin/main", $VALUE_MAIN
    COMMITS=$((COMMITS+VALUE_MAIN+VALUE_MASTER))
    cd ../../../
done;
echo $COMMITS
```

Listing 8.5: Count commits of `origin/main` and `origin/master` branch

In the following, based on the exported archives, the expected numbers of each term are discussed in more detail. As Tables 8.2 and 8.4 show, in summer terms (denoted with prefix *SS*) more students take the class than in winter terms (denoted with prefix *WS*). The high number of students in the summer term does not come as a surprise as it is scheduled for the summer term in the university's recommended path of study.

8.2.1 Individual Phase

With regard to the IP, Table 8.2 and 8.3 show the values for all available terms. As expected, due to the course design, there are no entries found for Issues or time tracking (Timelogs). For the sake of completeness these columns (*Issues* and *Timelogs*) are also shown in Table 8.2. The *Results* column shows the number of entries in the Excel file which is used to store the student's points achieved during the grading phase. In addition to the term-wise values of Table 8.2, in the course of the four terms examined in this study, 816 students attended the IP of the class. Since for every term a new Excel file is created, and students might retake the class, the number of the *Results* column is different to the number of attended students.

Special attention should be given to the *Empty repos* column or Table 8.3. This number indicates how many students were registered for the course at the beginning of the

semester but did not push any Code on the remote repository. To be more specific, a repository is treated as empty if no commits have been pushed to the `origin/main` or `origin/master` branch.

Notable differences are also found in the data about the summer term 2020: First of all, 399 students were registered to the class at the beginning of the term. In the exported GitLab projects, however, only 389 students were found. The Excel file, on the other hand, listed only 350 graded students. Hence, 39 out of the 49 students who did not complete or pass the class logged in at least once. As the user management of GitLab is based on a university wide authentication provider, the GitLab user is not created before the user first signs in[86].

Taking a closer look at Table 8.2, one can see that for both terms SS21 and WS21 the number of results is one higher than the number of identified GitLab users (414 vs. 413 and 171 vs. 170). This indicates again that at least one student did not log in. One user also committed Code into the repository (and therefore was graded), the other did not push any code into the codebase.

Term	Results	Issues	Timelogs	Commits	GitLab User
SS20	350	0	0	14265	389
WS20	151	0	0	4228	151
SS21	414	0	0	17523	413
WS21	171	0	0	6839	170
$\sum$	1086	0	0	42855	1123

Table 8.2: Expected numbers for the Individual Phase

Term	Results	Empty repos	Graded results	... thereof negative
SS20	350	23 (6.57%)	327	80 (24.46%)
WS20	151	25 (16.56%)	126	59 (46.83%)
SS21	414	33 (7.97%)	381	185 (48.56%)
WS21	171	28 (16.37%)	143	68 (47.55%)
$\sum$	1086	109 (10.04%)	977	392 (40.12%)

Table 8.3: Grading numbers of the Individual Phase

[86] https://docs.gitlab.com/15.0/ee/user/profile/account/create_accounts.html#create-users-through-authentication-integrations, Accessed: *23.01.2023*

8.2.2 Group Phase

Similarly to the IP, the extracted reference values for the GP are listed in Table 8.4.

Noteworthy discrepancies of extracted numbers are also found in this dataset. The dataset of summer term 2020, in particular, contains several inconsistencies:

Firstly, the total number of Timelogs is 22146. However, 5 are not from a user with the *Developer* role, so they are not considered.

The number of GitLab users is exactly 250 but as there is a user which is in two projects the unique number is 249. Moreover, since the COVID-19 pandemic allowed young people to volunteer for social or military service, one student dropped out after the IP but still appeared in GitLab as a user for the GP. In addition, a tutor is in one project listed with *Developer* role. Thus, the number of students graded after the Group Phase is three less than the number of users.

Otherwise, no more disparity was found.

Term	Projects	Issues	Timelogs	Commits	GitLab User	Graded students
SS20	41	5317	22141	25942	249	246
WS20	12	1338	4676	5977	67	67
SS21	34	4641	17471	22340	196	196
WS21	13	1126	5707	7239	75	75
$\sum$	100	12422	49995	61498	587	584

Table 8.4: Expected numbers for the Group Phase

8.3 Data Analysis

As already mentioned at the beginning of this thesis, the proposing Education Intelligence (EI) in a software engineering education context encloses the entire information required to support lecturers and supervisors in evaluating the student's performance. Therefore, the following analyses can also be used in a productive environment as a fact-based intelligence system for software engineering courses. Within the scope of the present thesis, Within the scope of this study, a detailed analysis of the collected data is provided in the following subsections so that RQ3 can be answered accordingly.

In addition to the visualization and analysis capabilities of Apache Superset, firstly, Python was used in combination with Matplotlib[87] to create the Violin plots and, secondly, Tableau[88] was used to generate advanced visualizations which exceed the capabilities of Apache Superset and Python.

[87] https://matplotlib.org/, Accessed: *23.01.2023*
[88] https://www.tableau.com/, Accessed: *23.01.2023*

8.3.1 Grade Analysis

Recall from Section 2.1 that students receive two grades, one grade for the Individual Phase and one for the Group Phase. The student attending the IP only receive feedback by means of the number of points they have achieved. Based on the *standard grade distribution*, their results, however, can be transformed into artificial grades, which allows easier comparison. In addition to the points for the submission of the assignment, the points of the entry test are added to the sum of the total achievable points.

Individual Phase Grades. A student can receive a maximum of 90 $(80 + 10)$ points in the IP. Based on the *standard grade distribution* of Table 2.1, the grading of the IP is structured as shown by Table 8.5. The sum of achieved submission points x_S and entry test points x_E are denoted as x: $x := x_S + x_E$

AT Grade	AT Code	ECTS Grade	Verbal	Definition
1	S1	A	Excellent	if $x \geq 77.5$ points
2	G2	B	Good	if $65.0 \leq x < 77.5$ points
3	B3	C	Satisfactory	if $52.5 \leq x < 65.0$ points
4	G4	D,E	Pass	if $40.0 \leq x < 52.5$ points
5	N5	F,FX	Fail	if $x_S < 40$ points

Table 8.5: Individual Phase grading scheme

The limits of the *Definition* column in Table 8.5 are constructed as follows: at least 40 points must be achieved, which means that $40 + 10$ points remain, resulting in $\frac{50}{4} = 12.5$ equal steps. Recall that it is required that the submission is graded with at least 40 points, otherwise a negative course certificate is issued. In the past, students were also required to achieve at least five points in the entry test. Since data on student's entry test points is incomplete, they are awarded with five points to be comparable. Only if the submission is positively graded, the additional entry test points are considered for the grading scheme.

Based on the explained grading scheme, the grades as shown in Table 8.6 are derivable for all submissions which were not empty. It demonstrates the distribution of each grade per term in percentages. Table 8.6a shows each grade in relation to all students *over all terms*, and Table 8.6b presents each grade in relation to all students *of the respective term*. During these four terms 977 submissions were graded.

It is evident that there is a change between the term *20SS* and the subsequent terms. Whereas in *20SS* the number of excellent and good grades is relatively high. The number of excellent grades fell below five percent, followed by a steady increase up to nearly ten percent. The situation was similar for good grades, with the proportion halving but remaining almost constant, as shown in Table 8.6b. The observed values can be traced to the fact that the summer term 2020 was the last semester without a grading scheme

Term	1	2	3	4	5
20SS	5.94	10.03	6.65	2.66	8.19
20WS	0.61	2.15	1.94	2.15	6.04
21SS	2.76	6.24	6.45	4.61	18.94
21WS	1.33	2.76	2.97	0.61	6.96
All	10.64	21.19	18.01	10.03	40.12

(a) Sum as fraction of total

Term	1	2	3	4	5
20SS	17.74	29.97	19.88	7.95	24.46
20WS	4.76	16.67	15.08	16.67	46.83
21SS	7.09	16.01	16.54	11.81	48.56
21WS	9.09	18.88	20.28	4.20	47.55

(b) Sum as fraction of rows

Table 8.6: Distribution of grades in Individual Phase (%)

designed for online-only submissions[89]. After this term the course staff team provided the tutors with a 70-page *grading plan* with the intention to standardize the assessment process. The new grading scheme resulted, on the one hand, in a high rate of negative grades (nearly about 50%) and, on the other hand, in a way lower ratio of excellent grades.

When visualizing the *All*-row of Table 8.6a in absolute numbers, as in Figure 8.1, it can be seen that the data does not follow a normal distribution, as argued in the past by scientists [3, 44, 113]. The illustration of Figure 8.1 becomes even more comprehensible if, instead of grades, the points x are directly considered, as shown in Table 8.7 and Figure 8.2.

For each term a sample based statistical analysis was done which resulted in a very low mode value. On the other hand, mean and median followed a similar evolution to the general grades shown in Table 8.6b: Term *20SS* differs from the following terms, whose values remain constant. However, the standard deviation s is different, it increased steadily by about five over time. The 25 and 75%-Quantiles Q_1 and Q_3 indicate a high spread of the points achieved by the students. In winter term 2021 25% even received less than 2 points on the submission. The skewness of Figure 8.1 towards the negative grade direction, can also be observed by the raw data in Table 8.7 where all values for each term and the overall value denote a Left-Skewed (Negatively Skewed) distribution.

The last two rows of Table 8.7 show the overall analysis without grouping by term. The population based statistical analysis substantiate the trend of each term with similar values. While students with zero points do receive a course certificate, when these students are excluded, as shown in the very last row, the picture is different. In this context the mode is 56, mean and median are relatively close to each other. To conclude, the *0-point-students* falsify the overall student performance.

[89] In previous terms, grading took place at the university where the course staff team was present and a much shorter grading plan was used. Since the summer term 2020 the grading procedure is done by each tutor at home.

Figure 8.2 illustrates the values of Table 8.7 per term using a Violin plot. The red dot indicates the mean, the black bar represents the Interquartile range (IQR) and the blue bars on top and bottom represent the minimum and maximum values for each plot. In addition, the dashed line indicates the value of 40, representing the minimum value to pass the Individual Phase. The distribution of points is easier to see in Figure 8.2 than in Table 8.7. The values around the minimum score, are very waisted for the 2021 semester, while they are almost unnoticeable for the 2020 semester.

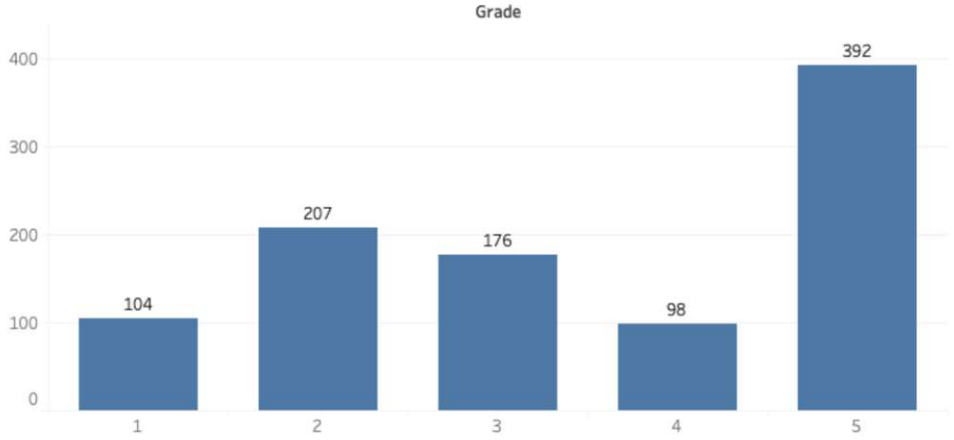


Figure 8.1: Individual Phase grades distribution

Term	Mode $\bar{m}$	Mean $\bar{x}$	Median $\tilde{x}$	Standard deviation s	Q_1	Q_3	Skewness g_1	Sample Size n
20SS	0	55.28	63	25.10	45.00	75.00	-1.02	327
20WS	0	39.02	48	27.28	13.25	61.75	-0.16	126
21SS	0	39.09	48	27.35	14.00	61.00	-0.03	381
21WS	0	39.14	48	30.64	2.00	67.50	-0.09	143
		μ	$\tilde{\mu}$	σ				N
All	0	44.51	52	28.13	19.00	69.00	-0.35	977
†[90]	56	52.26	56.50	22.88	31.00	71.00	-0.54	823

Table 8.7: Statistical analysis of Individual Phase points x

[90] Filtered results with 0 points on submission

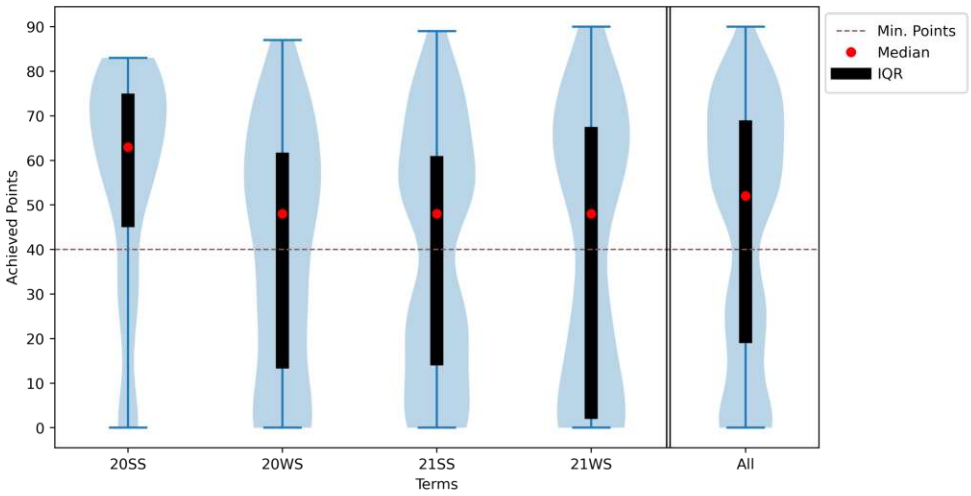


Figure 8.2: Individual Phase's achieved points per term (Violin plot)

Group Phase Grades. As for the IP, the distribution of the grades for the Group Phase can be analyzed as shown in Table 8.8. This table is structured as before: Table 8.8a shows each grade relative to all students *across all terms*, and Table 8.8b lists each grade relative to all students *of the respective term*. These tables only consider students who passed the IP, all other students are excluded, resulting in 584 students[91]. However, in this particular case only the grade is available. It can be seen that both subtables are very right-skewed (positively skewed, Table 8.10 g_1-column), having a high focus on excellent and good grades. In both terms of 2020, not a single student received a grade of 4 or 5, and in 2021, only a few students received these grades: three in the summer term and one in the winter term. Besides the skewness, it becomes evident that, with the exception of winter term 2020, the percentage values for each grade is relatively steady for 1s and 2s, and even decreasing for 3s. Due to the nature and design of the GP, the 4's and 5's are insignificant low as opposed to the overall values — 0.85% of the overall distribution.

A similar picture is drawn when statistically analyzing the grades, as shown in Table 8.9. The most common value is — as expected when considering Table 8.8 — a 1, and also the mean $\bar{x}$ respectively μ and the median $\tilde{x}$ respectively $\tilde{\mu}$ all show the grade of 1. Furthermore, the standard deviation s and σ is rather low, indicating that the values are very close to each other.

The distribution of the grades can also be determined either for each term, as in Figure 8.3, or per research division, as in Figure 8.4. Figure 8.3 simply represents a visualization of the absolute values of Table 8.8b. Figure 8.4, however, is more interesting in that

[91] $977 - 392 = 585 \neq 584$ (according to Table 8.3). As one student dropped out in SS20 due to mandatory military service during the first COVID-19 lockdown and, therefore, this student is missing in the data.

particular case. While the 2s in both departments are essentially identical, Research Division B has more 3s and Division A has about the same proportion more 1s. It is also notable that the negatively graded students exclusively were part of Research Division A.

Term	1	2	3	4	5
20SS	34.25	6.51	1.37	0.00	0.00
20WS	7.71	2.23	1.54	0.00	0.00
21SS	28.77	3.42	0.86	0.17	0.34
21WS	10.96	1.37	0.17	0.34	0.00
All	81.68	13.53	3.94	0.51	0.34

(a) Sum as fraction of total

Term	1	2	3	4	5
20SS	81.30	15.45	3.25	0.00	0.00
20WS	67.16	19.40	13.43	0.00	0.00
21SS	85.71	10.20	2.55	0.51	1.02
21WS	85.33	10.67	1.33	2.67	0.00

(b) Sum as fraction of rows

Table 8.8: Distribution of grades in Group Phase (%)

Term	$\bar{m}$	$\bar{x}$	$\tilde{x}$	s	g_1	n
20SS	1	1.22	1	0.49	2.16	246
20WS	1	1.46	1	0.72	1.21	67
21SS	1	1.21	1	0.61	3.81	196
21WS	1	1.21	1	0.60	3.30	75
		μ	$\tilde{\mu}$	σ		N
All	1	1.24	1	0.58	2.84	584

Table 8.9: Statistical analysis of Group Phase grades

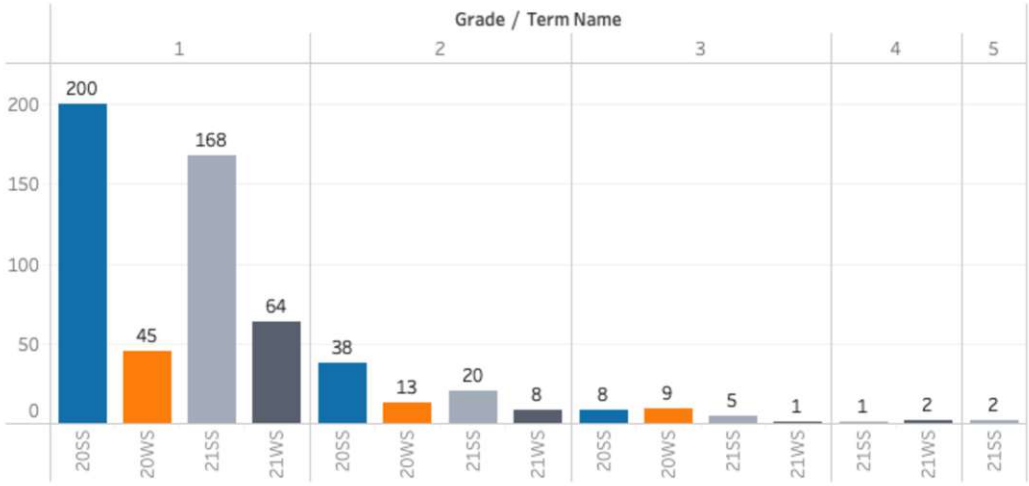


Figure 8.3: Group Phase grade distribution per term

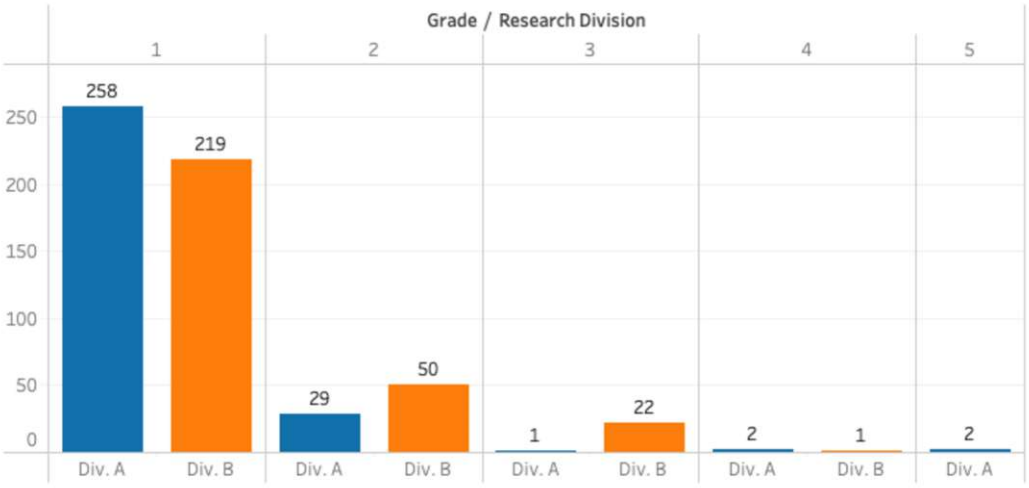


Figure 8.4: Group Phase grade distribution per research division

Final Grades. When, the data of the IP and GP are combined finally, the final grade can be calculated. The final value is a weighted calculation of the two grades: 0.25∗IP grade + 0.75∗GP grade. Thus, the value of the Individual Phase does not influence the final grade much, except for a 4. If that is the case the final grade is one lower than the GP grade, otherwise the GP grade also determines the final grade. Table 8.10 visualizes the grades as fraction of total *over all terms* and in relation to all students *of the respective term*.

