FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

Integration of Learning Analytics into Artemis

Stefan Waldhauser

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

Integration of Learning Analytics into Artemis

Integration von "Learning Analytics" in Artemis

Author:	Stefan Waldhauser
Supervisor:	Prof. Dr. Bernd Brügge
Advisor:	Dr. Stephan Krusche
Date:	26.03.2021

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

Munich, 26.03.2021 Stefan Waldhauser

Abstract

Learning analytics is a research field that tries to support the learning process by collecting and comparing data about learners and their learning contexts. Today, many learning management systems use learning analytics techniques because of the benefits they offer both instructors and students.

This thesis addresses several learning analytics related shortcomings of Artemis: Students need information about other students' performance as a guide to interpret their own performance. Although the data is available in the system, Artemis does not currently provide students with such a means of comparison. Artemis also does not support learning goals, i.e. competencies that students should master when completing a course. The implementation of learning analytics is hindered by the fact that large parts of the learning process take place outside Artemis, as it is currently not possible to include, for example, lecture recordings on the platform.

To address these shortcomings, we expanded Artemis in three areas: First, the lecture concept has been redesigned. Videos, notes, files, and exercises can be directly integrated into a lecture. Second, learning goals can be defined and linked to relevant learning material. The system calculates a student's progress towards goal mastery. Third, a learning analytics dashboard contains visualizations that provide students with the necessary context to interpret their performance in a course. The redesigned lectures and learning goals were used in the course *Patterns in Software Engineering*. 538 students actively participated in the course. It contains 5 learning goals and 12 lectures, with a total of 50 videos, 51 exercises, and 53 files. The new Artemis capabilities received positive feedback from students and instructors.

Zusammenfassung

Learning Analytics ist ein Forschungsgebiet, das versucht, den Lernprozess durch das Sammeln und Vergleichen von Daten über Lernende und deren Lernkontexte zu unterstützen. Heutzutage verwenden viele Lernmanagementsysteme Learning-Analytics-Techniken, weil sie sowohl den Lehrenden als auch den Lernenden Vorteile bieten.

Diese Arbeit befasst sich mit mehreren auf Learning Analytics bezogenen Unzulänglichkeiten von Artemis: Studierende benötigen Informationen über die Leistungen anderer Studierender als Orientierungshilfe, um ihre eigenen Leistungen zu interpretieren. Obwohl die Daten im System verfügbar sind, bietet Artemis den Studenten derzeit keine solche Vergleichsmöglichkeit. Artemis unterstützt auch keine Lernziele, d.h. Kompetenzen, die Studierende nach Abschluss eines Kurses beherrschen sollten. Die Implementierung von Learning Analytics wird dadurch erschwert, dass große Teile des Lernprozesses außerhalb von Artemis stattfinden, da es derzeit nicht möglich ist, z.B. Vorlesungsaufzeichnungen in die Plattform einzubinden.

Um diese Unzulänglichkeiten zu beheben, haben wir Artemis in drei Bereichen erweitert: Erstens wurde das Vorlesungskonzept neu gestaltet. Videos, Notizen, Dateien und Übungen können direkt in eine Vorlesung integriert werden. Zweitens können Lernziele definiert und mit relevantem Lernmaterial verknüpft werden. Das System errechnet den Fortschritt eines Studenten bei der Zielerreichung. Drittens enthält ein Learning-Analytics-Dashboard Visualisierungen, die den Studenten den nötigen Kontext bieten, um ihre Leistung in einem Kurs zu interpretieren. Die neu gestalteten Vorlesungen und Lernziele wurden im Kurs *Patterns in Software Engineering* verwendet. 538 Studenten nahmen aktiv an dem Kurs teil. Er enthält 5 Lernziele und 12 Vorlesungen, mit insgesamt 50 Videos, 51 Übungen und 53 Dateien. Die neuen Artemis-Funktionen erhielten positives Feedback von Studenten und Dozenten.

Contents

API	Application Programming Interface
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
LA	Learning Analytics
REST	Representational State Transfer
UML	Unified Modeling Language
URPS	Usability Reliability Performance Supportability

Chapter 1

Introduction

The nowadays commonly accepted definition of learning analytics (LA) was first offered at the 1st International Conference on Learning Analytics[1]:

> *"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."*

Compared to established analytical fields like business intelligence, LA is a relatively new discipline [Sie13]. For a long time, learning was difficult to analyze as students in traditional learning environments commonly left no central digital trail. They might research online, watch tutorial videos, or ask for help in a discussion forum, but the collected data is fragmented across multiple platforms and thus not easily accessible for analytics.

This changed with the wide adoption of learning management systems (LMSs) by higher learning institutions [Sie13] [CJB05]. Today, nearly every university in Germany uses one or more LMSs to support their courses [KTKK12]. These mostly web-based software applications deliver and manage all types of learning content, including lecture recordings, assignments, and lecture slides. Advanced LMSs allow students to solve different types of exercises directly within the application.

Data is generated, logged, and aggregated when a student uses a LMS. Depending on the specific system, the collected data may include navigational patterns, click patterns, time spent, social interactions, document and tool usage, artifacts produced and exercise results. This large amount of behavioral data can then be analyzed by the system and is often presented to the users in so-called learning analytics dashboards. Schwendimann et al. defined them as:

[1]`https://www.solaresearch.org/about/what-is-learning-analytics/`

"A learning dashboard is a single display that aggregates different indicators about learner (s), learning process(es) and/or learning context(s) into one or multiple visualizations." [SRTV+16]

Artemis[2] is an open-source LMS developed originally at the Chair for Applied Software Engineering at the Technical University of Munich [KS18]. Today it is used by several higher learning institutions in Germany and Austria.

Instructors can create courses with associated learning resources (e.g. lecture slides) and setup text, modeling, file-upload and programming exercises. Students can solve these exercises directly within the application. This allows instructors to use an interactive learning approach in their courses [KvFA17]. In this approach, lectures and exercises are combined into interactive classes to encourage active student participation. The subsequent correction of student submissions by teaching assistants is also managed in the application. Since 2020, Artemis also supports the conduction of graded online exams.

1.1 Problem

The lack of important frames of reference for students, and the inflexible lecture design are learning analytics related shortcomings of Artemis that are addressed in this thesis.

Missing Frames of Reference for Students

A frame of reference is a comparison point used by students when evaluating their performance in a course [Wis14]. Several kinds are possible and beneficial as a motivational tool to different types of student:

- **Performance-orientation**: Students compare their performance to the rest of the class: *"How am I doing compared to the rest of the class?"*

- **Mastery-orientation**: Students view their performance in the context of skill mastery: *"Have I mastered all the skills I should learn in this course?"*

- **Self-orientation**: Students compare their current performance to their performance earlier in the class: *"Have I improved myself?"*

[2] `https://github.com/ls1intum/Artemis`

The only performance data presented to students in the current Artemis version is their own current score in exercises. As no history of one's own performance is displayed, students find it difficult to use self-orientation. Students are also not able to compare their performance to the performance of other students. Thus, students can also not use performance-orientation. Finally, students can also not use mastery-orientation as Artemis does not support learning goals, i.e. competencies that students should master when completing a course.

Inflexible Lecture Design

The concept of a lecture in the current Artemis version is limited. A lecture has a title, a description and various attached files. In most cases, a lecture in Artemis is used simply as a place to attach lecture slides. Other LMSs are more flexible and allow instructors to provide students with multiple kinds of lecture content. Lecture recordings, notes and related exercises can be embedded directly on the lecture page. A good example of an LMS that offers more flexibility is edX[3], presented in Section 2.1. Since this is currently not possible in Artemis, content is often scattered across multiple websites. For example, the lecture recordings can be found on a video sharing platform, the slides on the chair website and the exercises in Artemis.

1.2 Motivation

In the following section, we explain our motivation to tackle the described shortcomings of Artemis.

Providing Multiple Frames of Reference to Students

Several studies in the field of educational research confirm that some students are motivated by performance or self-orientation while others are motivated by mastery-orientation [Pin00]:

Kim et al. [KJP16] and Corrin et al. [CDB15] have shown that being compared with the class average had a positive motivational effect on students below the average but no effect on highly-performing students. Contrasting results were obtained by Tan et al. [TYKJ16]: Highly-performing students were even more motivated by peer comparison, but low-performing students felt demoralized. Instead, these students preferred when their performance was compared to their performance earlier in the course or viewed in the

[3]`https://www.edx.org/`

context of skill mastery. These contrasting results show that it is important for Artemis to offer performance-orientation, self-orientation and mastery-orientation to the student and let the student pick whats motivates her or him the most.

The possibility of defining learning goals gives students the ability to use mastery-orientation and supports instructors that want to use constructive alignment.

Constructive alignment is a learning concept that focuses on learning goals, i.e. competencies, that students should acquire. First, teachers should define clear and realistic goals. Afterwards, exercises should be developed that can be used to test the mastery of these goals. After that should the actual lectures be designed. The design should ensure that the material supports the students in achieving the defined goals [BT11]. Various studies describe the positive effects of constructive alignment on learning outcome. A good overview can be found in [Kan14].

Increasing Lecture Design Flexibility

One advantage of a more flexible lecture design is that the more learning content can be made available in Artemis, the more precisely the student's learning behavior can be analyzed. For example, resource access (e.g. number of downloads, video view count) is a common metric for how engaged students are.

Embedding various content in a lecture also allows an instructor to divide a long lecture into smaller thematically related chunks. For example, a chunk could consist of a 15 minutes to 30 minutes video, the lecture notes and a matching exercise. A lecture of several hours can be presented in a more digestible way. This can also improve learning according to the cognitive theory of multimedia learning [May05]. The positive effect of short videos is also supported by studies such as one from Guo et al. [GKR14]. They showed that students find shorter videos much more engaging than long videos. Splitting up a lecture into small parts also supports the concept of interactive learning. A course design concept that has been shown to increase student participation by delivering small chunks of content and exercises in short cycles [KSBB17].

1.3 Objectives

The following section describes how we address the problems described in Section 1.1.

The first objective is to provide students with all three motivating frames of reference. This is achieved via two visualizations in a learning analytics dashboard:

The first visualization allows students to use performance-orientation, and self-orientation for motivation. Figure 1.1 shows a mock-up. The horizontal axis shows the various assignments ordered by release date. The vertical axis shows the achieved points. By displaying both the student's individual performance (blue line), the class average (green line), and the best performance in the class (dotted line), students can be motivated by self-orientation and performance-orientation.

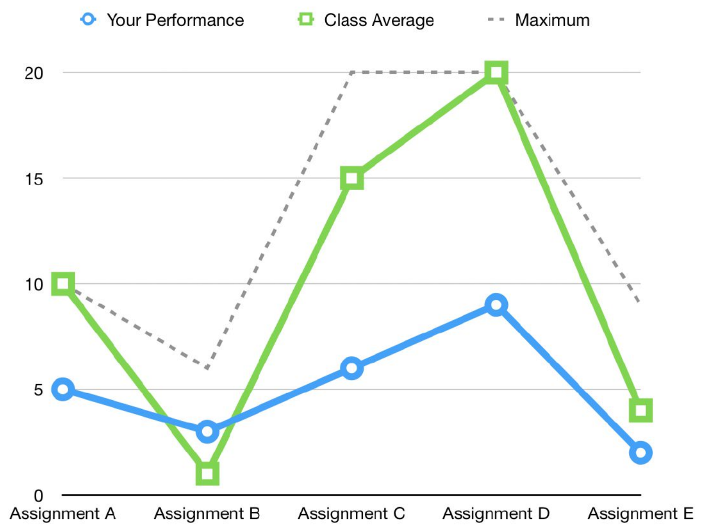


Figure 1.1: Mock-up of a visualization that gives students the ability to use performance and self-orientation

The second visualization allows students to use mastery orientation. This requires the introduction of learning goals in Artemis. Instructors can define learning goals, i.e. competencies, that students should master. The system will track a student's progress in completing those goals. Students can use a visualization to check their learning goal progress. Figure 1.2 shows a mock-up. The color and fill level of the progress bars represent the achieved progress.

The second objective is the lecture redesign. Instructors can embed notes, lecture recordings, relevant exercises and files in a lecture. More of the learning process can happen inside Artemis.

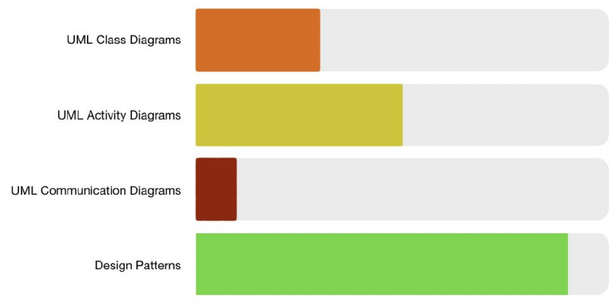


Figure 1.2: Mock-up of a visualization that gives students the ability to use mastery-orientation

1.4 Outline

This section provides an overview of the structure of this thesis:

Chapter 2 Related Work describes other systems that serve as sources of inspiration for the implementation in Artemis.

Chapter 3 Requirements Analysis is based on the requirements analysis document template described in [BD09]. We describe the current status of Artemis and the requirements of the proposed system. Also, the user interface of the proposed system is presented.

Chapter 4 System Design gives an overview of the design goals that guide our implementation and describes several implementation aspects.

Chapter 5 Object Design describes in detail how we achieved the performance design goal.

Chapter 6 Summary recaps our work. This chapter also includes a progress overview of our implementation and ideas for the future.

Chapter 2

Related Work

This chapter presents existing systems that serve as a source of inspiration for solving the problems described in Section 1.1.

2.1 edX

edX[1] is a non-profit platform for Massive Open Online Courses (MOOCs), founded in 2012 by the Harvard University and the Massachusetts Institute of Technology. More than 150 universities (including the Technical University of Munich), nonprofit organizations, and corporations offer over 2800 courses on the platform.[2] edX runs on the Open edX open-source software platform.[3]

The platform offers instructors the possibility to add several kinds of content to their MOOCs. Therefore it is a source of inspiration for the redesign of lectures in Artemis.

Figure 2.1 shows on the left side the view of a course in the edX editor (called edX Studio) and on the right side how a student would see this course. A course in edX is divided into sections. Sections (1) are divided into subsections (2), which in turn contain one or more units (3). A unit contains one or more components.

edX offers four basic types of components:

- **Discussion** components allow students to discuss with each other.

[1] `https://www.edx.org/`

[2] `https://www.edx.org/schools-partners`

[3] `https://open.edx.org/`

[4] Taken from `https://edx.readthedocs.io/projects/open-edx-building-and-running-a-course/en/open-release-juniper.master/developing_course/course_subsections.html`

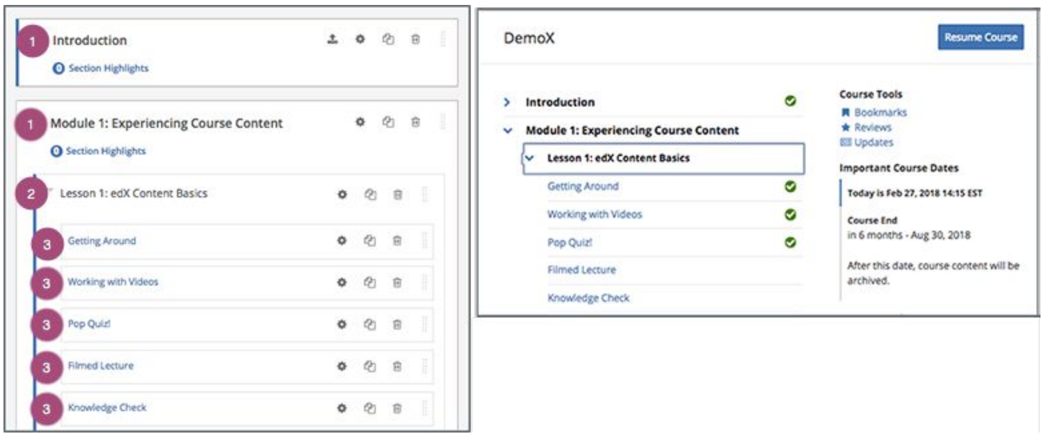


Figure 2.1: The course structure in edX[4]

- **HTML** components allow instructors to add formatted text and images.
- **Problem** components allow instructors to add different types of exercises, from simple multiple-choice quizzes to more complex exercises.
- **Video** components allow instructors to add videos.

Figure 2.2 shows an example of a unit page in edX containing a (1) video component, (2) a text component, (3) a problem component, and (4) a discussion component.

