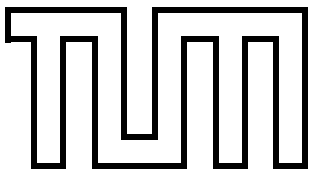


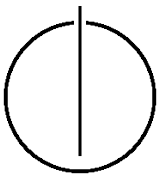
FAKULTÄT FÜR INFORMATIK

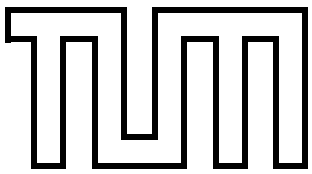
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatik

Teaching Analytics in Artemis

Dominik Fuchs





FAKULTÄT FÜR INFORMATIK

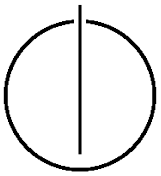
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatik

Teaching Analytics in Artemis

Lehranalytik in Artemis

Author:	Dominik Fuchs
Supervisor:	Prof. Dr. Bernd Brügge
Advisor:	Dr. Stephan Krusche
Date:	15. April 2021



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 15. April 2021 Dominik Fuchs

Abstract

Learning Management Systems become more and more relevant in modern teaching. One disadvantage of these systems is the lack of direct student feedback, which is normally created by the interaction between the student and the instructor during the lecture. Teaching analytics is a research field which thematizes this issue and focuses on enhancing the teaching quality by collecting, analyzing and presenting student performances in the course.

Artemis in its current state does not provide course analytics for instructors. Instructors are unable to get a deeper insight into the intermediate state of the course, which prevents them from dynamically adapting the lecture and exercises according to the students' weaknesses and strengths. Through this thesis, we want to address this problem and expand Artemis in a way that allows instructors to gather more information about the status of their students. We integrate new analytical functionalities on 3 levels:

First, we refactor an existing course management page in order to provide instructors with early access to the most relevant course analytics. Second, we present a course detail page which extends the information shown in the aforementioned page with additional data. Third, we introduce several new statistics pages to Artemis, which provide administrators with server-wide analytics and instructors with statistics regarding the overall course, exercises, exams and lectures, that they can use to optimize the course quality.

Zusammenfassung

Lernmanagement-Systeme gewinnen in der modernen Lehre immer mehr an Bedeutung. Ein Nachteil dieser Systeme ist das Fehlen von direktem Studentenfeedback, das normalerweise durch die Interaktion zwischen dem Studenten und dem Dozenten während der Vorlesung entsteht. Lehranalytik ist ein Forschungsfeld, das diese Problematik thematisiert und sich darauf konzentriert, die Qualität der Lehre durch das Sammeln, Analysieren und Darstellen von studentischen Leistungen in der Lehrveranstaltung zu verbessern.

Artemis bietet in seinem aktuellen Zustand keine Kursanalyse für Dozenten. Dozenten sind nicht in der Lage, einen tieferen Einblick in den Zwischenstand des Kurses zu bekommen, was sie daran hindert, die Vorlesung und die Übungen dynamisch an die Schwächen und Stärken der Studenten anzupassen. Mit dieser Arbeit wollen wir dieses Problem adressieren und Artemis so erweitern, dass Dozenten mehr Informationen über den Status ihrer Studenten erhalten können. Wir integrieren neue analytische Funktionalitäten auf 3 Ebenen:

Erstens überarbeiten wir eine bestehende Kursverwaltungsseite, um Dozenten einen frühen Zugriff auf die relevantesten Kursanalysen zu ermöglichen. Zweitens präsentieren wir eine Kursdetailseite, die die auf der oben genannten Seite angezeigten Informationen um zusätzliche Daten erweitert. Drittens führen wir mehrere neue Statistikseiten in Artemis ein, die Administratoren mit serverweiten Analysen und Dozenten mit Statistiken zum gesamten Kurs, zu Übungen, Prüfungen und Vorlesungen versorgen, die sie zur Optimierung der Kursqualität nutzen können.

Contents

1 Introduction **2**
- 1.1 Problem . . . 3
- 1.2 Motivation . . . 4
- 1.3 Objectives . . . 5
- 1.4 Outline . . . 6

2 Related Work **7**
- 2.1 edX Insights . . . 7
- 2.2 Canvas . . . 12

3 Requirements Analysis **16**
- 3.1 Overview . . . 16
- 3.2 Current System . . . 16
- 3.3 Proposed System . . . 19
  - 3.3.1 Functional Requirements . . . 20
  - 3.3.2 Nonfunctional Requirements . . . 22
- 3.4 System Models . . . 23
  - 3.4.1 Scenarios . . . 24
  - 3.4.2 Use Case Model . . . 25
  - 3.4.3 Analysis Object Model . . . 28
  - 3.4.4 User Interface . . . 30

4 System Design **38**
- 4.1 Overview . . . 38
- 4.2 Design Goals . . . 39
- 4.3 Subsystem Decomposition . . . 40

5 Object Design **43**
- 5.1 Statistics . . . 43

6 Summary **46**
6.1 Status 46
6.1.1 Realized Goals 46
6.1.2 Open Goals 49
6.2 Conclusion 49
6.3 Future Work 49

TA Teaching Analytics

LA Learning Analytics

LMS Learning Managenemt System

TUM Technical University Munich

GUI Graphical User Interface

API Application Programming Interface

CI Continuous Integration

VC Version Control

CSS Cascading Style Sheets

SCSS Sassy CSS

HTML Hypertext Markup Language

REST Representational State Transfer

UML Unified Modeling Language

ML Machine Learning

Chapter 1

Introduction

Teaching analytics (TA) is a term with increasing relevance in modern classes. To understand the meaning and origin of TA, we need to give two definitions. First, we define the broader term learning analytics (LA), where TA is derived from:

> *"Learning analytics is the measurement, collection, analysis, and reporting of data about learners and their contexts, for understanding and optimizing learning and the environments in which it occurs"* [Sie13].

Second, we describe teaching analytics:

> *"Teaching analytics is the application of learning analytics techniques to understand teaching and learning processes, and eventually enable supportive interventions"* [PSDJ16].

TA as a subcategory of LA specializes on enhancing the teaching quality by analyzing student performances in the course.

Prior to this trend, face-to-face communication between students and teachers made it difficult to generate meaningful data, as there is no digital footprint and therefore only the fraction of interactions that took place online can be taken into account by analytics tools. The reason for the rising demand for TA functionalities is the establishment of Learning Management Systems (LMS) into higher education [Sie13, KTKK12]. Artemis is one of a few LMS which the Technical University Munich (TUM) uses to provide online teaching to students and has therefore aspects which can be used to apply teaching analytics [KS18].

In Artemis, instructors are able to create courses, which contain lectures, exercises and exams. For lectures, instructors can either upload files for the students to download or directly create text or link videos for the students

to view. Regarding exercises and exams, instructors have the possibility to create problems which the student needs to solve in a defined amount of time. Artemis offers text-, modeling, file-upload and programming exercises on which the student can work directly in the application. Artemis also manages the subsequent assessment of student submissions by tutors.

Altogether, instructors are able to apply an interactive learning approach that also generates a large amount of data and is well suited for analytical purposes [KvFA17].

1.1 Problem

When a lecturer interacts with 20 to 30 students in the lecture hall and corrects the assignments himself, he gets to see very quickly how well the students understand the content. In the case of the computer science course "Introduction to Software Engineering" at the Technical University of Munich in the summer semester of 2021, a lecturer supervises more than 2000 students online. This poses new challenges, as the docent has to delegate responsibilities to tutors, no longer has direct student contact and therefore has difficulty overlooking students and their current status in the course.

Figure 1.1 displays an example: Students receive an average score of approximately 90 percent in exercise 1 and exercise 2, however, obtain an average score of 30 percent in another exercise 3. Due to the distribution of assessment to the tutors, instructors do not have insights into these results and therefore will not detect this issue. What instructors need are analytics regarding the students' performances.

Artemis in its current version does not offer such opportunities, neither for courses nor exercises. We will elaborate more on what the instructor can see at the moment in Section 3.2. The lack of analytical data prevents instructors from dynamically adapting the lecture and exercises according to the students' weaknesses and strengths. As a result, students are less likely to compensate their weaknesses and knowledge gaps emerge.

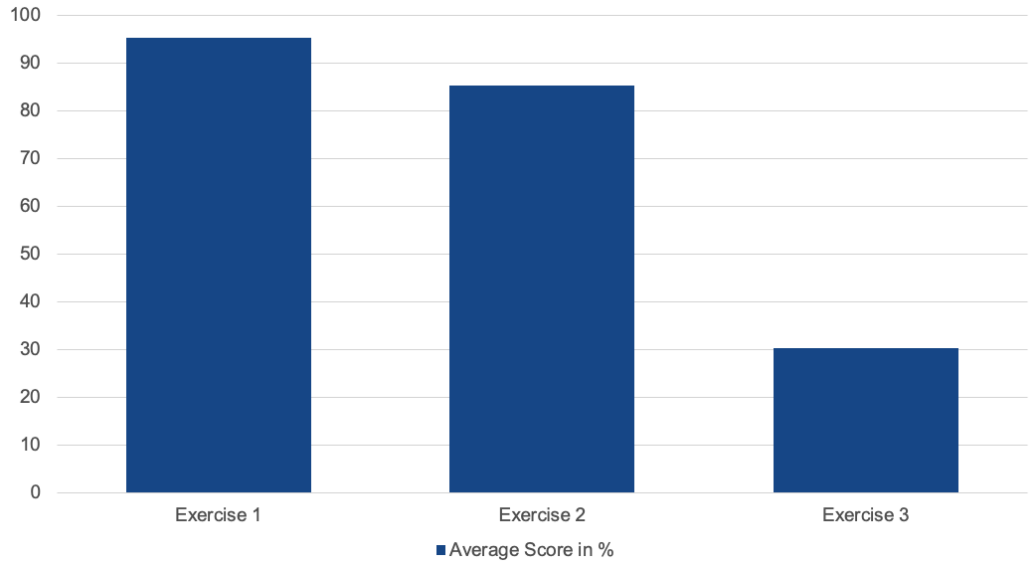


Figure 1.1: Scenario where students achieve a bad average score in one of three exercises

1.2 Motivation

Integrating learning analytics tools has proven to be an effective way to enhance student learning [OC12,Won17]. By introducing Artemis to teaching analytics approaches, instructors can also benefit by gaining detailed insight into students learning process that can be used to adapt the lecture and improve course quality [SL11].

Artemis is well suited for such an approach as it provides a large database with around 2000 carried out exercises, approximately 100 courses and nearly 15 000 users with about 2 million submissions in total. In the following, we explain how we want to approach the problems described in Section 1.1.

Improve and simplify course overview
By collecting, analyzing and visualizing relevant, course related information, we improve the transparency of the course for tutors and instructors while easing and optimizing the course management.

Provide independent, automated feedback for instructors
Instructors should have the possibility to receive course feedback at any time. As feedback can often be subjective, we want to use student results as a credible base of the analysis and provide instructors with result analytics which are independent of direct student feedback.

Improve student results
Instructors should be able to access in-depth analytics concerning different

aspects of the course. This helps to identify weaknesses of students that are, for instance, indicated by a low average score in a particular exercise. The instructors can reduce deficiencies by further practice in this particular field. The improved student comprehension lowers the number of students at-risk, reduces drop-out rates and improves student grades.

1.3 Objectives

The goal of this thesis is to introduce teaching analytics practices, which are helpful for instructors to get a deeper insight into the students results. Analytics are best presented by visualizations. We want to display statistics in different levels of analytical intensity so the instructor is able to choose how detailed the analysis should be. We derive 3 main objectives for our thesis:

Goal 1: Early and basic analytics for a quick overview of the course
Instructors should be able to get a brief insight into the most relevant information regarding the course. Therefore, we refactor the course management overview, which is the first contact point for instructors and add essential analytics visualizations.

Goal 2: More detailed statistics about the course with extended functionality
On the basis of goal 1, an instructor should have the possibility of gaining even deeper insights. In a re-iterated course management detail page, we build upon the metrics[1] shown in the new course management overview and extend the functionality so the instructor is able to get an more detailed analysis.

Goal 3: Elaborated statistics for specific course content
In order to get an in-depth look into specific course related entities, we want to supply the instructor with detailed knowledge about every little detail concerning the course. To achieve that, we provide elaborated statistics regarding the course, exercises, lectures and exams. As it is relevant for administrators how excessively Artemis is utilized by users, we treat Artemis as an entity and provide Artemis-spanning analytics.

[1] A metric is a quantifiable measure that is used to track and assess the status of a specific process (https://www.klipfolio.com/blog/kpi-metric-measure). In this particular context, a metric is a measurable information about the course

1.4 Outline

As described in [BD09], the outline of this thesis corresponds the main software development activities:

Chapter 2 Related Work uses comparable Learning Management Systems to demonstrate already existing teaching analytics aspects.

Chapter 3 Requirements Analysis summarizes the current system in terms of teaching analytic practices, accumulates and analyzes the requirements of the proposed system and provides system models to further evaluate the requirements according to the Analysis Document Template by Brügge, et al. [BD09].

Chapter 4 System Design follows the System Design Document Template in [BD09] and shows how the requirements are realized on the system design level.

Chapter 5 Object Design elaborates on how statistics are implemented into the proposed system.

Chapter 6 Summary recaps the implemented goals of this thesis and which requirements the implementation fulfills. Furthermore, we discuss potential future work on this topic.

Chapter 2

Related Work

In this chapter we highlight how other systems integrate teaching analytics. We demonstrate different visualization techniques and show in which way Artemis differs from them but also how Artemis uses similar approaches to solve the issues that arose in Section 1.1.