Although the evaluation of the course shows a high rate of negative grades, at approximately 40% as in Table 8.10a, the rate of excellent and good grades is about 15% higher than the negative ones. Therefore, the majority of the students' overall performance in this course can be considered passing, with emphasis on the very good grades.

The observed phenomenon, as aforementioned in the *Individual Phase Grades* paragraph, of the negative grades is also reflected in both subtables of Table 8.10. Since students who fail the IP receive a negative certificate and must not participate in the GP.

Figure 8.5 visualizes the values of Table 8.10b in absolute numbers per term.

With regard to the statistical analysis of Table 8.11, the trends of Table 8.10a and 8.10b are reflected in the mode value $\bar{m}$ and mean $\bar{x}$ respectively μ and also the standard deviation s and σ. The value of the median $\tilde{x}$ respectively $\tilde{\mu}$ represents the overall values as shown in Table 8.10a. Approximately half of the students received a grade better than a 3 and half received a grade worse. Another interesting trend can be seen in the skewness column g_1 where the overall evaluation as well as the winter term 2020 are positively-skewed and the subsequent terms are all negatively-skewed.

Term	1	2	3	4	5
20SS	12.90	11.26	1.02	0.00	8.29
20WS	1.74	3.89	1.23	0.00	1.23
21SS	7.88	10.85	1.02	0.10	19.14
21WS	3.68	3.38	0.41	0.20	6.96
All	26.20	29.38	3.68	0.31	40.43

(a) Sum as fraction of total

Term	1	2	3	4	5
20SS	38.53	33.64	3.06	0.00	24.77
20WS	13.49	30.16	9.52	0.00	46.83
21SS	20.21	27.82	2.62	0.26	49.08
21WS	25.17	23.08	2.80	1.40	47.55

(b) Sum as fraction of rows

Table 8.10: Distribution of final grades (%)

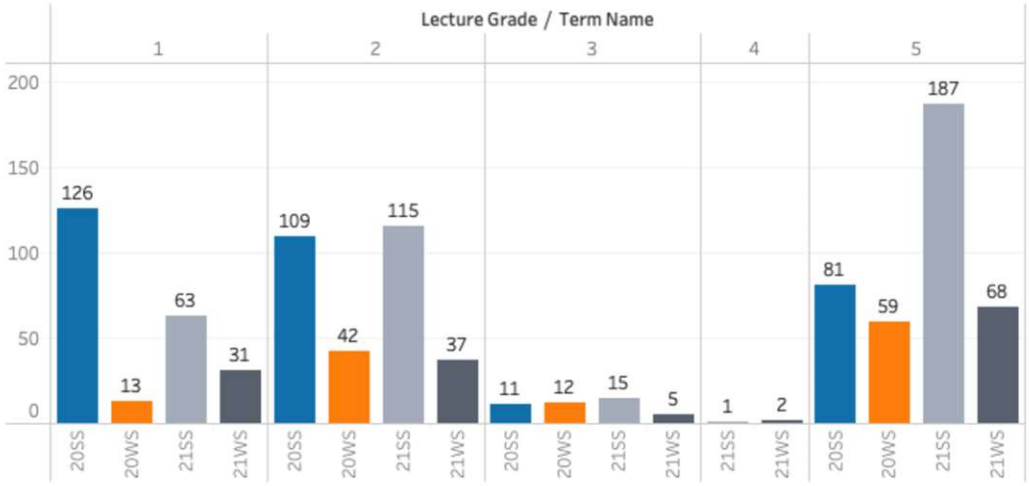


Figure 8.5: Distribution of grades per term

Term	$\bar{m}$	$\bar{x}$	$\tilde{x}$	s	g_1	n
20SS	1	2.39	2.0	1.58	0.86	327
20WS	5	3.37	3.0	1.61	-0.12	126
21SS	5	3.30	3.0	1.72	-0.13	381
21WS	5	3.23	3.0	1.77	-0.11	143
		μ	$\tilde{\mu}$	σ		N
All	5	2.99	2.0	1.72	0.18	977

Table 8.11: Statistical analysis of final grades

8.3.2 Time Spent Analysis

In the following section the time spent on the project by students during the IP, GP and in total are analyzed in more detail. By means of an easier comparison, the time planned for each stage t_p is divided into three categories in which x is the student's time:

- *Low*: C_L: $x < t_p * 0.9$
- *Target*: C_T: $t_p * 0.9 \leq x \leq t_p * 1.1$
- *High*: C_H: $x > t_p * 1.1$

The value of t_p is different in the following paragraphs: for the paragraph about the Individual Phase it is 40 hours, for the paragraph about the Group Phase it is 110 hours and for the last paragraph it is 150 hours.

Individual Phase. For the summer term 2020 does not exist any data about the IP time spent on the project anymore, so this term is ignored in the following tables. Thus, 650 students remain (see Table 8.7). Table 8.12 illustrates the distribution of time spent on the project for the available terms in relation to all students and the respective students in a semester. With regard to Table 8.12a, it can be concluded that the absolute majority of students lies within the highest category C_H. Moreover, 7.54% of all students (49 people in absolute numbers) did not manage to provide the timesheet in their submission. Table 8.12b indicates that for both terms in 2021 the relative number of students in the highest category decreased, while the lowest category increased. However, two-thirds of all students are still in category C_H.

Term	C_L	C_T	C_H	n.a.
20SS	/	/	/	/
20WS	1.85	2.00	14.31	1.23
21SS	9.85	6.62	37.54	4.62
21WS	3.38	2.77	14.15	1.69
All	15.08	11.38	66.00	7.54

(a) Sum as fraction of totals

Term	C_L	C_T	C_H	n.a.
20SS	/	/	/	/
20WS	9.52	10.32	73.81	6.35
21SS	16.80	11.29	64.04	7.87
21WS	15.38	12.59	64.34	7.69

(b) Sum as fraction of rows

Table 8.12: Distribution of the categorized time spent on the project during the Individual Phase (%)

When analyzing the time tracking records, as in Table 8.13, it becomes evident that the numbers highly differ from the planned time of 40 hours. Mean and median are approximately 20 hours higher than the planned value. The standard deviation value of 20 hours shows a high degree of dispersion within a semester between students. This is also reflected in the visualization of these numbers (without the 0-hour-students) using a

histogram, as in Figure 8.6. Based on the Freedman-Diaconis rule [49], the histogram is divided into 30 bins each seven hours wide and also shows a normal distribution around the median of 57.95 hours. Furthermore, the positive skewness g_1, as demonstrated in the ‡-row of Table 8.13, can be observed. The histogram also shows that there are some extreme outliers with more than 100 as well as more than 200 hours.

Term	$\bar{m}$	$\bar{x}$	$\tilde{x}$	s	g_1	n
20WS	92.00	67.62	61.75	28.81	1.50	118
21SS	0.00	55.87	55.00	27.01	0.32	351
21WS	42.00	56.63	56.00	23.68	-0.05	132
		μ	$\tilde{\mu}$	σ		N
All	0.00	58.35	57.00	27.02	0.57	601
‡[92]	56.0	60.46	57.95	25.07	0.95	580

Table 8.13: Statistical analysis of time spent on the project during the Individual Phase

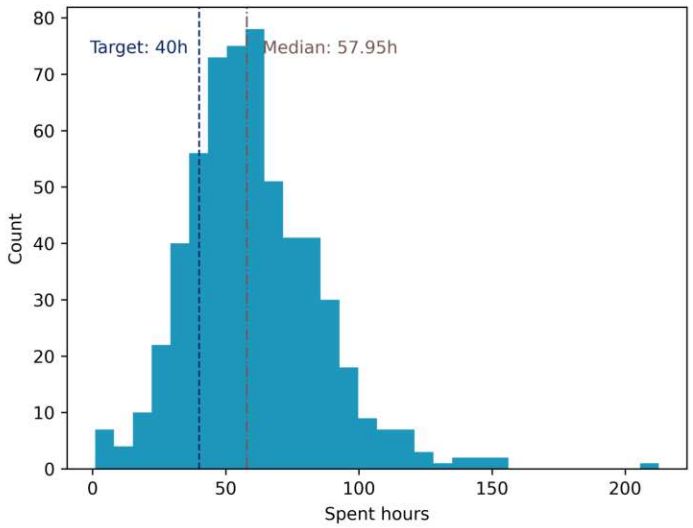


Figure 8.6: Histogram of the time spent on the project during the Individual Phase (‡[92])

[92] Excluding the students who spent zero hours on the Individual Phase

Group Phase. The distribution of time spent on the project during the Group Phase is slightly different compared to the Individual Phase. As demonstrated in Table 8.14 — in particular in the last row — the number of students within the target category C_T is approximately three times higher compared to Table 8.12a. In addition, the outlier in the lower category C_L as well as in C_H marginally decreased. Table 8.14b also shows steady values for each category over all terms. From this, it can be inferred that students did not put in more effort over time, as measured by hours.

Term	C_L	C_T	C_H	n.a.
20SS	5.31	15.07	21.75	0.00
20WS	0.86	4.28	6.34	0.00
21SS	5.31	11.47	16.61	0.17
21WS	1.54	3.94	7.36	0.00
All	13.01	34.76	52.05	0.17

(a) Sum as fraction of totals

Term	C_L	C_T	C_H	n.a.
20SS	12.60	35.77	51.63	0.00
20WS	7.46	37.31	55.22	0.00
21SS	15.82	34.18	49.49	0.51
21WS	12.00	30.67	57.33	0.00

(b) Sum as fraction of rows

Table 8.14: Distribution of the categorized time spent on the project during the Group Phase (%)

The statistical analysis of all terms, shown in Table 8.15, also demonstrates that the planned time of 110 hours is exceeded by about 20-25 hours (when referring to the mean and median), but is relatively persistent. However, the standard deviation value of about 30-35 is rather high. Another interesting value is the mode $\bar{m}$ which is 107 hours, and, thus, located nearly at the target. Figure 8.7 shows the histogram of all terms combined, having 31 bins with a width of nine hours. At this point, the skewness of 0.95 is again clearly observable as the values are normally distributed tending towards the 110 planned hours. Again, the histogram shows extreme outliers with more than 200 and even 300 hours spent on the GP.

Term	$\bar{m}$	$\bar{x}$	$\tilde{x}$	s	g_1	n
20SS	160.25	131.47	122.28	35.82	1.23	246
20WS	129.25	133.66	129.83	36.28	1.03	67
21SS	88.0	126.03	121.0	30.16	0.38	195[93]
21WS	116.0	136.35	129.21	35.46	0.63	75
		μ	$\tilde{\mu}$	σ		N
All	107.0	130.53	123.38	34.1	0.95	583

Table 8.15: Statistical analysis of time spent on the project during the Group Phase

[93] 196 students were officially part of the Group Phase. Although one student passed the Individual Phase, he dropped out before the start of the Group Phase.

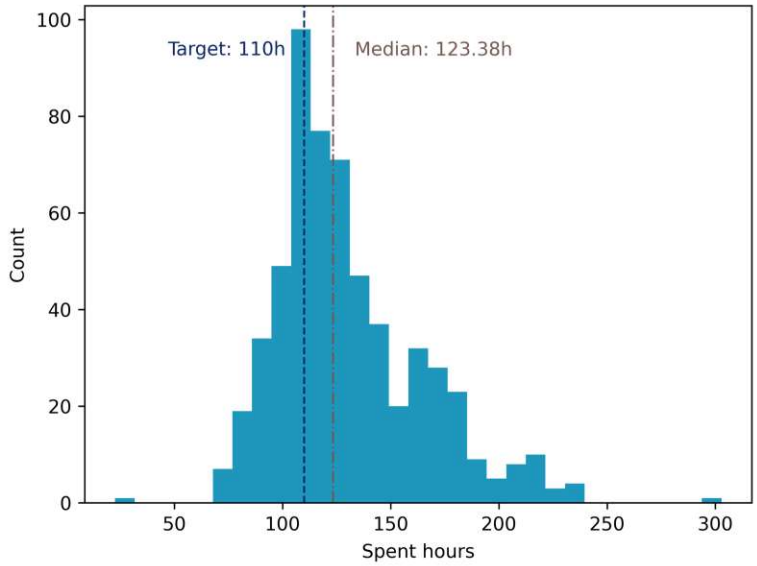


Figure 8.7: Histogram of the time spent on the project during the Group Phase

Overall. To calculate the total time spent on the project by students, the sum of both IP and GP is taken into account. Recall that, as shown in Table 8.11, there are 977 students who received a certificate, 327 out of these participated in the summer term 2020. Since there is no data about the IP for this term, the data is excluded from the following results, resulting in 650 remaining data points. Out of these records, all students who failed the IP are also excluded as, consequently, there are no records for the GP, resulting in 338 students. As shown in Table 8.14, again one student did not complete the Group Phase resulting in 337 students who can be analyzed.

Table 8.16 demonstrates the distribution of the time spent on the project in the categories C_L, C_T, C_H and $n.a.$ Considering the planned 150 hours for the class, it can be seen in Table 8.16a that only about one-fifth truly lies within $\pm 10\%$ of the expected time, and more than two-thirds put in more effort than planned. The relation between real and planned effort is even worse for both winter terms, as in Table 8.16b, and paltry better for the summer term. In addition to these values, it becomes evident that 3.56% (which corresponds to an absolute number of 12) of the students were not able to provide the IP timesheet, resulting in a $n.a$ categorization. These students exclusively took the class in summer term 2021.

The categories under observation become even more evident when viewing the concrete statistical parameters, as in Table 8.17. The mean and median values for all terms and, in general, are clearly above the planned time. In some cases, these values are up to 50 hours higher, which corresponds to 2 ECTS. Moreover, the standard deviation is also around 50 hours. The indicated skewness g_1 of 1.03 is also observable in the histogram

Term	C_L	C_T	C_H	n.a.
20WS	0.89	3.86	15.13	0.00
21SS	2.97	13.65	37.69	3.56
21WS	1.78	3.86	16.62	0.00
All	5.64	21.36	69.44	3.56

(a) Sum as fraction of totals

Term	C_L	C_T	C_H	n.a.
20WS	4.48	19.40	76.12	0.00
21SS	5.13	23.59	65.13	6.15
21WS	8.00	17.33	74.67	0.00

(b) Sum as fraction of rows

Table 8.16: Distribution of the categorized total time spent on the project (%)

Term	$\bar{m}$	$\bar{x}$	$\tilde{x}$	s	Q_1	Q_3	g_1	n
20WS	112.75	203.75	191.92	55.43	168.35	223.37	1.65	67
21SS	147.00	188.47	188.17	38.85	156.58	214.74	0.33	183
21WS	154.33	196.48	191.42	45.54	164.67	228.25	0.56	75
		μ	$\tilde{\mu}$	σ				N
All	149.75	193.47	190.0	44.49	162.33	217.50	1.03	325

Table 8.17: Statistical analysis of total time spent on the project

of Figure 8.8, which has 22 bins, each 16 hours wide. It also shows that the outliers are again over 300 and even 400 hours of total time spent. These high numbers result from the aggregation of previously observed outliers.

The histogram values of Figure 8.8 do not distinguish between terms and their trend over time. Thus, when visualizing the time spent on the project for each term using a Violin plot as in Figure 8.9, some trends can clearly be identified. Recall that for each plot in Figure 8.9 the red dot represents the mean value, the black bar represents the IQR, and the blue bars at the top and bottom represent the maximum and minimum values. In addition, the dashed vertical line indicates the planned 150 hours. The difference between minimum and maximum time definitely decreased during these terms. However, also in the winter term 2020 there can be found blatant outlier, as shown in Figure 8.8. The subsequent terms were much closer together when the minimum and maximum values are considered.

In addition to the term, the data can also be categorized regarding the respective research division, as demonstrated in Section 1.1. The overall trend of Figure 8.9 can also be detected in this chart, however, the range of the students' efforts is smaller for Research Division B (Div. B) than for Research Division A (Div. A). Based on the design of the course — keep in mind that the groups of Division A get a predefined project, whereas the groups of Division B need to come up with their own project ideas — it becomes evident that the groups working on the predefined project need a lot more time as the other groups.

8.3.3 Commit Time Analysis

In the following the commit timestamps are analyzed in more detail. For the IP the whole timespan is considered (since it is only about four weeks), for the GP only the weekday and daytime are analyzed and visualized in this chapter.

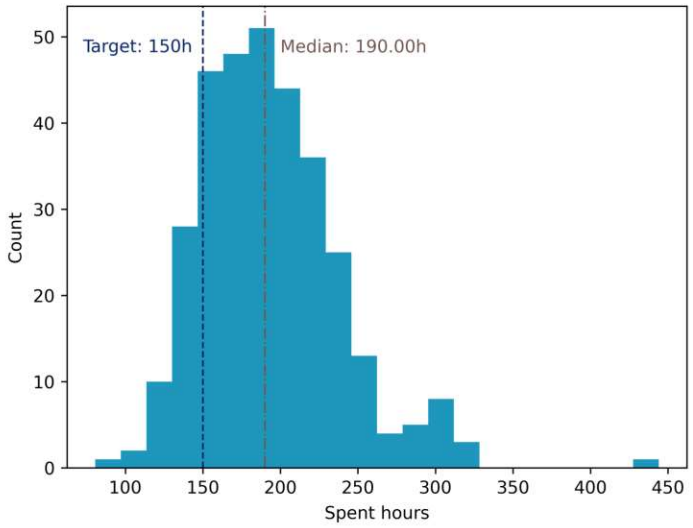


Figure 8.8: Histogram of the total time spent on the project

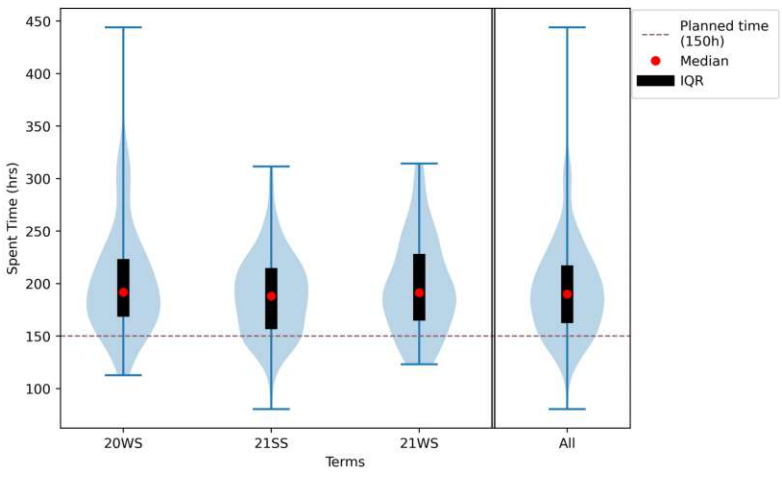


Figure 8.9: Total time spent on the project per term (Violin plot)

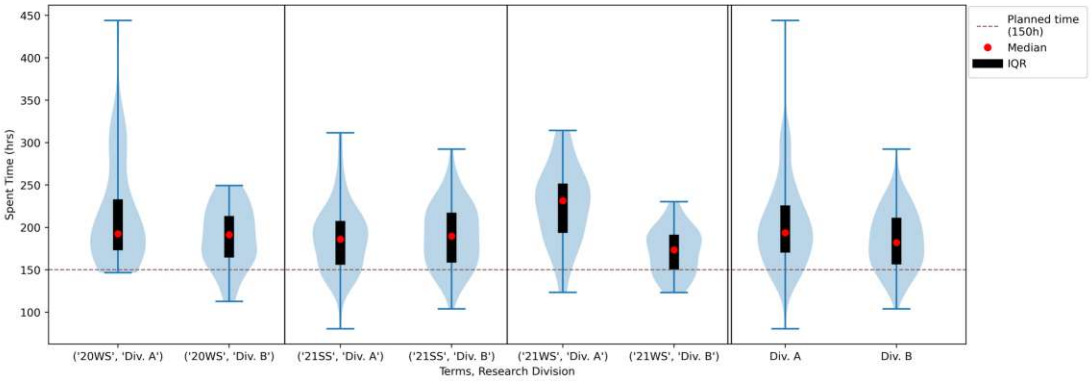


Figure 8.10: Total time spent on the project per research division (Violin plot)

Individual Phase

Figure 8.11 visualizes the course of the commits based on the timestamp provided by the commit. Hence, every commit which was authored later and, therefore, has a second, authored timestamp will also only have the creation timestamp taken into account. The x-axis shows the timeline, the y-axis represents the number of commits on a particular day. In addition, the thickness of the line indicates the number of individual students on that respective day, therefore, the thicker the line, the more individual students.