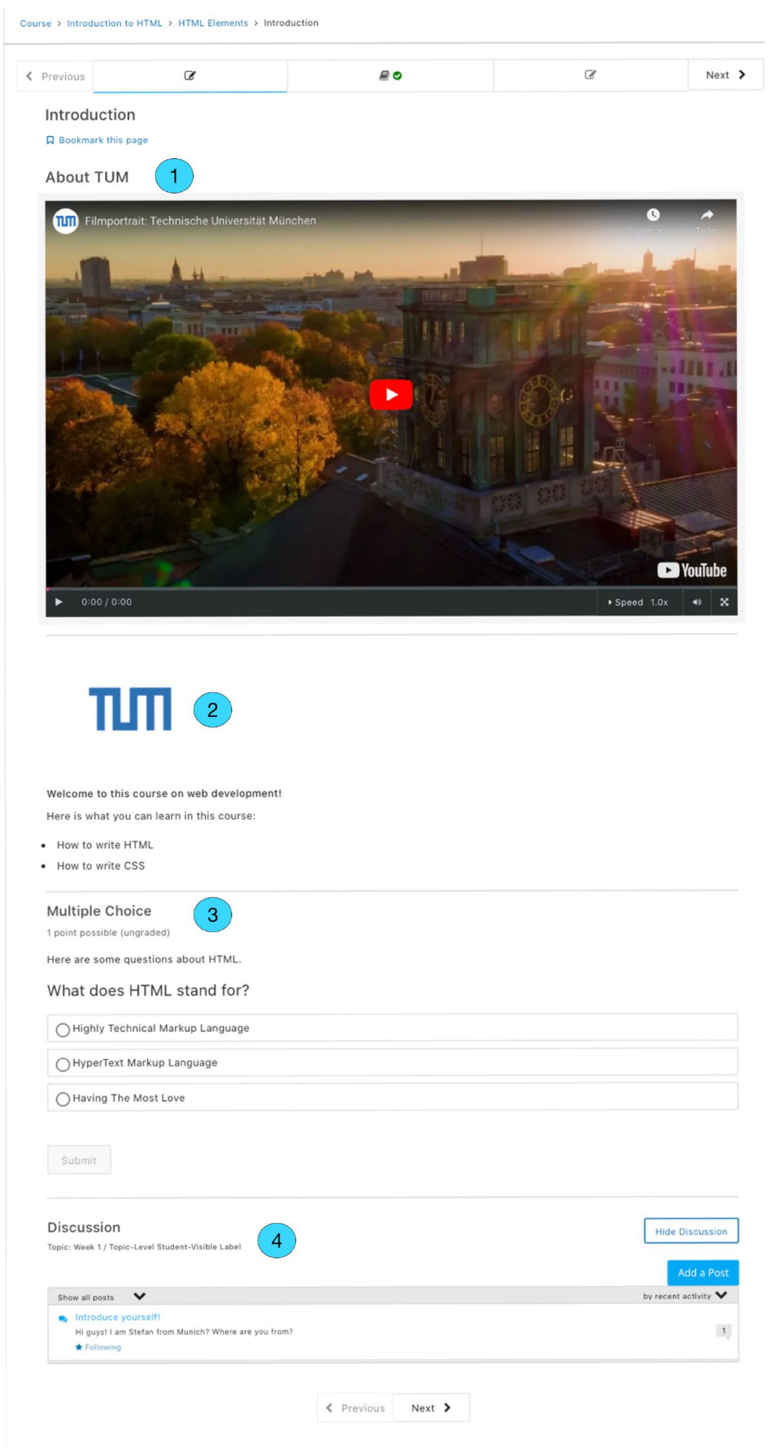
The different components can be added and ordered in the edX Editor.

Figure 2.3 shows how a video component can be created. An instructor has to give the component a title and provide an URL to the uploaded video. The video will then be embedded on the component page.

Figure 2.4 shows how an HTML component can be created. An instructor can enter the text and images using a "What You See Is What You Get" editor or even enter HTML directly. The HTML is rendered when the student opens the component.

Figure 2.5 shows how a simple problem component for a multiple choice quiz is created. Simple exercises can be defined using an edX studio-specific markup language that converts the markup syntax into exercises.

This modular approach allows the instructor to assemble the lecture page as desired. Thus edX can be used for a variety of teaching methods.



Introduction

About TUM

Welcome to this course on web development!

Here is what you can learn in this course:

- How to write HTML
- How to write CSS

Multiple Choice

1 point possible (ungraded)

Here are some questions about HTML.

What does HTML stand for?

- Highly Technical Markup Language
- HyperText Markup Language
- Having The Most Love

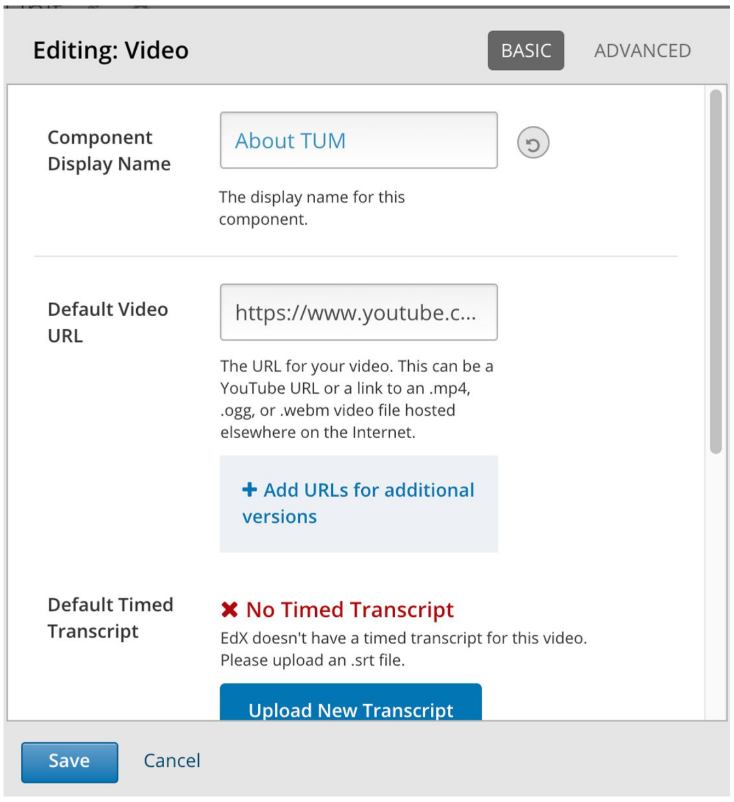
Discussion

Topic: Week 1 / Topic-Level Student-Visible Label

Introduce yourself!

Hi guys! I am Stefan from Munich? Where are you from?

Figure 2.2: Example of a unit page in edX


Editing: Video

Component Display Name: About TUM

The display name for this component.

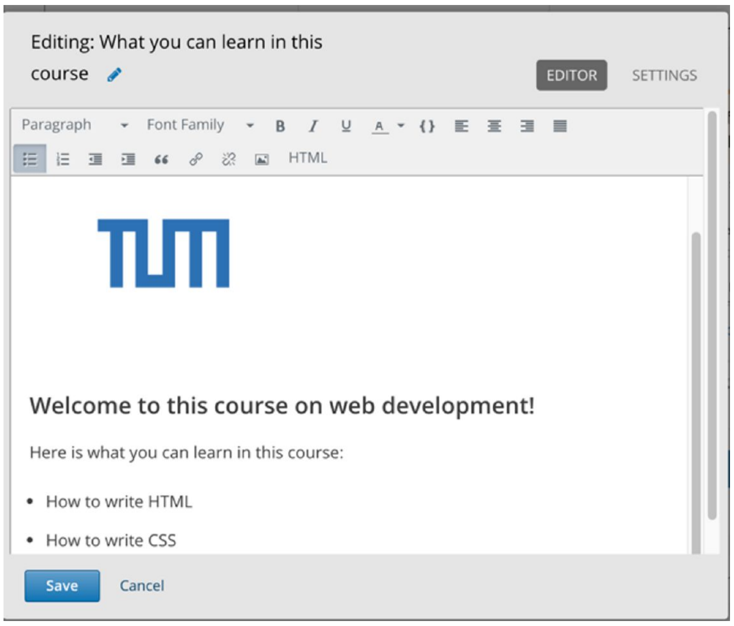
Default Video URL: https://www.youtube.c...

The URL for your video. This can be a YouTube URL or a link to an .mp4, .ogg, or .webm video file hosted elsewhere on the Internet.

Default Timed Transcript: ✖ No Timed Transcript

EdX doesn't have a timed transcript for this video. Please upload an .srt file.

Figure 2.3: Example of the video component editor in edX

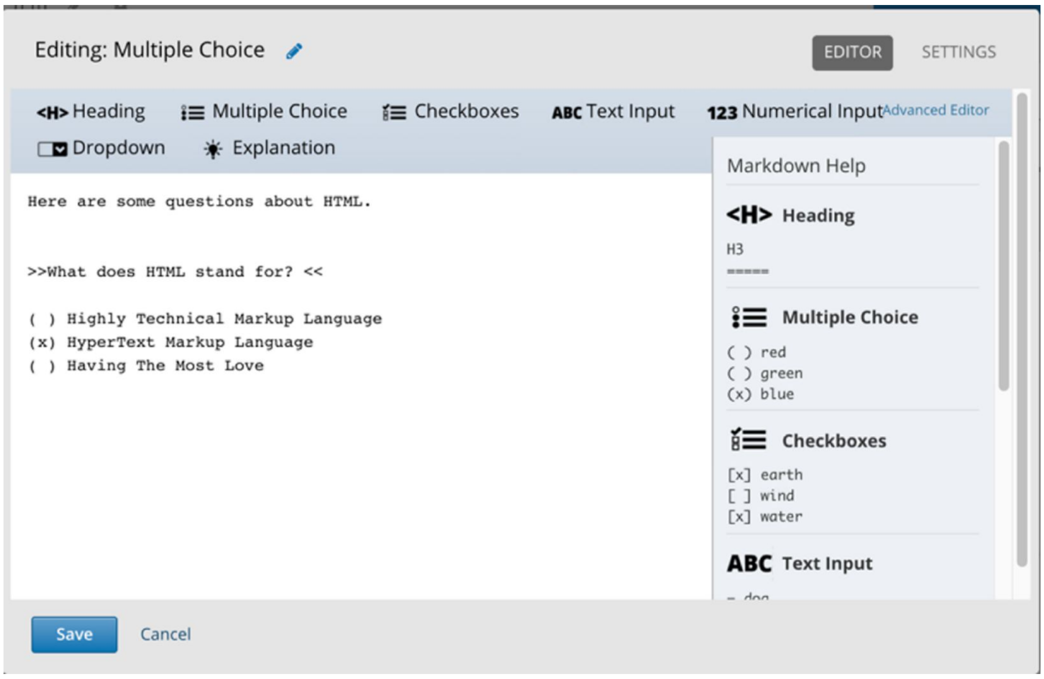

Editing: What you can learn in this course

Welcome to this course on web development!

Here is what you can learn in this course:

- How to write HTML
- How to write CSS

Figure 2.4: Example of the HTML component editor in edX



```
Here are some questions about HTML.


>>What does HTML stand for? <<

( ) Highly Technical Markup Language
(x) HyperText Markup Language
( ) Having The Most Love
```

<H> Heading

```
H3
=====
```

Multiple Choice

```
( ) red
( ) green
(x) blue
```

Checkboxes

```
[x] earth
[ ] wind
[x] water
```

ABC Text Input

Figure 2.5: Example of the multiple choice problem component editor in edX

2.2 Khan Academy

Khan Academy[5] is a non-profit organization founded in 2006 by Salman Khan. The website offers free courses in mathematics, computer science, natural sciences, and humanities in 46 languages. According to its own data[6], the website has more than 115 million registered users and nearly 20 million learners per month. As of 2020, the platform had 429 courses. This includes over 13,000 videos and over 74,000 exercises.[7] Instructors can use Khan Academy for teaching support. They can set up classrooms on the platform. This then allows them to assign Khan Academy courses to their students. Instructors can track their students' progress as they work through the material.

Although Khan Academy is not a learning management system in the classical sense, it is interesting for this thesis because it tries to implement mastery learning through learning analytics and heavy use of gamification. Therefore, it is a source of inspiration for the implementation of mastery orientation and learning goals in Artemis.

Mastery learning is a teaching technique popularized by Benjamin Bloom [Blo68] and closely related to the concept of constructive alignment mentioned in Chapter 1. Mastery learning is focused on personalized, self-paced learning for students. Class time is held constant in traditional teaching, accepting that the material's mastery level will vary among students. Some students will have a good understanding of the material, while others will have gaps in their learning. The goal of mastery learning is to vary the instructional time and achieve a largely constant mastery level of the material among all students. To achieve this, the material is broken down into small units with defined learning goals (competencies that the student should achieve). Students must demonstrate through assignments that they have achieved a sufficient mastery level (usually 80%) before moving on to the next unit. If students do not reach the desired mastery level, they are supported with further assignments, tutoring, and material. This instruct-assess cycle repeats itself until the student achieves mastery in the complete material [Akp20] [Dor20].

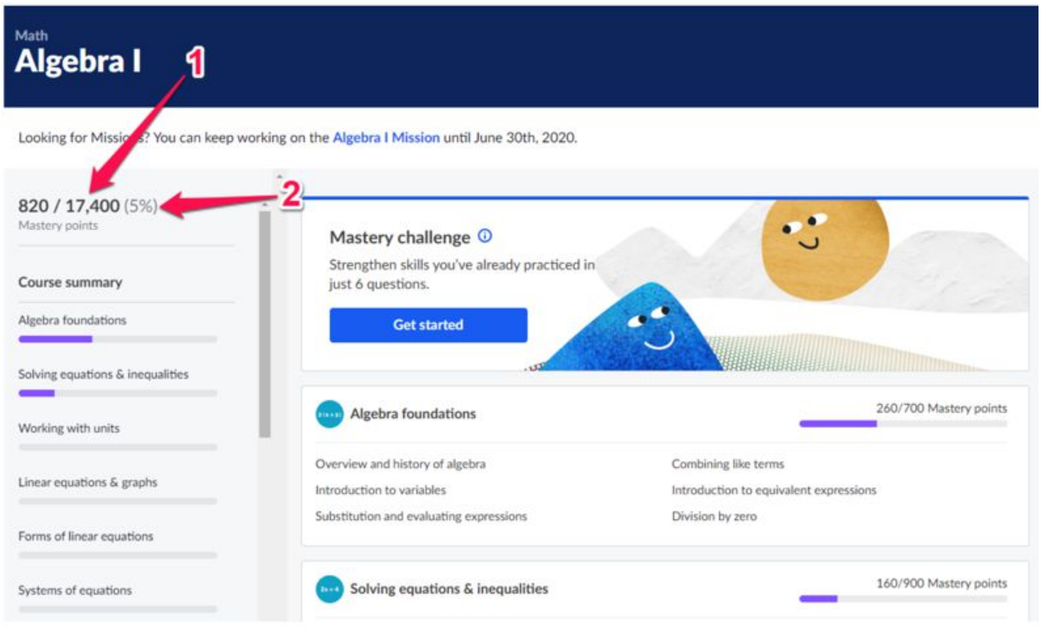
Mastery learning in Khan Academy is implemented using the concept of skills and mastery points. Skills are competencies that students should gain while working through a course unit. As the student practices skills and answers questions in quizzes, unit tests, and course challenges, the level for that skill increases (or decreases if the student answers questions incorrectly).

[5]`https://www.khanacademy.org/`

[6]`https://support.khanacademy.org/hc/en-us/articles/202483630`

[7]`https://khanacademyannualreport.org/`

Mastery points are earned by increasing one's skill level.


Looking for Missions? You can keep working on the Algebra I Mission until June 30th, 2020.

Strengthen skills you've already practiced in just 6 questions.
Math
Algebra I
1
2
820 / 17,400 (5%)
Mastery points
Course summary
Algebra foundations
Solving equations & inequalities
Working with units
Linear equations & graphs
Forms of linear equations
Systems of equations
Mastery challenge
Get started
Algebra foundations
260/700 Mastery points
Overview and history of algebra
Introduction to variables
Substitution and evaluating expressions
Combining like terms
Introduction to equivalent expressions
Division by zero
Solving equations & inequalities
160/900 Mastery points

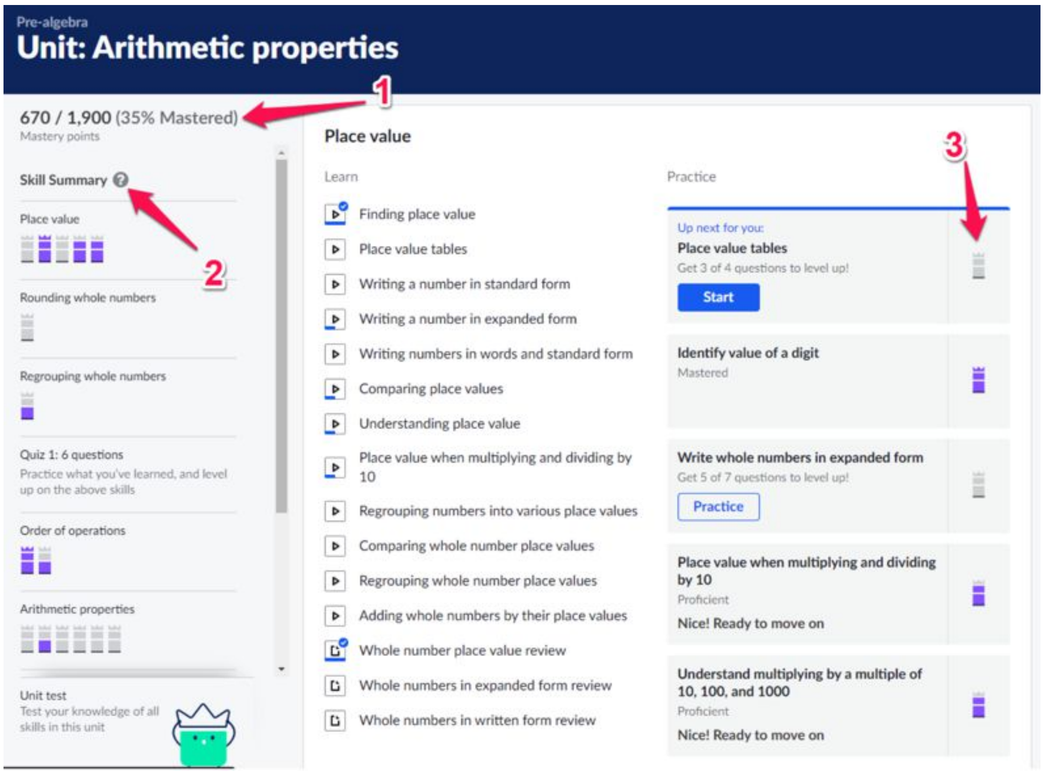
Figure 2.6: A course overview page in Khan Academy[8]

Figure 2.6 shows a course overview page in Khan Academy. The page shows (1) the overall fraction of mastery points that the student achieved and (2) the percentage of the course mastered. Students can see their mastery progress in each unit of the course as a progress bar on the left-hand side. The individual units and their lessons are listed on the right-hand side. A course (e.g., Algebra I) in Khan Academy consists of several units (e.g., Algebra foundations), which consist of several lessons (e.g., Introduction to variables). A lesson, on the other hand, consists of exercises, articles, and videos.