2.1 edX Insights

edX is one of the biggest Learning Management Systems worldwide with over 110 million enrollments[1]. Founded and developed by Harvard University and Massachusetts Institute of Technology, it is utilized by many universities, including the Technical University of Munich, and runs on the open-source software Open edX. edX Insights is an administration tool introduced by this particular platform. It provides instructors with information about their courses and its learners. Metrics provided by edX Insights are for example the number of course enrollments, the engagement with course content or submission related information like the number of correct submissions by students[2].

Weekly Learner Engagement

Figure 2.1 shows a line graph regarding the weekly learner engagement of a course. The graph counts the number of users for different aspects of content interactions. Four different metrics are shown on a weekly basis.

[1] https://www.classcentral.com/report/edx-top-1000-website/,https://press.edx.org/edx-passes-110-million-total-global-enrollments-up-29-million-year-over-year

[2] https://edx.readthedocs.io/projects/edx-insights/en/latest/Overview.html

These are the number of active users, the amount of user who watched a video, the amount of users who tried a problem and the number of users who participated in discussions. edX defines the number of active users as those who visited a course page at least once. The number of students who tried a problem are the ones who submitted at least one submission for a problem in this particular week. A participation in a discussion can be, for instance, a post, a response or a comment.

With regard to Artemis, we define terms differently. As Artemis currently cannot track the pages a user visited in a session, we define an active user as a person who actively participated in the course by handing in a submission. The number of active users and number of discussion participations are metrics we will also address in Artemis.

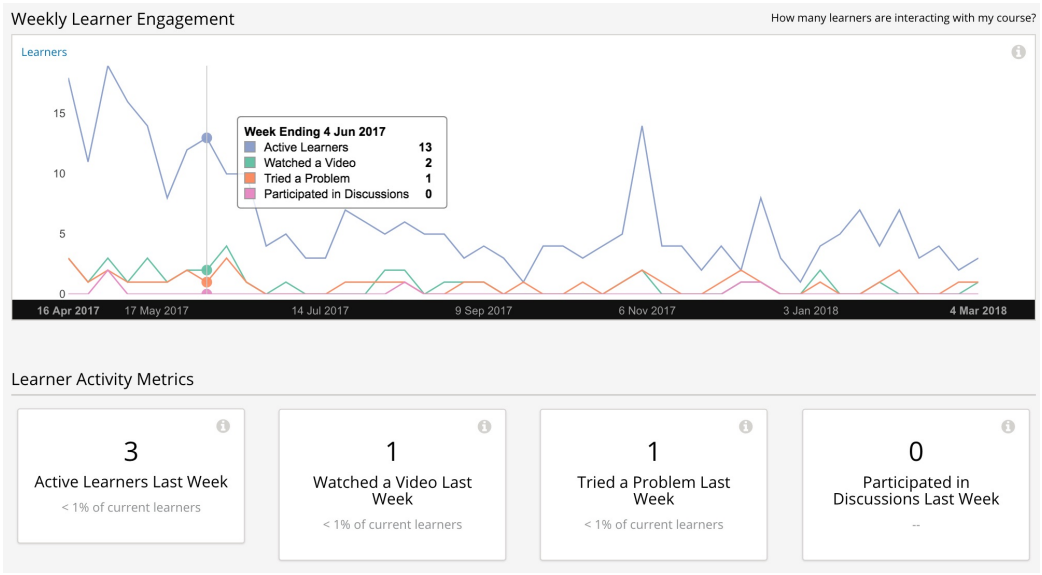


Figure 2.1: The Weekly Learner Engagement of a course in edX Insights[3]

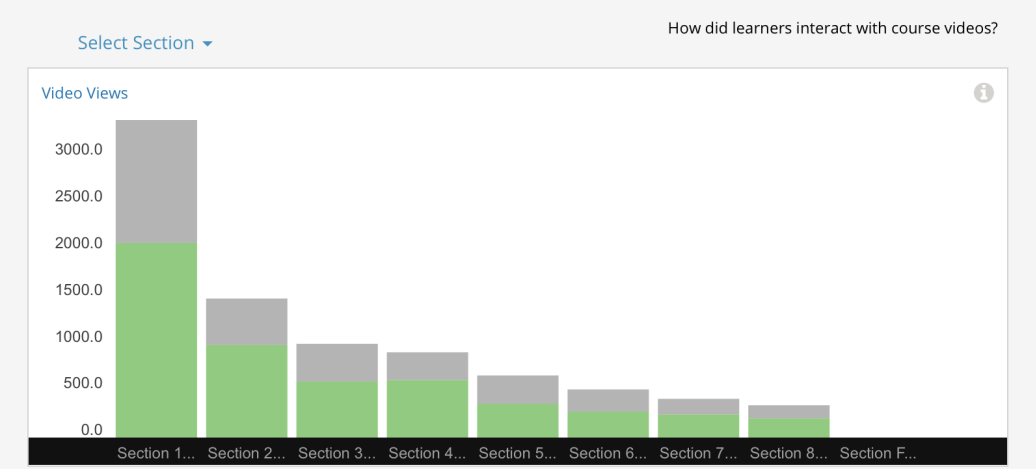
Engagement with course videos

Another possibility for instructors to improve lecture content is to analyze the students' engagement with course videos [HPM+20]. edX Insights provides multiple statistics regarding this. The bar chart in Figure 2.2 visualizes the number of views of all videos in a course section. The bar itself is divided into the amount of completed views, shown in green, and the number of incomplete views, shown in gray. A view counts as completed if the user

[3] https://help.appsembler.com/article/312-viewing-engagement-data-in-open-edx-insights

watches the whole video. edX Insights provides additional information below the graph. In a list, the instructor can see the title of the section together with the number of videos, the average number of complete and incomplete views of those videos and the completion rate in percentage.

If instructors want to investigate the origin for a high incompletion rate of views, there is the possibility to navigate into a specific section in order to inspect subsections as well as concrete videos to find out which lectures need to be improved in the future.



Video Views

Order	Section Name	Videos	Average Complete Views	Average Incomplete Views	Completion Percentage
1	Section 1 Project Organization and Management	7	2,082.7	1,315.4	61.3%
2	Section 2 Software Configuration Management	5	990.0	498.0	66.5%
3	Section 3 Object Oriented Programming	6	599.7	403.7	59.8%
4	Section 4 Requirements Analysis	6	616.3	296.3	67.5%
5	Section 5 System Design	6	361.7	303.0	54.4%
6	Section 6 Object Design	5	278.2	236.8	54.0%
7	Section 7 Testing	7	247.9	166.7	59.8%

Figure 2.2: Video views grouped by course sections in edX Insights

As instructors are experienced in their area of proficiency and students in lecture usually have many different levels of competence, it can happen that students do not understand things the instructor explained. An advantage of Learning Management Systems is the possibility to rewatch videos. It can be relevant for instructors which lecture parts students did not understand as they should improve those and either explain the content in simpler terms or elaborate on the subject in more detail. edX Insights is able to detect parts of videos which students watched more than once, as seen in Figure 2.3.

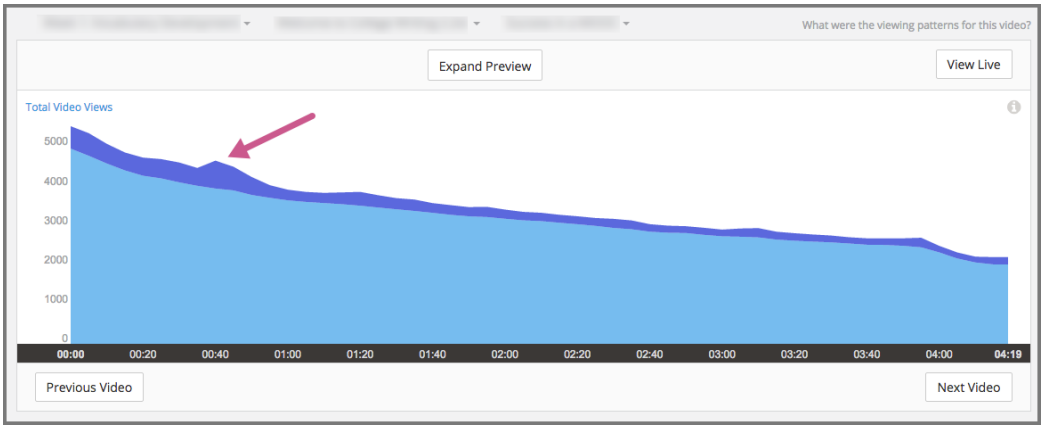


Figure 2.3: Video frames which are viewed more than once, highlighted in dark blue[4]

To improve lecture quality even further, instructors in edX Insights can identify at which point students stop watching the lecture. Figure 2.4 displays a steadily decreasing number of student views, which indicates that the instructor looses the students' attention and should work on the quality of the lecture in order to optimize student performance [Pur07].

In contrary to edX Insights, Artemis lacks video analytics offerings for instructors. Since edX provides its own video player, there are more possibilities to gather data. In Artemis, it is possible to embed videos into a lecture by providing a video URL [Wal21]. However, we currently do not collect any data regarding student views.

[4] `https://edx.readthedocs.io/projects/edx-insights/en/latest/engagement/Engagement_Video.html`

[5] `https://edx.readthedocs.io/projects/edx-insights/en/latest/engagement/Engagement_Video.html`

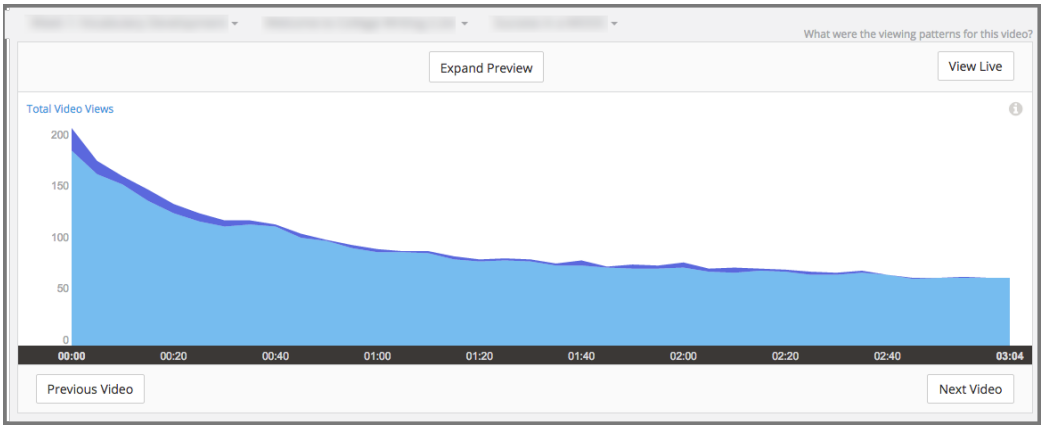


Figure 2.4: Number of video views slowly declining as students stop to watch the lecture[5]

Student submission analytics

In regards to submission analytics, edX Insights has similar constraints as the current Artemis version. A limitation of edX are the number of exercise types, where the only ones with analytical support are Checkbox, Dropdown, Multiple Choice, Numerical Input and Text Input problems[6]. For those, edX Insights provides statistics regarding the distribution of correct and wrong answers as well as the content of those, showcased in Figure 2.5. There is the possibility to create custom exercise types through a plugin architecture, called *XBlock* [Le16]. However, edX Insights does not support analytics for those exercises.

Artemis in its current state supports analytics for quiz exercise [Iss18], but not for modeling, text, file-upload or programming exercises, which we address with the proposed system. The complexity of problem statements and their answer limits us in revealing insights into the content of submissions. Taking text exercises as an example, it is currently not practical to visualize submission content due to the length and variety of different student submissions.

[6] https://edx.readthedocs.io/projects/edx-partner-course-staff/en/latest/exercises_tools/create_exercises_and_tools.html#common-problem-types

[7] https://edx.readthedocs.io/projects/edx-insights/en/latest/performance/Performance_Answers.html

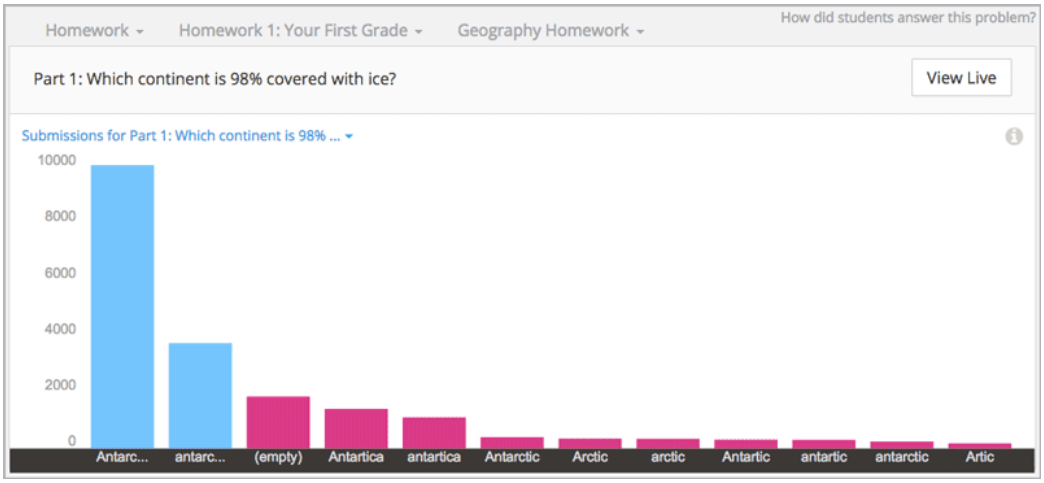


Figure 2.5: Submissions analytics of a text input problem in edX[7]

2.2 Canvas

Canvas is a cloud-based Learning Management System with integrated teaching analytics practices [TM18]. The platform provides instructors with a large variety of functionalities, similar to the ones we integrate in our proposed system.