Looking at the graphs of Figure 8.11, it can be seen clearly that at the beginning of each term there is little progress. After approximately the first quarter the number of commits increases. The number continues to rise until about three-fourths of the time has elapsed, followed by a steep increase lasting until the deadline. Each subfigure of Figure 8.11 has vertical lines denoting the deadline. However, in the summer term 2020 the initial deadline was postponed for seven days due to the first COVID-19 lockdown, which is why Figure 8.11a has two vertical lines. The first vertical line denotes the official date when the students were notified about the deadline extension. Figure 8.11a is interesting as the students back then did not continue to commit Code, which resulted in a decrease of the number of commits after the notification. Around the last quarter of the Individual Phase, the number of commits began to rise again.

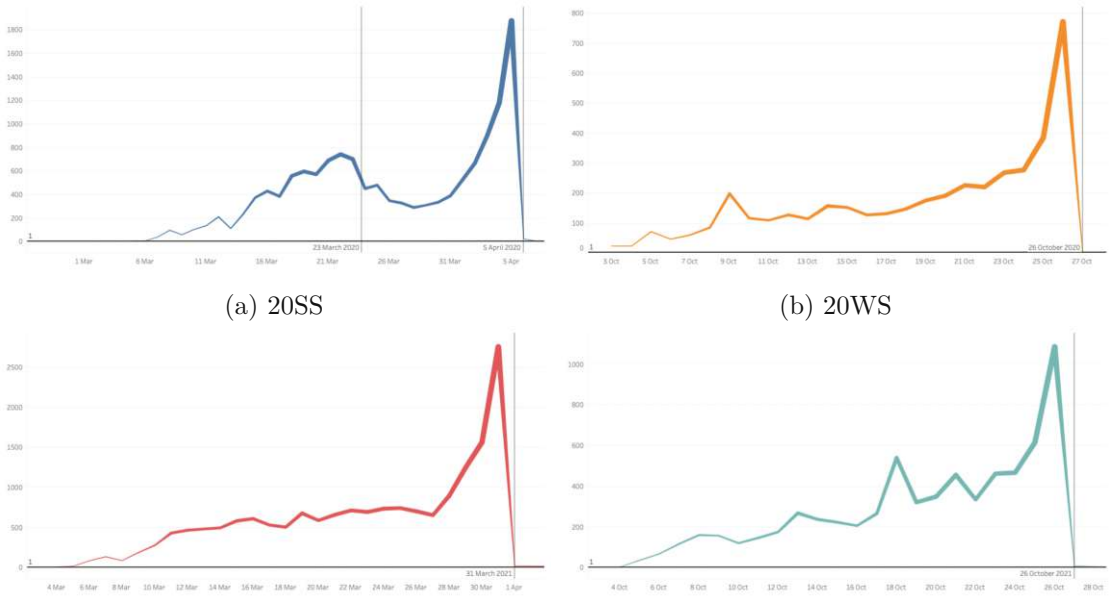
72 hours before the deadline. The charts of Figure 8.11 are of even greater interest when zooming into the peak and analyzing the last 72 hours before the final deadline. The course of the commits are visualized in Figure 8.12 which displays even more crucial features. It can be seen at which time the students work on the assignment: the peak values are reached exclusively during the day, while the number of commits decreases and reaches its lowest point at night. For all four terms the daily peak increased steadily and reached its highest (and last) peak in the last 24 hours.

Table 8.18 shows that the absolute number of commits in the last 72 hours before the deadline is reached, also deserves careful attention. The table illustrates that

approximately 30% of all IP commits are made. The value of the first column of Table 8.18 shows different values than Table 8.2, but the former only considers valid commits (before the deadline). The values of Table 8.2, on the other hand, take into account all commits made.

Term	$\sum$ Commits	Commits_{72h}	Ratio (%)
20SS	14150	3931	27.78
20WS	4205	1162	27.63
21SS	17538	5534	31.55
21WS	6824	2160	31.65
All	42717	12787	29.93

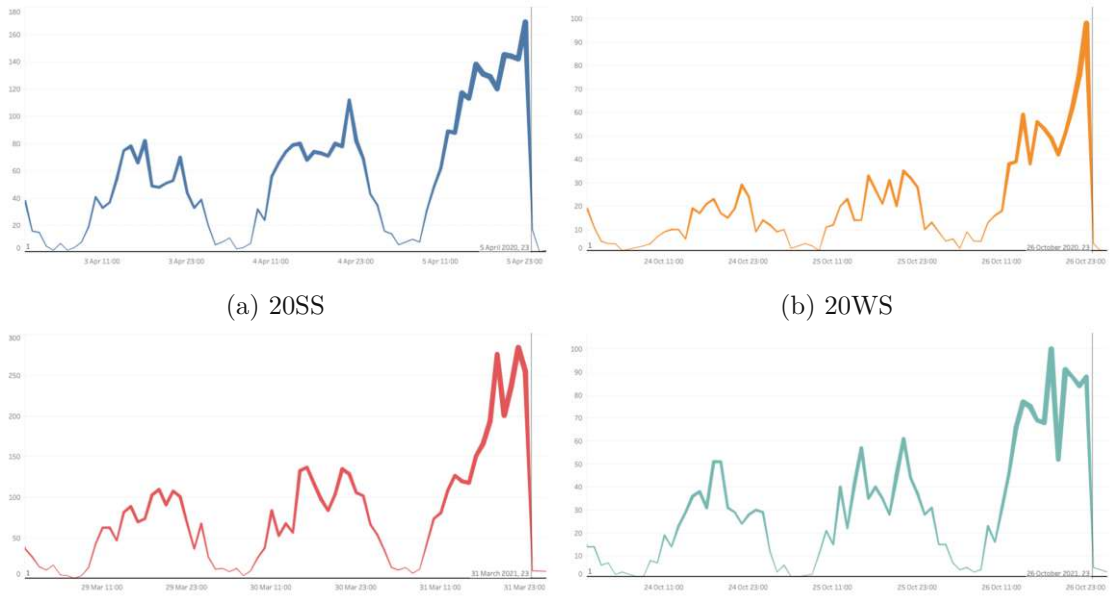
Table 8.18: Ratio of the last 72h to the total amount of commits



(a) 20SS (b) 20WS

(c) 21SS (d) 21WS

Figure 8.11: Individual Phase Commit timestamps



(a) 20SS (b) 20WS

(c) 21SS (d) 21WS

Figure 8.12: Last 72 hours before the deadline

Group Phase

The commit distribution of the Group Phase based on the weekday and daytime is visualized in Figure 8.13. The figure shows the daytime on the x-axis, based on the date of the creation of a commit, and the weekday on the y-axis. In addition, each square is colored according to the distinct number of students. The bluer the tile the smaller the number of students. The more orange the tile, the greater the number of students who committed on the respective day and at timeframe. The third dimension shown in Figure 8.13 is the size of a tile. The number of commits made on a particular weekday and daytime define the size.

According to Figure 8.13, Monday to Thursday early afternoon until about eight o'clock in the evening is a rather busy time, having the absolute peak on Thursday between three and four pm. Moreover, on Friday a relatively small number of commits are made and that students tend to stop working on the assignment earlier. On Saturday at the same time, students start later than on other days and also end earlier than on weekdays. On Sundays students seem to start working on the project shortly after noon and having a relatively steady number of commits between four and eight pm.

Figure 8.13: Commits per weekday and daytime, colored by no. of active students

8.3.4 Advanced Analysis

The combination of certain extracted features based on the commits during the GP resulted in a newly calculated measure, as shown in Listing 8.1. Where $\sum$ `Changes` is the sum of insertions and deletions (= changes) of all commits on day x, $\#|$`Students`$|$ is the number of distinct students who committed on day x and $\sum\ t_x$ is the sum of the time spent on the project on day x in hours. For example, for an arbitrary day x_a the calculation of $CTR(x_a)$ results in the ratio of changes per student per hours spent on the project. Hence, the value is also called Changes-Timelog-Ratio (CTR).

$$CTR(x) = \frac{\Sigma\ \texttt{Changes}}{\#\ |\texttt{Students}|} \over \Sigma\ t_x \tag{8.1}$$

Figure 8.14 shows two line plots for each term. The upper row always denotes the number of commits on a seven-day average. The second row displays the calculated CTR value as shown in Equation 8.1. Again, the thicker the line, the more different students worked during this time. The gray areas on top of each chart serve as a reference, indicating the review periods of the groups with their respective course assistant.

All figures in 8.14 start with a decreasing curve, which can be explained easily as before the first review meeting students are mostly required to complete project management tasks. After this initial phase, as be seen in Figure 8.14, particularly the terms of 2020 are relatively steady (at very high values) but sharply increase at the end. Summer term 2020 additionally has a brief peak just before the second review date. The 2021 terms indicate lower values as opposed to the previous year; the winter term also is, as a matter of fact, the one with the smallest increase at the end.

Ideally, the CTR value remains constant over time. However, all illustrations of Figure 8.14 at least have a slight increase at the end of the term. The idea of the CTR value is to visualize changes over time in the workload (= Source Code changes) per student in relation to the student's effort (= time spent on the project). Since these values are also aggregated by their seven-day average, increases at the end of a term can be interpreted as follows: The students realize that they underestimated the workload and the end of term is ahead. Based on the nature of Equation 8.1, the number of changes in relation to the time effort on the project must be remarkably higher to create a notable increase. Moreover, the lines of the CTR value in Figure 8.14 get thinner at the end of a term, which indicates a smaller number of students working at this time than during a term.

Figure 8.14b also shows a slight decrease, which according to Equation 8.1, should quite be the opposite: The changes decrease while the amount of time spent on the project increases. This might indicate that time is artificially added to align the total time spent within a group. To detect this kind of data manipulation an additional analysis of the time spent history must be carried out. This can be done, for example, by using the features presented in Figure 7.4.

However, it has to be kept in mind that by considering a relation of three different variables only, it cannot be proven if students manipulated their time. The CTR value is merely an indicator that there are measurable changes, compared to the rest of the term.

(a) 20SS

(b) 20WS

(c) 21SS

(d) 21WS

Figure 8.14: Group Phase CTR value

8.4 Expert Evaluation

In the following the evaluation of the prototype's usability and the thesis' initial hypotheses is presented. As the latter were also part of the first survey (Appendix C), in this survey the hypotheses were evaluated if the prototype matches the expert's expectations. Similar to the first survey of Section 5.2, the interviewer's screen was shared during the whole interview. To evaluate the usability, the most frequent use cases of the past were presented to the interviewee. This showcase lasted for about ten minutes at the beginning of each interview. The presented scenarios covered all possibilities to derive knowledge about groups, with respect to the available data. Thus, only the dashboard of the Group Phase was demonstrated.

8.4.1 Hypotheses evaluation

For the sake of simplicity, the following enumeration lists all hypotheses of the first survey (Appendix C). However, the hypotheses numbered XIII and XIV were not evaluated. These two would have required knowledge of the entire architecture that could not have been provided in the time available for the survey. For the complete expert evaluation survey see Appendix D.

I The visualization artifacts are portable and cross-platform

II The administration of the architecture should be possible within the existing GitLab infrastructure.

III Visualization artifacts should be viewable offline (that is without being hosted on a server)

IV It is important to select multiple semester to compare them against each other

V It is important to only show the current semester

VI It is important to quickly get an overview of all groups' project state of the current semester

VII Assuming you are responsible for multiple groups in the current semester, it is important to quickly get an overview of all of your group's project state

VIII Assuming you want to compare multiple groups, it is important to filter by research divisions

IX Assuming you want to compare multiple groups, it is important to filter by groups (only display a subset of all)

X It is important to customize the time granularity (i.e. display time series data per day or week)

XI It is important to customize the timeframe, for example zooming into the last X weeks while hiding the rest

XII It is important to filter by specific students

XIII A software repository visualization architecture for software engineering education should not require initial configuration

XIV A software repository visualization architecture for software engineering education should be configurable during operation

Each hypothesis could be rated from one to five[94] on the extent to which the expectation was met. Figure 8.15 shows the results as a box plot per question, the diamond shape indicates the average value. In general, eleven hypotheses were rated on average with at least *Satisfied*, the median was five (*Very satisfied*) with the exception of hypothesis XI. In addition, eight hypotheses were evaluated with no worse than a *Neither*. On average, all hypotheses were rated with approximately 4.5 which is exactly between *Satisfied* and *Very satisfied*.

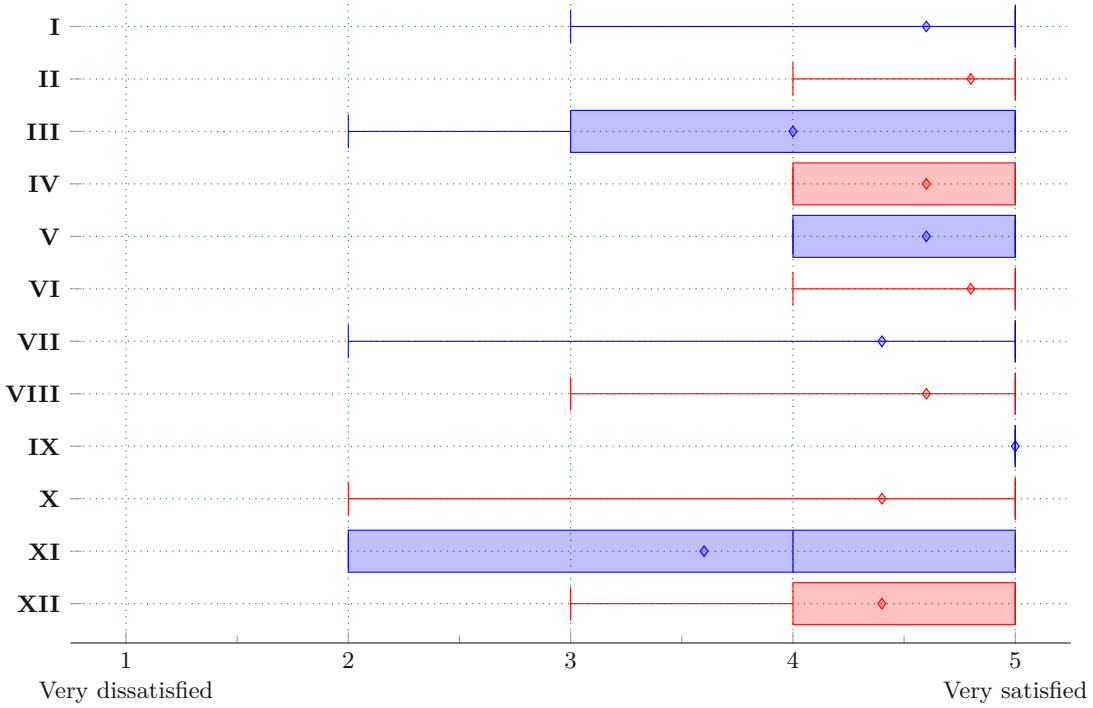


Figure 8.15: Hypotheses evaluation results[94]

8.4.2 System Usability Scale

The System Usability Scale (SUS), developed by Brooke et al. [15] in 1996 as a "quick and dirty" scale for surveys to swiftly assess a product or service's usability. The survey gives a single score on a scale that is easy to understand by a wide range of people, even if they don't know much about human factors or usability [6]. The scale ranges from zero

[94] 1 Very dissatisfied; 2 Dissatisfied; 3 Neither; 4 Satisfied; 5 Very satisfied

to 100, however Bangor et al. [6] designed a 7-point adjective grading scale to provide a more absolute assessment for SUS scores. These adjectives are: “Worst Imaginable”, “Awful”, “Poor”, “OK”, “Good”, “Excellent” and “Best Imaginable.”

Four of the five participated experts rated the prototype with at least 92.5 points, while one participant saw the usability more critical and rated it with 80. The average score of the prototype’s is 91.5 points, as shown in Figure 8.16. Hence, the proposed prototype can be rated as “Best Imaginable”, assuming the System Usability Scale is divided into seven parts of equal size. Overall, the feedback on the prototype was very positive. The main criticisms were that some parts of the UI were not as clear as they could be, for example, to select a timeframe a date picker should be provided. In addition, some also criticized that such an expert system should be complex since otherwise it may lack capabilities to derive certain (complex) knowledge. However, the SUS explicitly asks if one thought the system is easy to use.

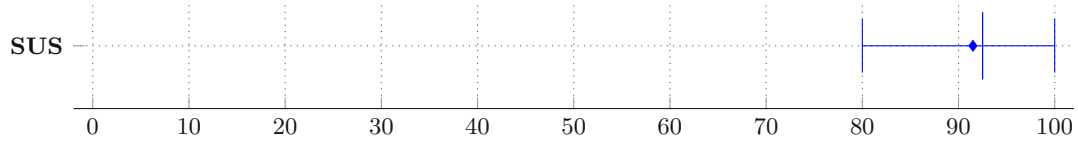


Figure 8.16: System Usability Score results

8.5 Discussion

In this section, firstly, Section 8.5.1 discusses the results of this work in light of the research questions defined in Section 1.3, and secondly, Section 8.5.2 further discusses the analyses of Section 8.3.

8.5.1 Answering the Research Questions

Recall from Section 1.3 that RQ1 can be divided in two parts: *RQ1a* and *RQ1b* are about the implementation of the data mining process and *RQ1c* is about the information needs of the experts. *RQ2* is about the implementation of the charts and dashboards and *RQ3* is about the data analysis tasks.

re RQ1. The implementation of the data mining process and the collection of the information needs of the experts was the first part of this work. As both sub-questions (*RQ1a* and *RQ1b*) were independent of *RQ1c*, they were worked on in parallel. The first two sub-questions, which are the first main research aspect, could be implemented to address duplicate, incomplete, or mislabeled data in a flexible and reliable form, as demonstrated in Chapter 6. In addition, it is also possible to speed up the processing of more than 1200 repositories and 100,000 commits (on the main/master branch only) by using the parallelization capabilities of the machine, as discussed in Section 8.1. Since this thesis builds on the ideas of existing research projects [53, 56], the expert’s information

needs were first derived from these two theses and formulated as hypotheses. These hypotheses were agreed with a median score of 4 and an average score of 3.92 out of 5 points on the Likert-scale [40] (as Figure 5.17 illustrates). The identified needs and requirements of the experts can be considered relevant and correct. Furthermore, the experts verbally shared personal opinions to better understand their results.

re RQ2. Various tools were reviewed for the implementation of the prototype, and Apache Superset was finally selected based on feedback from the expert interviews. The created diagrams and dashboards were then evaluated in a final survey (as already discussed in Section 8.4) to provide enough information for answering the second main question of the thesis (*RQ2a*): *What is an appropriate intelligence system for software engineering education*? The prototype was rated with a median score of 5 and an average score of 4.48 out of 5 on the Likert scale [40] (as shown in Figure 8.15). In addition to expert satisfaction, the SUS section of the expert evaluation resulted in a median score of 92.5 or an average score of 91.5 out of 100 points. Therefore, *RQ2b* can be answered with confidence: the prototype meets the experts' needs and perceptions of an intelligence tool in a software development class to a high degree.

re RQ3. The analysis was mainly divided into two parts focusing on different granularities, as shown in Section 8.3. To answer *RQ3a* for the features studied, a similar picture emerges in terms of students' grades and time spent: There are measurable differences within a term. Scores for the Individual Phase use the full range of the possible 90 points, with almost equal numbers of students scoring negatively as positively. At the same time the results in the Group Phase are very skewed towards good and excellent grades. In terms of time spent on the project during the course, the results are widely distributed. The IQR is relatively constant, but the absolute values vary greatly within a semester. There are students who spent less than 100 hours and some who spent almost 450 hours.