If a student clicks on an individual unit, they are taken to the unit overview page. Figure 2.7 shows a unit overview page in Khan Academy. The page shows (1) the absolute and relative amount of mastery points achieved in this unit, (2) the skill summary, which enables the student to see their progress in skills, and (3) the exercises that allow the student to practice skills. The skill progress is communicated through a crown symbol. The goal of a student is to fill the crown for each skill and thus achieve

[8]Taken from `https://support.khanacademy.org/hc/en-us/articles/115002552631-What-are-Course-and-Unit-Mastery-`

[9]Taken from `https://support.khanacademy.org/hc/en-us/articles/115002552631-What-are-Course-and-Unit-Mastery-`



Pre-algebra

Unit: Arithmetic properties

670 / 1,900 (35% Mastered)
Mastery points

Skill Summary

Place value

Rounding whole numbers

Regrouping whole numbers

Quiz 1: 6 questions
Practice what you've learned, and level up on the above skills

Order of operations

Arithmetic properties

Unit test
Test your knowledge of all skills in this unit

Place value

Learn

- Finding place value
- Place value tables
- Writing a number in standard form
- Writing a number in expanded form
- Writing numbers in words and standard form
- Comparing place values
- Understanding place value
- Place value when multiplying and dividing by 10
- Regrouping numbers into various place values
- Comparing whole number place values
- Regrouping whole number place values
- Adding whole numbers by their place values
- Whole number place value review
- Whole numbers in expanded form review
- Whole numbers in written form review

Practice

Up next for you:
Place value tables
Get 3 of 4 questions to level up!
Start

Identify value of a digit
Mastered

Write whole numbers in expanded form
Get 5 of 7 questions to level up!
Practice

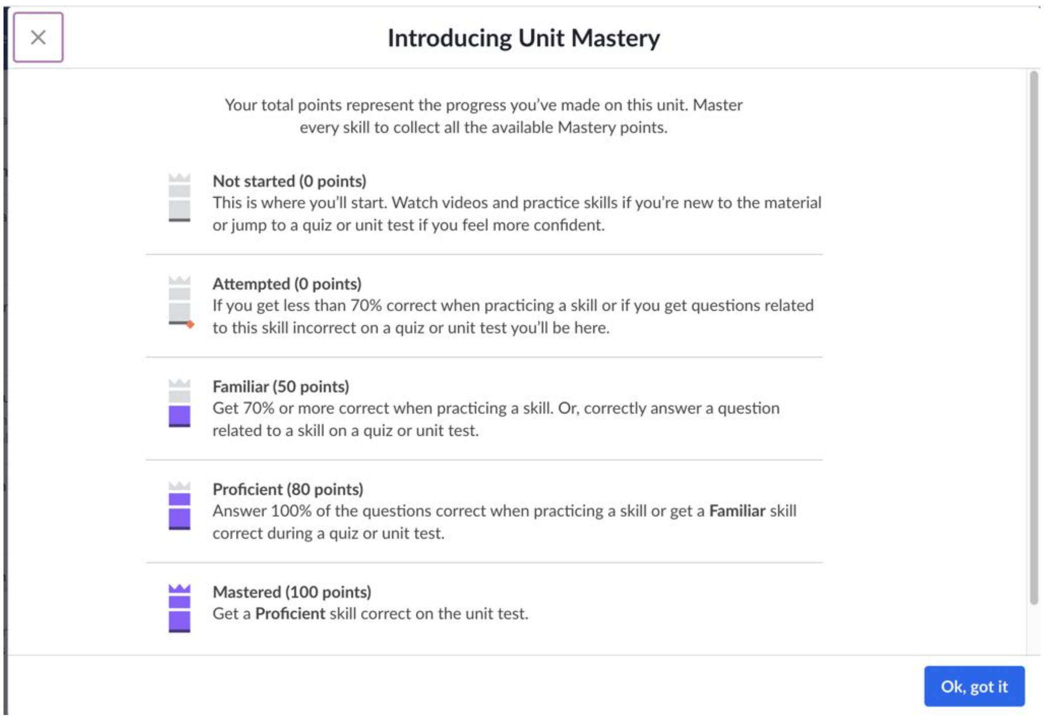
Place value when multiplying and dividing by 10
Proficient
Nice! Ready to move on

Understand multiplying by a multiple of 10, 100, and 1000
Proficient
Nice! Ready to move on

Figure 2.7: A unit overview page in Khan Academy[9]

mastery. The learning material (videos or articles) for each unit are listed in the middle of the page. The student can see his or her progress in working through the material using small progress bars and ticks.

When a student clicks on the question mark next to the skill summary, it is explained how the mastery calculation works. Figure 2.8 shows the explanation Khan Academy gives students about the different skill levels. A student can achieve up to 100 mastery points for each skill.


Introducing Unit Mastery

Your total points represent the progress you've made on this unit. Master every skill to collect all the available Mastery points.

Not started (0 points)
This is where you'll start. Watch videos and practice skills if you're new to the material or jump to a quiz or unit test if you feel more confident.

Attempted (0 points)
If you get less than 70% correct when practicing a skill or if you get questions related to this skill incorrect on a quiz or unit test you'll be here.

Familiar (50 points)
Get 70% or more correct when practicing a skill. Or, correctly answer a question related to a skill on a quiz or unit test.

Proficient (80 points)
Answer 100% of the questions correct when practicing a skill or get a **Familiar** skill correct during a quiz or unit test.

Mastered (100 points)
Get a **Proficient** skill correct on the unit test.

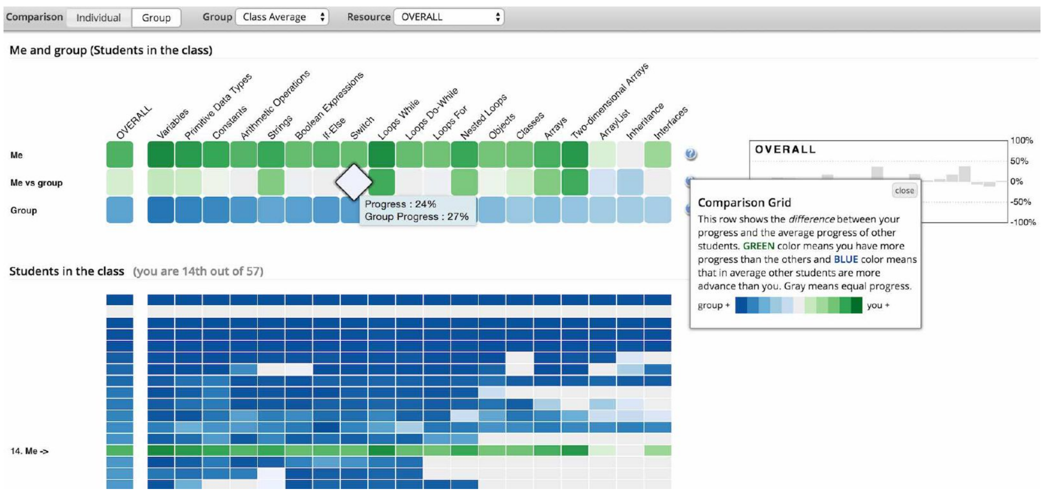
Figure 2.8: An explanation of the skill mastery levels in Khan Academy

2.3 Mastery Grids

An interesting learning visualization approach is Mastery Grids [LGHB14] developed by the Personalized Adaptive Web Systems (PAWS) group at the University of Pittsburgh[10]. Using this visualization, students can use self-orientation, mastery-orientation, and performance-orientation. This visualization is as a source of inspiration for the implementation of the learning analytics dashboard in Artemis.

Figure 2.9 shows a Mastery Grids example. Course content is organized into topics (e.g., Variables), displayed as columns in the grid. The first row labeled ”me” shows the student's topic by topic progress using shades of green color—the darker the color, the higher the progress. The last row labeled ”group” shows the rest of the class's aggregated progress in blue color shades. The second row, labeled ”me vs. group,” presents a differential color that compares the student's progress to the overall class progress. The student has higher progress than the rest of the class, where the second-row cells are green, but the class has higher progress where the second-row cells are

[10]http://adapt2.sis.pitt.edu/wiki/Mastery_Grids_Interface
[11]Taken from http://adapt2.sis.pitt.edu/wiki/Mastery_Grids_Interface



Progress : 24%
Group Progress : 27%

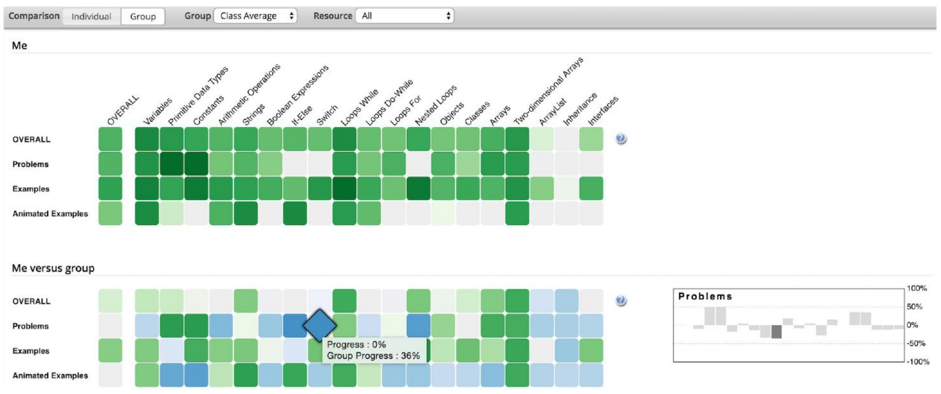
Comparison Grid

This row shows the *difference* between your progress and the average progress of other students. GREEN color means you have more progress than the others and BLUE color means that in average other students are more advance than you. Gray means equal progress.

Figure 2.9: Mastery Grids visualization with overall resources shown[11]

blue. The student has the same progress as the rest of the class in topics with the light gray color. If a student hovers the mouse over a topic cell, the student can see the personal progress and the group's progress in a tool-tip and a neighboring bar chart. The student can also see the ordered individual progress of all students in a second grid at the bottom. The student with the highest progress is at the top of the list, the student with the lowest progress is at the bottom of the list.

Mastery Grids also offers a more detailed view that shows the progress categorized by content type (e.g., Problems, Examples). An example is shown in Figure 2.10.



Progress : 0%
Group Progress : 36%

Figure 2.10: Mastery Grids visualization with all resources shown[12]

[12]Taken from `http://adapt2.sis.pitt.edu/wiki/Mastery_Grids_Interface`

Chapter 3

Requirements Analysis

This chapter follows the process of requirements elicitation and requirements analysis described in [BD09]. Section 3.1 provides a summary of the planned system. Then we begin by analyzing the current system in Section 3.2, before describing the proposed system's requirements in detail in Section 3.3. This chapter concludes with Section 3.4, where we present the user interface and develop the application domain's system models.

3.1 Overview

As stated in Section 1.3, we want to achieve two goals with the proposed system:

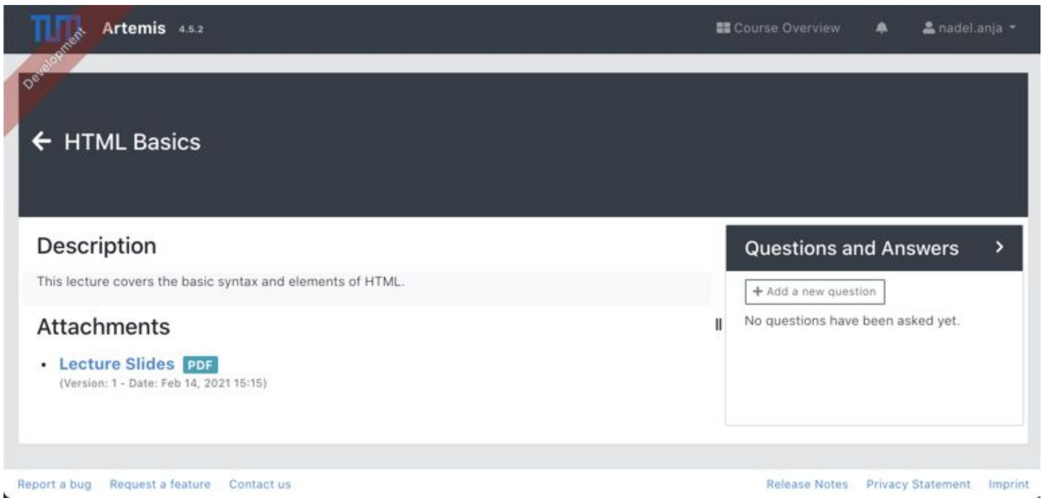
We want to motivate students to do better in a course by providing them with all three frames of reference (self, performance, and mastery) using a new learning analytics dashboard. The mastery frame of reference requires the implementation of learning goals. Success is achieved if instructors can define learning goals and students can use the three frames of reference to motivate themselves.

We also want to redesign the concept of a lecture in Artemis. In the proposed system, instructors can embed various content directly in a lecture. Success is achieved if instructors can include recordings, notes, related exercises, and files in a lecture.

3.2 Current System

We do refer to version 4.5.2 as the current Artemis version.

A lecture in the current version consists of a title, a description, and file attachments. Students can post questions on the lecture page, which tutors can then answer. Figure 3.1 shows an example of a lecture page.



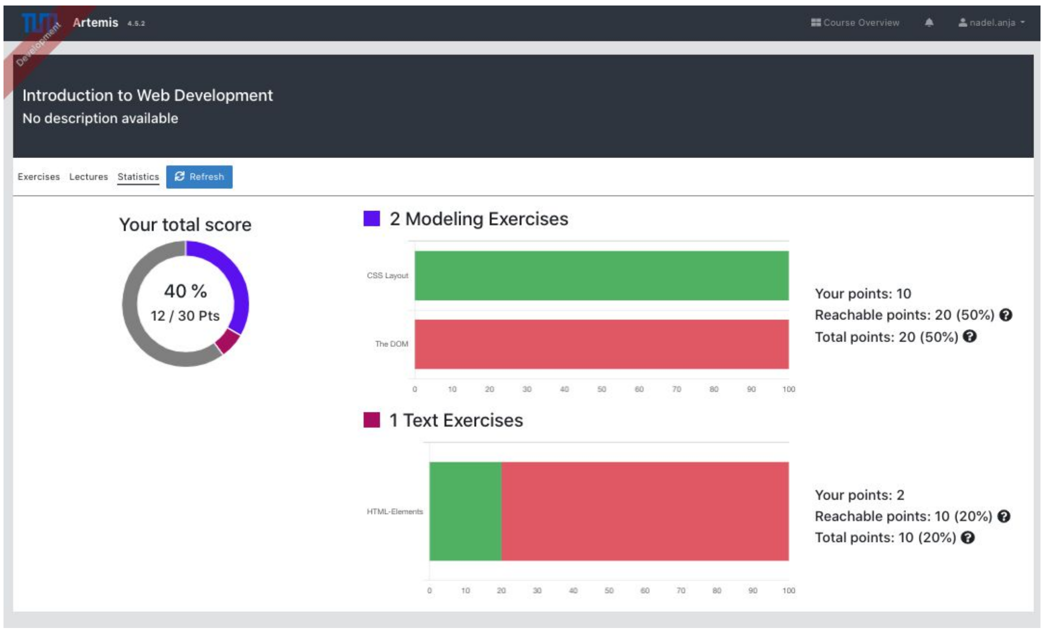
This lecture covers the basic syntax and elements of HTML.

(Version: 1 - Date: Feb 14, 2021 15:15)

No questions have been asked yet.

Figure 3.1: A lecture page in Artemis version 4.5.2

Artemis offers students in the current version a course statistics page, which shows their momentary achieved points in exercises. They cannot compare themselves with the rest of the course or see if their performance has improved or deteriorated. Figure 3.2 shows an example of the course statistics page. There is also no way for instructors to define learning goals in the system.



Introduction to Web Development
No description available

Your total score
40 %
12 / 30 Pts

2 Modeling Exercises
Your points: 10
Reachable points: 20 (50%)
Total points: 20 (50%)

1 Text Exercises
Your points: 2
Reachable points: 10 (20%)
Total points: 10 (20%)

Figure 3.2: The course statistics page in Artemis version 4.5.2

3.3 Proposed System

When an instructor creates a lecture in the proposed system, they can add notes, recordings, related files, and exercises. The student can then access this content via the lecture page.