In Figure 2.6 we demonstrate how Canvas visualizes different average grades in a course. Below the general course average grade (1), Canvas displays a dot chart (2) containing the average grades of different exercises. The site allows the instructor to filter for specific exercise types (3) as well as specific course sections and students (4).

[8]https://community.canvaslms.com/t5/Canvas-Basics-Guide/What-is-New-Analytics/ta-p/73

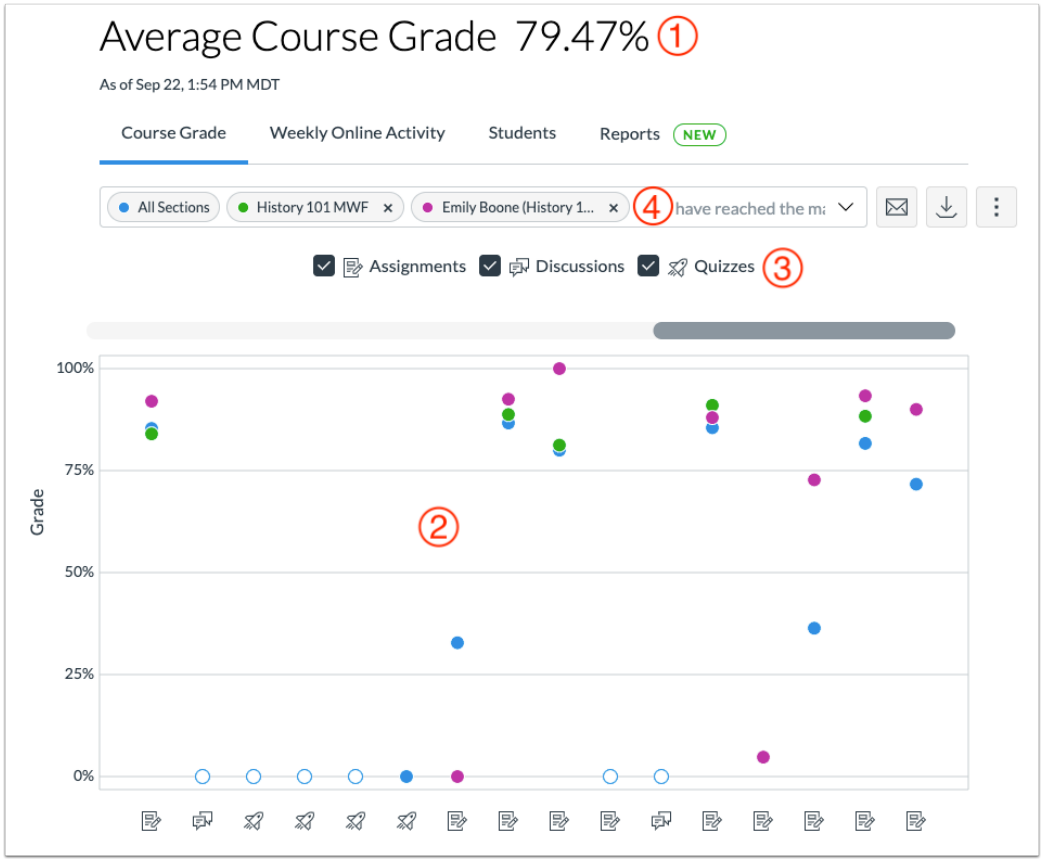


Figure 2.6: The course average score in canvas[8]

Canvas has additional metrics regarding course interactions. As we show in Figure 2.7, the instructor gets further in-depth insights into courses. An instructor is able to see when students interact with the course in terms of page views and participations (1). The *Submissions* graph (2) provides the instructor with information regarding submissions in each course exercise. *Submissions* can additionally differ between on time, late and missing submissions from students.

As the average score in Figure 2.6 does not reflect the grade distribution among the students, Canvas provides an additional statistic visualizing this data. The *Grades* chart (3) lets instructors evaluate whether there is a noticeable large gap between good and bad performing students or whether the majority of students achieved grades close to the average. This can be useful for lecture evaluations, as a large amount of bad performing students could imply a weak point in the lecture, even if many students acquired good grades as well.

Canvas extends the analytical input for instructors by providing the aforementioned metrics for each individual student (4). This includes pages views, participations, submissions and the current score. Especially the average score and the possibility of sorting allows the instructor to identify students at-risk which is an important step in improving student results.

Canvas' analytics share some similarity with the statistics we integrate in the proposed system. As the average score and the other previously mentioned datasets are fundamental indications for course weaknesses, we provide the majority of these metrics to the instructor in Artemis. A limitation that we currently face in Artemis is the fact that the system does not track which pages the user visits. Since this is not in the scope of our work due to time reasons, we can not visualize page views to the instructor.

[9] `https://a6199-661633.cluster76.canvas-user-content.com/courses/6199~20914/files/6199~661633/course%20files/Embedding/Learning%20Analytics/CanvasCommunityCanvasAnalyticswhatis.htm?download=1&inline=1`

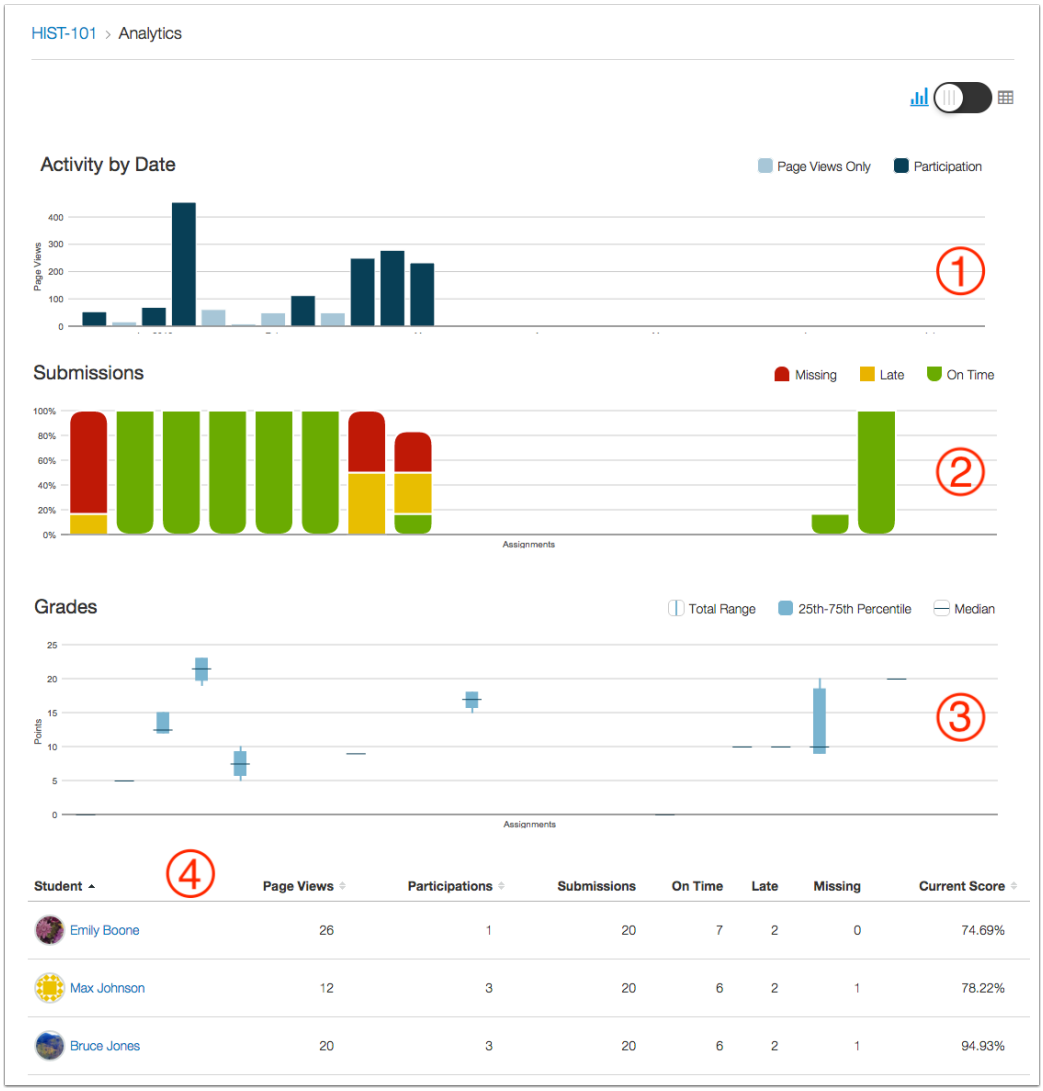


Figure 2.7: Course Analytics in Canvas[9]

Chapter 3

Requirements Analysis

We base this section on [BD09] and introduce two aspects of software development, requirements elicitation and requirements analysis. Both are of high importance in order to create a well defined basis for the upcoming chapters. First, we give a short overview on how the proposed system improves the quality of the current system. Second, we summarize what the current system is able to do in terms of teaching analytics aspects. Third, we provide UML system models in order to further specify the proposed system.

3.1 Overview

Like already mentioned in Section 1.2, we want to improve the instructors possibilities to overlook the course. As instructors currently have only very little insights into the students achievements, we want to integrate various statistics into Artemis that will help instructors evaluate the current status of the course and their students. These statistics are grouped into single views to provide relevant information about the students progress in exercises, the workload that lies on them and how well they do in those exercises.

Furthermore, we want to provide a stepwise information illustration, which means that instructors should be able to acquire very basic information about the course status very early on, while also being able to specifically collect course related metrics on a deeper level.

3.2 Current System

We specify Artemis version 4.7.6 as the current Artemis version.

Statistics

As mentioned in Section 1.1 an instructor has only little opportunities to gather course information. An instructor can navigate into the *instructor course dashboard* to obtain basic information about the system, like shown in Figure 3.1. The view displays information concerning the course status, like the assessment progress. What is missing, however, are deeper insights into the students' achievements, as for example an average score that indicates how well the students did in finished exercises. The instructor should be able to have more insight into the assessment process as well, for instance whether tutors almost failed to assess all submission in time or whether they easily assessed all submission days before assessment due date.

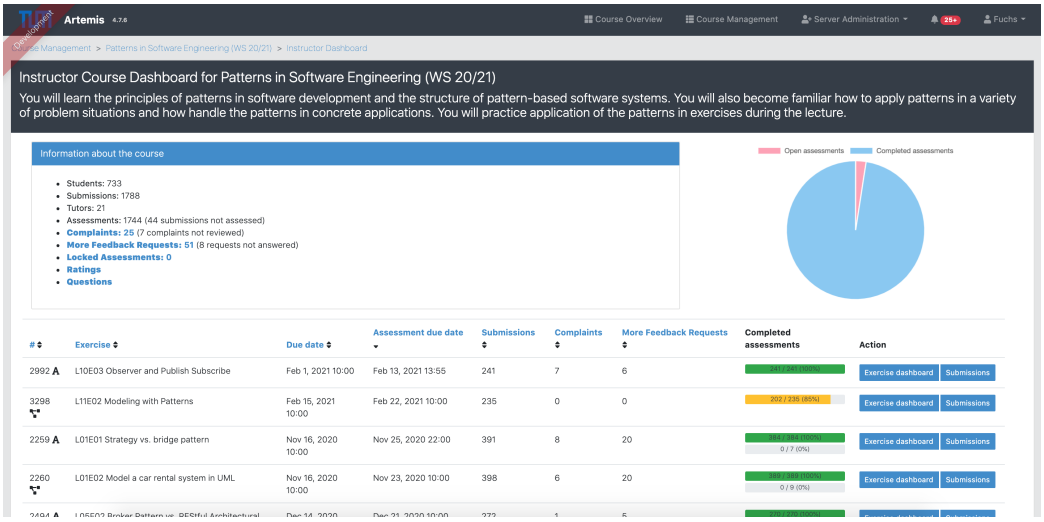


Figure 3.1: Instructor Course Dashboard in the current Artemis version

Another possibility for instructors is to access the *instructor exercise dashboard*, displayed in Figure 3.2. This interface shows the instructor some information regarding an exercise, like assessments and submissions. It is important to know how many assessments are done, but again a time constraint is missing.

Course Management

In the current system the course management overview is mainly used for navigational purposes. As seen in Figure 3.3, Artemis displays courses in a list, displaying basic information and providing links for tutor and instructor navigation.