Comparing the analyses of individual terms with others, *RQ3b* can be answered in the same way: there are measurable differences between different terms and research divisions. During the IP, students of the summer term of 2020 performed slightly better (in terms of grades) than in subsequent terms. The number of achieved points in the Individual Phase was high in the first term and remained constant at about 50 thereafter. However, the IQR slightly widened over time, and the relative number of students who achieved a negative score remained constant at about 48%. The results in Group Phase, on the other hand, remained relatively constant over time, with the exception of the winter term 2020. In terms of time spent on the project, as shown in Figure 8.9, there is a noticeable trend of the median value being constant at around 190 hours. The maximum value was constant in both terms of 2021 at around 300 hours. The IQR did not change either over time. Another interesting fact emerges when the data are further broken down by research division, as shown in Figure 8.10. It can be seen that students of *Div. B* spent less time than students of *Div. A*.

8.5.2 Interpretation of the Results of the Data Analysis

The results of Section 8.3 can be discussed and interpreted beyond the boundaries of the previously answered research questions.

Points and Grades. First of all, the results presented in Section 8.3.1 might suggest that the class has a very high failure rate, as about 40%[95] of all submissions of the IP are graded negatively. On the other hand, however, 95%[96] of all students manage to pass the IP no later than the second attempt. According to the course staff, the roots of these values are the new *online* format: before the pandemic all submissions were reviewed in so-called *lab-sessions* on site at the university. If a student did not show up, they were not graded nor was their work evaluated (regardless of how many submissions they had made at that time). Since the introduction of the online format, submitted work is assessed as soon as at least one submission has been pushed to the remote repository. As a result, the number of assessed submissions is much higher than it was years ago, and even students who were not sure they would pass the exam before the pandemic are now assessed.

The second aspect of the grade analysis is the high number of excellent grades in the GP. Despite the high failure rate in the IP, with a probability of more than 80% students will be graded with an 1 in the GP when they eventually pass the exam. Then, the final grade is usually equals also the grade of the final certificate, as already explained in the *Final Grades* paragraph of Section 8.3. More precisely, 543 of 977 graded students, representing about 55% of all students at the time, received a course certificate with a 1 or 2[97]. Furthermore, as shown in Table 8.4, 584 students attended the GP during this period. Since students can only receive a positive certificate if they have passed the GP, these 543 students must be put in proportion to the 584 students of the GP. Thus, the probability is approximately 93%[97] of receiving a 1 or 2 for the class when passing the IP.

Time spent. The second major part of Section 8.3 which is about the time spent by students on the project can be summarized as follows: *the students' effort is bigger than expected and, thus, it is not valued enough.* This starts with the IP, where the median is 57.95 hours, almost 18 hours (or 44%) higher than planned. The situation is slightly better for the GP where the median value of 123.38 hours is only about 14 hours (or 21%) higher than planned. However, when these numbers are broken down by semester of study and research department, a different picture emerges. As can be seen in Figure 8.10, the median value of students of *Div. B* is lower (and also closer to the expected value) than the value of students of *Div. A*. At the same time, the students of *Div. A* invest at least more time (if the minimum values are compared), but at the same time the maximum value of *Div. B* is significantly lower in comparison. This result was

[95] 392 out of 977 submissions were graded negatively
[96] 767 of 810 students; 582 on the first attempt and 185 on the second attempt
[97] 256 (43.84%) got a 1; 287 (49.14%) got a 2

generally very surprising because students usually want to avoid *Div. B*, as they often do not have a suitable project idea and do not want to have additional workload. On the other hand, the already planned extension of the assignment of *Div. A* should be carefully thought through. The general tenor of the students that the overall effort is too high is supported by the results of Section 8.3.2.

Commit Time. With regard to the third main analysis stage, the analysis of commit timestamps reveals highly interesting as well as insignificant findings. The latter could simply be summed up as *confirming the expected*. To start with these, the easiest aspect to identify is represented in Figure 8.11. All instructors in the course, as well as the assignment for the IP, recommend starting early as the assignment is not designed for a late start with a heavy workload. However, it is observed, and this is also confirmed by the values of Table 8.18, that students do not follow this approach. To set the numbers of Table 8.18 in relation: The average period of the IP was about 29 days. Within the first 26 days 70.07% (or 29.930) of all valid commits[98] were made (which is equivalent to about 1151 per day). On the other hand, within the last 3 days about 4262 commits are created per day, which is about 3.7 times higher.

One surprising insight that has emerged is that, unlike the comparatively short period of the IP, during the GP the groups do not commit the code just before the meetings with the lecture assistant (who is also responsible for grading the respective group).

The third interesting finding obtained is the one described in Section 8.3.4. The formula of Equation 8.1, which takes changes, students, and effort in hours into account, serves as a first attempt to relate these otherwise unrelated characteristics. However, this work lacks a mathematical proof confirming that the result of $CTR(x)$ is correct under all circumstances. As discussed at the end of Section 8.3.4, the regression of this value is suggested to serve as an initial indicator and needs to be improved and further explored in the future.

8.5.3 Threats to Validity

This section describes the threats to the validity of the expert evaluations.

Number of Participants. The evaluation of this thesis only used five people and was both quantitative and qualitative. This sample may be too small to generalize the quantitative results of the evaluation, as this would usually require a larger number of randomly chosen participants. According to Francis et al. [48], at 13, the minimum sample size, it is likely that almost all the beliefs about Attitude, Subjective Norm, and Perceived Behavioral Control will be covered.

Two out of seven staff members from Research Division A and one out of five from Division B participated in the survey. In relative numbers, they represent 28% respectively 20%,

[98] Bear in mind that a commit is considered as *valid* if it is made before the deadline

and overall, they represent 25% of all class staff core members. The number of tutors varies from term to term, so it cannot be considered absolute. At the time of the interviews, the two participating tutors accounted for about 15% of all tutors.

Participant Selection. All interviewees were part of the (core) staff team for the class at TU Wien at the time of the interview. As this prototype tool is intended to be used in the respective class, the amount of possible participants was very limited. Thus, there is definitely a selection bias and should be considered if these results are mapped to other classes. In the future, when new versions of the established prototype are used for research, participants should also be chosen from other, similar classes to generate a more uniform result.

Performance and Usability Issues. The prototype's underlying project, Apache Superset, sometimes was not as performant as needed to render the number of charts created in the dashboards. That is, it resulted in timeout's or other connectivity issues. However, in case it happened, all these charts could have been restored quickly.

CHAPTER 9

Conclusion

In this work a comprehensive prototype was proposed for aggregating and visualizing various data from different sources generated during a term in a software engineering class. With the help of a data store and a visualization tool, the possibility of data preparation and analysis was demonstrated. This topic unites three main scientific fields: Mining Software Repositories, Data Visualization and Software Engineering Education. Data-driven software engineering systems have, however, mostly been overlooked in previous research. Existing solutions mainly focus on two of these fields, leaving the space for a niche. For this reason, the term Education Intelligence was suggested to fill the gap of a missing scientific method in software engineering education. Although the system was created with the intention of assisting course staff in an educational context, its broad usefulness is not confined to software engineering education only.

The literature and technology reviews revealed that there are already existing solutions for educational purposes which all, however, focus on different aspects in education. There is hardly any research conducted regarding MSR in an educational context. The same accounts for data visualization, whose focus is mostly on business or scientific needs.

Based on the dataset exported from GitLab, the literature and technology research, conceptual visualizations were created and a solution for the Extract, Transform, Load (ETL) tasks was developed. In order to address the first research question (*RQ1*) about the expert's information needs and a suitable process and architecture for data collection, the relevant information is described in Section 5.3 and Chapter 6. As this thesis is built upon previous scientific research [53, 56], the initially derived hypotheses had to be evaluated and the concepts had to be rated in expert interviews. Based on the results of these interviews, which were conducted with the undergraduate software engineering course staff at the TU Wien, the ETL implementation could be tailored to the identified needs.

The feedback provided in the first interviews served as a solid basis for the prototype refinement round. In an interview experts evaluated the Apache Superset based prototype of Chapter 7 and also validated the hypotheses about a suitable intelligence system for a software engineering class. In Section 8.4, the insights gained from the second round of interviews were presented. These insights were used to determine the level of satisfaction that the experts had with the implementation of charts and dashboards (*RQ2*). On a Likert scale [40], the experts rated the system an average of 4.48 out of 5 in terms of appropriateness. The system's usability was evaluated as 91.5 out of 100 points.

In order to answer the data analysis related questions as specified for *RQ3*, Section 8.5 discusses the findings of Section 8.3 greater detail. This thesis examined student's grades and time spent of both course phases (Individual Phase and Group Phase) and on class level. On the one hand, there were significant differences within a term between groups and students, with some extreme outliers being measured. On the other hand, when comparing terms as a whole, these outliers were balanced out by the overall results, leading to smaller differences. However, the data for both cases indicates that the expected effort in terms of time for both phases exceeded the expectations of the course staff.

Future Work

Due to this study's time constraints, not all ideas and analyses proposed by experts could be realized. Thus, this thesis has given rise to many aspects in need of further investigation. In the following, possible future research in this field is outlined.

As this thesis is built upon the GitLab archived data, there is currently no *live* monitoring possible. GitLab provides two options for live monitoring, based on a modified program: The data can either be accessed during CI pipeline runs or based on Webhooks[99]. As both options definitely have advantages and disadvantages, no suggestion can be made without more research.

As mentioned at the beginning of this thesis, the idea of creating a EI system is based on the "Portfoliotrix" [53] and "Binocular" [56] systems. The capabilities of these two systems in combination with the visualizations proposed by this study allows lecturers to gain even more and also different insights.

Section 3.1 describes the experience report by Pérez and Rubio [95] about Project Based Learning (PBL) in software engineering classes. The application of PBLs show significant improvements over time, as discussed. The proposed analyses and visualizations might serve as a sound starting point for a similar PBL format in the future. However, this also requires some course redesign.

Lastly, from a technical perspective, due to the fact that the architecture was implemented as a proof of concept, the system should also be deployable in a state-of-the-art manner,

[99] https://docs.gitlab.com/ee/user/project/integrations/webhooks.html, Accessed: *23.01.2023*

interacting with the GitLab system in the existing Kubernetes environment. Furthermore, the prototype implementation can still be improved at some points for a more stable and faster data transformation. The loading process of the stored information is currently based on PostgreSQL. However, since various data sources and future metrics might not work well in a relational database environment, Apache Drill[100] should be considered as a data connector application.

[100] https://drill.apache.org/, Accessed: *23.01.2023*

List of Figures

List of Tables

List of Algorithms

List of Listings

Acronyms

API Application Programming Interface. 10, 24, 28, 67, 73

BI Business Intelligence. 2, 4, 8, 23, 24

CI Continuous Integration. 2, 124

CPU Central processing unit. 71, 92

CRUD Create, Read, Update, and Delete. 10

CSV Comma-separated values. 26, 63, 71, 72, 75, 92, 93

ECTS European Credit Transfer and Accumulation System. 10, 107

EI Education Intelligence. 1, 2, 5, 83, 96, 123, 124

ETL Extract, Transform, Load. 6, 8, 63, 77, 123

GP Group Phase. 2, 4, 10, 11, 70–73, 83, 84, 89, 96, 97, 100, 102, 104, 106, 107, 109, 112, 113, 116, 119–121, 124

HTML Hypertext Markup Language. 23, 82

HTTP Hypertext Transfer Protocol. 10, 82

IDE Integrated Development Environment. 26, 29, 34

IP Individual Phase. 2, 4, 10, 11, 63, 64, 70–73, 79, 89, 94, 96, 97, 99, 100, 102, 104–107, 109–111, 119–121, 124

IQR Interquartile range. 16, 99, 108, 119

ITS Issue Tracking System. 8

JSON JavaScript Object Notation. 23, 27, 64, 66, 68–70, 93, 131

LoC Lines of Code. 40

MR Merge request. 64–66, 93, *Glossary:* Merge request

MSR Mining Software Repositories. 1, 8, 19, 20, 32, 84, 123

MV Materialized View. 35, 77–79, 81

ORM Object–relational mapping. 68

PBL Project Based Learning. 22, 124

REST Representational state transfer. 9, 10

SI Software Intelligence. 20

SQL Structured Query Language. 9, 10, 26, 66, 78, 79, 81, 82

SUS System Usability Scale. 117–119

UI User Interface. 2, 10, 26, 66, 118

US User Story. 3, 21

VCS Version Control System. 1, 8

XML Extensible Markup Language. 23, 82

Glossary

ELK Stack The abbreviation *ELK* stands for Elasticsearch[101], Logstash[102], and Kibana[103], three open source projects. Elasticsearch is a data analytics and search engine. Logstash is a server-side data processing pipeline and transfers it to a *stash* like Elasticsearch. In Elasticsearch, Kibana allows users to view data using charts and graphs[104] . 26

Git Distributed version control system . 5, 8, 19, 21, 23, 30, 34, 53, 55, 63, 68, 72, 73, 77, 84, 92, 94

GitLab GitLab is a source code hosting platform[105]. 4, 20, 30, 33, 34, 54, 55, 63, 64, 66–68, 71–76, 83, 93–96, 116, 123–125, 131

Merge request A merge request is a way of integrating a source branch A into a target branch B. Such a merge request may include an unlimited number of commits. When the (optional) reviewer approves the merge request, the code will be merged into the targeted branch . 64, 134

Timelog A timelog is defines as a single entry of spent time, and can either be positive (adding time) or negative (removing time)[106]. 64, 65, 84, 94, 96

[101] https://www.elastic.co/elasticsearch/, Accessed: *23.01.2023*
[102] https://www.elastic.co/logstash/, Accessed: *23.01.2023*
[103] https://www.elastic.co/kibana/, Accessed: *23.01.2023*
[104] Source: https://www.elastic.co/what-is/elk-stack, Accessed: *23.01.2023*
[105] https://gitlab.com/, Accessed: *23.01.2023*
[106] https://docs.gitlab.com/15.0/ee/user/project/time_tracking.html, Accessed: *23.01.2023*

Bibliography

Print Resources

[1] Shari L Kitchenham Barbara A. and Pfleeger. "Personal Opinion Surveys". In: ed. by Janice, Sjøberg Dag I K Shull Forrest, and Singer. Springer London, 2008, pp. 63–92. ISBN: 978-1-84800-044-5. DOI: 10.1007/978-1-84800-044-5_3. URL: https://doi.org/10.1007/978-1-84800-044-5_3.

[2] Efthimia Aivaloglou and Anna van der Meulen. "An Empirical Study of Students' Perceptions on the Setup and Grading of Group Programming Assignments". In: *ACM Trans. Comput. Educ.* 21.3 (2021). DOI: 10.1145/3440994. URL: https://doi.org/10.1145/3440994.

[3] Noah Arthurs et al. "Grades are not Normal: Improving Exam Score Models Using the Logit-Normal Distribution". In: *EDM*. 2019.

[5] Adrian Bachmann et al. "The Missing Links: Bugs and Bug-Fix Commits". In: *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE '10. Santa Fe, New Mexico, USA: Association for Computing Machinery, 2010, 97–106. ISBN: 9781605587912. DOI: 10.1145/1882291.1882308. URL: https://doi.org/10.1145/1882291.1882308.

[6] Aaron Bangor, Philip T. Kortum, and James T. Miller. "An Empirical Evaluation of the System Usability Scale". In: *International Journal of Human-Computer Interaction* 24 (6 July 2008), pp. 574–594. ISSN: 1044-7318. DOI: 10.1080/10447310802205776.

[7] Elisa Baniassad et al. "STOP THE (AUTOGRADER) INSANITY: Regression Penalties to Deter Autograder Overreliance". In: ACM, Mar. 2021, pp. 1062–1068. ISBN: 9781450380621. DOI: 10.1145/3408877.3432430.

[8] Olga Baysal, Reid Holmes, and Michael W Godfrey. "Situational awareness: personalizing issue tracking systems". In: *2013 35th International Conference on Software Engineering (ICSE)*. IEEE. 2013, pp. 1185–1188. DOI: 10.1109/ICSE.2013.6606674.

[9] Dane Bertram et al. “Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams”. In: *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. 2010, pp. 291–300. DOI: 10.1145/1718918.1718972.

[11] Michael Blaha, David LaPlant, and Erica Marvak. “Requirements for repository software”. In: *Proceedings Fifth Working Conference on Reverse Engineering (Cat. No. 98TB100261)*. IEEE. 1998, pp. 164–173. DOI: 10.1109/WCRE.1998.723186.

[12] Jose A. Blakeley, Per-Ake Larson, and Frank Wm Tompa. “Efficiently updating materialized views”. In: *ACM SIGMOD Record* 15 (2 June 1986), pp. 61–71. ISSN: 0163-5808. DOI: 10.1145/16856.16861.

[13] Alejandro Bogarín, Rebeca Cerezo, and Cristóbal Romero. “A survey on educational process mining”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8 (1 Jan. 2018), e1230. ISSN: 19424787. DOI: 10.1002/widm.1230.

[14] David Bowen et al. “Team Skills of Engineers–Do We Teach What Industry Wants?” In: *Proceedings, International Conference on Engineering Education*. 2004, pp. 16–21.

[15] John Brooke et al. “SUS-A quick and dirty usability scale”. In: *Usability evaluation in industry* 189.194 (1996), pp. 4–7.

[16] Kevin Buffardi. “Assessing Individual Contributions to Software Engineering Projects with Git Logs and User Stories”. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE ’20. Portland, OR, USA: Association for Computing Machinery, 2020, 650–656. ISBN: 9781450367936. DOI: 10.1145/3328778.3366948. URL: https://doi.org/10.1145/3328778.3366948.

[19] Stephanie Carlisle. “Software: Tableau and Microsoft Power BI”. In: *Technology/Architecture + Design* 2 (2 July 2018), pp. 256–259. ISSN: 2475-1448. DOI: 10.1080/24751448.2018.1497381.

[20] Donald D. Chamberlin. “Early History of SQL”. In: *IEEE Annals of the History of Computing* 34.4 (2012), pp. 78–82. DOI: 10.1109/MAHC.2012.61.

[21] Donald D. Chamberlin and Raymond F. Boyce. “SEQUEL: A Structured English Query Language”. In: *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*. SIGFIDET ’74. Ann Arbor, Michigan: Association for Computing Machinery, 1974, 249–264. ISBN: 9781450374156. DOI: 10.1145/800296.811515. URL: https://doi.org/10.1145/800296.811515.

[22] Hock Chuan Chan, Hong Jun Lu, and Kwok Kee Wei. “A Survey of SQL Language”. In: *Journal of Database Management* 4 (4 Oct. 1993), pp. 4–16. ISSN: 1063-8016. DOI: 10.4018/jdm.1993100101.

[23] Miguel Morales Chan et al. “Perceived usefulness and motivation students towards the use of a cloud-based tool to support the learning process in a Java MOOC”. In: *Proceedings of the International Conference MOOC-MAKER*. 2017, pp. 73–82.

[24] Kenneth J. Chapman and Stuart van Auken. “Creating Positive Group Project Experiences: An Examination of the Role of the Instructor on Students’ Perceptions of Group Projects”. In: *Journal of Marketing Education* 23 (2 Aug. 2001), pp. 117–127. ISSN: 0273-4753. DOI: 10.1177/0273475301232005.

[25] K.K. Chaturvedi, V.B. Sing, and Prashast Singh. “Tools in Mining Software Repositories”. In: IEEE, June 2013, pp. 89–98. ISBN: 978-0-7695-5045-9. DOI: 10.1109/ICCSA.2013.22.

[26] Surajit Chaudhuri, Umeshwar Dayal, and Vivek Narasayya. “An overview of business intelligence technology”. In: *Communications of the ACM* 54 (8 Aug. 2011), pp. 88–98. ISSN: 0001-0782. DOI: 10.1145/1978542.1978562.

[27] Jian Chen et al. “Assessing Teamwork Performance in Software Engineering Education: A Case in a Software Engineering Undergraduate Course”. In: *2011 18th Asia-Pacific Software Engineering Conference*. 2011, pp. 17–24. DOI: 10.1109/APSEC.2011.50.

[28] Li-Pang Chen. “Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python”. In: *Technometrics* 63 (2 Apr. 2021), pp. 272–273. ISSN: 0040-1706. DOI: 10.1080/00401706.2021.1904738.

[29] Edgar Frank Codd. “A Relational Model of Data for Large Shared Data Banks”. In: *Commun. ACM* 13.6 (1970), 377–387. ISSN: 0001-0782. DOI: 10.1145/362384.362685. URL: https://doi.org/10.1145/362384.362685.