To provide students with mastery-orientation, the proposed system introduces the concept of learning goals. A learning goal represents a competency that students are expected to master in the course. Instructors can define these goals and connect them to relevant lecture content. The system calculates a student's progress in achieving the learning goals, and this information is available in the learning analytics dashboard.

In the dashboard, students are able to see the development of their personal performance, the average course performance, and the best performance in the course with the help of a visualization. This allows them to use self-orientation and performance-orientation.

Instructors can use the dashboard to track the course's performance and average learning goal progress.

The following sections list the functional and non-functional requirements of the proposed system. The requirements are derived from the user's view.

3.3.1 Functional Requirements

This section lists the functional requirements (FRs) of the proposed system. The FRs are described independently of the concrete implementation. We divide the FRs into three parts. First, we list those for the lecture redesign. Those for the learning goals and finally those for the learning analytics dashboard.

Lecture Redesign

FR1.1 **Provide lecture recordings to students**: An instructor can provide students with lecture recordings by adding them to a lecture. A student can watch the lecture recordings on the lecture page.

FR1.2 **Provide lecture notes to students**: An instructor can provide students with lecture notes by adding formatted text and images to a lecture. A student can read the lecture notes on the lecture page.

FR1.3 **Provide files to students**: An instructor can provide students with files by attaching them to a lecture. A student can download the files on the lecture page.

FR1.4 **Provide related exercises to students**: An instructor can mark exercises related to a lecture. A student can see the related exercises on the lecture page.

FR1.5 **Track lecture completion progress of students**: Instructors can track how far their course is in working through a lecture: Have they watched all the videos, read all the notes, etc.? Students can see what they still need to do within a lecture.

FR1.6 **Set release date**: An instructor can give individual lecture content a release date. The content will be hidden for students on the lecture page until the specified date.

Learning Goals

FR2.1 **Define learning goals**: An instructor can define learning goals, i.e., competencies students should master. He or she can then link the goals to relevant lecture content. Students can see the goals of the course and the linked lecture content.

FR2.2 **Progress in learning goals**: A student can progress in a learning goal by working through the linked content. A learning goal is considered completed if the student has completed all the linked content (i.e. watched all videos, read all notes, downloaded all files and completed all exercises).

FR2.3 **Define learning paths**: An instructor can link learning goals to create learning paths. For example, the learning goals "CSS" and "HTML" can be defined as recommended prerequisites for the learning goal "JavaScript."

Learning Analytics Dashboard

FR3.1 **View development of own exercise performance**: Students can see how their performance in exercises has developed over the duration of the course. This enables students to use self-orientation.

FR3.2 **View course exercise performance development**: Instructors and students can see how the average and best performance of students in exercises has developed over the course duration. This enables students to use performance-orientation and gives instructors an overview of how their class is performing.

FR3.3 **View own learning goal progress**: Students can see their own progress in achieving each learning goal. This enables students to use mastery-orientation.

FR3.4 **View course learning goal progress**: Students and instructors can see the average progress of the course for each learning goal.

3.3.2 Non-functional Requirements

This section describes the requirements of the proposed system that are not related to functionality. We categorize the non-functional requirements using the URPS (Usability Reliability Performance Supportability) model. This model is used by the Unified Process [JBR99] and is described in [BD09].

Lecture Redesign

NFR1.1 **Usability (Efficiency)**: Students see all the lecture content directly on the lecture page without having to navigate to other pages.

NFR1.2 **Reliability (Robustness)**: An instructor should not lose any data when he or she is in the process of adding content to a lecture and the connection to Artemis is lost.

NFR1.3 **Supportability (Adaptability)**: A developer should be able to add new types of lecture content in the future without having to touch the code of the existing lecture content types.

NFR1.4 **Supportability (Learnability)**: The various types of lecture content should have a common look and interaction pattern.

NFR1.5 **Performance (Response Time)**: Loading a lecture page with 5 lecture recordings, 5 lecture notes, and 5 related exercises should take no more than 0.5 seconds.

Learning Goals

NFR2.1 **Usability (Efficiency)**: It should not take an instructor more than three clicks to create a new learning goal.

NFR2.2 **Usability (Efficiency)**: An instructor should be able to connect a learning goal with multiple pieces of lecture content in one step (i.e., without having to navigate between several pages).

NFR2.3 **Usability (Learnability)**: The user interface for learning goals should visualize the progress in a comprehensive way.

Learning Analytics Dashboard

NFR3.1 **Performance (Response Time)**: Loading the learning analytics dashboard page for a course with 2000 students, 50 exercises, and 10 learning goals should not take more than 3 seconds.

NFR3.2 **Usability (Learnability)**: The information on the learning analytics dashboard should be understandable for students without having to read any manual.

3.4 System Models

This section uses various types of system models to provide a functional specification of the proposed system. These models are independent of the actual implementation.

3.4.1 Scenarios

Scenarios provide an informal description of specific system functionality from an actor's point of view [BD09]. Visionary scenarios describe sophisticated functionality that is ideal but hard to realize in this thesis's scope. On the other hand, demo scenarios describe functionality that can be realistically implemented.

Visionary Scenario: Adaptive Learning

The following visionary scenario describes how adaptive learning is used to deliver a customized learning experience. In this case, Artemis dynamically adapts the difficulty level of exercises to the respective student.

It is the middle of the third semester. Simon is studying computer science at the Technical University of Munich. He attends the course "Introduction to Algorithms," which uses Artemis. The next lecture, Simon wants to work on is on the topic of graph algorithms. After watching the lecture recordings and reading the notes, he starts the graph algorithms exercise sequence. First, he is given a simple coding exercise. Simon needs several hints, attempts, and much time to solve the exercise. The system analyses Simon's solution (How many mistakes did he make? How long did it take him? How many attempts did he need? How many hints did he use?) and decides that Simon is not yet ready to solve exercises of medium difficulty. Therefore, he is given a new easy exercise next. In the course of further easy exercises, Simon gets better and better until the system finally presents medium difficulty exercises. This adaptive process continues until Artemis recommends Simon to move on to the next topic, as he has already mastered graph algorithms. Simon follows the recommendation and starts the next topic.

Visionary Scenario: Detecting At-Risk Students

The following visionary scenario describes how enhanced learning analytics could help students who are identified as being at risk of dropping out of a course.

It is the middle of the first semester. Christian is studying computer science at the Technical University of Munich. He attends the course "Introduction to Programming," which uses Artemis. He found the assignments relatively easy in the beginning, and he was able to complete them well. However, in the last two weeks, the topics have become more complicated, and he can no longer keep up well. Since he is demotivated, he neglects the course. He is doing worse and worse in the assignments and hardly deals with the lecture material. Artemis detects the change in Christian's behavior (e.g., decreased interaction with the lecture material and deteriorating results). The system calculates that he has a high probability of dropping out of the course soon. Therefore, the system informs professor Stephan by email. Stephan then opens Artemis and looks at the data on Christian. As he agrees with the system that Christian is in danger of dropping the course, he contacts Christian by e-mail and tells him that he is welcome to come to his weekly office hours to talk about problems with the course material. Christian did not know about the weekly office hours and accepts the offer. The tips he receives motivate him, his performance improves again, and he successfully completes the course.

Demo Scenario: Creating an Online Lecture With Different Types of Content

The following demo scenario describes how an instructor can use the more flexible lecture design.

It is a week before the start of the semester, and Stephan is an instructor for the course "Basics of Web Development" at the Technical University of Munich. Due to the global coronavirus pandemic, the lecture will be held completely online this semester. He has decided to use Artemis and is currently setting everything up. He has already created the course and the first exercises. Only the first lectures are still missing. He is currently preparing the first lecture about the basics of HTML. He already recorded himself holding this multiple-hour-long lecture. Instead of just uploading the whole multiple-hour video to YouTube and the slides to Artemis, he wants to use the new lecture format.

As he has read that it is pedagogically useful to divide an online lecture into several thematically self-contained chunks, he splits the video and the lecture slides. For example, the first chunk deals with HTML syntax, the second with the Document Object Model, and so on. After splitting the multiple-hour lecture into smaller parts, he creates the lecture in Artemis. He decides on the following structure for each part of the lecture: First the lecture recording, then lecture notes containing the information from the slides, and finally a related exercise so students can immediately test if they have understood the material. For the video, he uploads the video to YouTube and inserts the URL into Artemis. He uses the material in his lecture slides to create the lecture notes in Artemis. For the exercise, he selects the course exercises he wants to integrate into the lecture. He also uploads the full lecture slides as a PDF for people who want everything in one file.

It is the first lecture, and Max, a student, opens the lecture page in Artemis. He sees the video, lecture notes, and related exercises embedded in the lecture page. As the videos are all relatively short (30 minutes maximum), he is motivated to start working on the lecture. He also finds it great to practice the things he has learned right away in the linked exercises. Since he prefers to read longer texts as PDFs, he downloads the slides and reads them on his tablet.

Demo Scenario: Viewing the Student Learning Analytics Dashboard

The following demo scenario describes how different types of students can use the student learning analytics dashboard to motivate themselves.

It is the middle of the semester, and the students Alex, Bob and Kristina are taking the course "Basics of Web Development."

Kristina is an outstanding student with an enormous amount of ambition. She wants to be one of the best students in every course. That is why she is pleased that she can check at any time in Artemis how she is doing compared to her fellow students. She opens the learning analytics dashboard and sees that she had not the best performance in the last two exercises. This motivates her to do even better, and she resolves to study even more.

Bob is a hands-on student. He wants to work as a web developer one day and is therefore interested in whether he has mastered all the skills that the course is supposed to teach. He opens the learning analytics dashboard and sees that he is already advanced in the learning goal HTML but still lags in the learning goal CSS. Therefore, he plans to focus on achieving this learning goal in the next learning session at the weekend.

Alex has little experience in web development and does not want to be demotivated by comparing himself to other students with much more experience. His goal is to continuously work on his skills. He opens the learning analytics dashboard and sees that he has got better scores in the exercises this week compared to last week. This encourages to continue studying and get even better.

3.4.2 Use Case Model

This section describes the functional requirements listed in Section 3.3.1 using UML use case diagrams. Use case diagrams model the system's functionality at a high level of abstraction from the so-called black box view of the user [BD09]. Only those use cases are defined that an external user can perceive and whose execution brings him or her a recognizable benefit.

The models describe what use cases the system offers and not how they are implemented in the system.

Lecture Redesign

Figure 3.3 shows the use cases for the redesigned lectures. Only files (e.g., lecture slides as PDFs) can be added to a lecture in the current system. The proposed system offers more flexibility in the content that can be added to a lecture. Instructors can provide students with recordings, notes, and related files. They can also suggest related exercises. Students who open the lecture page can watch the recordings, read the notes, download the related files and do the suggested exercises. The system measures a student's interaction with the content (e.g., how much of a recording a student has already watched) and calculates the student's progress in working through the lecture. A student can see which parts he or she still has to work on to complete the lecture. An instructor can track the average progress of students as they work through the lecture.

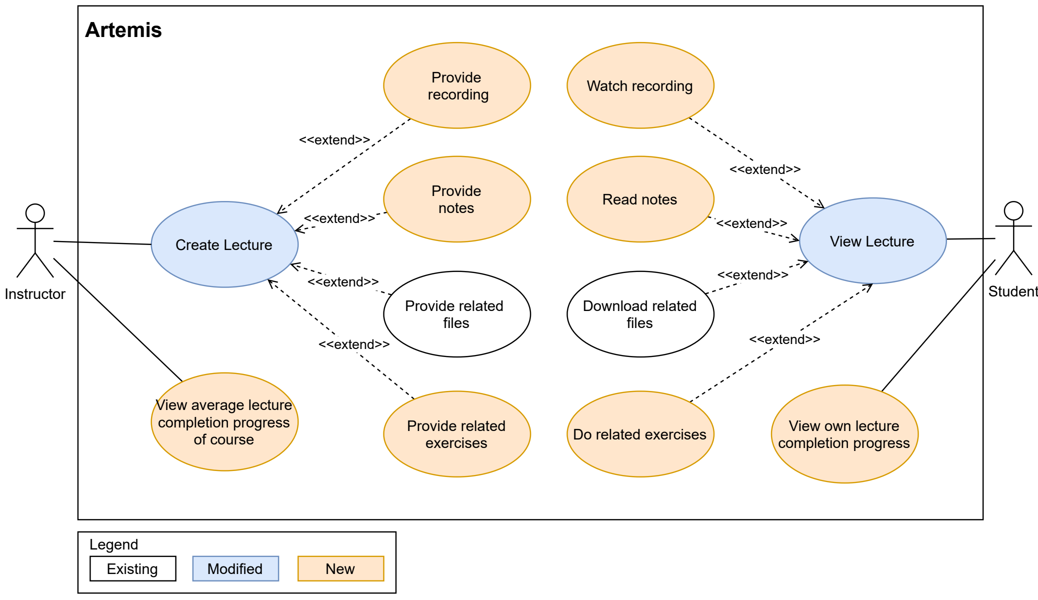


Figure 3.3: Use case diagram of the lecture redesign in the proposed system

Learning Goals

Figure 3.4 shows the use cases for the learning goals. An instructor can define learning goals for a course and link those to lecture content. When the student works through a lecture's content, the student can see what learning goals this content is related to. By working through the linked content (e.g., completing relevant exercises), the student increases his or her progress in the learning goal. A learning goal is considered completed when the student completes all the connected material.

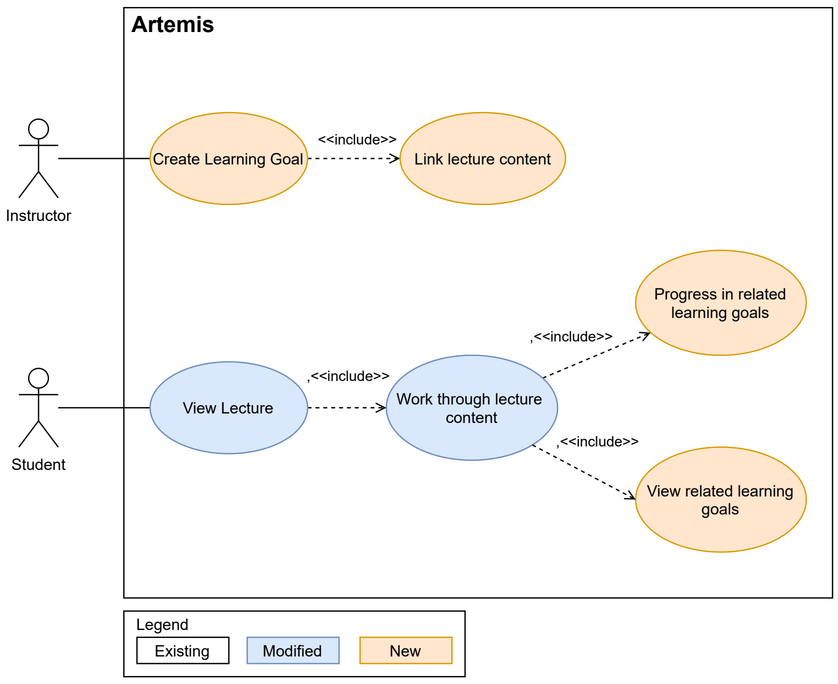


Figure 3.4: Use case diagram of the learning goals in the proposed system

Learning Analytics Dashboard

Figure 3.5 shows the use cases for the learning analytics dashboard. Students can use the dashboard to compare the development of their own performance in exercise with the course's average and best performance. They can also see their progress towards completing the learning goals. Instructors can use the dashboard to check the average progress and performance of their students.

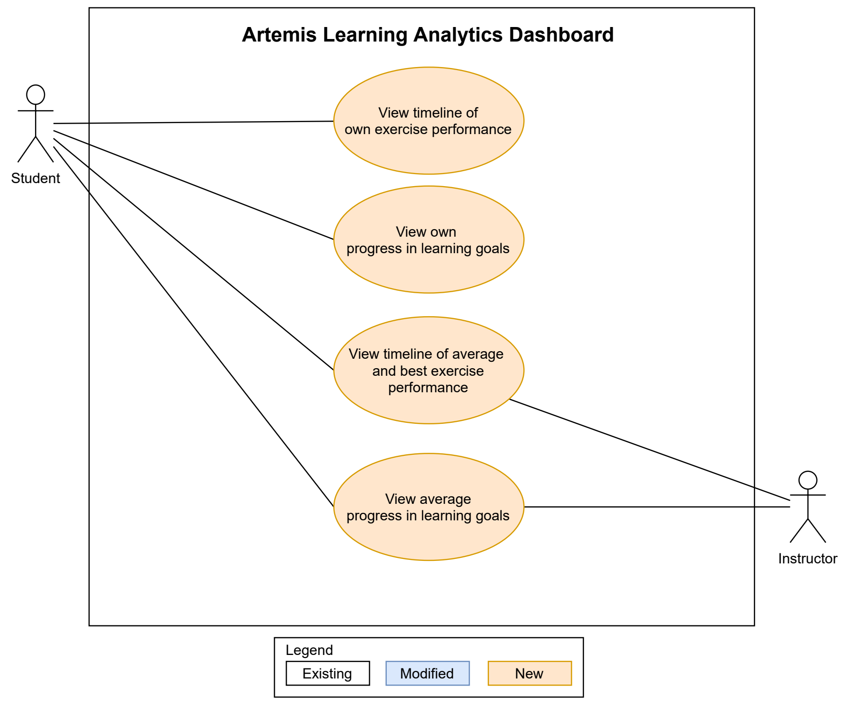

Artemis Learning Analytics Dashboard
Student
View timeline of own exercise performance
View own progress in learning goals
View timeline of average and best exercise performance
View average progress in learning goals
Instructor
Legend
Existing
Modified
New

Figure 3.5: Use case diagram of the learning analytics dashboard in the proposed system

3.4.3 Analysis Object Model

This subsection discusses the analytics object model with the help of UML class diagrams. It represents the main concepts that are visible to users when using the system [BD09]. Implementation-specific details like return types and access modifiers are omitted from the UML models, as the analysis object model is on the application domain level.