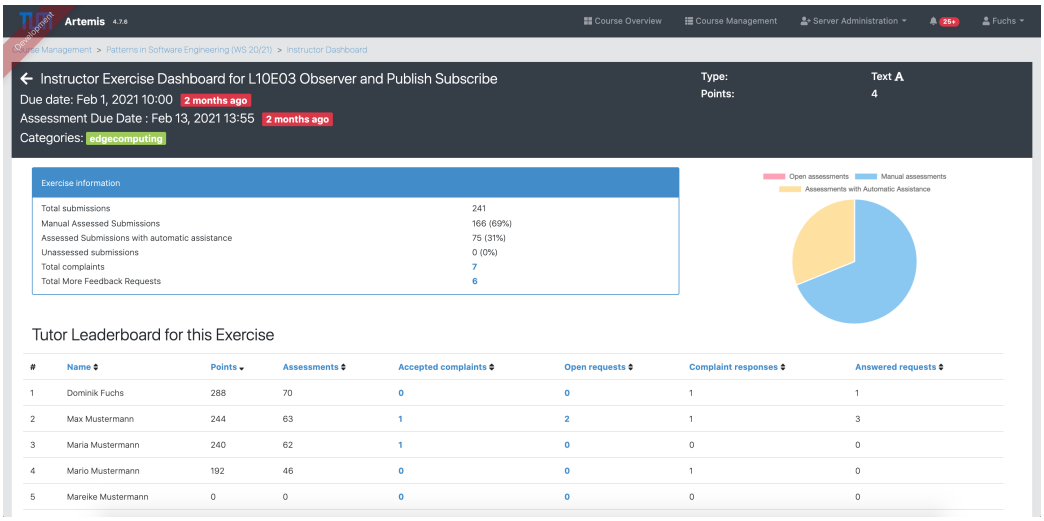


Figure 3.2: Instructor Exercise Dashboard in the current Artemis version

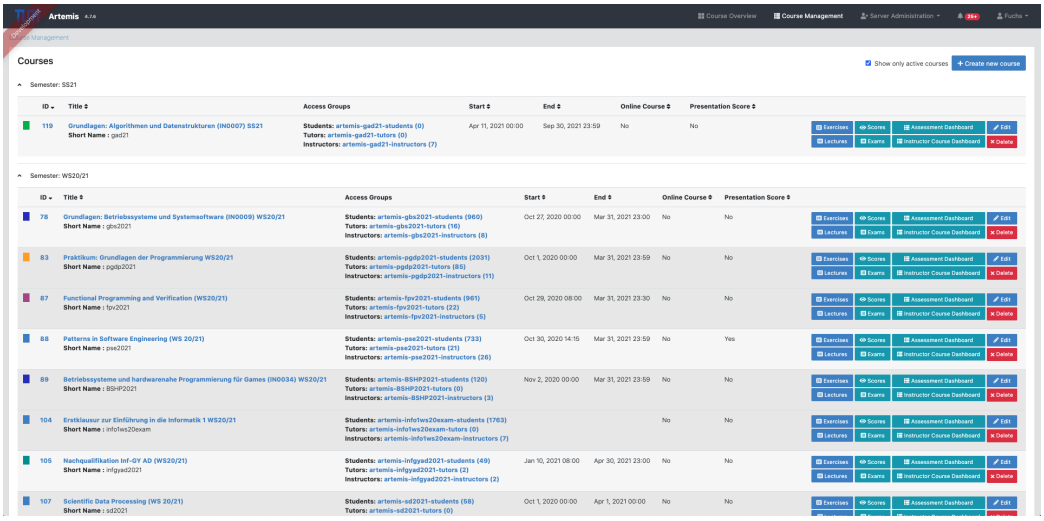


Figure 3.3: The course management overview in the current system

The current course management detail view as depicted in Figure 3.4 builds upon the data shown in the course overview and extends the displayed data by further aspects. Instructors can use this interface to examine basic course information and to navigate into tutor and instructor pages.

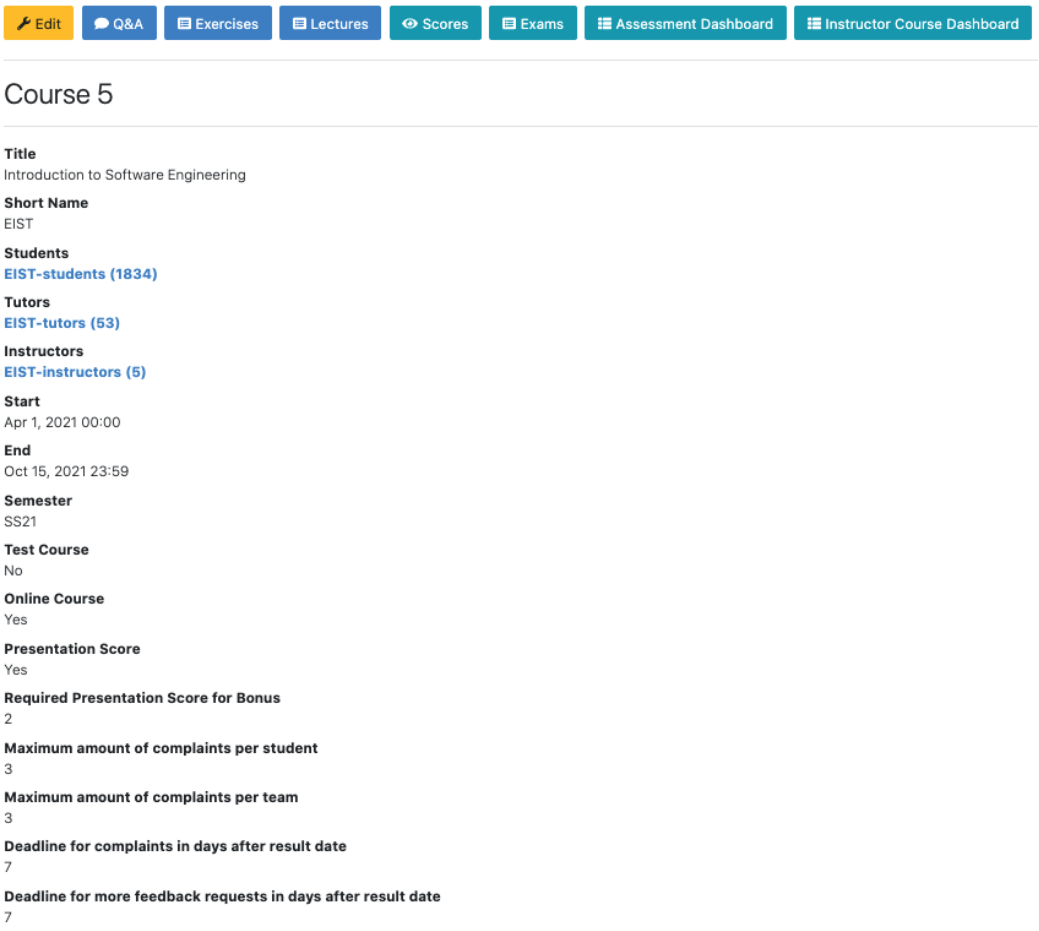


Figure 3.4: Current course management detail page as instructor

3.3 Proposed System

In the proposed system, the instructor is able to utilize many new metrics to improve course progress tracking. We introduce newly added metrics like average student score and the number of active users, which the user can view very fast by looking at the refactored course management overview as well as statistics like number of logged-in users, active tutors or conducted exams, which the instructor can observe by accessing one of the newly added

statistics views. In the proposed system, we introduce Artemis-wide *user statistics*, course-ranging *course statistics*, exercise-wide *exercise statistics* as well as *exam statistics* and *lecture statistics*.

As mentioned in the overview, we want to provide early access to course statistics in the course management overview. However, through the course management detail page, the instructor is able to extend the overview's visualizations and therefore obtain more knowledge on exercises and the students progress.

There is a defined hierarchy structure in Artemis. There are students, tutors, instructors and administrators. Students are not involved in course management and are therefore not relevant for this thesis. As tutors have the least privileges out of the management roles, an instructor can do everything a tutor is able to do for the respective course plus additional rights regarding instructor-related responsibilities. An administrator monitors the entire web application and has therefore instructor rights for each course plus additional permissions to overlook the server. With that in mind, declaring functionalities for tutors in the following sections applies for instructors as well and features for instructors also apply for administrators.

The following subsections list the functional and nonfunctional requirements of the proposed system, originated from the user's perspective.

3.3.1 Functional Requirements

This subsection lists the functional requirements (FRs) of the proposed system. [BD09] describes FRs as the interaction between the system and its environment, whereas an environment in this particular context is depicted as the user and other external systems. The FRs are independent of their actual implementation.

First, we introduce the FRs for newly added statistics in Artemis. Secondly, we demonstrate FRs for the refactored course management and after that state specific FRs for the course management overview and course management detail view.

Statistics

These FRs are aiming to define possible user interactions with statistics views. The general idea and outline of those are present multiple times in the scope of Artemis, for example as user statistics, the course statistics or as exercise statistics.

FR1 **Open global Artemis statistics**: An administrator can open different Artemis-related statistics like number of logged-in users and active users, amount of conducted exams and metrics about exercises and exams.

FR2 **Open course statistics**: Tutors are able to open different course statistics concerning their course (number of active users, tutor ratings, the average student score in the course and in different exercises and statistics about exams).

FR3 **Access exercise statistics**: A tutor is capable of opening different exercise statistics for each exercise like the number of active students, the average student score or the students' score distribution.

FR4 **Open exam statistics**: A tutor can open different statistics (average student score, grade distribution and number of registrations and participations) about an exam.

FR5 **Access lecture statistics**: An instructor can open different lecture statistics about questions asked and the students' progress in lecture videos and notes.

Course management

For the course management, we want to refactor multiple views in order to give them more relevance in the workflow. Both the course management overview and the course management detail view should display more data so that the user does not need to access several subpages to acquire the information. These two types of views share some similar functionalities.

FR6 **View active users**: The tutor can see how many students actively participated in the course.

FR7 **View exercises**: The tutor can see a list of exercises from the course

FR8 **View exercise progression**: The tutor can see the progression in an exercise (number of exercise participations, amount of assessed and unassessed submissions).

FR9 **View average score of exercise**: The tutor can view the average score of an exercise

Course management overview

The course management overview gives an outline over the courses the user has access to. This view can be accessed, if the user holds tutor or instructor rights for at least one course.

FR10 **View every accessible course**: To have a quick outline over every accessible course, the user can see courses where he holds at least tutor rights arranged among each other.

Course management detail view

Through this thesis, we refactor the course management detail view, which currently only shows basic course information and is used only very rarely. By displaying more interesting information, this view is integrated more deeply into the instructor's workflow.

FR11 **Open tutor statistics**: A tutor can open statistics (the assessment progress and the number of addressed and not addressed complaints and more feedback requests) about the course.

FR12 **View average score in course**: A tutor is able to see the students' current average score in the course.

FR13 **Search for exercise**: A tutor is able to manually search for a specific exercise

3.3.2 Nonfunctional Requirements

We list the nonfunctional requirements (NFRs) which, in contrast to the functional aspects mentioned above, lay down quality requirements of the proposed system. We categorize according to the URPS[1] model, which is described in [BD09] and is also used in the Unified Process in [JBR99].

With regards to the data which has to be processed, we define a reference value of a course as one with 2000 students, 40 exercises and corresponding participations and submission from these students.

[1]URPS, derived from FURPS+, constitutes an acronym using the first letter of each spect of the model: Functionality, Usability, Reliability, Performance and Supportability. The + stands for any additional subcategories that may arise

Statistics

NFR1 **Performance - Response time**: All data should be displayed within 4 seconds after accessing the page

NFR2 **Supportability - Extensibility**: If new requirements occur, a developer can easily add new graphs without having to create new classes.

NFR3 **Reliability - Security**: At no time should there be any information about students achievements on the client

Course management overview

NFR4 **Usability - Efficiency**: After logging into Artemis, the user can access the course exercises in 2 interactions via the course management overview

NFR5 **Performance - Response time**: When accessing the course management overview, the course information is displayed in 2 seconds

NFR6 **Usability - Learnability**: Every button without a description has a tooltip which explains the buttons usage

NFR7 **Supportability - Adaptability**: The system adapts to different screen sizes

Course management detail view

NFR8 **Performance - Response time**: When accessing the course management detail view, the course information is displayed in 2 seconds

3.4 System Models

In this section we demonstrate through different system models how we integrate the innovations into the planned system. This is still independent of its concrete implementation and aims to give a better understanding of the interactions between the components among each other as well as how the user interacts with the system.

3.4.1 Scenarios

In the following, we provide two scenarios, which are an informal and very concrete descriptions of a system's feature from the viewpoint of an actor [BD09]. First, we demonstrate a demo scenario, which focuses on innovations proposed by this thesis. Second, we show a potential futuristic feature of this system, which is however not feasible during this thesis.

Demo Scenario: Evaluating an informal student complaint as an instructor

The instructor Alice and her tutors have their weekly meeting, discussing the next week's schedule of the course "Software Engineering for Beginners", carried out in Artemis. Bobby, a tutor, mentions that his students recently complained in his weekly tutorial that the latest modeling exercise, "Bridge Patterns in the real world" was too difficult. After the meeting, Alice wants to make sure whether these statements are wrong or justified. She navigates to the course management detail page in order to figure out the average student's score in her course, which amounts to 78%. This means that the average student achieved 78% of the maximum points currently reachable in the course.

After this, Alice opens the course statistics page where she can find the average students score for every single exercise. The value for this particular exercise equals 85%.

Alice concludes that the accusations, the exercise "Bridge Patterns in the real world" would be too hard are incorrect. Because the specific exercise average score is actually higher than the overall course score, Alice decides to not make any changes in her following lectures.

Visionary Scenario: Detecting potential lack of time for assessment

The Artemis course "Patterns in Software Engineering" consists of several weekly assignments, which the students have to hand in until Sunday midnight. The assessment of those exercises takes place in the following week, where the students work on the next tasks and the tutors mark the latest submissions.

Harry, the instructor of the course, can usually focus on preparing new lectures and creating new exercises. Currently, it's the 6th week into the semester and Harry's tutors have some time shortage due to midterm exams, which take place in other lecture they attend. As a result, the tutors only managed to do circa 50% of the assessments they usually do until now. On Thursday evening, 3 days before the assessment due date, Artemis discovers

that only 60% of the assessment is done. Because statistically only 20 to 30% of the assessments are done on the weekend, Artemis automatically sends the instructor a notification, warning about the lack of assessments. Artemis provides the prediction that only 80% of the submissions will be graded until the assessment due date, which would result in students receiving delayed feedback and therefore a shorter preparation time for upcoming exercises and lectures.

Harry perceives the notification and messages the tutors in their usual communication tool, that they need to speed up the assessment process. In the following days, the tutors organize assessment sessions where they are able to resolve issues immediately and therefore finish the exercise assessment until Sunday afternoon.