[30] Carol L. Colbeck, Susan E. Campbell, and Stefani A. Bjorklund. “Grouping in the Dark”. In: *The Journal of Higher Education* 71 (1 Jan. 2000), pp. 60–83. ISSN: 0022-1546. DOI: 10.1080/00221546.2000.11780816.

[31] Robert Gravlin Cooper, Scott J Edgett, and Elko J Kleinschmidt. “Portfolio management for new products”. In: (2001).

[32] Julio César Cortés Ríos et al. “A Methodology for Using GitLab for Software Engineering Learning Analytics”. In: *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 2019, pp. 3–6. DOI: 10.1109/CHASE.2019.00009.

[33] Michelle Craig et al. “Listening to Early Career Software Developers”. In: *J. Comput. Sci. Coll.* 33.4 (2018), 138–149. ISSN: 1937-4771.

[34] Ilka Datig and Paul Whiting. “Telling your library story: tableau public for data visualization”. In: *Library Hi Tech News* 35 (4 Aug. 2018), pp. 6–8. ISSN: 0741-9058. DOI: 10.1108/LHTN-02-2018-0008.

[35] Vladan Devedzic et al. “Metrics for Students’ Soft Skills”. In: *Applied Measurement in Education* 31 (4 Oct. 2018), pp. 283–296. ISSN: 0895-7347. DOI: 10.1080/08957347.2018.1495212.

[37] Philipp Dumbach et al. “Exploration of Process Mining Opportunities In Educational Software Engineering - The GitLab Analyser”. In: *EDM*. 2020.

[39] Suvi Elonen and Karlos A. Artto. “Problems in managing internal development projects in multi-project environments”. In: *International Journal of Project Management* 21 (6 Aug. 2003), pp. 395–402. ISSN: 02637863. DOI: 10.1016/S0263-7863(02)00097-2.

[40] Robert Wall Emerson. “Likert Scales”. In: *Journal of Visual Impairment & Blindness* 111 (5 Sept. 2017), pp. 488–488. ISSN: 0145-482X. DOI: 10.1177/0145482X1711100511.

[42] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. “Knowledge Discovery and Data Mining: Towards a Unifying Framework”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, 82–88.

[43] Michael Felderer and Fabian Jeschko. “A Process for Evidence-Based Engineering of Domain-Specific Languages”. In: ACM, June 2018, pp. 169–174. ISBN: 9781450364034. DOI: 10.1145/3210459.3210479.

[44] Lynn Fendler and Irfan Muzaffar. “THE HISTORY OF THE BELL CURVE: SORTING AND THE IDEA OF NORMAL”. In: *Educational Theory* 58 (1 Feb. 2008), pp. 63–82. ISSN: 0013-2004. DOI: 10.1111/j.1741-5446.2007.0276.x.

[46] Roy Thomas Fielding and Richard N. Taylor. “Architectural Styles and the Design of Network-Based Software Architectures”. AAI9980887. PhD thesis. 2000. ISBN: 0599871180.

[48] Jill J. Francis et al. “What is an adequate sample size? Operationalising data saturation for theory-based interview studies”. In: *Psychology & Health* 25 (10 Dec. 2010), pp. 1229–1245. ISSN: 0887-0446. DOI: 10.1080/08870440903194015.

[49] David Freedman and Persi Diaconis. “On the histogram as a density estimator:L2 theory”. In: *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 57 (4 1981). ISSN: 00443719. DOI: 10.1007/BF01025868.

[50] Michael Friendly. “Milestones in the History of Data Visualization: A Case Study in Statistical Historiography”. In: *Classification — the Ubiquitous Challenge*. Ed. by Claus Weihs and Wolfgang Gaul. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 34–52. ISBN: 978-3-540-28084-2.

[51] Rose F. Gamble and Matthew L. Hale. “Assessing individual performance in Agile undergraduate software engineering teams”. In: *2013 IEEE Frontiers in Education Conference*. 2013, pp. 1678–1684. DOI: 10.1109/FIE.2013.6685123.

[53] Patric Genfer et al. “Visualizing Metric Trends for Software Portfolio Quality Management”. In: *2021 Working Conference on Software Visualization (VISSOFT)*. 2021, pp. 88–99. DOI: 10.1109/VISSOFT52517.2021.00018.

[54] Carlo Ghezzi and Dino Mandrioli. “The Challenges of Software Engineering Education”. In: 2006, pp. 115–127. DOI: 10.1007/11949374_8.

[55] Nicolas E. Gold and Jens Krinke. “Ethics in the mining of software repositories”. In: *Empirical Software Engineering* 27 (1 Jan. 2022), p. 17. ISSN: 1382-3256. DOI: 10.1007/s10664-021-10057-7.

[56] Johann Grabner et al. “Combining and Visualizing Time-Oriented Data from the Software Engineering Toolset”. In: *2018 IEEE Working Conference on Software Visualization (VISSOFT)*. 2018, pp. 76–86. DOI: 10.1109/VISSOFT.2018.00016.

[58] Ashish Gupta, Inderpal Singh Mumick, and V. S. Subrahmanian. “Maintaining views incrementally”. In: *ACM SIGMOD Record* 22 (2 June 1993), pp. 157–166. ISSN: 0163-5808. DOI: 10.1145/170036.170066.

[60] P.E. Hadjidoukas et al. “torcpy: Supporting task parallelism in Python”. In: *SoftwareX* 12 (July 2020), p. 100517. ISSN: 23527110. DOI: 10.1016/j.softx.2020.100517.

[61] Matthew Hale, Noah Jorgenson, and Rose Gamble. “Predicting individual performance in student project teams”. In: *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)*. 2011, pp. 11–20. DOI: 10.1109/CSEET.2011.5876078.

[62] Maria Halkidi et al. “Data mining in software engineering”. In: *Intelligent Data Analysis* 15.3 (2011), pp. 413–441.

[63] Ahmed E. Hassan. “The road ahead for Mining Software Repositories”. In: IEEE, Sept. 2008, pp. 48–57. ISBN: 978-1-4244-2654-6. DOI: 10.1109/FOSM.2008.4659248.

[64] Ahmed E. Hassan and Tao Xie. “Software Intelligence: The Future of Mining Software Engineering Data”. In: *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*. FoSER ’10. Santa Fe, New Mexico, USA: Association for Computing Machinery, 2010, 161–166. ISBN: 9781450304276. DOI: 10.1145/1882362.1882397. URL: https://doi.org/10.1145/1882362.1882397.

[65] Marc Hesenius et al. “Towards a Software Engineering Process for Developing Data-Driven Applications”. In: *2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*. 2019, pp. 35–41. DOI: 10.1109/RAISE.2019.00014.

[66] Mike Hintze and Khaled El Emam. “Comparing the benefits of pseudonymisation and anonymisation under the GDPR”. In: *Journal of Data Protection & Privacy* 2.2 (2018), pp. 145–158.

[67] John Hunt. “Multiprocessing”. In: 2019, pp. 363–376. DOI: 10.1007/978-3-030-25943-3_31.

[69] David Jackson. “A software system for grading student computer programs”. In: *Computers & Education* 27.3 (1996), pp. 171–180. ISSN: 0360-1315. DOI: https://doi.org/10.1016/S0360-1315(96)00025-5. URL: https://www.sciencedirect.com/science/article/pii/S0360131596000255.

[71] An Ju and Armando Fox. "TEAMSCOPE: Measuring Software Engineering Processes with Teamwork Telemetry". In: *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE 2018. Larnaca, Cyprus: Association for Computing Machinery, 2018, 123–128. ISBN: 9781450357074. DOI: 10.1145/3197091.3197107. URL: https://doi.org/10.1145/3197091.3197107.

[72] Eirini Kalliamvakou et al. "The Promises and Perils of Mining GitHub". In: *Proceedings of the 11th Working Conference on Mining Software Repositories*. MSR 2014. Hyderabad, India: Association for Computing Machinery, 2014, 92–101. ISBN: 9781450328630. DOI: 10.1145/2597073.2597074. URL: https://doi.org/10.1145/2597073.2597074.

[73] Yasutaka Kamei and Andy Zaidman. "Guest editorial: Mining software repositories 2018". In: *Empirical Software Engineering* 25 (3 May 2020), pp. 2055–2057. ISSN: 1382-3256. DOI: 10.1007/s10664-020-09817-8.

[74] D.A. Keim. "Information visualization and visual data mining". In: *IEEE Transactions on Visualization and Computer Graphics* 8.1 (2002), pp. 1–8. DOI: 10.1109/2945.981847.

[75] D.A. Keim et al. "Challenges in Visual Data Analysis". In: *Tenth International Conference on Information Visualisation (IV'06)*. 2006, pp. 9–16. DOI: 10.1109/IV.2006.31.

[76] Iman Keivanloo et al. "A Linked Data platform for mining software repositories". In: *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. 2012, pp. 32–35. DOI: 10.1109/MSR.2012.6224296.

[77] Tobias Kuipers and Joost Visser. "A Tool-based Methodology for Software Portfolio Monitoring". In: *Software Audit and Metrics*. 2004.

[78] Khoa Le, Caslon Chua, and Rosalind Wang. "Mining Software Engineering Team Project Work Logs to Generate Formative Assessment". In: *2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW)*. 2017, pp. 78–83. DOI: 10.1109/APSECW.2017.19.

[79] Ronald J. Leach. "Using Metrics to Evaluate Student Programs". In: *SIGCSE Bull.* 27.2 (June 1995), 41–43. ISSN: 0097-8418. DOI: 10.1145/201998.202010. URL: https://doi.org/10.1145/201998.202010.

[80] PS Lokhande et al. "Efficient way of web development using python and flask". In: (2015).

[81] H. P. Luhn. "A Business Intelligence System". In: *IBM Journal of Research and Development* 2 (4 Oct. 1958), pp. 314–319. ISSN: 0018-8646. DOI: 10.1147/rd.24.0314.

[82] Ami Marowka. "Python accelerators for high-performance computing". In: *The Journal of Supercomputing* 74 (4 Apr. 2018), pp. 1449–1460. ISSN: 0920-8542. DOI: 10.1007/s11227-017-2213-5.

[83] Petito Michele, Francesca Fallucchi, and Ernesto William De Luca. “Create Dashboards and Data Story with the Data & Analytics Frameworks”. In: 2019, pp. 272–283. DOI: 10.1007/978-3-030-36599-8_24.

[84] Stephen R. Midway. “Principles of Effective Data Visualization”. In: *Patterns* 1 (9 Dec. 2020), p. 100141. ISSN: 26663899. DOI: 10.1016/j.patter.2020.100141.

[85] Nuthan Munaiah et al. “Curating github for engineered software projects”. In: *Empirical Software Engineering* 22.6 (2017), pp. 3219–3253. DOI: 10.1007/s10664-017-9512-6.

[86] Sandeep Nagar. “Introduction to Python Basics”. In: Apress, 2018, pp. 13–30. DOI: 10.1007/978-1-4842-3204-0_2.

[87] Henrik R Nagel. “Scientific visualization versus information visualization”. In: *Workshop on state-of-the-art in scientific and parallel computing, Sweden*. Citeseer. 2006, pp. 8–9.

[88] Solomon Negash and Paul Gray. “Business Intelligence”. In: Springer Berlin Heidelberg, 2008, pp. 175–193. DOI: 10.1007/978-3-540-48716-6_9.

[89] Hung Nguyen et al. “PANDORA: Continuous mining software repository and dataset generation”. In: *EEE International Conference on Software Analysis, Evolution and Reengineering (SANER2022). IEEE*. 2022.

[90] Robert Nisbet, Gary Miner, and Ken Yale. *Chapter 6 - Accessory Tools for Doing Data Mining*. Ed. by Robert Nisbet, Gary Miner, and Ken Yale. 2018. DOI: https://doi.org/10.1016/B978-0-12-416632-5.00006-2. URL: https://www.sciencedirect.com/science/article/pii/B9780124166325000062.

[91] M.C. Ferreira de Oliveira and H. Levkowitz. “From visual data exploration to visual data mining: a survey”. In: *IEEE Transactions on Visualization and Computer Graphics* 9.3 (2003), pp. 378–394. DOI: 10.1109/TVCG.2003.1207445.

[93] 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)*. 2018, pp. 1–5. DOI: 10.1109/FIE.2018.8658983.

[94] Arnold L. Patton and Monica McGill. “Student Portfolios and Software Quality Metrics in Computer Science Education”. In: *J. Comput. Sci. Coll.* 21.4 (2006), 42–48. ISSN: 1937-4771.

[95] Beatriz Pérez and Ángel L. Rubio. “A Project-Based Learning Approach for Enhancing Learning Skills and Motivation in Software Engineering”. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE ’20. Portland, OR, USA: Association for Computing Machinery, 2020, 309–315. ISBN: 9781450367936. DOI: 10.1145/3328778.3366891. URL: https://doi.org/10.1145/3328778.3366891.

[96] Alex Radermacher and Gursimran Walia. “Gaps between industry expectations and the abilities of graduates”. In: ACM Press, 2013, p. 525. ISBN: 9781450318686. DOI: 10.1145/2445196.2445351.

[97] Sebastian Raschka, Joshua Patterson, and Corey Nolet. “Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence”. In: *Information* 11 (4 Apr. 2020), p. 193. ISSN: 2078-2489. DOI: 10.3390/info11040193.

[99] Robert Richards. *Representational State Transfer (REST)*. 2006. DOI: 10.1007/978-1-4302-0139-7_17.

[100] Daniel Rubio. “Jinja Templates in Django”. In: Apress, 2017, pp. 117–161. ISBN: 978-1-4842-2786-2. DOI: 10.1007/978-1-4842-2787-9_4.

[101] Jim Rudd, Ken Stern, and Scott Isensee. “Low vs. High-Fidelity Prototyping Debate”. In: *Interactions* 3.1 (1996), 76–85. ISSN: 1072-5520. DOI: 10.1145/223500.223514. URL: https://doi.org/10.1145/223500.223514.

[102] Larry J. Shuman, Mary Besterfield-Sacre, and Jack McGourty. “The ABET “Professional Skills” - Can They Be Taught? Can They Be Assessed?” In: *Journal of Engineering Education* 94 (1 Jan. 2005), pp. 41–55. ISSN: 10694730. DOI: 10.1002/j.2168-9830.2005.tb00828.x.

[103] Andy Siddaway. “What is a systematic literature review and how do I do one”. In: *University of Stirling* 1.1 (2014), pp. 1–13.

[104] Tamanna Siddiqui and Ausaf Ahmad. “Data Mining Tools and Techniques for Mining Software Repositories: A Systematic Review”. In: 2018, pp. 717–726. DOI: 10.1007/978-981-10-6620-7_70.

[105] Daniel Simon, Kai Fischbach, and Detlef Schoder. “Application Portfolio Management—An Integrated Framework and a Software Tool Evaluation Approach”. In: *Communications of the Association for Information Systems* 26 (2010). ISSN: 15293181. DOI: 10.17705/1CAIS.02603.

[106] J.E. Sims-Knight et al. “Teams in software engineering education”. In: *32nd Annual Frontiers in Education*. Vol. 3. 2002, S3G–S3G. DOI: 10.1109/FIE.2002.1158712.

[107] J.E. Sims-Knight et al. “Teams in software engineering education”. In: *32nd Annual Frontiers in Education*. Vol. 3. 2002, S3G–S3G. DOI: 10.1109/FIE.2002.1158712.

[108] Francisco Zigmund Sokol, Mauricio Finavaro Aniche, and Marco Aurélio Gerosa. “MetricMiner: Supporting researchers in mining software repositories”. In: *2013 IEEE 13th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. 2013, pp. 142–146. DOI: 10.1109/SCAM.2013.6648195.

[109] Davide Spadini, Maurício Aniche, and Alberto Bacchelli. “PyDriller: Python framework for mining software repositories”. In: *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*. New York, New York, USA: ACM Press, 2018, pp. 908–911. ISBN: 9781450355735. DOI: 10.1145/3236024.3264598. URL: http://dl.acm.org/citation.cfm?doid=3236024.3264598.

[110] Diomidis Spinellis. “Git”. In: *IEEE software* 29.3 (2012), pp. 100–101. DOI: 10.1109/MS.2012.61.

[111] KR Srinath. “Python–the fastest growing programming language”. In: *International Research Journal of Engineering and Technology* 4.12 (2017), pp. 354–357.

[113] Robert J. Sternberg. “The School Bell and The Bell Curve. Why They Don’t Mix”. In: *NASSP Bulletin* 80 (577 Feb. 1996), pp. 46–56. ISSN: 0192-6365. DOI: 10.1177/019263659608057710.

[114] Michael Stonebraker et al. “On rules, procedure, caching and views in data base systems”. In: *ACM SIGMOD Record* 19 (2 May 1990), pp. 281–290. ISSN: 0163-5808. DOI: 10.1145/93605.98737.

[116] Fabian Trautsch et al. “Addressing problems with replicability and validity of repository mining studies through a smart data platform”. In: *Empirical Software Engineering* 23 (2 Apr. 2018), pp. 1036–1083. ISSN: 1382-3256. DOI: 10.1007/s10664-017-9537-x.

[117] John Rodney Turner. “The handbook of project-based management: improving the processes for achieving strategic objectives”. In: (1999).

[118] Olivier Vandecruys et al. “Mining software repositories for comprehensible software fault prediction models”. In: *Journal of Systems and Software* 81 (5 May 2008), pp. 823–839. ISSN: 01641212. DOI: 10.1016/j.jss.2007.07.034.

[119] Panos Vassiliadis. “A Survey of Extract-Transform-Load Technology”. In: *International Journal of Data Warehousing and Mining* 5 (3 July 2009), pp. 1–27. ISSN: 1548-3924. DOI: 10.4018/jdwm.2009070101.

[120] J.M. Verner et al. “Guidelines for industrially-based multiple case studies in software engineering”. In: IEEE, Apr. 2009, pp. 313–324. ISBN: 978-1-4244-2864-9. DOI: 10.1109/RCIS.2009.5089295.

[121] M. Vidoni. “A systematic process for Mining Software Repositories: Results from a systematic literature review”. In: *Information and Software Technology* 144 (Apr. 2022), p. 106791. ISSN: 09505849. DOI: 10.1016/j.infsof.2021.106791.

[122] Matthias Weiß. “A Lightweight and Integrated Software Repository Mining and Visualisation Approach for Software Engineering Education”. MA thesis. Technische Universität Wien, 2022. DOI: https://doi.org/10.34726/hss.2022.102022.

[124] C.E. Wills. “Group-based software engineering in an introductory computer science course”. In: *Proceedings. 1998 International Conference Software Engineering: Education and Practice (Cat. No.98EX220)*. 1998, pp. 26–33. DOI: 10.1109/SEEP.1998.707630.

[125] Tao Xie, Jian Pei, and Ahmed E. Hassan. “Mining Software Engineering Data”. In: *29th International Conference on Software Engineering (ICSE'07 Companion)*. 2007, pp. 172–173. DOI: 10.1109/ICSECOMPANION.2007.50.

[126] Andy Zaidman et al. “Mining Software Repositories to Study Co-Evolution of Production & Test Code”. In: *2008 1st International Conference on Software Testing, Verification, and Validation*. 2008, pp. 220–229. DOI: 10.1109/ICST.2008.47.

[127] Nazatul Nurlisa Zolkifli, Amir Ngah, and Aziz Deraman. “Version control system: A review”. In: *Procedia Computer Science* 135 (2018), pp. 408–415. DOI: 10.1016/j.procs.2018.08.191.

Book References

[4] Henning Baars and Hans-Georg Kemper. *Business Intelligence & Analytics – Grundlagen und praktische Anwendungen*. Springer Fachmedien Wiesbaden, 2021. ISBN: 978-3-8348-1958-1. DOI: 10.1007/978-3-8348-2344-1.

[10] Robert M. Bethea. *Statistical Methods for Engineers and Scientists*. CRC Press, Apr. 2018. ISBN: 9781351414388. DOI: 10.1201/9780203738580.

[17] Field Cady. *The data science handbook*. John Wiley & Sons, 2017. ISBN: 9781119092940.

[18] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, eds. *Readings in Information Visualization: Using Vision to Think*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. ISBN: 1558605339.