Figure 3.6 shows the most important attributes and methods of the entities and their relationships with each other.

Instructors can add *lectures* to a course and different types of *lecture content* in turn to a lecture. When a *student* engages with *lecture content*, Artemis tracks the interaction and determines the *content completion progress*. The condition when a *lecture content* can be considered completed is different for each type of content.

An instructor can define *learning goals* for a *course* in the proposed system. *Learning goals* represent competencies that a student should master. The instructor can link goals to relevant *lecture content*. The *content completion progress* of the linked *lecture content* is used to calculate a student's *progress in mastering a learning goal*. A *learning goal* is considered mastered when the student has completed all the linked *lecture content*. Instructors can combine different *learning goal*s to form *learning paths*. Different types of relationships can be defined. For example, one *learning goal* can be marked as a recommended prerequisite for another *learning goal*.

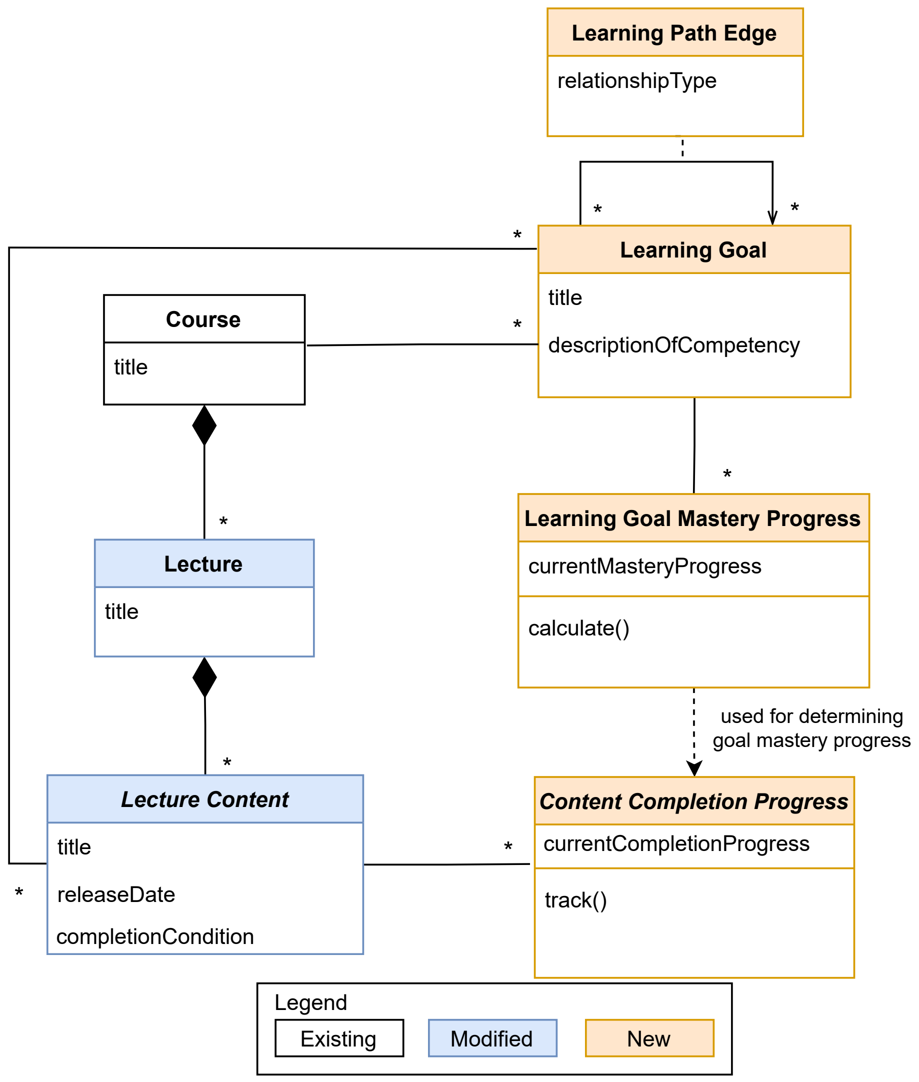


Figure 3.6: Analysis object model of the proposed system

Figure 3.7 shows the different *lecture content* types and the associated progress measurements. In the current system, only *files* can be added to a *lecture*. In the proposed system, *video recordings* and *notes* can now also be added. *Exercises* can be marked as belonging to a *lecture*. A *video recording* is considered completed when the student has watched it completely. A *note* is considered completed when it has been read in full by the student. A *file* is considered completed when it has been downloaded at least once by the student. For an *exercise*, the score achieved can be used as a progress measure. An *exercise* could be considered completed when the student has achieved more than 90 % of the points. The results of exercises already exist in the system, so no new tracking mechanism is needed.

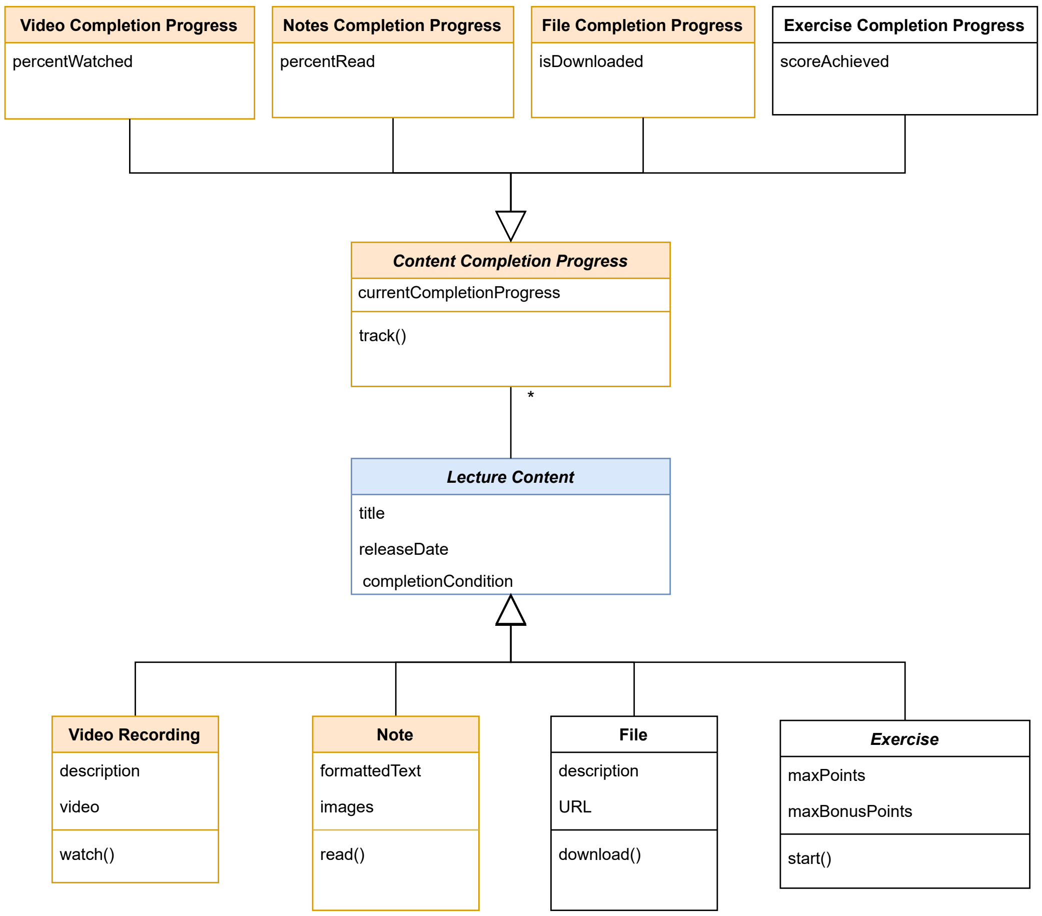


Figure 3.7: Detailed analysis object model of the new types of lecture content and their progress measurements in the proposed system

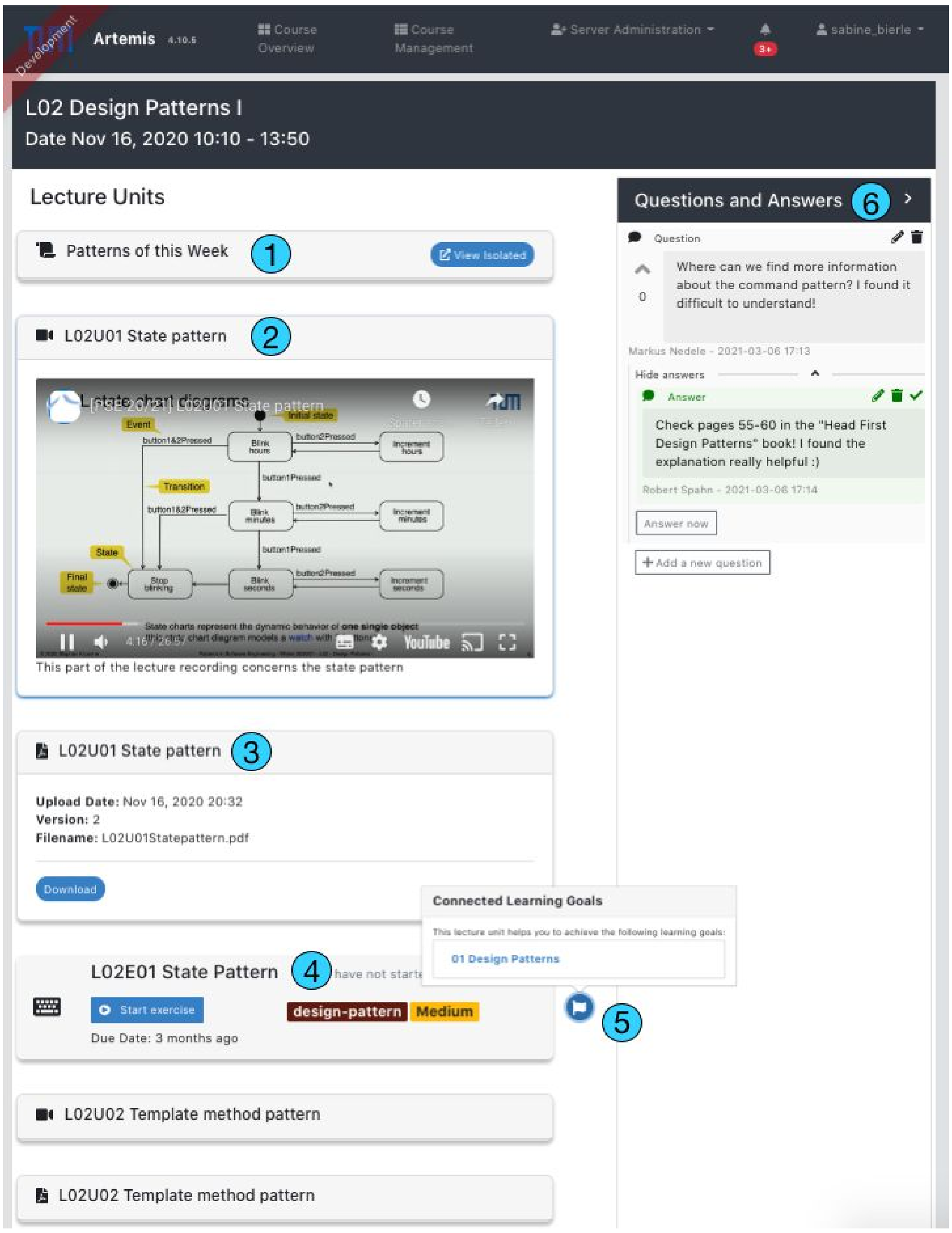
3.4.4 User Interface

This section describes the user interface of the proposed system. As the proposed system is already implemented at the time of writing, we present the new system's actual interface and not mock ups.

Lecture Redesign

Student View

Figure 3.8 shows how a lecture page with different types of content looks from a student's perspective. The content can be seen in a column on the left. Highlighted in the figure is a lecture note (1), a lecture recording (2), the downloadable lecture slides (3), and a relevant exercise (4). The different types of content have a similar appearance and can all (except for exercises) be expanded and collapsed by clicking on the title. In the example, the lecture recording (2) and the file (3) are expanded, the note (1) is still collapsed. Students can recognize if the content is linked to a learning goal by a flag symbol (5). They can obtain more information by clicking on the flag. As in the current system, students can ask questions about the lecture content (6), which other users can then answer.



L02 Design Patterns I

Date Nov 16, 2020 10:10 - 13:50

Lecture Units

Patterns of this Week

L02U01 State pattern

This part of the lecture recording concerns the state pattern

L02U01 State pattern

Upload Date: Nov 16, 2020 20:32
Version: 2
Filename: L02U01Statepattern.pdf

L02E01 State Pattern

have not starte

Due Date: 3 months ago

Connected Learning Goals

This lecture unit helps you to achieve the following learning goals:

01 Design Patterns

L02U02 Template method pattern

L02U02 Template method pattern

Questions and Answers

Question

0

Where can we find more information about the command pattern? I found it difficult to understand!

Markus Nedele - 2021-03-06 17:13

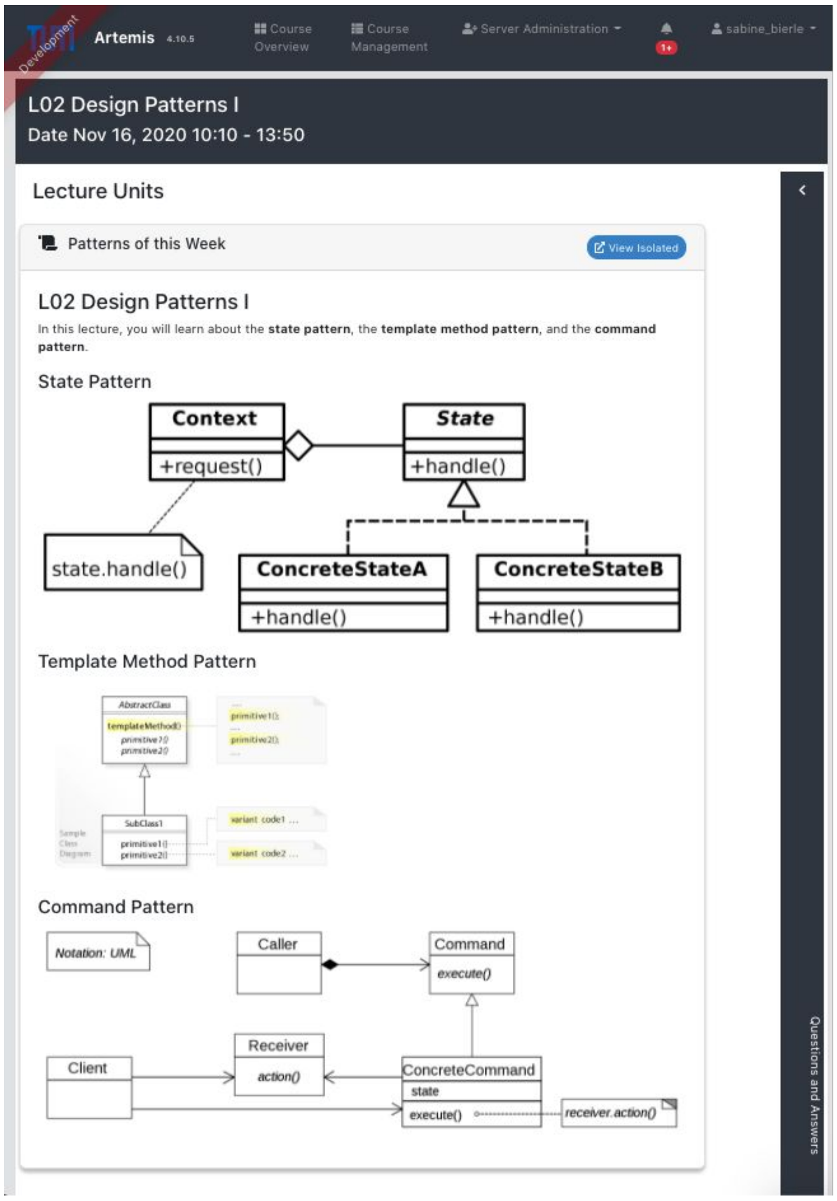
Answer

Check pages 55-60 in the "Head First Design Patterns" book! I found the explanation really helpful :)

Robert Spahn - 2021-03-06 17:14

Figure 3.8: Example of a lecture page in the proposed system

Figure 3.9 shows the lecture note again in expanded form. It contains formatted text and embedded images. The student can click on "View Isolated" to open the note on a separate page for better printing and reading.



L02 Design Patterns I
Date Nov 16, 2020 10:10 - 13:50

Lecture Units

Patterns of this Week

L02 Design Patterns I

In this lecture, you will learn about the **state pattern**, the **template method pattern**, and the **command pattern**.