3.4.2 Use Case Model

In the next subsection we want to visualize the FRs proposed in Section 3.3.1 using use case diagrams. We define the therein contained use cases according to [BD09]. The actors in this particular case are limited to tutors, instructors and administrators, since these are the roles in Artemis we focus on in this thesis.

Statistics

Figure 3.5 displays the use cases for opening statistics in Artemis. A tutor can open different exam statistics, course statistics as well as exercise statistics. Referring to the annotation at the beginning of Section 3.3, this implies that these interactions with the tutor are also possible for instructors and administrators. An instructor can additionally open several lecture metrics, as the lecture management is only accessible for instructors and not for tutors. A functionality exclusively for administrators is the access to the user statistics.

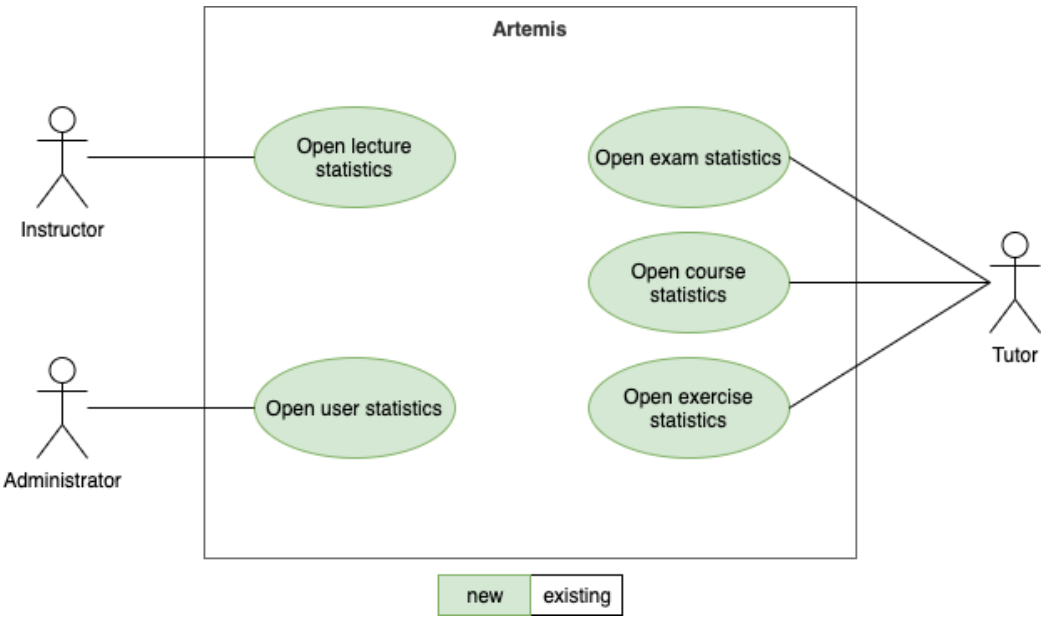


Figure 3.5: Use case diagram of the statistics pages in the proposed system

Course Management

For the second use case model depicted in Figure 3.6 we illustrate the interaction possibilities with the improved course management of the proposed system. With the advanced course management, tutors and roles with higher privileges can view the number of active users in the course. Tutors can directly see a list of course exercises with information like the title and important dates like the release date, due date and the assessment due date. Tutors are also able to view information about the exercise progression, which includes the number of student participations in the exercise and the amount of assessed and unassessed submissions. To get an idea of how well students did in past exercises, tutors can see the average scores of those.

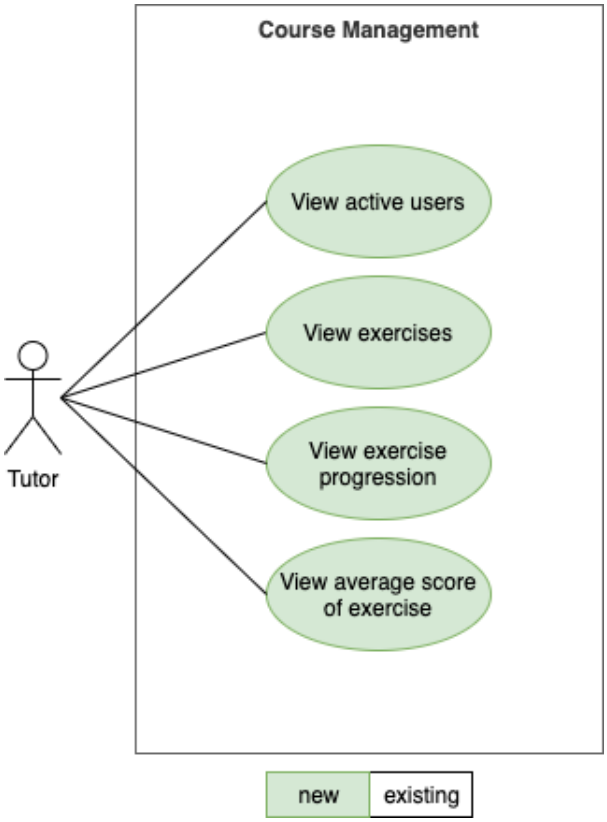


Figure 3.6: Use case diagram of the course management

3.4.3 Analysis Object Model

In the next subsection, we will deal with the analysis object model of the proposed system. In order to show the structural setup of the application domain, we provide an UML class diagram and elaborate on the changes in further detail. The analysis object model operates on a user-level and aims to identify abstract concepts that the user interacts with [BD09], which is why we omit unnecessary implementation details like access modifiers and return types.

In Figure 3.7 we can see the structural integration of statistics views into Artemis. We picture already existing entities in white, whereas newly added items are colored in green. A *course* with its regular attributes and methods has a number of unique *course statistics*, with each of them visualizing one specific metric. *Lectures*, *exercises* and *exams* as part of a course share a similar structure and have specific statistics. The specialized statistics inherit from a *statistic* superclass, which provides basic mutual attributes and methods all statistics share. We will elaborate more on the *statistic*'s timespan and how it utilized in Section 3.4.4. A *chart* takes over the presentation of the metric and displays the data with the help of labels.

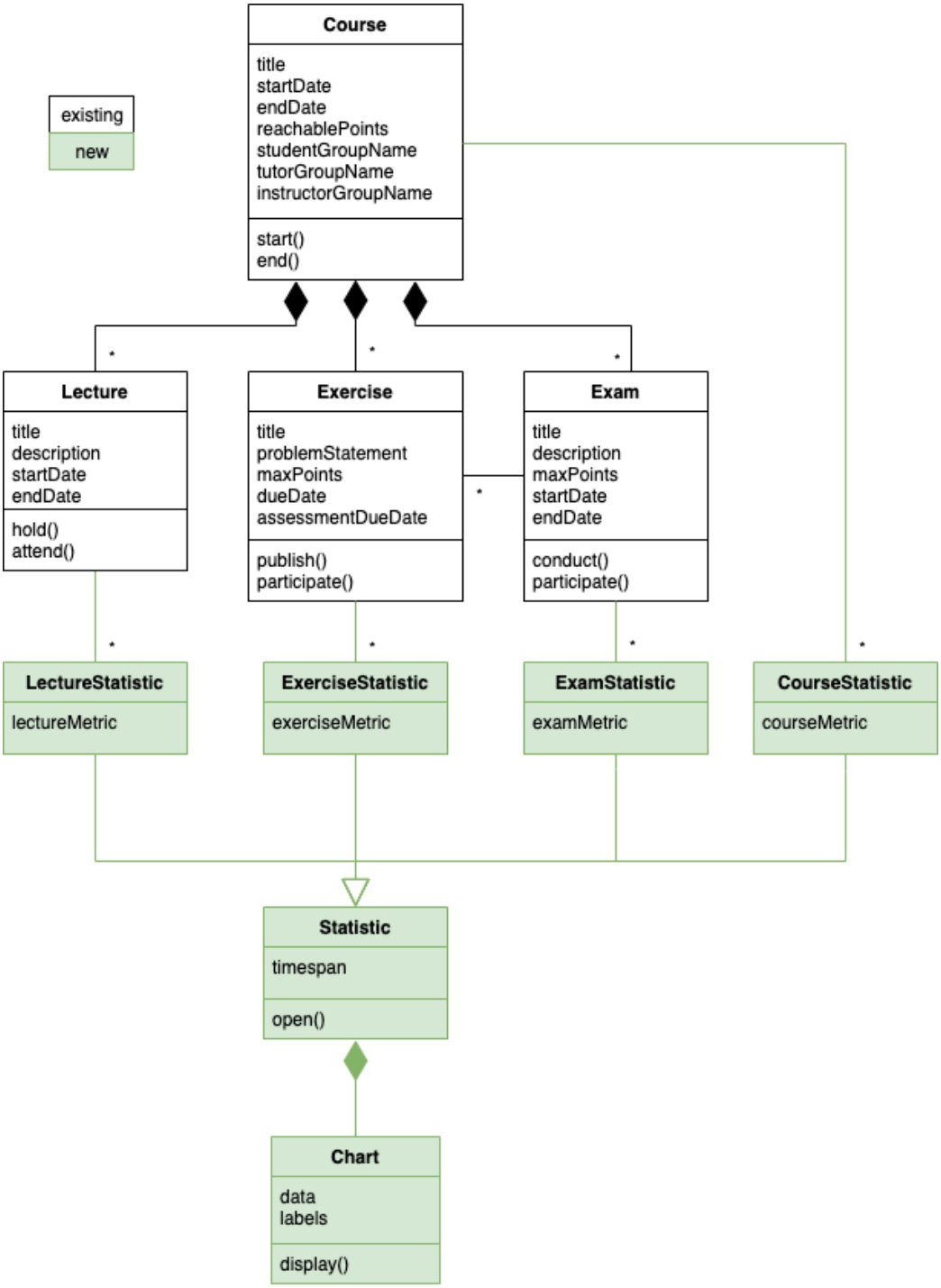


Figure 3.7: Analysis object model of the proposed system

3.4.4 User Interface

In the user interface subsection, we visualize previously analyzed innovations of the proposed system. Successively, we will show a representative view and elaborate on this in more detail. To distinguish more easily between page elements, we use marks in the following figures.

Statistics

First, we demonstrate the new user statistics view. It is only accessible by administrators and should give them an overview of the system in terms of the usages by different roles. As the user statistics page is already completely implemented, Figure 3.8 shows the finalized interface of this integration.

The design of the page is simple as the focus is supposted to lie on the diagrams. By using the buttons marked with (1) it is possible to change the timespan of which the data is shown. The following span types are available:

- **Day**: Shows the data throughout the day - considered hourly
- **Week**: Shows the data during the last 7 days - considered daily
- **Month**: Shows the data during the last month, depending on the last month's length - considered daily
- **Quarter**: Shows the data of the last 12 weeks - considered weekly
- **Year**: Shows the data during the last 12 months - considered monthly

By default, the active timespan is *Week* as seen in Figure 3.8. Figure 3.9, Figure 3.10, Figure 3.11 and Figure 3.12 display the remaining options and also visualize more metrics which the interface provides.

An additional element of this page is the ability to switch time periods. By clicking one of the arrows besides every graph (2)(3), the user can either view the previous time frame or the following one. While the arrows imply changes to only one graph, switching timespans with (1) affects all charts. After an interaction with the user, all affected graphs automatically refresh and show the wanted information.

Using this functionality, it can be quite hard to keep track of the currently displayed time period. That is why below every graph, we show the start and the end date of the timeframe (4).

The user statistics have several metrics visualized to the administrator. Since the appearance of the charts are identical except for data and title, we representatively demonstrate the number of submissions, the amount of

active users and the number of logged-in users in Figure 3.8. For the sake of completeness we list all metrics shown in the user statistics in the following:

- **Number of submissions done**: The administrator is able to view how intensely students work on exercises by the number of submissions made.
- **Number of active users**: The administrator can view how many students have been actively working on exercises.
- **Number of logged-in users**: The administrator can identify how many users recently logged in.
- **Number of released exercises**: To schedule exercise releases, the administrator can see how many exercise releases there are at a specific time.
- **Number of exercises due**: To know how many exercises close at a specific time, the administrator can view the number of exercises due dates
- **Number of exam conductions**: To overview how many exams take place at once, the administrator can overview the amount of exams which will take place or have taken place.
- **Number of students which participated in an exam**: To check how many students actually participated in exams, the administrator can see how many exam participations exist.
- **Number of exam registrations**: To estimate how big exam conductions are, the administrator can view how many students are registered to exams at a particular point.
- **Number of active tutors**: To get an overview at what time tutors assess student submissions, the administrator can see this the number of active tutors at a particular time.
- **Number of created results**: To overview at what time tutors create the most or the fewest submissions, the administrator can see the temporal distribution of created results.
- **Number of created feedback**: Because it is interesting to evaluate how many feedbacks are created in a particular time, the administrator can see the amount of feedback creations.