[36] Directorate-General for Education, Youth, Sport and Culture (European Commission). *ECTS users' guide 2015*. Publications Office, 2017. DOI: doi/10.2766/87192. URL: https://op.europa.eu/s/wAMi.

[41] Eric Evans and Eric J Evans. *Domain-Driven Design: Tacking Complexity In the Heart of Software*. USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN: 0321125215.

[45] Alberto Ferrari and Marco Russo. *Introducing Microsoft Power BI*. Microsoft Press, 2016.

[47] Ben Forta. *SQL in 10 Minutes, Sams Teach Yourself*. en. 4th ed. Indianapolis, IN: Sams Publishing, 2012. ISBN: 9780672336072.

[57] Robert Grant. *Data Visualization*. Chapman and Hall/CRC, Dec. 2018. ISBN: 9781351781756. DOI: 10.1201/9781315201351.

[59] Werner Gurker. *Statistik und Wahrscheinlichkeistheorie using R*. TU Verlag, 2015. ISBN: 9783903024793.

[68] Wolfgang Karl Härdle, Sigbert Klinke, and Bernd Rönz. *Introduction to Statistics*. Springer International Publishing, 2015. ISBN: 978-3-319-17703-8. DOI: 10.1007/978-3-319-17704-5.

[70] David W Johnson and Roger T Johnson. *Learning together and alone: Cooperative, competitive, and individualistic learning*. Prentice-Hall, Inc, 1987.

[92] Jan Palach. *Parallel programming with Python*. Packt Publishing Ltd, 2014. ISBN: 9781783288397.

[98] G. Recht. *Bundesdatenschutzgesetz (BDSG)*. G. Recht, 2014. ISBN: 978-1-500-10002-5.

[115] Alexandru Telea. *Data visualization: principles and practice*. New York: CRC Press, 2014. ISBN: 9780429074080. DOI: 10.1201/b17217.

[123] Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014. ISBN: 978-3-662-43838-1. DOI: 10.1007/978-3-662-43839-8.

Online References

[38] IBM Cloud Education. *ETL (Extract, Transform, Load)*. Apr. 2020. URL: https://www.ibm.com/topics/etl (Accessed: *23.01.2023*).

[52] *GENERAL DATA PROTECTION REGULATION (GDPR)*. European Commission. May 25, 2018. URL: https://gdpr-info.eu/art-4-gdpr/ (Accessed: *23.01.2023*).

[112] National Institute of Standards and Technology. *NIST/SEMATECH e-Handbook of Statistical Methods*. Apr. 2012. DOI: https://doi.org/10.18434/M32189. URL: https://www.itl.nist.gov/div898/handbook/ (Accessed: *23.01.2023*).

Appendices

APPENDIX A

PostgreSQL Views

```sql
SELECT _grp.committer_student_matriculation_number AS
    student_matriculation_number,
    _grp.term_id,
    _grp.term_name,
    count(_grp.commit_hash) AS grp_commits_count,
        CASE
            WHEN count(DISTINCT _grp.commit_hash) < 63 THEN '< 63 (Q1)'
            WHEN count(DISTINCT _grp.commit_hash) < 96 THEN '< 96 (Q2)'
            WHEN count(DISTINCT _grp.commit_hash) < 139 THEN '< 139 (Q3)'
            ELSE '>= 139 (Q4)'
        END AS grp_commits_cat
FROM "group".grp_commit_user_mview _grp
WHERE _grp.committer_student_matriculation_number IS NOT NULL
GROUP BY _grp.committer_student_matriculation_number, _grp.term_id,
    _grp.term_name
```

Listing A.1: grp_commits_cat View

```sql
SELECT
    _issue.id AS issue_id,
    _issue.issue_id AS issue_gitlab_id,
    _issue.issue_internal_id AS issue_gitlab_internal_id,
    _issue.title AS issue_title,
    _issue.created_at AS issue_created_at,
    _issue.updated_at AS issue_updated_at,
    _issue.closed_at AS issue_closed_at,
    _project.project_id AS project_id,
    _project.project_name_filterable AS project_name_filterable,
    _project.project_full_name AS project_full_name,
    _project.project_short_name AS project_short_name,
    _term.id AS term_id,
    _term.term_name AS term_name,
    _project.project_research_division AS project_research_division
```

```
16 FROM "gitlab".gitlab_project_issue _issue
17 LEFT JOIN "group".grp_project_mview _project ON _project.project_id =
       _issue.project_id
18 JOIN "csv".term _term ON _term.id = _project.project_term_id
```

Listing A.2: grp_issue_project_view View

```
1  CREATE VIEW "public".student_grades_view AS
2  SELECT
3    sub.*,
4    CASE
5      WHEN sub.grp_grade_num IS NOT NULL THEN
           ROUND(sub.ind_grade_num*0.25+sub.grp_grade_num*0.75)
6      ELSE 5
7    END AS lecture_grade
8  FROM (SELECT
9    _ind_repos.student_matriculation_number,
10   _ind_repos.term_name,
11   _ind_repos.result_points,
12   CASE
13        WHEN _ind_repos.result_points < 40 THEN '5 (N5)'
14        WHEN _ind_repos.result_points < 50 THEN '4 (G4)'
15        WHEN _ind_repos.result_points < 60 THEN '3 (B3)'
16        WHEN _ind_repos.result_points < 70 THEN '2 (G2)'
17        WHEN _ind_repos.result_points >= 70 THEN '1 (S1)'
18     END AS ind_grade_str,
19     CASE
20        WHEN _ind_repos.result_points < 40 THEN 5
21        WHEN _ind_repos.result_points < 50 THEN 4
22        WHEN _ind_repos.result_points < 60 THEN 3
23        WHEN _ind_repos.result_points < 70 THEN 2
24        WHEN _ind_repos.result_points >= 70 THEN 1
25     END AS ind_grade_num,
26   CASE
27     WHEN _grp_upv.grade = 5 THEN '5 (N5)'
28     WHEN _grp_upv.grade = 4 THEN '4 (G4)'
29     WHEN _grp_upv.grade = 3 THEN '3 (B3)'
30     WHEN _grp_upv.grade = 2 THEN '2 (G2)'
31     WHEN _grp_upv.grade = 1 THEN '1 (S1)'
32   END AS grp_grade_str,
33   _grp_upv.grade AS grp_grade_num
34 FROM "individual".ind_results_non_empty_repos _ind_repos
35 LEFT OUTER JOIN "group".grp_user_project_mview _grp_upv ON
36   _ind_repos.student_matriculation_number =
         _grp_upv.student_matriculation_number AND
37   _ind_repos.term_id = _grp_upv.term_id
38 ) AS sub
39 ORDER BY "lecture_grade";
```

Listing A.3: student_grades_view View

```
1 SELECT
```

```
    _timelog.id AS timelog_id,
    _timelog.timelog_id AS timelog_timelog_id,
    _timelog.time_spent AS timelog_time_spent,
    _timelog.spent_at AS timelog_spent_at,
    _timelog.issue_id AS timelog_issue_id,
    _upv.user_id,
    _upv.student_matriculation_number,
    _upv.student_firstname,
    _upv.student_lastname,
    _upv.user_username,
    _upv.user_email,
    _upv.project_id,
    _upv.project_name_filterable,
    _upv.project_full_name,
    _upv.project_short_name,
    _upv.term_id,
    _upv.term_name,
    _upv.project_research_division
FROM "gitlab".gitlab_timelog _timelog
LEFT JOIN "group".grp_user_project_mview _upv ON
    _upv.user_id = _timelog.user_id AND
    _upv.project_id = _timelog.project_id
```

Listing A.4: `grp_user_timelog_view` View

APPENDIX B

PostgreSQL Materialized Views

```sql
SELECT
    _commit.id AS commit_id,
    _commit.authored_date AS commit_authored_date,
    _commit.committed_date AS commit_committed_date,
    _commit.commit_id AS commit_hash,
    _commit.committer_id AS commit_committer_id,
    _commit.author_id AS commit_author_id,
    _commit.project_id AS commit_project_id,
    _branch.id AS branch_id,
    _branch.name AS branch_name
FROM "gitlab".git_commit _commit
JOIN "gitlab".git_branch_commit_association _ass ON _ass.commit_id = _commit.id
JOIN "gitlab".git_branch _branch ON _branch.id = _ass.branch_id
```

Listing B.1: `commit_branch_mview` Materialized View

```sql
SELECT
    _project.id AS project_id,
    _project.full_name AS project_full_name,
    COALESCE(_group_project.short_name, split_part(_project.full_name, '/', 3))
        AS project_short_name,
    _group_project.research_division AS project_research_division,
    _project.term_id AS project_term_id,
    concat(_term.term_name, ', ', COALESCE(_group_project.short_name,
        split_part(_project.full_name, '/', 3))) AS project_name_filterable
FROM "gitlab".gitlab_project _project
JOIN "gitlab".gitlab_group_project _group_project ON _group_project.id =
    _project.id
JOIN "csv".term _term ON _term.id = _project.term_id
```

Listing B.2: `grp_project_mview` Materialized View

```
SELECT
    _student.id AS student_id,
    _student.matriculation_number AS student_matriculation_number,
    _student.firstname AS student_firstname,
    _student.lastname AS student_lastname,
    _term.id AS term_id,
    _term.term_name AS term_name,
    _ipr.id AS result_id,
    _ipr.points AS result_points
FROM "csv".student _student
JOIN "csv".student_term_association _sta ON _student.id = _sta.student_id
JOIN "csv".term _term ON _term.id = _sta.term_id
JOIN "csv".individual_phase_result _ipr ON _ipr.student_id = _student.id AND
    _ipr.term_id = _term.id
```

Listing B.3: `student_term_result_mview` Materialized View

```
SELECT
    _user.id AS user_id,
    _user.username AS user_username,
    _user.email AS user_email,
    _student.matriculation_number AS student_matriculation_number,
    _student.firstname AS student_firstname,
    _student.lastname AS student_lastname,
    _project.project_id AS project_id,
    _project.project_name_filterable AS project_name_filterable,
    _project.project_full_name AS project_full_name,
    _project.project_short_name AS project_short_name,
    _term.id AS term_id,
    _term.term_name AS term_name,
    _project.project_research_division AS project_research_division,
    _ass.grade AS grade
FROM "gitlab".gitlab_project_user _user
JOIN "gitlab".gitlab_project_user_association _ass ON _ass.user_id = _user.id
JOIN "group".grp_project_mview _project ON _project.project_id = _ass.project_id
LEFT OUTER JOIN "csv".student _student ON _student.id = _user.student_id
JOIN "csv".term _term ON _term.id = _project.project_term_id
```

Listing B.4: `grp_user_project_mview` Materialized View

```
SELECT _cbv.commit_id AS commit_id,
    _cbv.commit_authored_date AS commit_authored_date,
    _cbv.commit_committed_date AS commit_committed_date,
    _cbv.commit_hash AS commit_hash,
    _upv_committer.user_id AS committer_user_id,
    _upv_committer.student_firstname AS committer_student_firstname,
    _upv_committer.student_lastname AS committer_student_lastname,
    _upv_committer.student_matriculation_number AS
        committer_student_matriculation_number,
    _upv_committer.user_username AS committer_user_username,
    _upv_committer.user_email AS committer_user_email,
    _upv_author.user_id AS author_user_id,
```

```
12      _upv_author.student_firstname AS author_student_firstname,
13      _upv_author.student_lastname AS author_student_lastname,
14      _upv_author.student_matriculation_number AS
            author_student_matriculation_number,
15      _upv_author.user_username AS author_user_username,
16      _upv_author.user_email AS author_user_email,
17      _cbv.branch_id AS branch_id,
18      _cbv.branch_name AS branch_name,
19      _project.project_id AS project_id,
20      _project.project_name_filterable AS project_name_filterable,
21      _project.project_full_name AS project_full_name,
22      _project.project_short_name AS project_short_name,
23      _term.id AS term_id,
24      _term.term_name AS term_name,
25      _project.project_research_division AS project_research_division
26  FROM "public".commit_branch_mview _cbv
27  JOIN "group".grp_project_mview _project ON _project.project_id =
        _cbv.commit_project_id
28  LEFT JOIN "group".grp_user_project_mview _upv_committer ON
29      _cbv.commit_committer_id = _upv_committer.user_id AND
30      _project.project_term_id = _upv_committer.term_id AND
31      _upv_committer.project_id = _project.project_id
32  LEFT JOIN "group".grp_user_project_mview _upv_author ON
33      _cbv.commit_author_id = _upv_author.user_id AND
34      _project.project_term_id = _upv_author.term_id AND
35      _upv_author.project_id = _project.project_id
36  JOIN "csv".term _term ON _term.id = _project.project_term_id
```

Listing B.5: `grp_commit_user_mview` Materialized View

```
1   SELECT
2       _cuv.*,
3       _commit_stats.insertions AS commit_insertions,
4       _commit_stats.deletions AS commit_deletions,
5       _commit_stats.files AS commit_files
6   FROM "group".grp_commit_user_mview _cuv
7   JOIN "gitlab".git_commit_stats _commit_stats ON _cuv.commit_id =
        _commit_stats.commit_id
```

Listing B.6: `grp_commit_stats_user_mview` Materialized View

APPENDIX C

Expert Interview Survey

SEPM Analytics Survey

*Required

Demographic

A person's visual perception can be influenced by a variety of factors.
Because this prototype is primarily based on different visualization concepts, gathering additional demographic information may be beneficial for the evaluation.
The data collected will only be used to evaluate the visualization prototype.
It is optional to respond to these questions.

1. What is your age?

2. What is your gender?

Mark only one oval.

- ◯ Female
- ◯ Male
- ◯ Prefer not to say
- ◯ Other: _____

General Questions

These question should identify the stakeholders in the area of student group projects.

3. How long have you been working in software engineering education? *

Mark only one oval.

- ◯ < 1 year
- ◯ 1-2 years
- ◯ 2-5 years
- ◯ 5-10 years
- ◯ 10-15 years
- ◯ 15-20 years
- ◯ > 20 years

4. What is your current role in this course? *

Tick all that apply.

- ☐ Lecturer
- ☐ Assistant
- ☐ Admin
- ☐ Tutor
- ☐ Other: _____

5. As an administrator: How many groups are you administrating during a semester? Please skip this question if you are not an administrator.

Mark only one oval.

- ◯ 0
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8
- ◯ 9
- ◯ 10
- ◯ > 10
- ◯ All groups

6. As a tutor: How many groups are you supervising during a semester? Please skip this question if you are not a tutor.

Mark only one oval.

- ◯ 0
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8
- ◯ 9
- ◯ 10
- ◯ > 10
- ◯ All groups

7. As an assistant: How many groups are you supervising during a semester?
Please skip this question if you are not a lecturer/assistant.

Mark only one oval.

- ◯ 0
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8
- ◯ 9
- ◯ 10
- ◯ > 10
- ◯ All groups

8. As a lecturer: How many groups are you supervising during a semester? Please skip this question if you are not a lecturer/assistant.

Mark only one oval.

- ◯ 0
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8
- ◯ 9
- ◯ 10
- ◯ > 10
- ◯ All groups

Grading and Comparing

9. Which of the following information is considered for grading student groups? *

Tick all that apply.

- ☐ Code Quality (e.g. Test Coverage, Successful/Failed Tests, etc)
- ☐ Quality of implemented features
- ☐ Working Hours (cumulated)
- ☐ Lines of Code (LoC)
- ☐ Git-Contribution (Number of Commits)
- ☐ Role conformance
- ☐ Theoretical knowledge
- ☐ Other: _____

10. Which information is considered for grading students *individually*? *

_____

11. Do you usually compare groups against each other? *

Mark only one oval.

- ◯ Yes
- ◯ No

12. If you answered the previous question with **no**: Why did you not compare/rank groups against each other in the past?

_____

_____

_____

_____

_____

Group Comparing & Ranking

The following questions are not relevant for grading; comparing and ranking groups is intended to serve as information basis to derive the current status of groups.

13. How many groups would you like to/need to be able to display, rank, and/or compare? *

_____

14. Which categories would you need to rank groups regarding their project state & progress? *

Tick all that apply.

- ☐ Code Quality (e.g. Test Coverage, Successful/Failed Tests, etc)
- ☐ Quality of implemented features
- ☐ Working Hours (cumulated)
- ☐ Lines of Code (LoC)
- ☐ Git-Contribution (Number of Commits)
- ☐ Role conformance
- ☐ Theoretical knowledge
- ☐ Other: _____

Current situation

15. How do you currently collect information about the spent time per student? *

16. How do you currently collect information about the workload distribution? *

17. How do you currently collect information about the Git contribution? *

Hypotheses

18. A software repository visualization architecture for software engineering education should not require initial configuration *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

19. A software repository visualization architecture for software engineering education should be configurable during operation *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

20. It is sufficient for grading to consider git-related data from main/master and dev branch only *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

21. The visualization artifacts are portable and cross-platform *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

22. The administration of the architecture should be possible within the existing GitLab infrastructure. *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

23. Visualization artifacts should be viewable offline (i.e. without being hosted on a server) *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

24. It is important to **select multiple semester** to compare them against each other *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

25. It is important to **only** show the current semester *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

26. It is important to quickly get an overview of **all groups'** project state of the current semester *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

27. Assuming you are responsible for multiple groups in the current semester, it is important to quickly get an overview of **all of your** group's project state *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

28. Assuming you want to *compare multiple groups*, it is important to *filter by research divisions* *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

29. Assuming you want to *compare multiple groups*, it is important to *filter by groups* (only display a subset of all) *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

30. It is important to customize the time granularity (i.e. display time series data per day or week) *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

31. It is important to customize the timeframe, e.g. zooming into the last X weeks while hiding the rest *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

32. It is important to *filter by specific students* *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

Available Views

In the following, all visualizations which are shown are already part of an implemented data visualization dashboard.

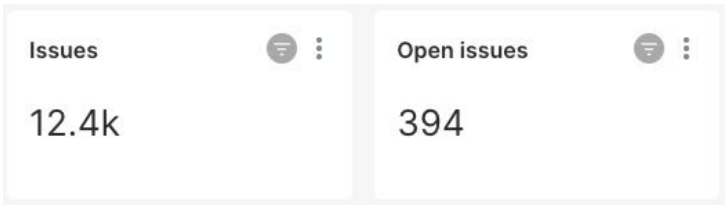
General visualizations

These visualizations show data, which is basically general information about the available data

Please note that, for all shown graphs, it is possible to filter by term, group, research division and students. Due to the amount of different filter variations, unless stated differently, no filter is applied.
If one is applied, it can be seen in the top right corner's filter icon indicating the number of applied filter (if it is gray, nothing is applied)

33. Consider the following tiles, how useful is the presented information for you? *

Based on your currently selected filter, that tile shows the number of all issues and open ones which match your filter(s)



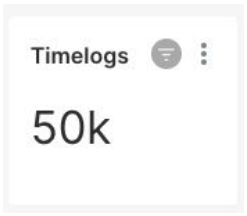
Mark only one oval.

	1	2	3	4	5	
Not useful at at all	◯	◯	◯	◯	◯	Extremely useful

34. Consider the following tile, how useful is the presented information for you? *

A timelog is one entry in GitLab created by the */spent* command.

Based on your currently selected filter, that tile shows the number of timelogs which match your filter(s)



Mark only one oval.

	1	2	3	4	5	
Not useful at at all	◯	◯	◯	◯	◯	Extremely useful

35. Consider the following tile, how useful is the presented information for you? *

A timelog is one entry in GitLab created by the */spent* command.