State Pattern

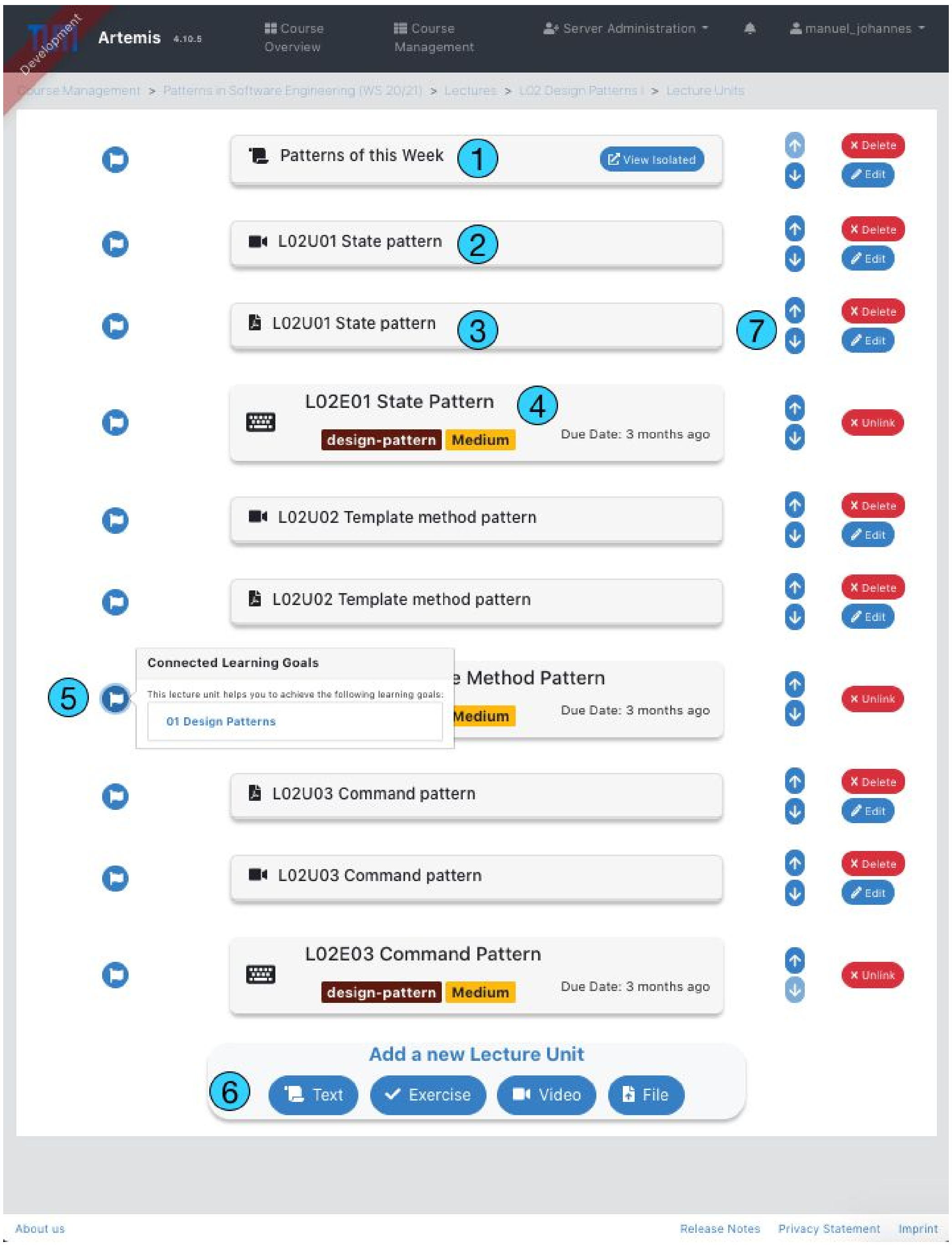
Template Method Pattern

Command Pattern

Figure 3.9: Example of a lecture note in the proposed system

Instructor View

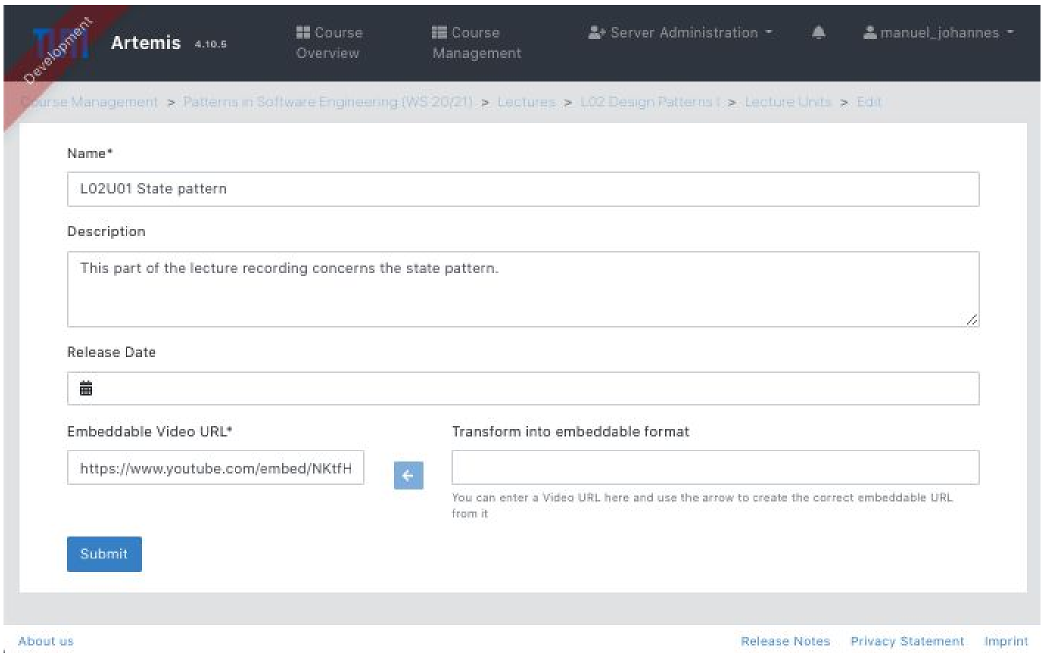
Figure 3.10 shows how instructors can modify lecture content in the proposed system. In the figure, the lecture note (1), the recording (2), the presentation slides (3), and the relevant exercise (4) are again marked. The instructor can change the content order by using the arrows (7). The buttons on the right side of the arrows can be used to edit or delete content. By clicking on the flag (5), the instructor can see the linked learning goals. New content can be added by clicking on the corresponding button in the toolbar at the bottom of the screen (6). The corresponding creation page of the content type will open.



This lecture unit helps you to achieve the following learning goals:

Figure 3.10: Example of lecture content editing in the proposed system

Figure 3.11 shows how videos can be added to a lecture in the proposed system. The instructor can give the video a title, a description, and a release date. Most video sharing platforms offer a special embeddable version of the video URL to embed a video on other websites. The instructor can enter such an embeddable URL directly or input a standard video URL and let Artemis generate an embeddable version from it by clicking on the arrow.


Development
Artemis 4.10.5
Course Overview
Course Management
Server Administration
manuel_johannes
Course Management > Patterns in Software Engineering (WS 20/21) > Lectures > L02 Design Patterns I > Lecture Units > Edit

Name*

L02U01 State pattern

Description

This part of the lecture recording concerns the state pattern.

Release Date

Embeddable Video URL*

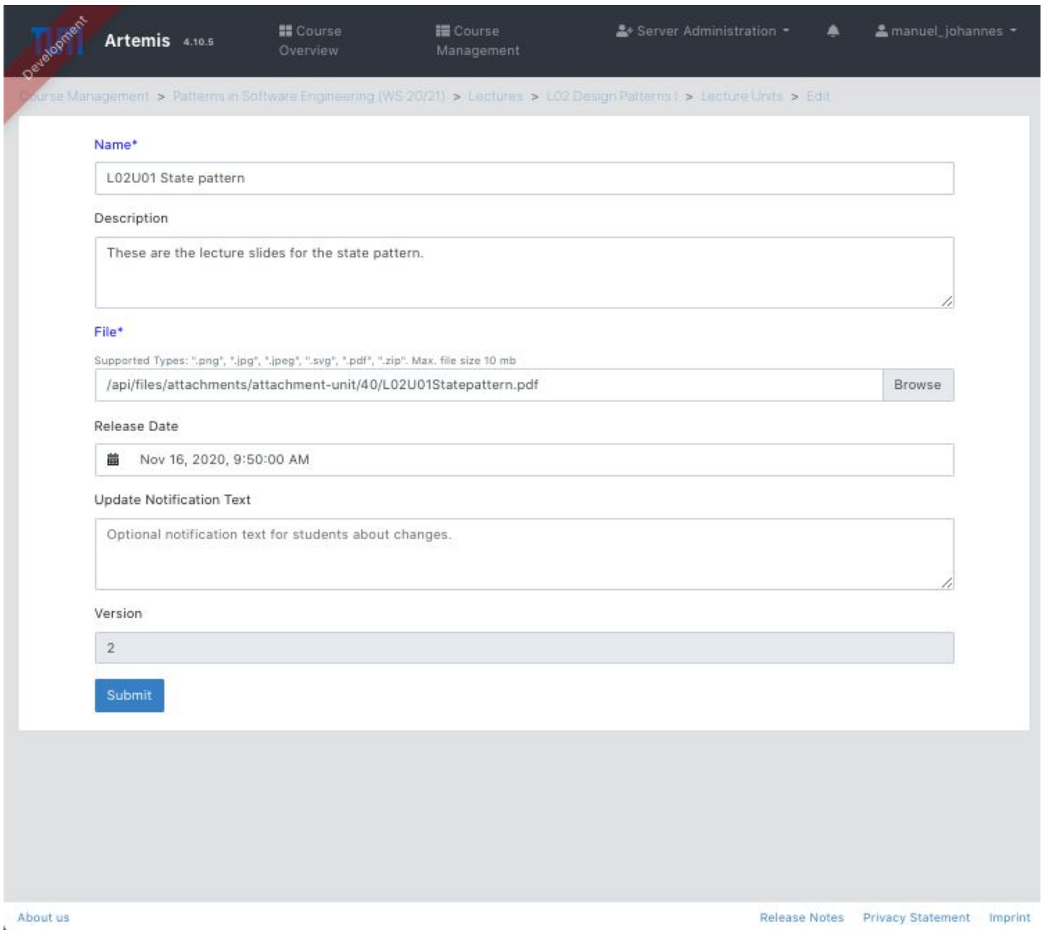
https://www.youtube.com/embed/NKtfH

Transform into embeddable format

You can enter a Video URL here and use the arrow to create the correct embeddable URL from it

Figure 3.11: Example of how a video can be added to a lecture in the proposed system

Figure 3.12 shows how a file can be uploaded and added to a lecture in the proposed system. The instructor can give the file a title, a description, and a release date. If an optional notification text is entered, Artemis will notify all registered students in the course. The version field increments when the instructor makes changes to the uploaded file.


Development
Artemis 4.10.5
Course Overview
Course Management
Server Administration
manuel_johannes
Course Management > Patterns in Software Engineering (WS 20/21) > Lectures > L02 Design Patterns I > Lecture Units > Edit

Name*

L02U01 State pattern

Description

These are the lecture slides for the state pattern.

File*

Supported Types: ".png", ".jpg", ".jpeg", ".svg", ".pdf", ".zip". Max. file size 10 mb

/api/files/attachments/attachment-unit/40/L02U01Statepattern.pdf

Release Date

Nov 16, 2020, 9:50:00 AM

Update Notification Text

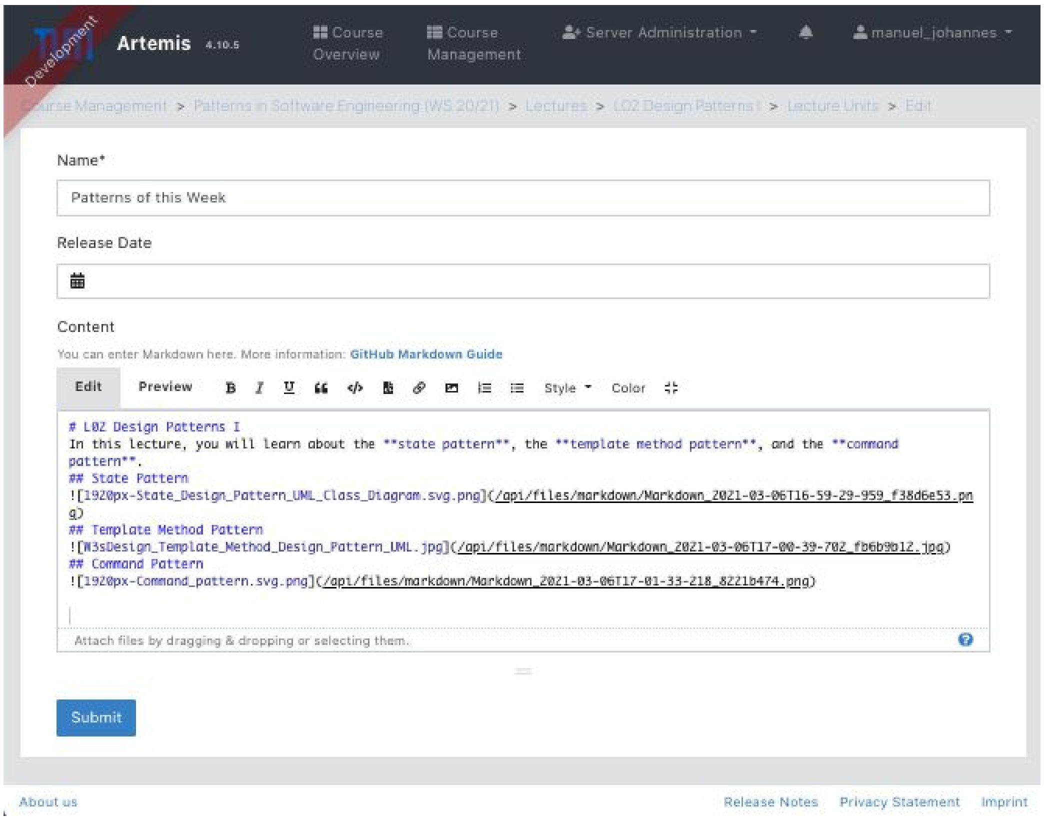
Optional notification text for students about changes.

Version

2

Figure 3.12: Example of how a file can be added to a lecture in the proposed system

Figure 3.13 shows how a note can be added to a lecture in the proposed system. The instructor can give the note a title and a release date. Text formatting is defined using Markdown[1] syntax. Images can be uploaded and embedded in the text by dragging and dropping them on the editor. The content will be converted to HTML for presentation when the student opens the lecture page.


Artemis 4.10.5
Development
Course Overview
Course Management
Server Administration
manuel_johannes
Course Management > Patterns in Software Engineering (WS 20/21) > Lectures > L02 Design Patterns I > Lecture Units > Edit

Name*

Patterns of this Week

Release Date

Content

You can enter Markdown here. More information: GitHub Markdown Guide

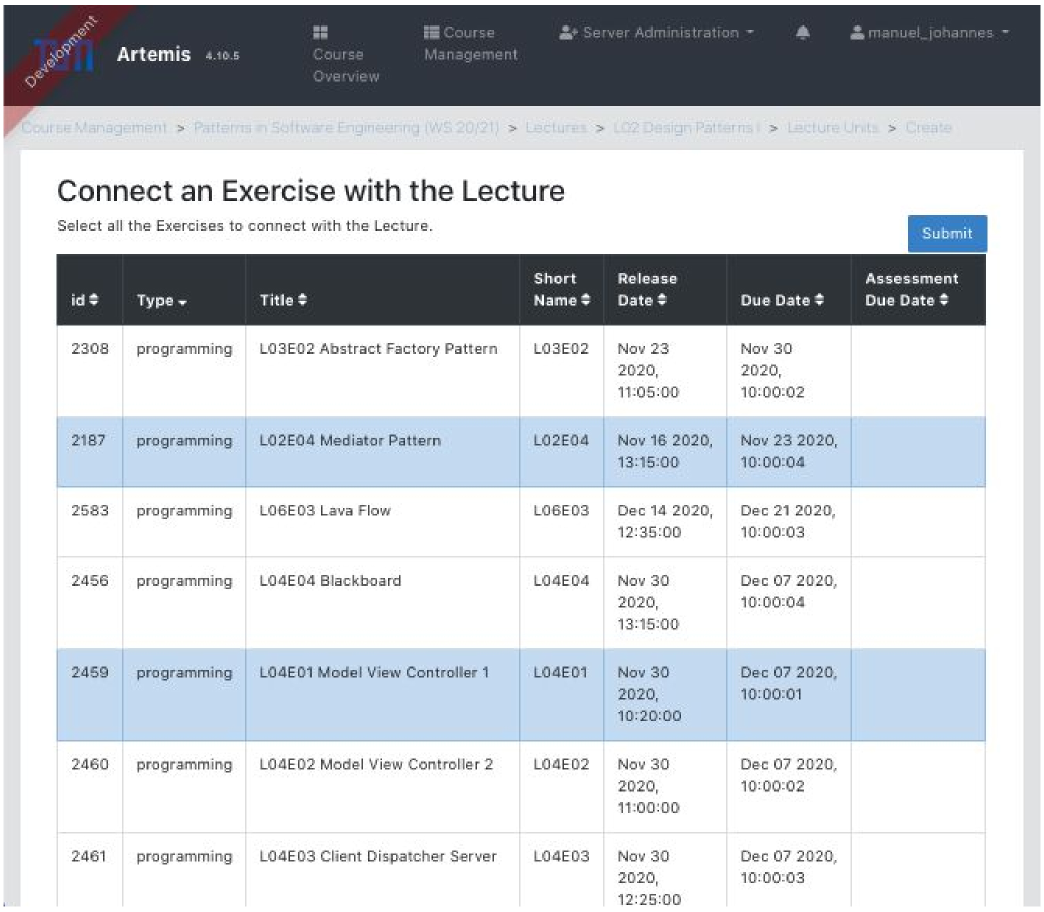
```
# L02 Design Patterns I
In this lecture, you will learn about the **state pattern**, the **template method pattern**, and the **command
pattern**.
## State Pattern
![1920px-State_Design_Pattern_UML_Class_Diagram.svg.png](/api/files/markdown/Markdown_2021-03-06T16-59-29-959_f38d6e53.pn
g)
## Template Method Pattern
![W3sDesign_Template_Method_Design_Pattern_UML.jpg](/api/files/markdown/Markdown_2021-03-06T17-00-39-702_fb6b9b12.jpg)
## Command Pattern
![1920px-Command_pattern.svg.png](/api/files/markdown/Markdown_2021-03-06T17-01-33-218_8221b474.png)
```

Attach files by dragging & dropping or selecting them.