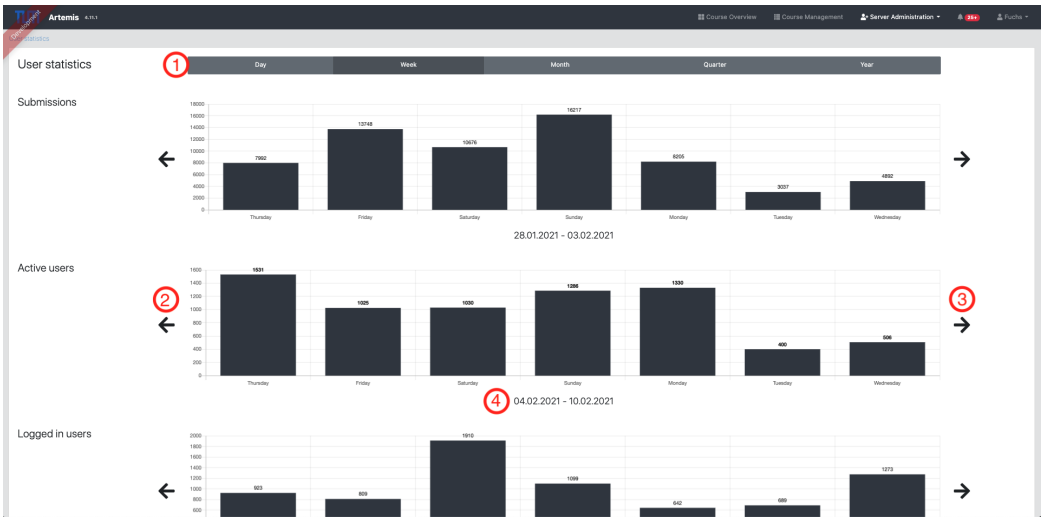


Figure 3.8: The user statistics page showing the amount of submission, active users and logged-in users, each in different weeks

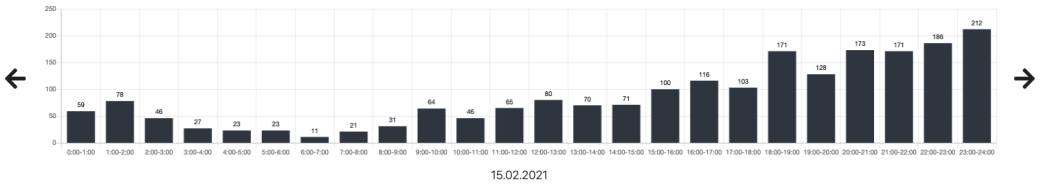


Figure 3.9: The distribution of submissions made throughout a day in Artemis

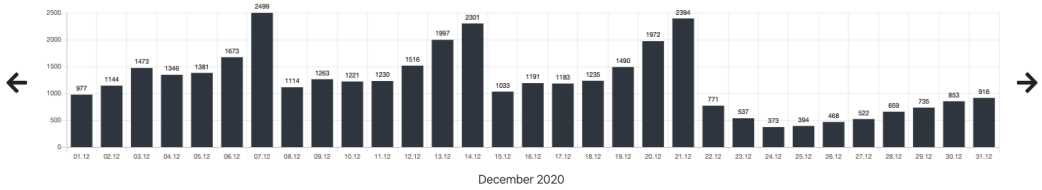


Figure 3.10: The number of active users during December 2020

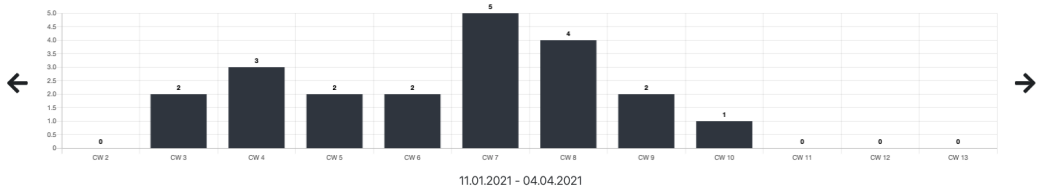


Figure 3.11: The number of conducted exams during the exam phase of WS 20/21

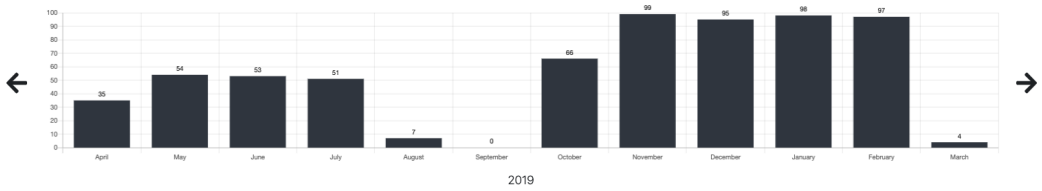


Figure 3.12: The number of distinct active tutors from April 2020 until March 2021

Other statistic pages

For the integration of statistic pages into Artemis, we use an modification of a *Specialization* approach [BDDS02]. This means that while providing a relatively general overview over all Artemis-wide courses in the user statistics, we specify further into separate course exclusive statistics pages and from there on into separate exercise, lecture and exams statistics pages as well. On the one hand a general Artemis-wide statistics interface supports administrators in identifying bandwidth problems or scheduling exam conductions and on the other hand some metrics like the average score are not meaningful on a universal Artemis-wide level and should be rather included only in course statistics.

In order to support specialization, we provide the same metrics as shown in the user statistics, but also extend the information visualization so Artemis can display special content related metrics as well. For course statistics, this covers for example the just mentioned average scores of students in homework and also question related information like questions asked or questions answered. Artemis adds these metrics to the list of displayed graphs, like the average scores graph shown in Figure 3.13.

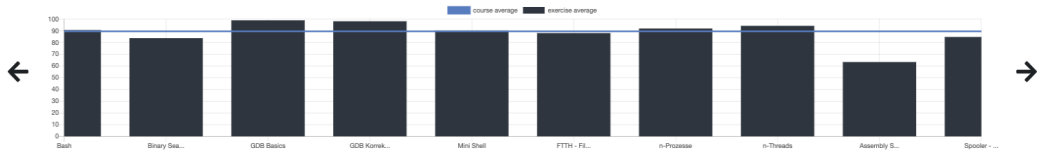


Figure 3.13: Additional metrics for the course statistics page. The blue line indicates the average course score while the bars present average scores of particular exercises

Course Management Overview

We want to enrich the current course management overview in order to give it more use than just navigation, which is now the case. In Figure 3.14 we demonstrate the course management overview of the proposed system.

Here Artemis arranges the courses among each other, but provides more information regarding the courses' progress during the semester. Artemis now displays a course as a tile and in this particular tile, we integrate an exercise list where the user is able to see the most relevant exercises (1). They are categorized in 4 different groups.

Beginning with the group on the very top of the list, are the *Released Soon* exercises, containing the ones which have not yet started. The second group of this list are the exercises *Currently In Progress*, which contain exercises the students are participating at the moment. The last two exercise groups are the tasks which are currently assessed by the tutors, called *Currently In Assessment* and exercises which are completely finished, including assessment. In order to not make the list too long and confusing, the *Currently in progress* group is the only one not collapsed by default and we limit the future exercises to the ones starting during the next week. Past exercises show the latest 5 and indicate with *4 of 4* (2) how many past exercise the course potentially contains.

Considering a specific exercise row, Artemis shows the exercises' title and dates (3), which is also done in the current system, but now features analytics data. Since exercises have different relevant aspects depending on the state, we show different metrics for each exercise group previously discussed. Following statistics are shown for the respective group (4):

- **Future Exercises**: For upcoming exercises we find there is not really a necessary metric to display, so we do not display anything.

- **Currently In Progress**: For exercises which the students work on at the moment, Artemis displays the number of submission in relation to the number of students in the course.

- **Currently In Assessment**: For exercises which the tutors currently assess, Artemis shows the number of assessments in relation to the amount of student submissions.

- **Past Exercises**: For exercises with already finished assessment, Artemis displays the average student points in relation to the maximum reachable points of the exercise.

On the right side of the aforementioned, the user can directly navigate into an exercise subpage in order to edit it, view the exercise scores or enter the submissions page (or respectively for programming exercises go to the grading page).

Next to the exercise list we present a simplified version of the active users chart taken from the statistics views. We minimize it so it shows the number of active students of the last 4 weeks in this course.

Just like in the current system's course management overview, the tutor is able to navigate to *Exercises*, *Exams* and *Assessment Dashboard*, whereas an instructor can additionally access *Lectures*, *Scores*, the *Instructor Course Dashboard* and also the course group management page. A newly added navigational element is the *Statistics* button, which open course statistics for this particular course.

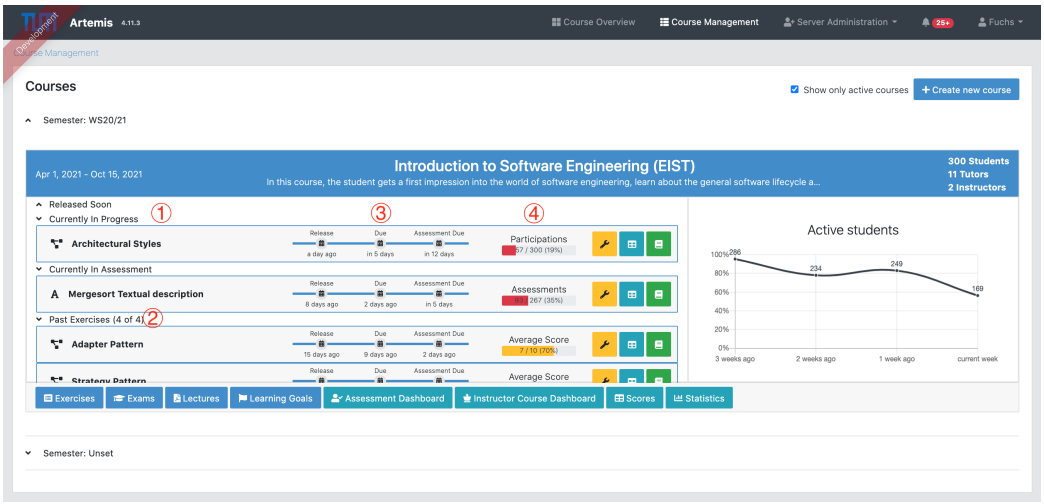


Figure 3.14: A tile representing a course in the Course management overview

Course Management Detail Page

We do not create the course management detail page from scratch but instead improve an already existing page. We want to extend the ability of this page in order to help the instructor or tutor to get a better overview of the course by introducing new features.

We demonstrate an advanced interface in Figure 3.15. The idea is to build upon the data displayed in the tile of the overview. The instructor gets a first impression of the course status through the tile in the overview page and if deeper insights are needed, has the possibility to move into the detail page by clicking on the tile header to access it.

At the top of the interface, we provide buttons for navigational purposes. Below, Artemis shows the number of active students throughout the last 4 weeks and offers the opportunity to adapt the time period as it is possible in the user statistics view (1). Artemis provides an exercise list similar to the one displayed on the overview page (2) but this time including every exercise of the course and the possibility to search for specific exercises through a search bar on top of the list (3). The progress bars (4) are able to display different metrics depending on the current status. Future exercises do not display any metric as there is none to show. Exercises which the student work on at the moment visualize the number of participations. Exercises which are currently assessed by tutors display the number of participations and the progress of the assessment. For exercises which have a passed assessment due date we show the number of participations and the average score in the exercise.

A detail view exclusive integration are the doughnut charts marked with (5), which offer 4 different metrics regarding the tutor and student progress. First, Artemis displays the percentage of overall assessments in relation to the amount of exercise submission. Second and third, metrics regarding tutor feedback in terms of complaints and more feedback requests are given, with a special focus on how many of those the tutors already addressed and how many are still unanswered. Fourth, we show the average student's score throughout the whole course.

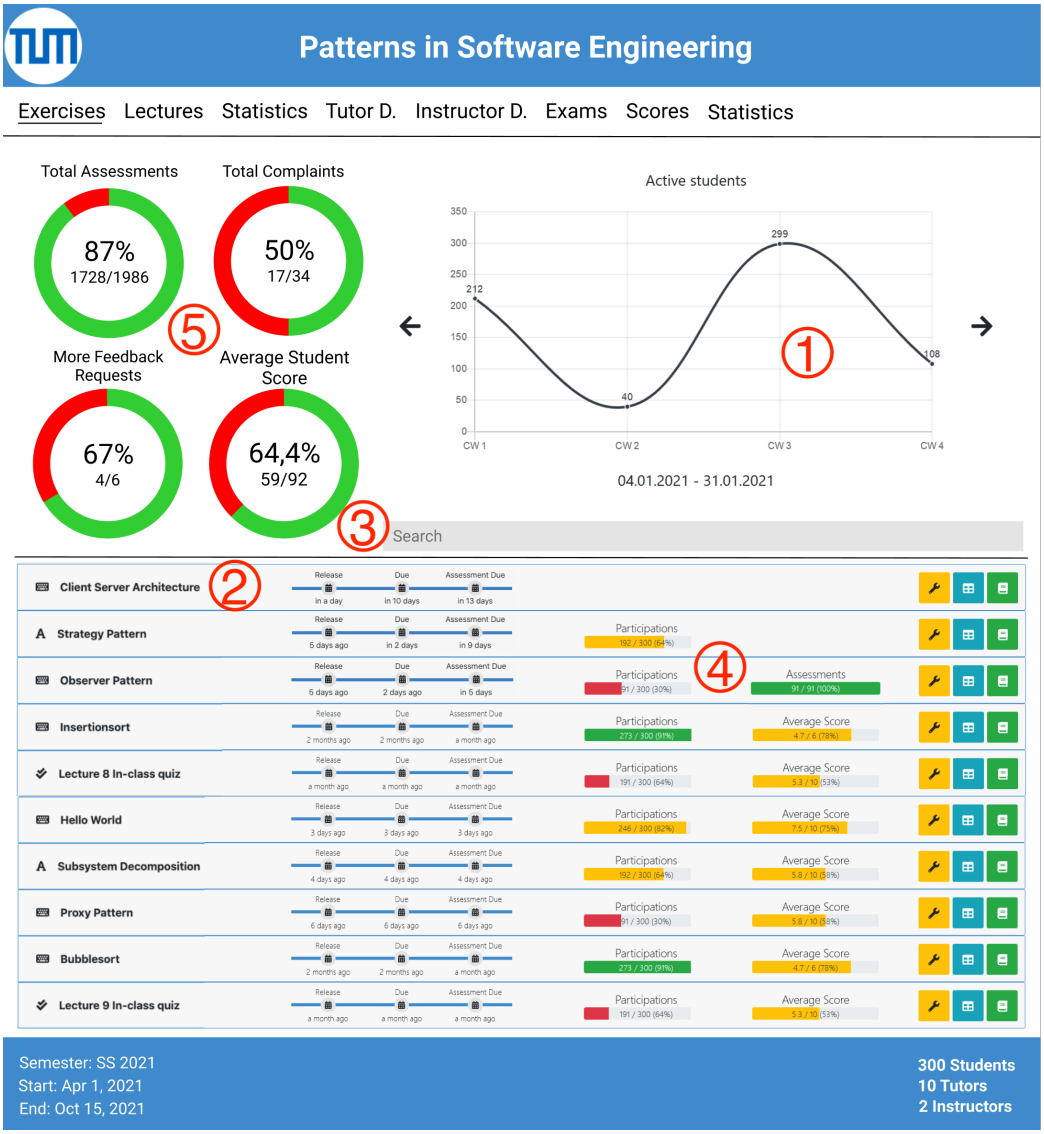


Figure 3.15: Proposed course management detail page with advanced metrics

Chapter 4

System Design

In the following chapter we introduce system design models based on the analysis model of the prior requirements analysis. The goal of this chapter is to bridge the gap between the application domain which we discussed in Chapter 3 and the solution domain. While the analysis model depicts the system only from the actors' point of view, system design alters the viewpoint and also takes in account how the system should be realized in terms of the internal structure [BD09]. We provide an overview of the proposed system, define design goals and create a subsystem decomposition. As a guideline on how to accomplish this, we follow the System Design Document Template presented in [BD09].