Based on your currently selected filter, that tile shows the trace of timelogs per week which match your filter(s)

Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

36. Consider the following tiles, how useful is the presented information for you? *

Based on your currently selected filter, that tile shows the number of students which match your filter(s)

Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

37. Consider the following tiles, how useful is the presented information for you? *

Hint: A "Commit orphan" is a Commit where neither the Committer nor the Author could be assigned to any student in that group and semester

Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

38. Based on the previous shown tiles, which information would you like to see visualized in such a way? *

Repository-related visualizations

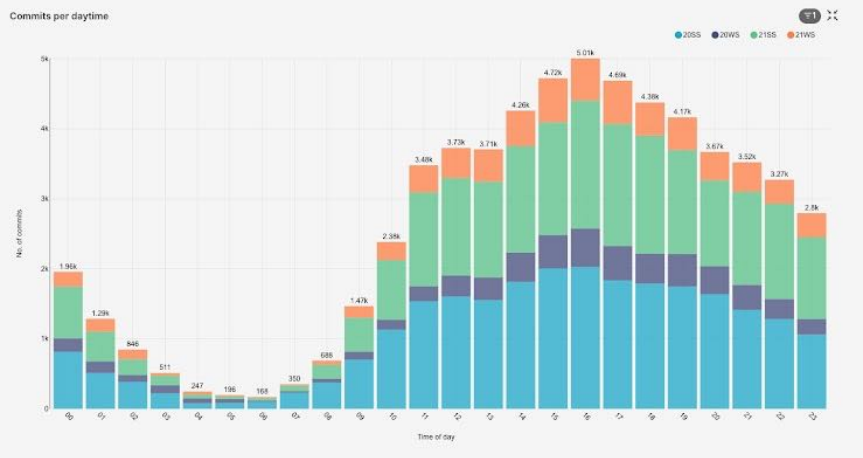
The following visualization are about the available data of the repositories which can be extracted out of the git repository

39. Consider the following chart, how useful is the presented information for you? *

This chart shows the distribution of commits (y-axis) per daytime (x-axis) in 24h format.

Please note that this chart shows all semester.

Please note that, as a user, you can filter to only display relevant groups.



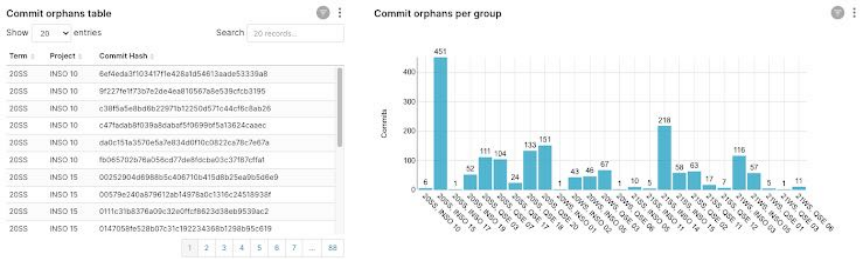
Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

40. Consider the following table and chart, how useful is the presented information for you? *

The table on the left shows the semester (term), project and commit hash of a Commit orphan and the bar chart on the right side shows the amount of orphans per group.

Recall, a "Commit orphan" is a Commit where neither the Committer nor the Author could be assigned to any student in that group and semester

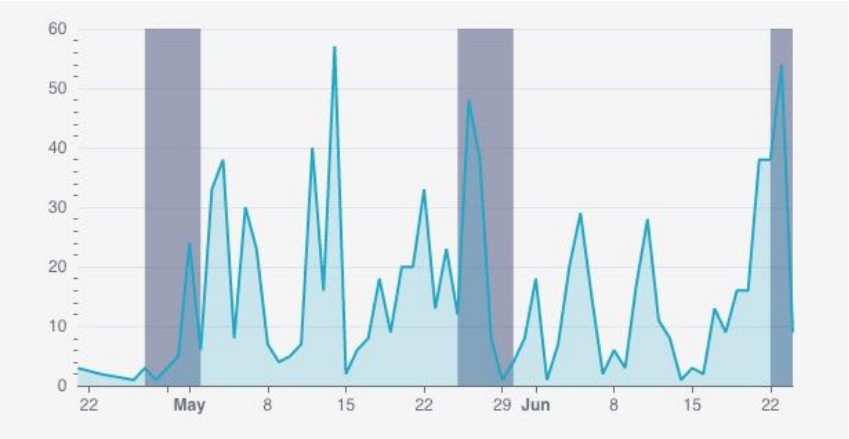


Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

41. *Consider the following chart, how useful is the presented information for you?* *

This chart shows the timeline on the x-axis and the *sum* of Commits per day on the y-axis of **one group**.
The vertical bars indicate the weeks when a Management Review was planned.

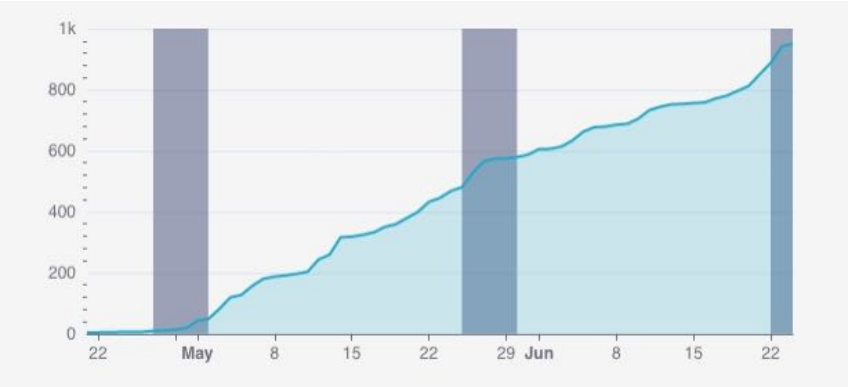


Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

42. *Consider the following chart, how useful is the presented information for you?* *

This chart shows the timeline on the x-axis and the *cumulative sum* of Commits per day on the y-axis of **one group**.
The vertical bars indicate the weeks when a Management Review was planned.



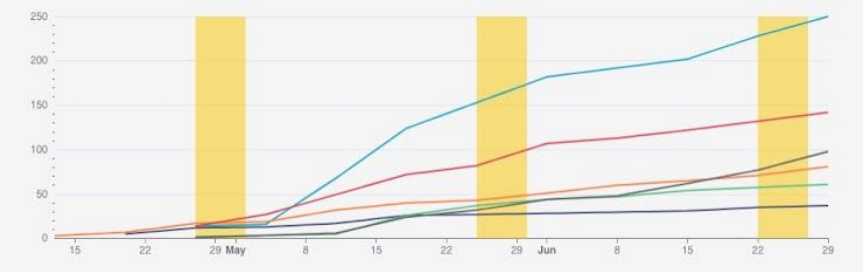
Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

43. *Consider the following chart, how useful is the presented information for you?* *

This chart shows the timeline on the x-axis and the *cumulative sum* of Commits per week per student of *one group* on the y-axis.
Students are distinguished by the color.

The vertical bars indicate the weeks when a Management Review was planned.



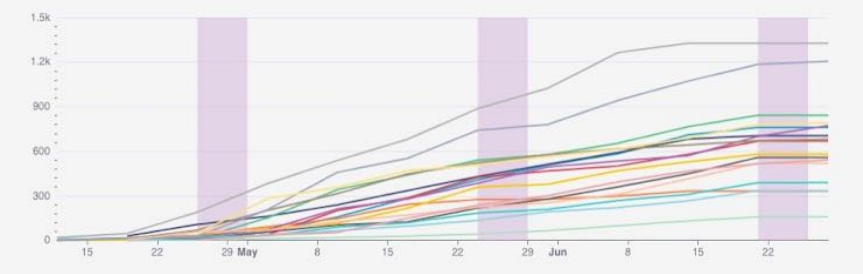
Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

44. *Consider the following chart, how useful is the presented information for you?* *

This chart shows the timeline on the x-axis and the *cumulative sum* of Commits per week *per group* on the y-axis.
Groups are distinguished by the color.
The vertical bars represent the week when a Management Review was planned.

Please note that this chart *shows all groups of a semester*. As a user, you can filter to only display relevant groups.

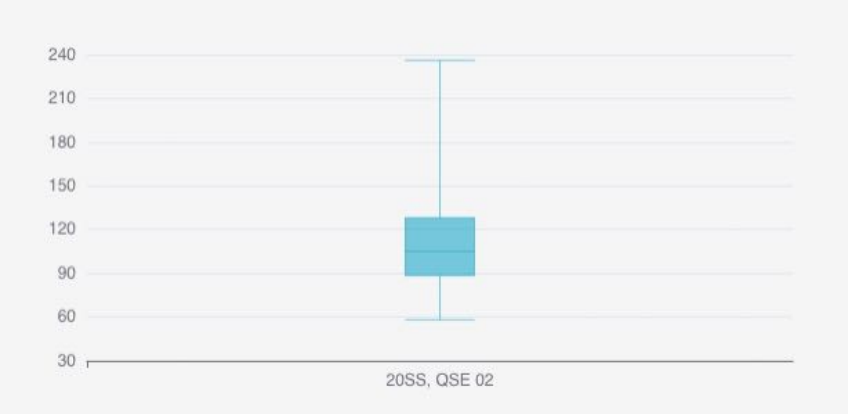


Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

45. *Consider the following chart, how useful is the presented information for you?* *

This box plot visualizes the Commits per student.



Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

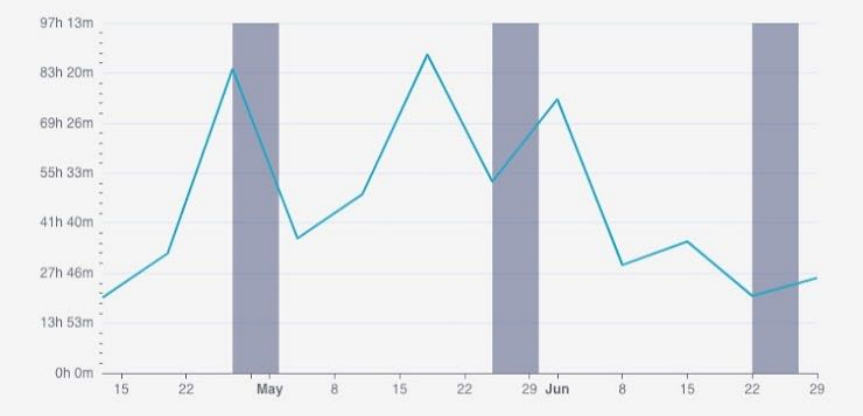
Project Management related Visualizations

The following visualizations are about project management related data like spent time per issue, etc. and is typically directly stored in GitLab.

46. *Consider the following chart, how useful is the presented information for you?* *

The chart shows the timeline on the x-axis and the *sum (in hours)* of **one** group's spent time on a *weekly* basis.

The vertical bars indicate the weeks when a Management Review was planned.



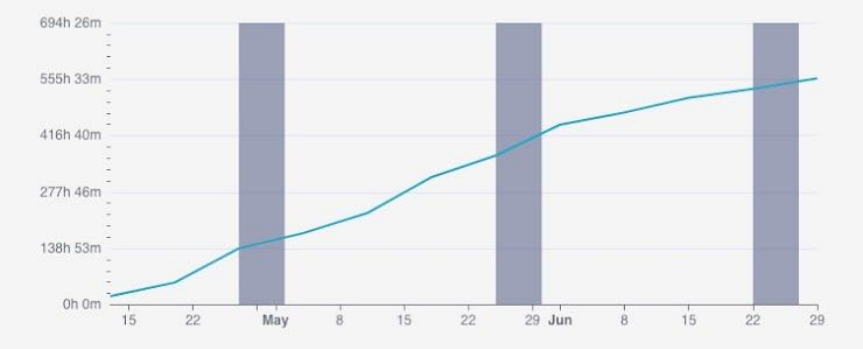
Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

47. *Consider the following chart, how useful is the presented information for you?* *

The chart shows the timeline on the x-axis and the *cumulative sum (in hours)* of **one** group's spent time on a *weekly* basis.

The vertical bars indicate the weeks when a Management Review was planned.



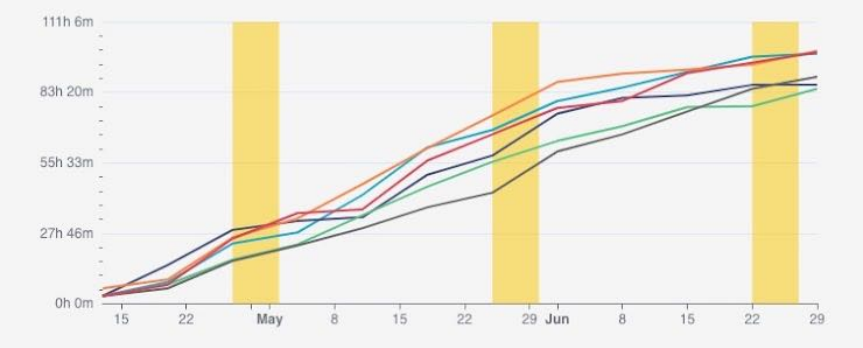
Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

48. *Consider the following table and chart, how useful is the presented information for you?* *

The chart shows the timeline on the x-axis and the *cumulative sum (in hours)* of **each student**'s spent time on a *weekly* basis.

The vertical bars indicate the weeks when a Management Review was planned.



Mark only one oval.

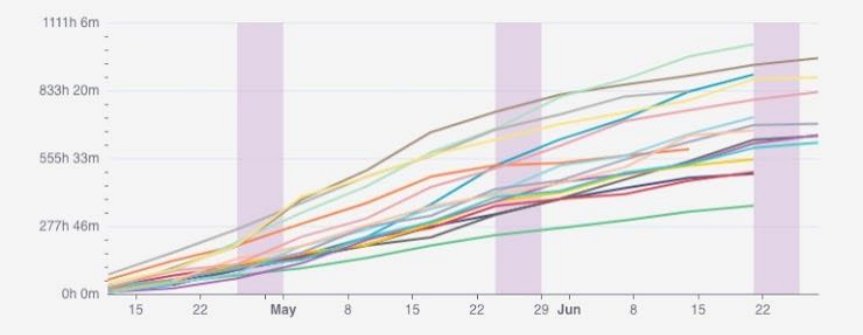
	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

49. *Consider the following table and chart, how useful is the presented information for you?* *

The chart shows the timeline on the x-axis and the *cumulative sum (in hours)* of **all groups**' spent time on a *weekly* basis.

The vertical bars indicate the weeks when a Management Review was planned.

Please note that, as a user, you can filter to only display relevant groups.



Mark only one oval.

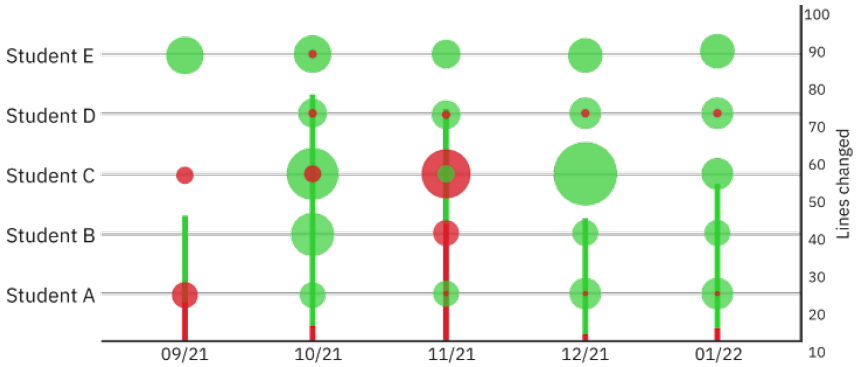
	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Extremely useful

Future views

In the following, images are shown visualizing certain information which can be extracted from students repositories (and its Code respectively), but are not yet available.

50. Consider the image below, how important is a visualization of the correlation of student's commits and its changes (in % line changed) per week? *

The image shows on the left-hand side the name of the students, on the right-hand side there's a scale of lines changed in percent.
The bottom x-axis shows the week. The stacked bars from the bottom represent the percentage of lines changed, meaning red are deleted lines and green are added lines. Furthermore, each circle for each student represents the percentage of contribution, the bigger the cycle the bigger the changes (green = added, red = deleted).



Mark only one oval.

	1	2	3	4	5	
Not important at all	◯	◯	◯	◯	◯	Very important

51. Which information is missing in this frame below, you would need to have an overview about the current project state? *

The shown visualization combines on a per-group basis multiple source into different charts. The coloring is chosen to be consistent per student.

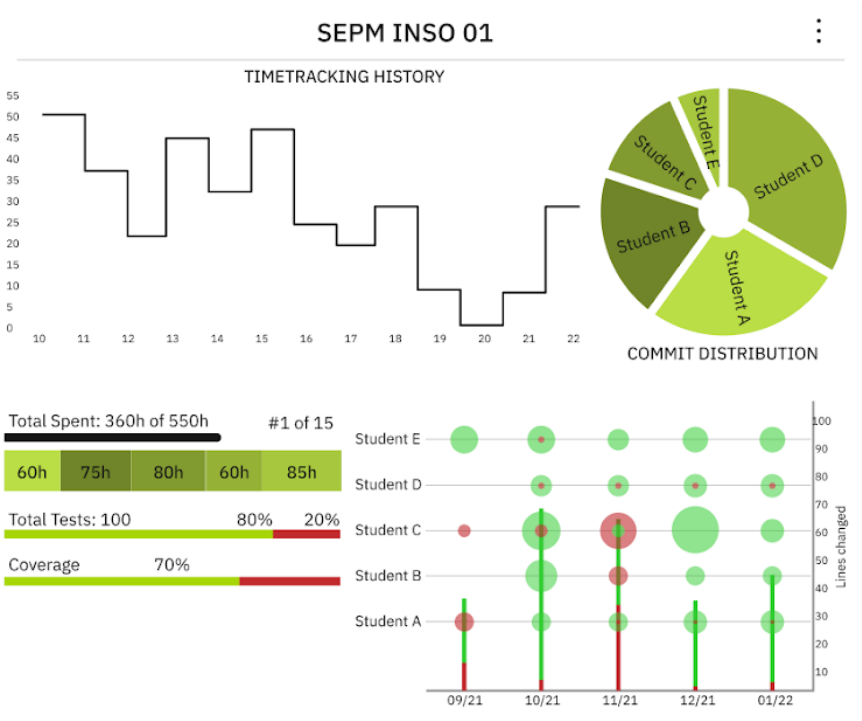
The top left chart shows the overall trace of the spent time on a weekly basis in hours.

The top right chart displays the distribution of Commits per student as a pie chart. The bigger the slice, the more Commits were made by the student.

At the bottom left, there are multiple sources combined:

- The first line shows the sum of spent hours as a bar chart in relation to the planned 550 hours (110h*5 students) and a ranking compared to all other groups (this group is currently at the first place of 15 groups in this semester)
- The second row shows the absolute amount of time spent per student, where each color matches the top left chart's colors.
- The third row indicates the total number of Tests and their current success-failure-ratio
- The fourth row displays the current Test-Coverage based on the available Tests

The bottom right chart is the integration of the student's Commit correlation of the previous question.



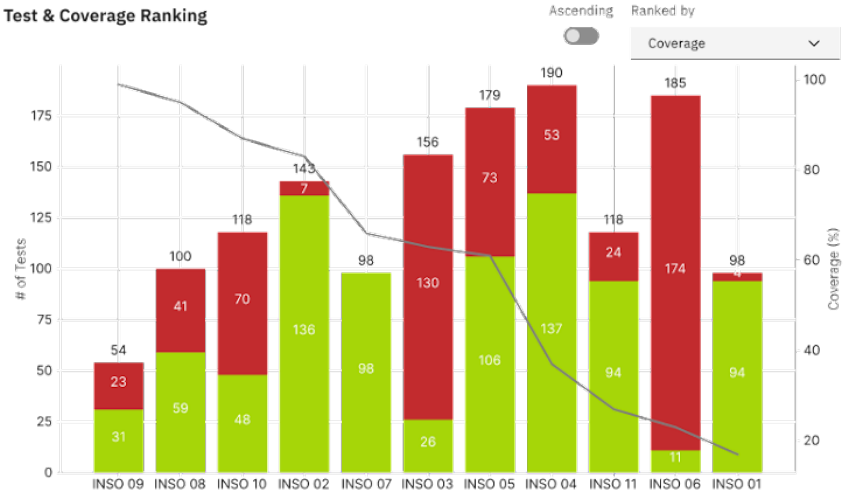
52. Consider the image below, comparing groups by their number of tests and coverage. *

How important are the listed ranking categories for you?

The following chart shows exemplarily a ranking of groups based on their coverage. By the dropdown at the top right, *Tests (Green)*, *Tests (Red)*, *Coverage*, *Tests (sum)* and *Group* can be selected

Legend

- The x-axis defines the group
- The left y-axis (bar chart) represents the number of tests, where the green part represents the successful tests and the red part the failed tests. Additionally, the respective number and sum of tests is shown
- The right y-axis (line chart) represents the current Coverage of the available tests in percent.