Figure 3.13: Example of how a note can be added to a lecture in the proposed system

[1]https://daringfireball.net/projects/markdown/

Figure 3.14 shows how exercises are added to a lecture in the proposed system. The instructor has to select the corresponding exercises in the table in order to add them to the lecture.


Development
Artemis 4.10.5
Course Overview
Course Management
Server Administration
manuel_johannes
Course Management > Patterns in Software Engineering (WS 20/21) > Lectures > L02 Design Patterns I > Lecture Units > Create
Submit

Connect an Exercise with the Lecture

Select all the Exercises to connect with the Lecture.

id	Type	Title	Short Name	Release Date	Due Date	Assessment Due Date
2308	programming	L03E02 Abstract Factory Pattern	L03E02	Nov 23 2020, 11:05:00	Nov 30 2020, 10:00:02	
2187	programming	L02E04 Mediator Pattern	L02E04	Nov 16 2020, 13:15:00	Nov 23 2020, 10:00:04	
2583	programming	L06E03 Lava Flow	L06E03	Dec 14 2020, 12:35:00	Dec 21 2020, 10:00:03	
2456	programming	L04E04 Blackboard	L04E04	Nov 30 2020, 13:15:00	Dec 07 2020, 10:00:04	
2459	programming	L04E01 Model View Controller 1	L04E01	Nov 30 2020, 10:20:00	Dec 07 2020, 10:00:01	
2460	programming	L04E02 Model View Controller 2	L04E02	Nov 30 2020, 11:00:00	Dec 07 2020, 10:00:02	
2461	programming	L04E03 Client Dispatcher Server	L04E03	Nov 30 2020, 12:25:00	Dec 07 2020, 10:00:03	

Figure 3.14: Example of how exercises can be added to a lecture in the proposed system

Learning Goals

Student View

Figure 3.15 shows the learning analytics dashboard section where students can see the learning goals of the course. Each learning goal is shown in the form of a card. The cards are arranged dynamically depending on the size of the user's screen. In the example, the instructor has defined five learning goals for the course. The student's learning goal mastery progress is shown using circular progress bars.

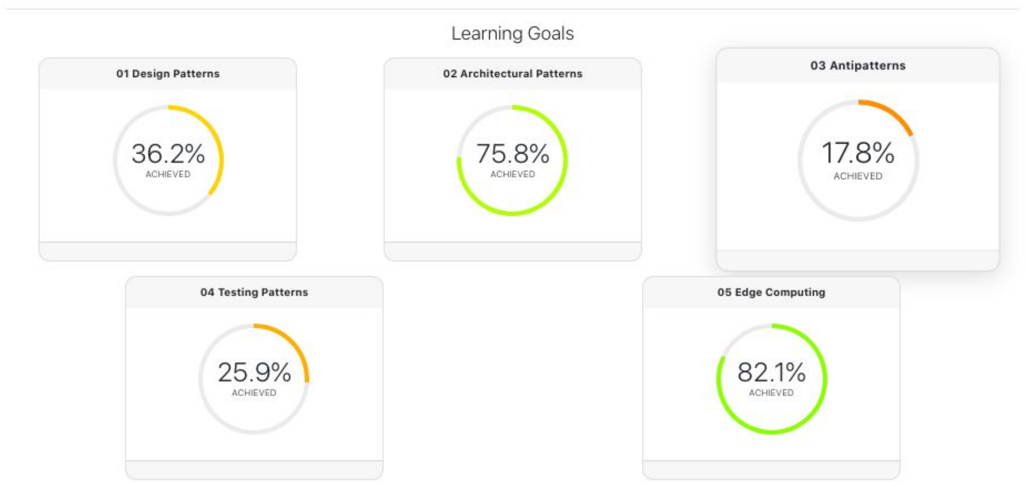
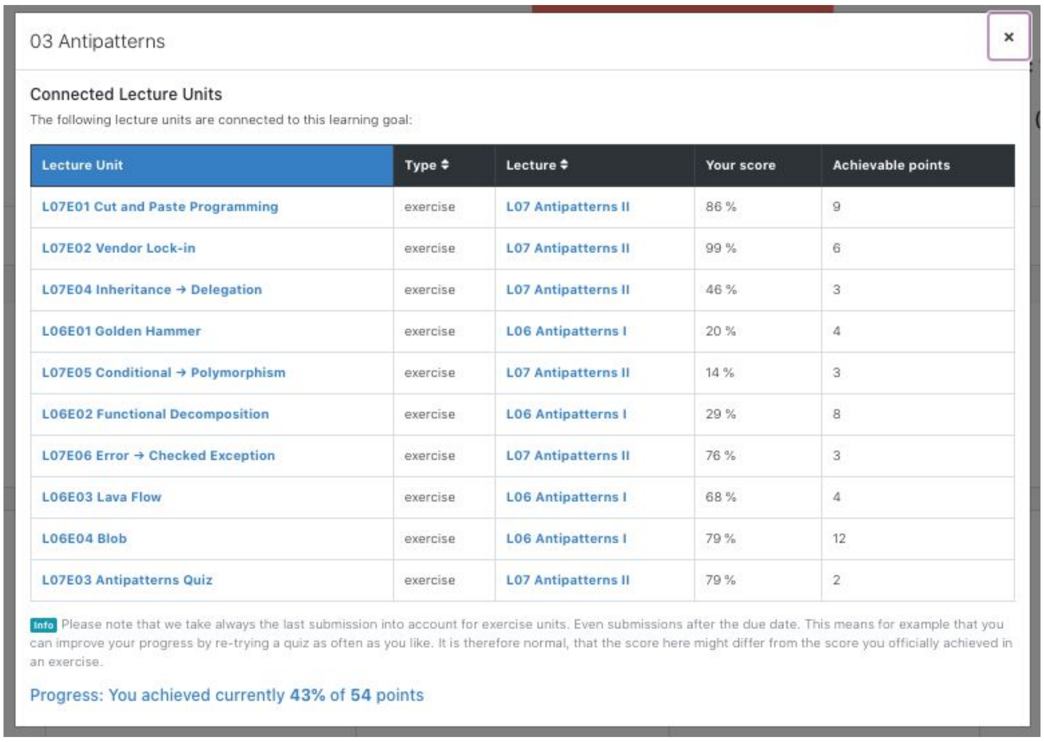


Figure 3.15: Example of the learning goal section in the learning analytics dashboard in the proposed system

When a student clicks on a goal card, a modal window opens with more information. An example is shown in Figure 3.16. The student can see to which lecture content the learning goal is linked to. In the example, these are only exercises. The student gets an overview of his or her lecture content completion progress. This progress is used to calculate the mastery progress of the learning goal.



03 Antipatterns

×

Connected Lecture Units

The following lecture units are connected to this learning goal:

Lecture Unit	Type ⬍	Lecture ⬍	Your score	Achievable points
L07E01 Cut and Paste Programming	exercise	L07 Antipatterns II	86 %	9
L07E02 Vendor Lock-in	exercise	L07 Antipatterns II	99 %	6
L07E04 Inheritance → Delegation	exercise	L07 Antipatterns II	46 %	3
L06E01 Golden Hammer	exercise	L06 Antipatterns I	20 %	4
L07E05 Conditional → Polymorphism	exercise	L07 Antipatterns II	14 %	3
L06E02 Functional Decomposition	exercise	L06 Antipatterns I	29 %	8
L07E06 Error → Checked Exception	exercise	L07 Antipatterns II	76 %	3
L06E03 Lava Flow	exercise	L06 Antipatterns I	68 %	4
L06E04 Blob	exercise	L06 Antipatterns I	79 %	12
L07E03 Antipatterns Quiz	exercise	L07 Antipatterns II	79 %	2

Info Please note that we take always the last submission into account for exercise units. Even submissions after the due date. This means for example that you can improve your progress by re-trying a quiz as often as you like. It is therefore normal, that the score here might differ from the score you officially achieved in an exercise.

Progress: You achieved currently **43%** of **54** points

Figure 3.16: Example of the modal that opens when a student clicks on a learning goal in the proposed system

Instructor View

Figure 3.17 shows an example of the learning goal management page. The page has a similar look to the learning goal section in the learning analytics dashboard, but the progress displayed is the course's average goal mastery progress.

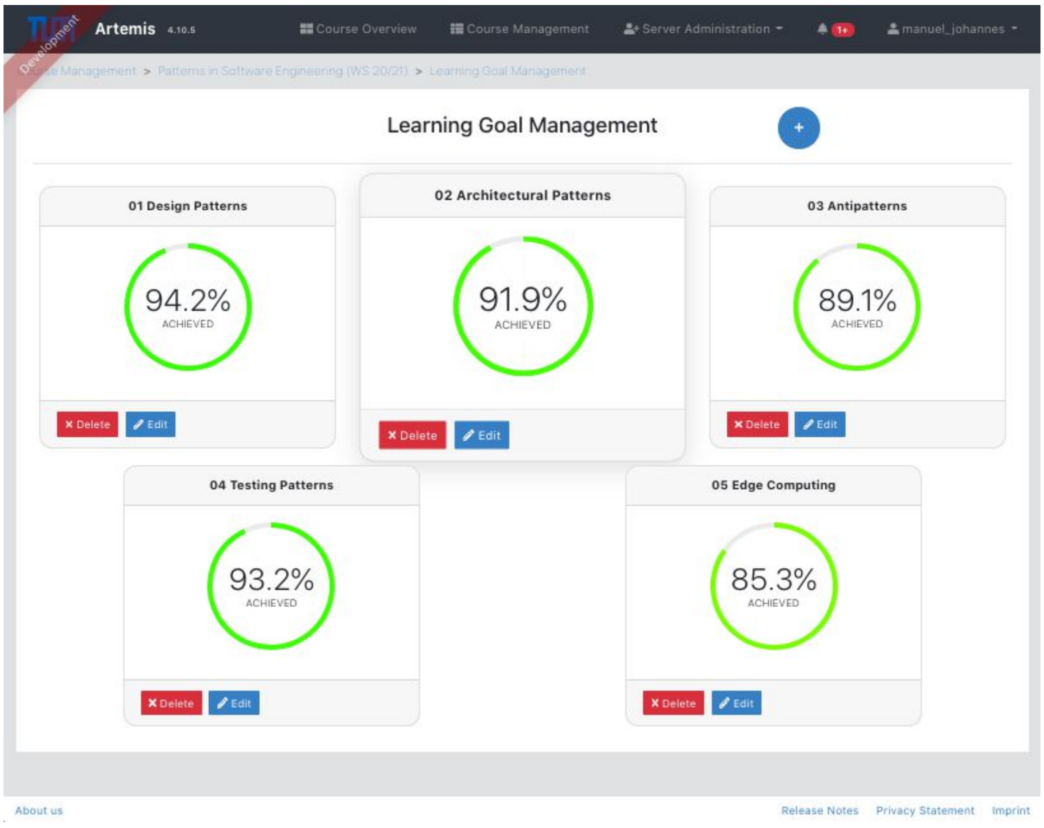
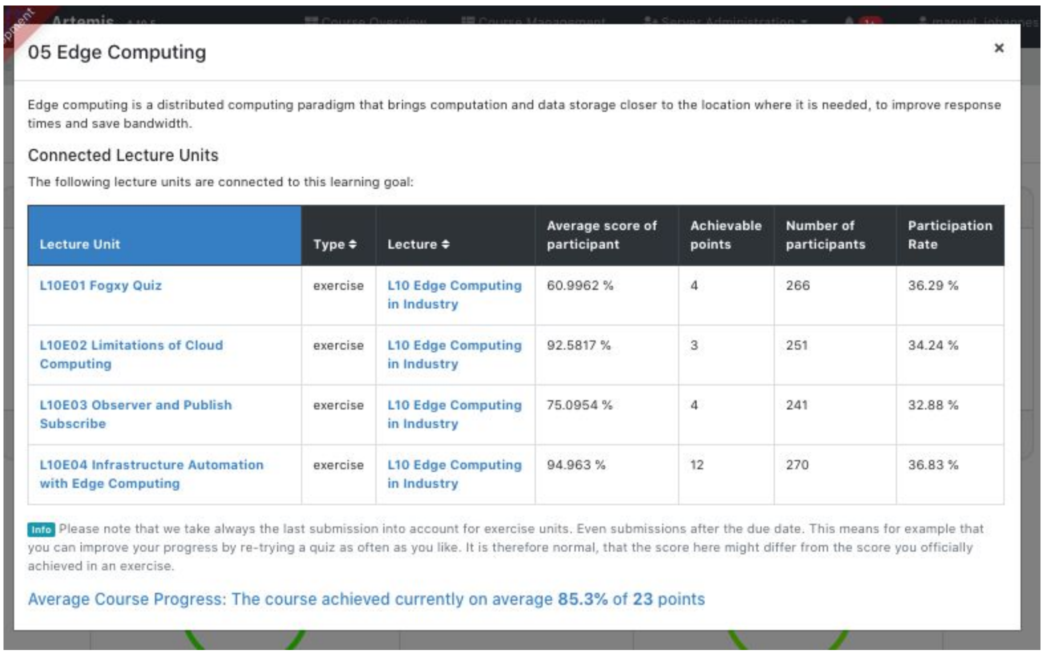


Figure 3.17: Example of the learning goal management page in the proposed system

When an instructor clicks on a goal card, a modal opens with more information. Figure 3.18 shows an example. The instructor can see the average completion progress of the linked lecture content. The instructor can also determine how many students have interacted in some form with the respective content.



05 Edge Computing

Edge computing is a distributed computing paradigm that brings computation and data storage closer to the location where it is needed, to improve response times and save bandwidth.

Connected Lecture Units

The following lecture units are connected to this learning goal:

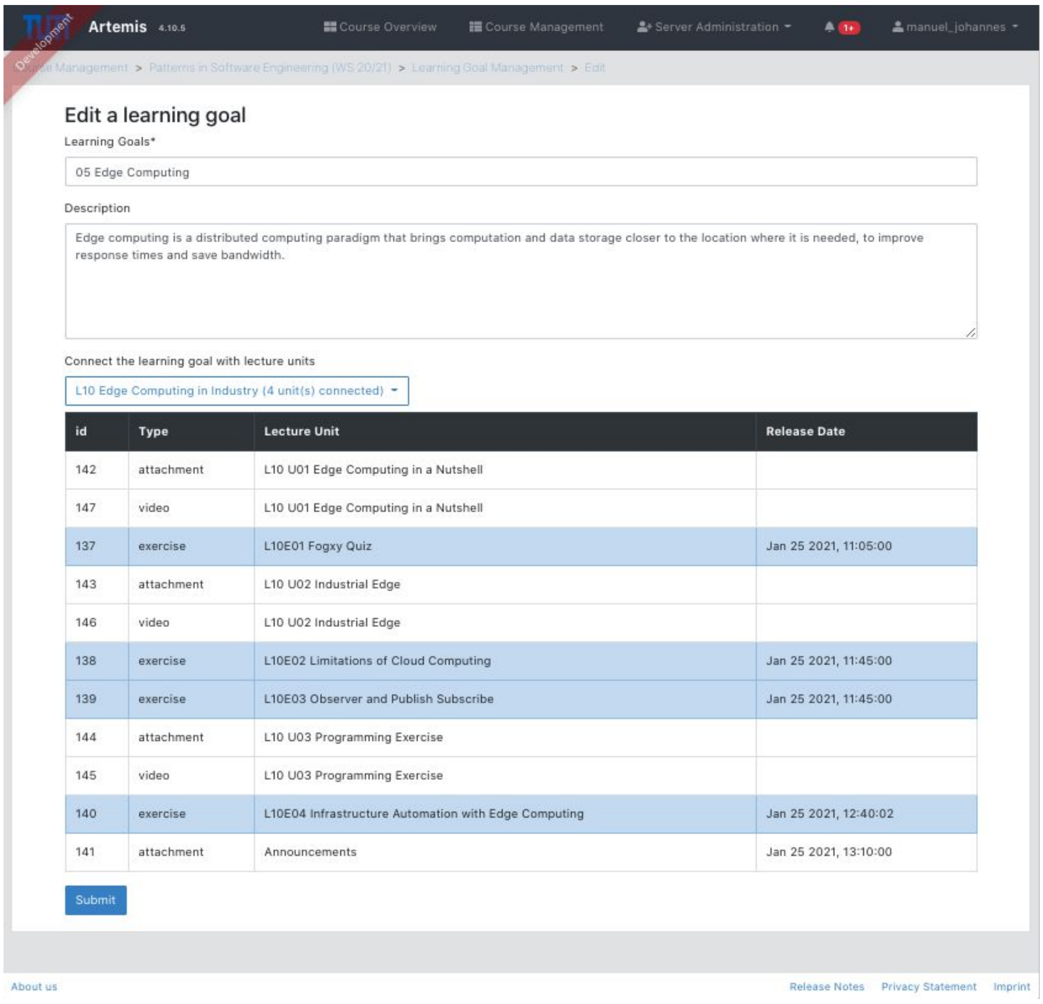
Lecture Unit	Type	Lecture	Average score of participant	Achievable points	Number of participants	Participation Rate
L10E01 Fogxy Quiz	exercise	L10 Edge Computing in Industry	60.9962 %	4	266	36.29 %
L10E02 Limitations of Cloud Computing	exercise	L10 Edge Computing in Industry	92.5817 %	3	251	34.24 %
L10E03 Observer and Publish Subscribe	exercise	L10 Edge Computing in Industry	75.0954 %	4	241	32.88 %
L10E04 Infrastructure Automation with Edge Computing	exercise	L10 Edge Computing in Industry	94.963 %	12	270	36.83 %

Info Please note that we take always the last submission into account for exercise units. Even submissions after the due date. This means for example that you can improve your progress by re-trying a quiz as often as you like. It is therefore normal, that the score here might differ from the score you officially achieved in an exercise.