4.1 Overview

We do not change any core functionality or technology in the proposed system. Artemis still uses a client server architecture as architectural style, a special case of layered architecture. Characteristics of this style is the communication of the end user only with the client through a graphical user interface (GUI), which then processes the requests, sends it to the server, receives a server response and displays this response to the user. The implementation of the client, also called *Application Client*, is in SCSS, HTML and Typescript. Additionally, we use frameworks like Angular for Typescript and Bootstrap for CSS. The client interacts with the *Application Server* using Websockets and a REST-based Application Programming Interfaces. The implementation of the *Application Server* is mainly based on Java and we make use of the frameworks Hibernate for object relational mapping and Spring. MySQL is the relational database for the system and besides the *Application Server*, the server also communicates with a Continuous Integration server (CI) and

the Version Control Server (VC).

4.2 Design Goals

In this section, we want to map nonfunctional requirements taken from Section 3.3.2 and map them to design goals. Design goals are an important aspect of system design as they crucially guide decision-making and describe goals for system optimization. Because design goals can also conflict with each other, there are design goal trade-offs which we need to consider during feature integration. Following listed design goals are sorted from high priority to low priority.

1. **Performance**: On the one hand a page that the user must explicitly open will only be used if it is quickly accessible (NFR1, NFR5, NFR8). On the other hand, a page which is used very frequently must provide good performance otherwise it will slow down the user's workflow immensely [Bar10]. Therefore, we need to achieve good performance when collecting data.

2. **Usability**: When analyzing data, especially a large amount of data, it is of high importance to clearly define what data is important to the user and how this data is displayed so the user does not get overwhelmed. Interfaces should be easy to use by having a clear structure and different features should also be (visually) separated from each other. The time until the user is able to use the page to its full potential should be as low as possible (NFR6). As users operate on different monitors with different screen sizes, the pages should also adapt to that (NFR7).

3. **Adaptability**: When defining which data is relevant and should therefore be displayed by Artemis, a certain amount of subjectivity cannot be prevented. Thus, we want it to be easily manageable to add or remove metrics which might become relevant or erase less important in the future (NFR2).

Trade-Offs

Like mentioned in the overview, it is possible that design goals work against each other. In the following, we highlight some of those conflicts and elaborate why we choose one over another.

1. **Functionality vs. Usability**: As stated in the design goals, it is important to evaluate which data is relevant and which data can be left unmentioned. There is a very high risk that a page is getting too complex and too unclear when too much data is provided at once. Still, the user should have access to as much information as possible. Since it is also feasible to distribute information into several subpages, we have the opportunity to provide the most important data in the overview of the course, while getting into more detail in the detail page, therefore not displaying too much information on the basic overview page while also giving the user the opportunity to acquire deeper knowledge if needed.

2. **Rapid development vs. Functionality**: Regarding the time limitations this thesis has, we need to evaluate how much we can achieve in 4 months. Having this timeframe in consideration, basic analytics should be done first: Providing a general user statistics page for administrator does provide a fundamental basis for website analytics and is feasible in the time constraints we have. We place light-weighted and very specific views like an exam statistics page or a lecture statistics page to the end of the working period as these interfaces are less crucial for identifying course problems and can be dropped from the proposed system to keep the schedule.

4.3 Subsystem Decomposition

Next, we want to distribute the proposed system into its subsystem and further describe them. A subsystem as a part of the system encapsulates the behaviour and status of the classes lying within and has well-defined services available for other subsystems. The goal is to create a system architecture where subsystem are loosely coupled and replaceable in order to allow changes [BD09]. As already mentioned in Section 4.1 and also displayed in Figure 4.1, Artemis uses a client server architecture that decomposes the system into a client subsystem and a server subsystem, which communicate with each other via REST APIs, pictured in the diagram as a "lollipop" between server and client. Based on Figure 4.1, we will elaborate on these subsystems in more detail in the next subsections.

Server Subsystem Decomposition

Server-sided, new statistics pages need a corresponding API which offers the possibility to fetch statistics from the server. As we do not have such a

service yet, we introduce a new *statistics* subsystem on the server. This subsystem provides a *statistics service*, which the client components use to collect information about the metrics. The statistics subsystem uses other server components like the *course* and *exercise* subsystem for data collection. Since we stated in the nonfunctional requirements that for privacy reasons we do not send user related data to the client, we compute the data needed for a graph on the server and only send back the raw number which the graph is going to display without any additional information. This also saves bandwidth and therefore shortens the time until metrics are shown to the user.

For the *course management overview* and for the *course management detail view* we need basic as well as advanced course information, which is fetched from the *course service* API. This subsystem already exists and we will only extend it by some further functionality regarding our special use case. Furthermore, the course subsystem depends on the server internal exercise interfaces.

The just mentioned *exercise* subsystem handles exercise specific inquiries. In this case, we want to gather information regarding the exercises in the exercise list. The exercise subsystem does not communicate with the newly implemented client features directly through APIs, but rather collects information for the course subsystem which then forwards this information to the client.

Client Subsystem Decomposition

For the client-side system decomposition we first introduce a new view - the *statistics view*. As the statistics for course, exercise, exam, lecture and the global user statistics basically all share the same purpose and have a similar underlying structure we generalize them into one subsystem. In this context, we build a chart that is so universal that every statistics component of the subsystem can use it. The different components fetch the content for the graphs through the *statistics service* REST API provided by the server.

The *course management detail view* already exists in the current system and is therefore only refactored. In the proposed system it consists of 2 major subcomponents, the *exercise list* and the *course management detail statistics*. It requests the needed information from the server via the *course service* API. The course management detail statistics contain 2 subcomponents which visualize metrics regarding the status of the course, an active users chart and a doughnut chart, which are omitted for reasons of clarity and comprehensibility. Furthermore, while the data Artemis displays with the doughnut charts is provided by the course management detail view subsystem, the ac-

tive users chart fetches requested data on its own as the information can change based on user input. The possibility to access the course statistics creates a dependency between *course management detail view* and *statistics view*.

Similar to the course management detail view, the *course management overview* exists in the current system and is getting refactored. A newly added subsystem of it is the *course tile*, which represents a course on the page. Every tile in the course management overview consists of an exercise list and an active users chart, which are therefore subsystems of a *course tile* but omitted for clarity reasons. The data displayed in this view is fetched once when accessing the page by requesting the *course service* REST API. As there is the possibility to navigate into the course statistics and course management detail view from the overview, we create a dependency between the *course management overview* and the *statistics view* and a dependency between the *course management overview* and the *course management detail view*.

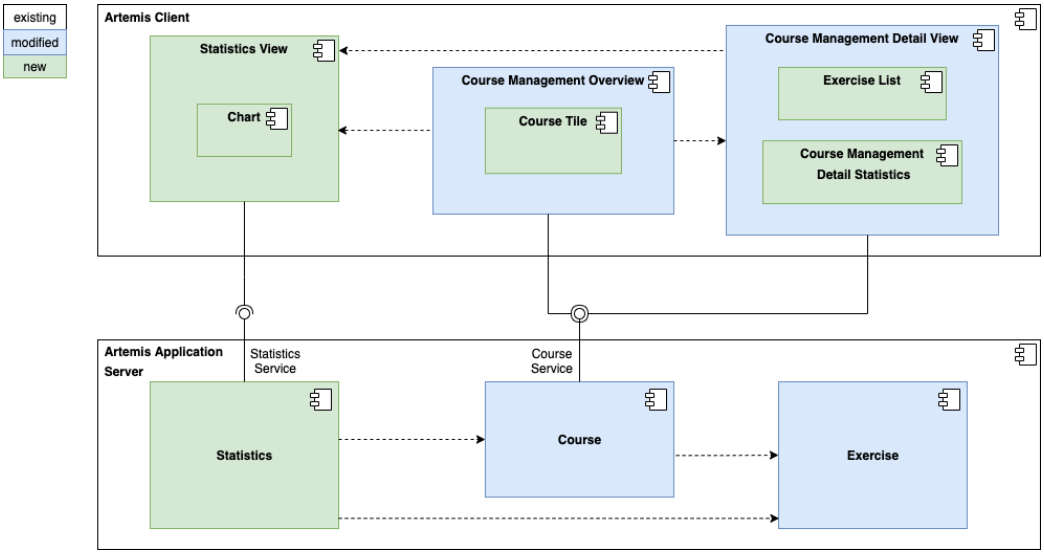


Figure 4.1: Subsystem Decomposition of the proposed system

Chapter 5

Object Design

In this chapter, we want to close the gap between the analysis and the system design by refining solution domain objects introduced in Chapter 4 and elaborating on specific implementation details of the proposed system.

5.1 Statistics

Figure 5.1 shows a class diagram, which models the actual implementation of the statistics into the proposed system. Since the statistics views all have similar visualizations and behavior towards the user, we take advantage of this and implement the graph in such a way that each component can reuse it. For this reason, the *statistics view* classes contains only very little information. The *user statistics view*, for instance, has only 3 tasks.

The first one is to store the metrics we want to display to the user. For this purpose, we introduce the new enumeration *Metric* which contains every metric available, independent of the view. We representatively model 6 possible values in Figure 5.1. We pick the metrics we want to display in the view from the *Metric* enumeration and store it in an array in the *StatisticsView*.

Second, we define the timespan, called *currentSpan*, which is managed globally for the whole view. For this value, we add the new enumeration *SpanType* that can be either *Day*, *Week*, *Month*, *Quarter* or *Year*. Per default, *currentSpan* is set to *Week* for every statistics view. When the user chooses a different span, the method *onTabChanged()* handles the change request.

Third, we define the current *view* the statistics are in. We use it to tell the graph component which data should be fetched from the server. Therefore, values for *view* can either be *Artemis*, *Course*, *Exercise*, *Exam* or *Lecture*.

An additional value in any statistics page except the user statistics is the ID of the entity the statistics belong to, for instance, the lectureId for the *lecture statistics view*.

For the chart invocation, we create a graph for each element stored in the *metrics* array of the *statisticsView* instance. As parameters, we forward the information the graph needs to know:

1. *Metric*: Tells the graph which metric will be shown.
2. *CurrentSpan*: Tells the graph which timespan is currently chosen.
3. *View*: Tells the graph in which statistics view it is in. This is needed when the graph must fetch new data due to user input like timespan changes.
4. *EntityId*: An entityId is the id of the course, exercise, lecture or exam the statistics are referring to. The graph uses the id to fetch new data from the server. The entityId of a graph located in the user statistics is not set.

As described in Section 3.4.4, a graph itself consists of several units. The arrows besides the chart are handled by each graph individually and are internally indicated by the *periodIndex*. Since you can theoretically do an infinite number of clicks onto an arrow, we handle it by assigning an Integer to it. Therefore, the *periodIndex* describes the deviation from the starting point. The method *switchTimeSpan()* takes care of the user input in terms of arrow clicks and updates the graph accordingly.