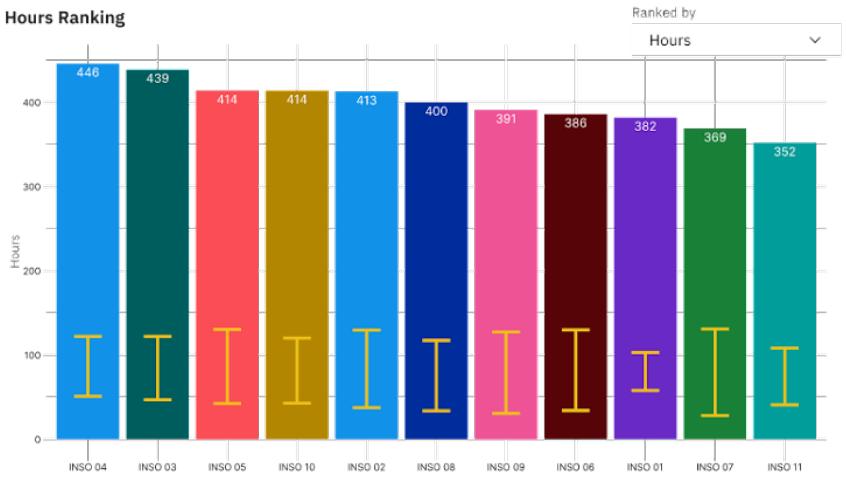
Mark only one oval per row.

	1 (Not important at all)	2	3	4	5 (Very important)
Tests (Green)	◯	◯	◯	◯	◯
Tests (Red)	◯	◯	◯	◯	◯
Coverage	◯	◯	◯	◯	◯
Tests (sum)	◯	◯	◯	◯	◯
Group	◯	◯	◯	◯	◯

53. Consider the image below, ranking groups by their spent hours. *

How important is it for you that the sum of spent hours is displayed in combination with the standard deviation of the student?

The yellow lines inside the bars show the standard deviation of the student's book hours. The narrower min and max value are, the shorter the vertical line.
The standard deviation is calculated per group.

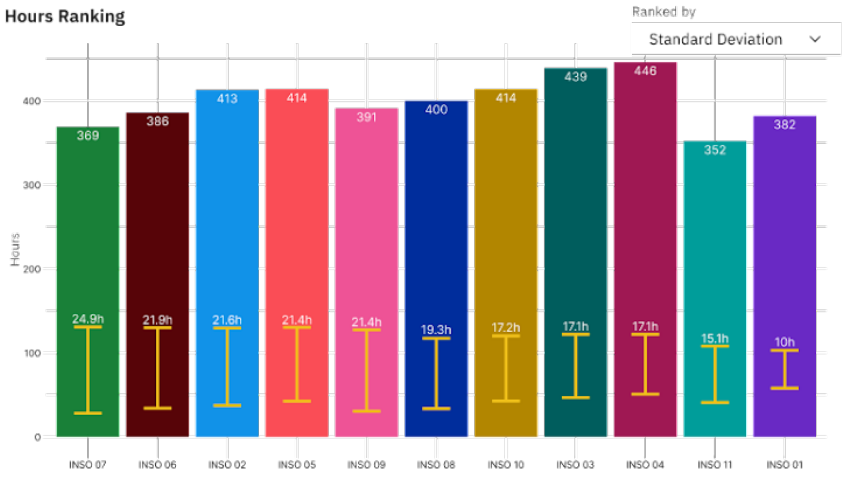


Mark only one oval.

	1	2	3	4	5	
Not important at all	◯	◯	◯	◯	◯	Very important

54. Consider the image below, how important is it for you to rank groups by the standard deviation of the student's spent hours? *

The standard deviation is calculated per group.

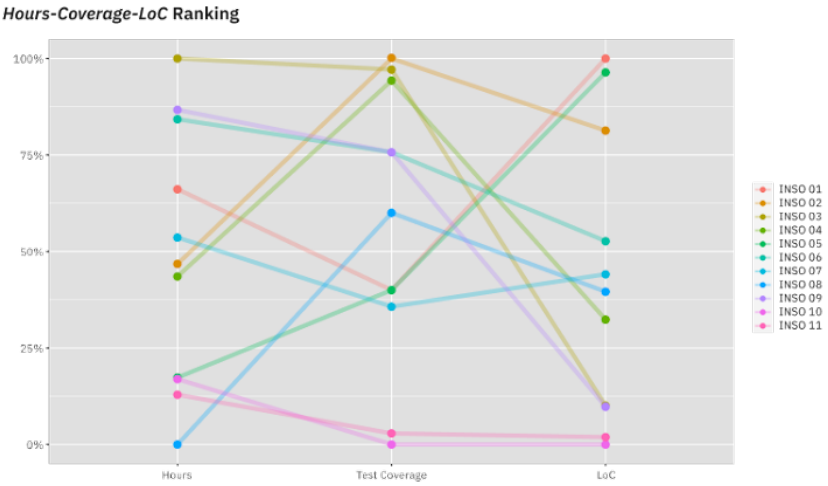


Mark only one oval.

	1	2	3	4	5	
Not important at all	◯	◯	◯	◯	◯	Very important

55. Consider the image below, how important is it for you to rank groups by hours, test coverage and LoC in a correlating way? *

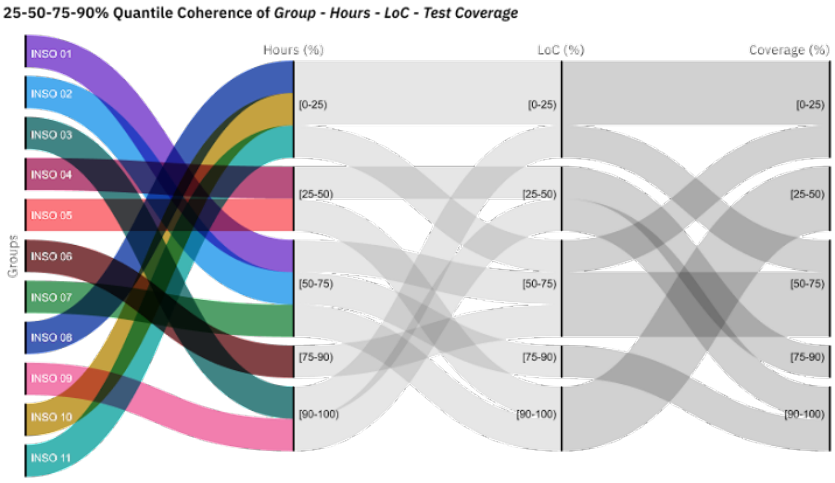
100% is corresponding to the maximum, 0% to the minimum value of the groups



Mark only one oval.

	1	2	3	4	5	
Not important at all	◯	◯	◯	◯	◯	Very important

56. Consider the image below showing all groups and their corresponding quantiles (25-50-75-90%) of Hours, LoC and Coverage. How useful is this diagram for you? *

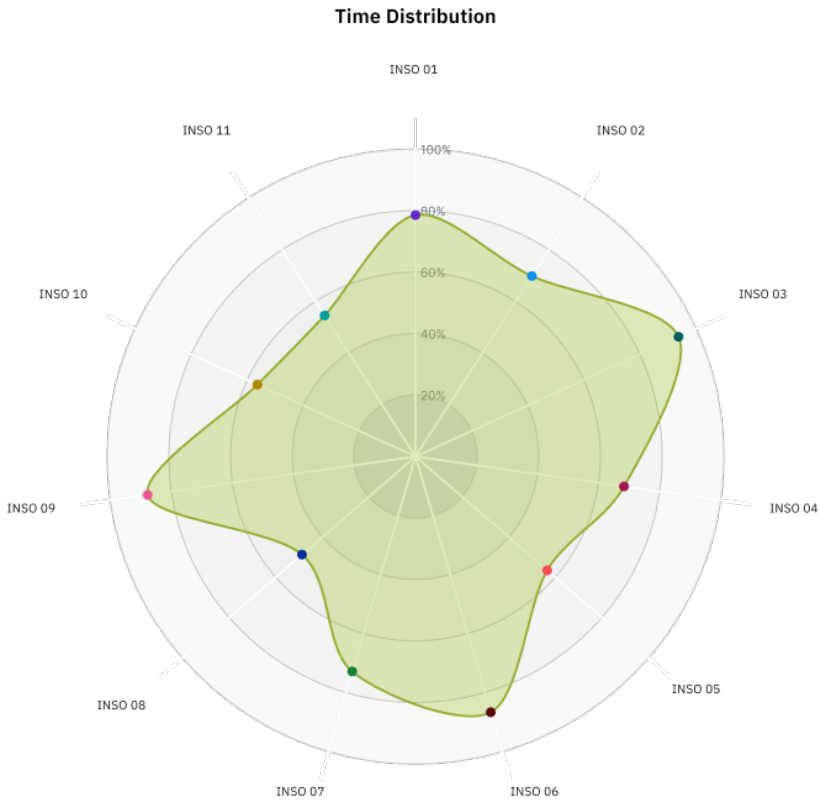


Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Very useful

57. Consider the image below, how useful do you find this radar chart? *

100% corresponds to the max. intended hours per group (110h per student)

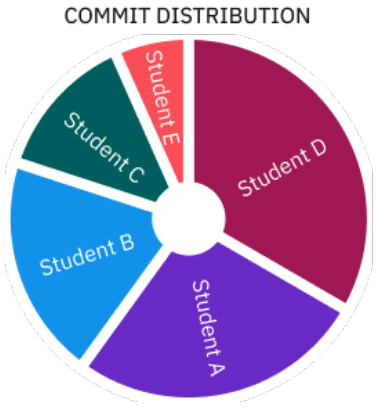


Mark only one oval.

	1	2	3	4	5	
Not useful at all	◯	◯	◯	◯	◯	Very useful

58. Consider the commit distribution pie-chart below, how important is that information for you? *

The bigger the slice, the more commits are assigned to one student.

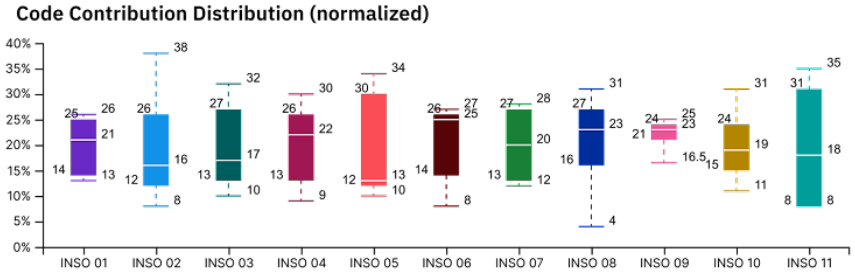


Mark only one oval.

	1	2	3	4	5	
Not important at all	◯	◯	◯	◯	◯	Very important

59. Consider the image below, showing the distribution of code ownership (in percent) displayed as a box plot. How important is that information for you? *

Hint: "normalized" means that the number of commits is relative to the sum of Commits within a group



Mark only one oval.

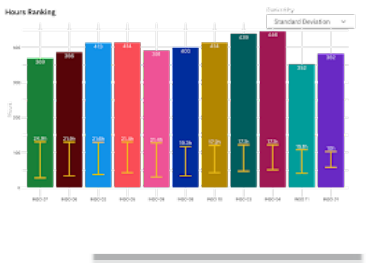
	1	2	3	4	5	
Not important at all	◯	◯	◯	◯	◯	Very important

Similar Views Comparison

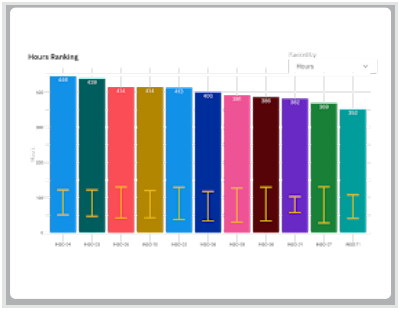
In the following, similar visualizations of the same underlying data are presented. Please select the one(s) you would prefer the most.

60. Regarding time tracking, please select the top 3 most informative graphs.

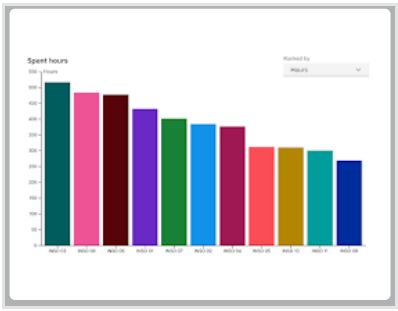
Tick all that apply.



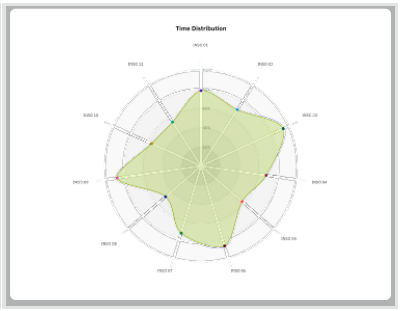
☐ Ranked by Standard Deviation



☐ Standard deviation shown, ranked by hours



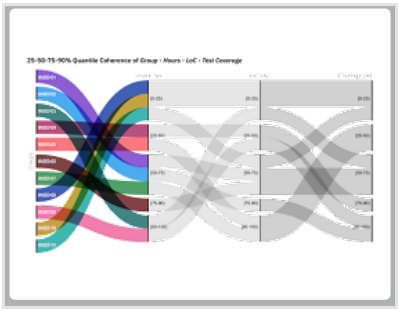
☐ Ranked by hours



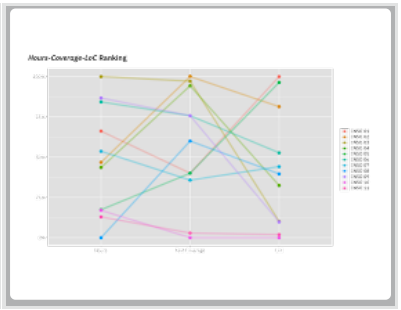
☐ Radar chart

61. Please select the more informative graph.

Mark only one oval.



◯ Group-Hours-LoC-Coverage_Quantiles

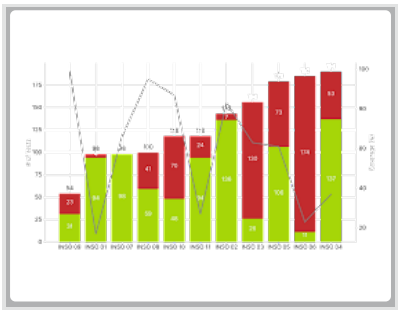


◯ Hours-Coverace-LoC Ranking

62. Which graph is more informative for you? *

Mark only one oval.

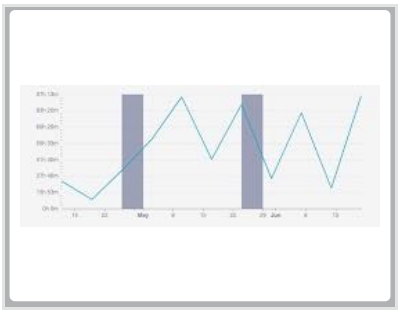
◯ Ranking with test quartiles shown at the top



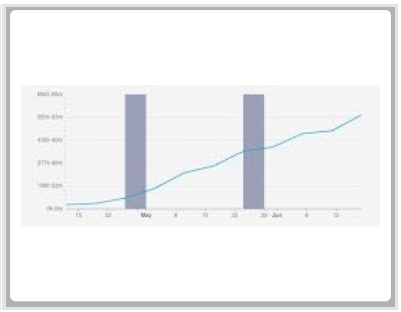
◯ Ranking without test quartiles

63. Consider the following two charts, as already shown before, which one do you prefer? *

Mark only one oval.



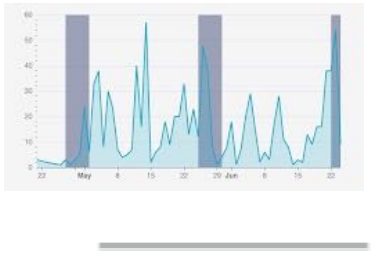
◯ Sum of spent time per week

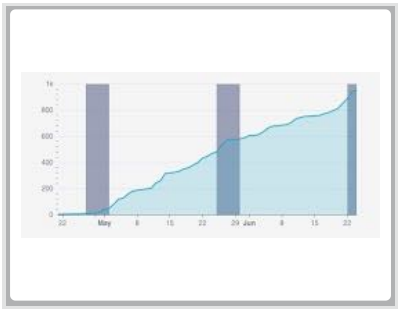


◯ Cumulative of spent time per week

64. Consider the following two charts, as already shown before, which one do you prefer? *

Mark only one oval.





◯ Sum of Commits per day

◯ Cumulative sum of Commits per day

Fullscreen Views

65. Considering the image below, showing an overview of all groups state. Which information is missing for you to draw any conclusion?

66. Considering the image below, showing a compare view of all groups. Which information is missing for you to draw any conclusion?

67. Considering the image below, showing a ranking view of all groups. Which information is missing for you to draw any conclusion?

APPENDIX D

Expert Evaluation Survey

SEPM Analytics Evaluation

*Required

Demographic

A person's visual perception can be influenced by a variety of factors.
Because this prototype is primarily based on different visualization concepts, gathering additional demographic information may be beneficial for the evaluation.
The data collected will only be used to evaluate the visualization prototype.
It is optional to respond to these questions.

1. What is your age?

_____

2. What is your gender?

Mark only one oval.

- ◯ Female
- ◯ Male
- ◯ Prefer not to say
- ◯ Other: _____

Hypotheses Evaluation

3. The prototype fulfills the requirement to be used out-of-the box *

Hypothesis: A software repository visualization architecture for software engineering education should not require initial configuration

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

4. The configuration possibilities are sufficient *

Hypothesis: A software repository visualization architecture for software engineering education should be configurable during operation

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

5. Filtering by a specific branch works as you would expect *

Hypothesis: It is sufficient for grading to consider git-related data from main/master and dev branch only

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

6. The portability and cross-platform capabilities work as you would expect *

Hypothesis: The visualization artifacts are portable and cross-platform

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

7. The architecture can be used within the existing infrastructure. *

Hypothesis: The administration of the architecture should be possible within the existing GitLab infrastructure.

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

8. Viewing the artifacts offline works as you would expect *

Hypothesis: Visualization artifacts should be viewable offline (i.e. without being hosted on a server)

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

9. Comparing multiple semesters against each other works as you would expect *

*Hypothesis: It is important to **select multiple semester** to compare them against each other*

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

10. Filtering the projects to **only** show the current semester works as you would expect *

*Hypothesis: It is important to **only** show the current semester*

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

11. The provided overview of **all of your** group's project state provides all information you need *

*Hypothesis: Assuming you are responsible for multiple groups in the current semester, it is important to quickly get an overview of **all of your** group's project state*

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

12. Filtering *by research divisions* works as you would expect *

Hypothesis: Assuming you want to compare multiple groups, it is important to filter by research divisions

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

13. Filtering *by groups* (only display a subset of all) works as you would expect *

Hypothesis: Assuming you want to compare multiple groups, it is important to filter by groups (only display a subset of all)

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

14. Customizing the time granularity (i.e. display time series data per day or week) works as you would expect *

*Hypothesis: It is important to customize the time granularity (i.e. display time series data per * day or week)*

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

15. Zooming into the last X weeks while hiding the rest works as you would expect *

Hypothesis: It is important to customize the timeframe, e.g. zooming into the last X weeks while hiding the rest

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

16. Filtering by *specific students* works as you would expect *

Hypothesis: It is important to filter by specific students

Mark only one oval.

	1	2	3	4	5	
Very dissatisfied	◯	◯	◯	◯	◯	Very satisfied

17. Anything else I would like to add ...

System Usability Scale (SUS)

18. *

Mark only one oval per row.

	1 (Strongly disagree)	2	3	4	5 (Strongly agree)
I think that I would like to use this system frequently.					
I found the system unnecessarily complex.					
I thought the system was easy to use.					
I think that I would need the support of a technical person to be able to use this system.					
I found the various functions in this system were well integrated.					
I thought there was too much inconsistency in this system.					
I would imagine that most people would learn to use this system very quickly.					
I found the system very cumbersome to use.					
I felt very confident using the system.					
I needed to learn a lot of things					

before I could get going with this system.