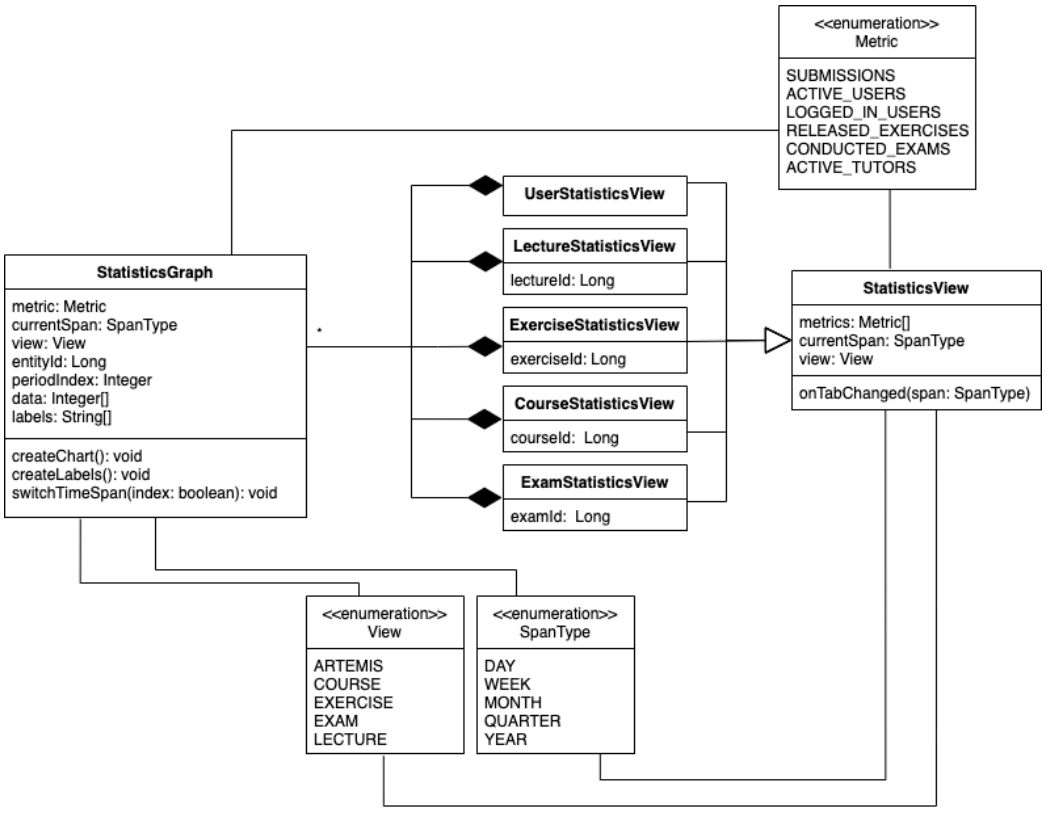


Figure 5.1: Class diagram showing implementation details of the proposed system regarding the statistics views

Chapter 6

Summary

This chapter concludes the thesis by presenting the current status of the proposed system and its realized goals in Section 6.1.1, as well as the unrealized goals in Section 6.1.2. In Section 6.2 we summarize the results of this thesis and address how teaching analytics could be further improved by future work in Section 6.3.

6.1 Status

In this section, we match the current status of the system to the requirements stated in Section 3.3. We divide the requirements into the following three categories:

- ● **Implemented**: The requirement is completely implemented.
- ◐ **Partially Implemented**: The requirement is only partially implemented, and further work is needed.
- ○ **Not implemented**: The implementation of the requirement has not been started.

6.1.1 Realized Goals

Table 6.1 gives an overview of the currently implemented and not implemented functional requirements. Table 6.2 lists all nonfunctional requirements and states whether or not we managed to implement them.

Functional Requirement	Status
Statistics	
FR1 Open global Artemis statistics	●
FR2 Open course statistics	●
FR3 Access exercise statistics	●
FR4 Open exam statistics	●
FR5 Access lecture statistics	○
Course management	
FR6 View active users	●
FR7 View exercises	●
FR8 View exercise progression	●
FR9 View average score of exercise	●
Course management overview	
FR10 View every accessible course	●
Course management detail view	
FR11 Open tutor statistics	●
FR12 View average score in course	●
FR13 Search for exercise	●

Table 6.1: Overview of the implementation status of the functional requirements (● fully implemented, ◐ partially implemented, ○ not implemented)

Non-functional Requirement	Status
Statistics	
NFR1 Performance - Response time	●
NFR2 Supportability - Extensibility	●
NFR3 Reliability - Security	●
Course management overview	
NFR4 Usability - Efficiency	●
NFR5 Performance - Response time	●
NFR6 Usability - Learnability	●
NFR7 Supportability - Adaptability	●
Course management detail view	
NFR8 Performance - Response time	●

Table 6.2: Overview of the implementation status of the nonfunctional requirements (● fully implemented, ◐ partially implemented, ○ not implemented)

We implemented 20 of the 21 requirements which we announced in Section 3.3. We managed to integrate teaching analytics in several locations

into Artemis. Administrators now have the possibility to access user statistics (FR1), giving an analytical overview of Artemis. Instructors can open course statistics, which present metrics regarding various aspects of the course content (FR2) as well as exercise and exam statistics (FR3, FR4).

Regarding the course management, we accomplished to improve the functionality of different course management pages. We managed to provide the tutors and instructors with information on how many users actively participated in the course (FR6), which Artemis extends with the possibility to adapt time constraints in the course management detail page. Tutors and instructors are able to view a list of exercises (FR7) and can have a look at the exercise participations, the exercise assessment progress and the average score of those exercises (FR8, FR9). We also accomplished the additional requirement elicited for the course management overview page, where tutors and instructors can view every course they have access to in a tile below each other (FR10).

The course management detail page is implemented and meets its full requirements. Tutors can view their progress concerning their responsibilities in the course. Artemis displays the average student score, the assessment progress and the number of complaints and more feedback requests, as well as how many of those are already addressed and how many are still unevaluated (FR11, FR12). As an additional functionality regarding the list of exercises, tutors can search for specific exercises from the course (FR13).

For the 4 out of 5 statistics pages we managed to realize all of the 8 nonfunctional requirements as seen in Table 6.2.

We hold the response restrictions when accessing statistics pages (NFR1). Through the GraphType implementation, developers can easily add additional metrics in the future by adding a value to the enumeration (NFR2). In terms of security, we have managed to restrict all computations to the server, so that no information about student performances is passed to the client (NFR3).

We successfully implemented all nonfunctional requirements concerning the course management overview. A tutor is able to access the course exercises within 2 interactions by using the course management overview (NFR4). We also remain under the response time constraint which we specified of 2 seconds when accessing the course management overview (NFR5). We fulfill nonfunctional requirement 6 and 7 (NFR6, NFR7) in the given time as we provide a tooltip for every button without a title and an interface that is adaptable to the screen size.

In terms of nonfunctional requirements of the course management detail view, we achieved the performance requirement (NFR 8) as we currently provide a fast execution time of the implemented functionality.

6.1.2 Open Goals

In this subsection, we discuss the open goals that we did not manage to incorporate into the system. Due to time reasons, we failed to fulfill 1 out of 21 requirements. The proposed system does not show specific lecture statistics (FR5). As lectures currently do not provide a reasonable amount of data which we could use to analyze its usage, we would need to spend more time creating new metadata, which is not feasible in our time constraint.

6.2 Conclusion

With this work, we have integrated some major teaching analytics aspects into Artemis. Previously, teachers lacked analytical functionalities that would allow them to detect weaknesses of their lecture or exercises. An instructor is now able to open statistics for several different course components and draw conclusions from them. We provide administrators with Artemis-wide analytics which can be used for server surveillance.

The new course management overview eases and enriches the workflow of tutors and instructors and offers early course analytics, which was not available before. Through the re-worked detail page, we are able to add more value to it so tutors and instructors utilize it more often in the future.

With the innovations we provide, instructors should be able to improve the lecture and exercise content in a way that allows students to work more intensively on tasks which they have problems with.

6.3 Future Work

As we introduce teaching analytics to Artemis, we managed to build an analytical foundation for upcoming work. However, there are a lot of other opportunities to extend the teaching analytics capabilities of Artemis:

Introduce new metadata

In this thesis, we are relatively limited by the metadata which is already created beforehand. A next step for further teaching analytics integration would be to expand data generation in Artemis, like, for instance, for lecture interactions: Currently, lectures do not create much data in terms of student usage. There is no data generated on how often a lecture PDF is downloaded or how often student click on embedded links. Furthermore, including the possibility of lecture livestreams into Artemis would integrate exercises into lecturing even better. An instructor could have the ability to directly append in-class exercises which are displayed to the viewer through

a pop-up notification, like it is currently possible with quizzes. Through this, we could display live exercise participation analytics and also identify live viewer statistics regarding the lecture livestream. Instructors could then analyze at which points students typically leave lectures.

Instructor alerts

Artemis could have the possibility to alert instructors on specific events. These would include when average exercise grades fall below an threshold. Instructors could then specify an average score threshold and if average exercise results are below the boundary after the assessment due date, Artemis automatically notifies instructors about this event. Through this innovation, we prevent the possibility of instructors overlooking bad exercise results.

Exam live analytics

Instructors currently do not have much control and hardly any insight when exams are conducted. To give instructors a certain overview on what is currently happening in their exam, a live exam view could support instructors. In this interface, instructors would have the possibility to view metrics such as the number of already submitted exams, exam participations and which exercise group students work on at that moment.

Integration of Machine Learning into Artemis

Using Machine Learning (ML) best practices to further improve Artemis' teaching and student results is another potential area of improvement [VCRCPP20, KJÐ18]. A use case adjusted to the Artemis system would be to guide tutor assessments as discussed in Section 3.4.1. Furthermore, a ML algorithm could support instructors in their work and adapt exercises according to the students' strengths and weaknesses. The algorithm could identify a bad average exercise score and automatically suggest exercises to the instructor which help students in their understanding of the problematic content. For instance, if a programming exercise contains an interface implementation which the students only rarely succeeded in, Artemis should detect this weakness and propose an exercise to the instructors that addresses this particular aspect.

List of Figures

1.1 Scenario where students achieve a bad average score in one of three exercises 4

2.1 The Weekly Learner Engagement of a course in edX Insights . 8
2.2 Video views grouped by course sections in edX Insights 9
2.3 Video frames which are viewed more than once, highlighted in dark blue[1] 10
2.4 Number of video views slowly declining as students stop to watch the lecture[2] 11
2.5 Submissions analytics of a text input problem in edX[3] 12
2.6 The course average score in canvas[4] 13
2.7 Course Analytics in Canvas[5] 15

3.1 Instructor Course Dashboard in the current Artemis version . 17
3.2 Instructor Exercise Dashboard in the current Artemis version . 18
3.3 The course management overview in the current system 18
3.4 Current course management detail page as instructor 19
3.5 Use case diagram of the statistics pages in the proposed system 26
3.6 Use case diagram of the course management 27
3.7 Analysis object model of the proposed system 29
3.8 The user statistics page showing the amount of submission, active users and logged-in users, each in different weeks 32
3.9 The distribution of submissions made throughout a day in Artemis 32
3.10 The number of active users during December 2020 32
3.11 The number of conducted exams during the exam phase of WS 20/21 32
3.12 The number of distinct active tutors from April 2020 until March 2021 33

3.13 Additional metrics for the course statistics page. The blue line indicates the average course score while the bars present average scores of particular exercises 33
3.14 A tile representing a course in the Course management overview 35
3.15 Proposed course management detail page with advanced metrics 37

4.1 Subsystem Decomposition of the proposed system 42

5.1 Class diagram showing implementation details of the proposed system regarding the statistics views 45

List of Tables

6.1 Overview of the implementation status of the functional requirements (● fully implemented, ◐ partially implemented, ○ not implemented) 47
6.2 Overview of the implementation status of the nonfunctional requirements (● fully implemented, ◐ partially implemented, ○ not implemented) 47

Bibliography

[Bar10] Scott Barber. How fast does a website need to be?, 2010.

[BD09] Bernd Bruegge and Allen H Dutoit. Object–oriented software engineering. using uml, patterns, and java. *Learning*, 5(6):7, 2009.

[BDDS02] Pieter Bekaert, Geert Delanote, Frank Devos, and Eric Steegmans. Specialization/generalization in object-oriented analysis: Strengthening and multiple partitioning. In *International Conference on Object-Oriented Information Systems*, pages 34–43. Springer, 2002.

[HPM+20] Raza Hasan, Sellappan Palaniappan, Salman Mahmood, Ali Abbas, Kamal Uddin Sarker, and Mian Usman Sattar. Predicting student performance in higher educational institutions using video learning analytics and data mining techniques. *Applied Sciences*, 10(11):3894, 2020.

[Iss18] Moritz Issig. Development of an interactive live quiz component with instant statistics in artemis. 2018.

[JBR99] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley Longman Publishing Co., Inc., USA, 1999.

[KJĐ18] Danijel Kučak, Vedran Juričić, and Goran Đambić. Machine learning in education-a survey of current research trends. *Annals of DAAAM & Proceedings*, 29, 2018.

[KS18] Stephan Krusche and Andreas Seitz. Artemis: An automatic assessment management system for interactive learning. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 284–289, 2018.

[KTKK12] Carola Kruse, Thanh-Thu Phan Tan, A. Koesling, and M. Krüger. Strategies of lms implementation at german universities. 2012.

[KvFA17] Stephan Krusche, Nadine von Frankenberg, and Sami Afifi. Experiences of a software engineering course based on interactive learning. In *SEUH*, pages 32–40, 2017.

[Le16] Hong Le. Interactive computer science exercises in edx. 2016.

[OC12] Martin M Olmos and Linda Corrin. Learning analytics: A case study of the process of design of visualizations. 2012.

[PSDJ16] Luis P Prieto, Kshitij Sharma, Pierre Dillenbourg, and María Jesús. Teaching analytics: towards automatic extraction of orchestration graphs using wearable sensors. In *Proceedings of the sixth international conference on learning analytics & knowledge*, pages 148–157, 2016.

[Pur07] Patrick Purcell. Engineering student attendance at lectures: Effect on examination performance. In *International conference on engineering education–ICEE 2007*, volume 2008, pages 699–717, 2007.

[Sie13] George Siemens. Learning analytics: The emergence of a discipline. *American Behavioral Scientist*, 57(10):1380–1400, 2013.

[SL11] George Siemens and Phil Long. Penetrating the fog: Analytics in learning and education. *EDUCAUSE review*, 46(5):30, 2011.

[TM18] Sarah Taylor and Pablo Munguia. Towards a data archiving solution for learning analytics. In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*, pages 260–264, 2018.

[VCRCPP20] William Villegas-Ch, Milton Román-Cañizares, and Xavier Palacios-Pacheco. Improvement of an online education model with the integration of machine learning and data analysis in an lms. *Applied Sciences*, 10(15):5371, 2020.

[Wal21] Stefan Waldhauser. Integration of learning analytics into artemis. 2021.

[Won17] Billy Tak Ming Wong. Learning analytics in higher education: an analysis of case studies. *Asian Association of Open Universities Journal*, 2017.