Average Course Progress: The course achieved currently on average **85.3% of 23 points**

Figure 3.18: Example of the modal that opens when an instructor clicks on a learning goal in the proposed system

A new learning goal can be created by clicking on the plus button on the management page. Changing or deleting a goal is done with the buttons at the bottom of each card. Figure 3.19 shows an example of the learning goal edit page. The goal can be linked to lecture content by selecting the content in the table at the bottom of the screen. In the example, the learning goal is linked to four exercises from the lecture "L10 Edge Computing in Industry".


Development
Artemis 4.10.5
Course Overview
Course Management
Server Administration
1+
manuel_johannes
Course Management > Patterns in Software Engineering (WS 20/21) > Learning Goal Management > Edit

Edit a learning goal

Learning Goals*

05 Edge Computing

Description

Edge computing is a distributed computing paradigm that brings computation and data storage closer to the location where it is needed, to improve response times and save bandwidth.

Connect the learning goal with lecture units

L10 Edge Computing in Industry (4 unit(s) connected)

id	Type	Lecture Unit	Release Date
142	attachment	L10 U01 Edge Computing in a Nutshell	
147	video	L10 U01 Edge Computing in a Nutshell	
137	exercise	L10E01 Fogxy Quiz	Jan 25 2021, 11:05:00
143	attachment	L10 U02 Industrial Edge	
146	video	L10 U02 Industrial Edge	
138	exercise	L10E02 Limitations of Cloud Computing	Jan 25 2021, 11:45:00
139	exercise	L10E03 Observer and Publish Subscribe	Jan 25 2021, 11:45:00
144	attachment	L10 U03 Programming Exercise	
145	video	L10 U03 Programming Exercise	
140	exercise	L10E04 Infrastructure Automation with Edge Computing	Jan 25 2021, 12:40:02
141	attachment	Announcements	Jan 25 2021, 13:10:00

Figure 3.19: Example of the learning goal editor in the proposed system

Exercise Performance Chart

Figure 3.20 shows the exercise performance chart. This chart can be found in the learning analytics dashboard. On the horizontal axis, the exercises are listed in ascending order according to their release date. Using the scroll bar below the chart, a user can scroll through the exercises of the course. The vertical axis shows the achieved result in percent. A student can use the visualization to determine how the development of their own performance (blue) compares to the development of the average (red) and best (green) performance in the course. By clicking on a data point, the student can navigate to the respective exercise page for more information.

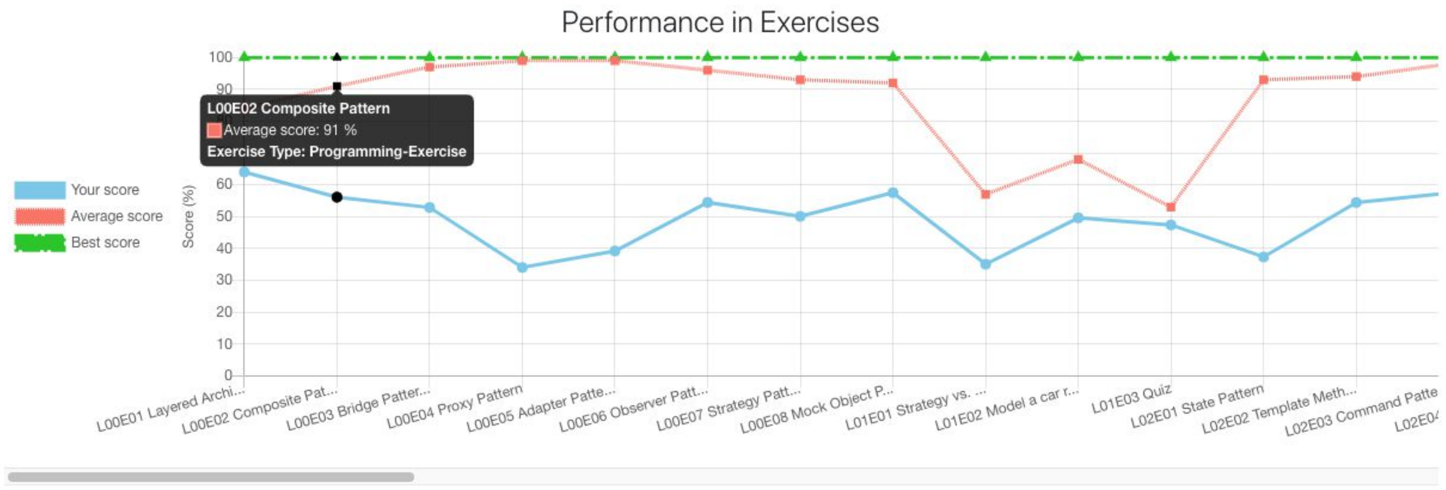


Figure 3.20: Example of the exercise performance chart in the proposed system

Chapter 4

System Design

This chapter presents the mapping of the application domain concepts from Chapter 3 to the solution domain. It follows the System Design Document Template described in [BD09].

The proposed system does not change the core technologies used for the implementation of Artemis. Artemis uses a client-server architecture. In the client, the Typescript-based web application framework Angular is mainly used. In the server, the application framework Spring Boot with the programming language Java is used. MySQL is used as a relational database, with Hibernate as an object-relational mapping tool. Client and server communicate via REST APIs and web sockets.

4.1 Design Goals

In this section, we present the design goals that guide the implementation of the proposed system. These goals are derived from the non-functional requirements described in Section 3.3.2. Design goals are an important basis for decision-making in development, as there are always various alternative implementation options [BD09]. The design goals are presented in descending order of priority:

1. **Performance**: Students will not regularly use the learning analytics dashboard if the page does not load quickly (NFR 3.1). The redesigned lecture page will only be positively received if students can access the content without long loading times (NFR 1.5). Therefore achieving good performance is the most important design goal.

2. **Usability**: It is a challenge to get existing users to use new features. Good usability can help to increase the adoption rate. Therefore, high

usability for both students and instructors is an important design goal in the lecture redesign (NFR 1.1), the learning goals (NFR 2.2 and NFR 2.3), and the learning analytics dashboard (NFR 3.2).

We also have made some design goal trade-offs:

- **Functionality vs. Usability:** As mentioned above, high usability is a crucial for achieving a high adoption rate of new features. That is why we choose a simple and understandable user interface over a complex user interface with many configuration options.
- **Rapid Development vs. Functionality:** At the end of the 5-month work period, the new system's core features should be fully usable. Therefore, it is important to implement the core features first. Non-core functionality can be dropped from the proposed system to meet the schedule.

4.2 Subsystem Decomposition

To make the complexity of the solution domain manageable, the system is divided into subsystems. Subsystems are groupings of related solution domain classes. Each subsystem is described by the services it provides to other subsystems. Services are a grouping of several operations with a common goal [BD09]. This section describes the subsystem decomposition of the proposed system.

To create the subsystem decomposition, we look at the use cases, functional requirements, and non-functional requirements described in Chapter 3. The design goals described in Section 4.1 help us to decide between implementation alternatives.

Figure 4.1 shows the subsystem decomposition of the proposed system. In the upper half of the figure, the client's subsystem decomposition can be seen, and in the lower half, that of the server. Client and server communicate via REST APIs. The services that offer a REST API can be seen in the area between client and server.

4.2.1 Server Subsystem Decomposition

When we look at the functional requirements and use cases described in Chapter 3, we can identify the need for a lecture content subsystem that deals with the new types of content that can be added to a lecture. Instead

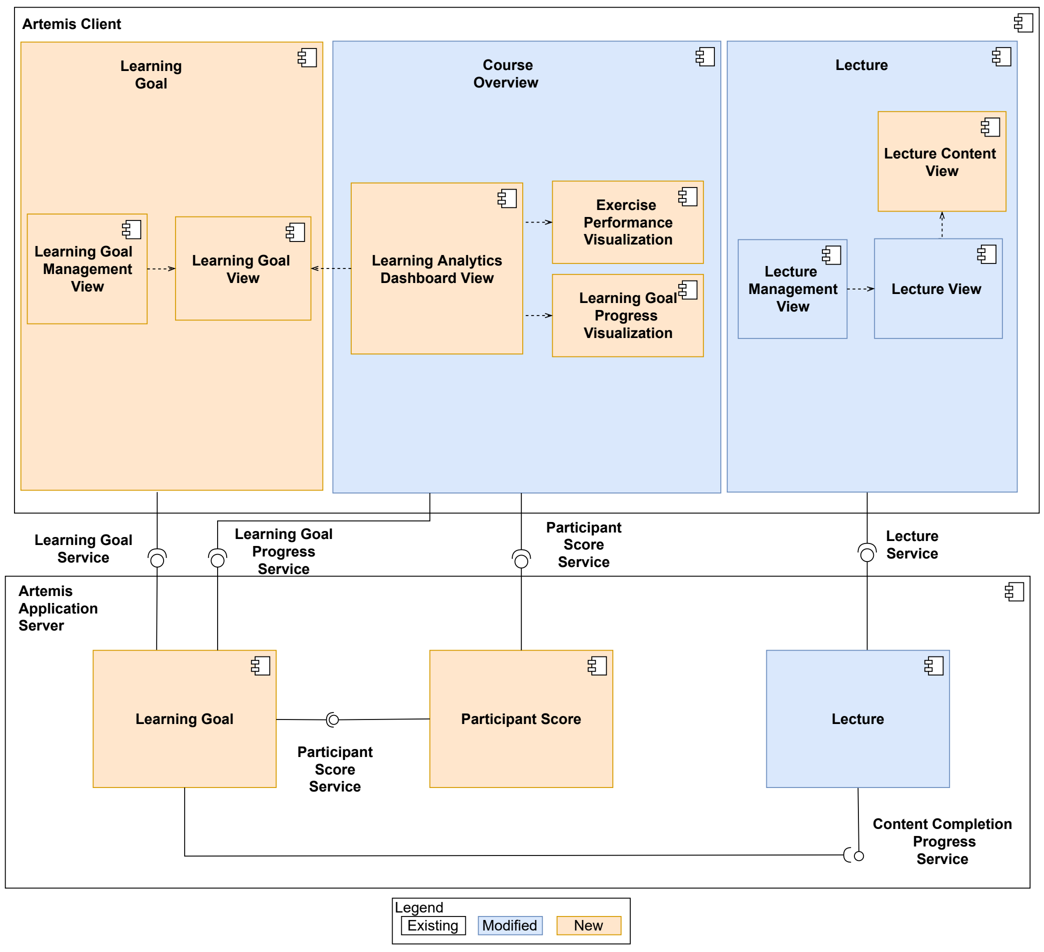

Artemis Client
Learning Goal
Learning Goal Management View
Learning Goal View
Course Overview
Learning Analytics Dashboard View
Exercise Performance Visualization
Learning Goal Progress Visualization
Lecture
Lecture Content View
Lecture Management View
Lecture View
Learning Goal Service
Learning Goal Progress Service
Participant Score Service
Lecture Service
Artemis Application Server
Learning Goal
Participant Score
Lecture
Participant Score Service
Content Completion Progress Service
Legend
Existing
Modified
New

Figure 4.1: Subsystem decomposition of the proposed system

of introducing a new subsystem, it makes sense to integrate the content-related functionality into the existing *lecture subsystem*. This subsystem is connected to the database and is responsible for retrieving, creating, updating, and deleting lectures and their content. This management and retrieval functionality is part of the *lecture service* that the subsystem offers to the client. The subsystem is also responsible for tracking the content completion progress. When a student engages with lecture content (e.g., watches a lecture recording), data about this interaction (e.g., how much of the video the student has watched) is sent from the client to the server via the *lecture service*. The lecture subsystem on the server then uses this interaction data to calculate a student's progress in completing lecture content. This progress information is offered to the new *learning goal subsystem* via the *content completion progress service*. It is important to note that the *lecture subsystem* provides progress information for all lecture content types except exercises. The high-performance delivery of information about students' results in exercises is a relatively complex task that is also important for exercises that are not connected to lectures (e.g., exam exercises). Therefore the new *participant score subsystem* was created.

We can also identify the need for a *learning goal subsystem.* This subsystem is connected to the database and is responsible for retrieving, creating, updating, and deleting learning goals. This management and retrieval functionality is part of the *learning goal service* that the subsystem offers to the client. As described in the requirements analysis, the calculation of a student's progress in mastering a learning goal is calculated from the student's progress in completing all content linked to the learning goal. A learning goal is considered completed when the student has completed all linked content. The *learning goal subsystem*'s task is to calculate the *learning goal progress* and offer it to the client via the *learning goal progress service.* The progress information is used in the learning goal section of the learning analytics dashboard. The learning goal subsystem receives the progress information for exercises from the *participant score subsystem.* It receives the progress information for the other types of content from the *lecture subsystem.*

As mentioned before, the new *participant score subsystem*'s task is the high-performance delivery of information about achieved scores in exercises to various parts of Artemis, including the learning analytics dashboard in the client. In the dashboard, this information is used to generate the exercise performance visualization. We call this subsystem participant score subsystem as it deals with the scores a participant (a student or a team of students) has achieved in exercises. The inner workings of this subsystem are explained in more detail in Chapter 5.

4.2.2 Client Subsystem Decomposition

The *learning analytics dashboard view* is included in the existing *course overview subsystem.* The learning analytics dashboard consists of two sections. One section deals with learning goals and displays the *learning goal view* combined with the *learning goal progress visualization* for each goal. In this section, students can see their progress in mastering the learning goals of the course. Instructors can see the average progress of all students in mastering the learning goals. A second section contains the *exercise performance visualization* that students can use to determine how the development of their own performance in exercises compares to the development of the average and best performance in the course.

For each lecture content type, a new *lecture content view* is implemented and added to the client's existing *lecture subsystem*. When a student opens the *lecture view*, all the embedded *lecture content views* are also loaded. The current system's *lecture management view* is extended with management capabilities for the new lecture content types.

Finally, there are new views to display and manage learning goals. These are included in a new *learning goal subsystem* in the client. As mentioned above, we decided to display the learning goals and their progress in the learning analytics dashboard.

4.3 Persistent Data Management

This section describes the mapping of application domain objects (see Section 3.4.3) to the solution domain's entities. We describe here the entities, i.e., Java classes, that are being mapped to the relational MySQL database in the Artemis application server by the object-relational mapping tool Hibernate[1]. To improve clarity, we have divided the description of the solution domain model into two figures. Figure 4.2 shows the most important changes of the proposed system. The new *ParticipantScore* entity is simplified in the figure for the sake of clarity. It is explained in detail in Figure 4.3.

We have decided to use the term lecture unit in the solution domain to refer to the various kinds of content an instructor can add to a lecture. The abstract class *LectureUnit* is the superclass of all content types that an instructor can add to a lecture. The connection of *Lecture* to *LectureUnit* is ordered, meaning an instructor can determine the order in which the lecture units are presented to students when they open a lecture page. *LectureUnit* is an abstract class to increase the supportability of the system. Future developers can implement new types as subclasses. All lecture units should have a release date and a name; therefore, these attributes are moved to the common superclass. We also decided that instructors can give lecture units points. For calculating a student's progress in mastering a learning goal, the progress in completing related lecture units is used. A lecture unit's points indicate how much the progress in a particular lecture unit is weighted in the learning goal progress calculation.

TextUnits represent written notes, *VideoUnits* represent video recordings, *AttachmentUnits* represent uploaded files, and *ExerciseUnits* represent exercises related to a lecture.

[1]`https://hibernate.org/`

[2]Attributes of *Student*, *Course*, *Lecture*, *Attachment*, *Exercise*, and *ParticipantScore* are fully or partially omitted for the sake of clarity. The model also only considers course exercises for simplicity and not exam exercises.

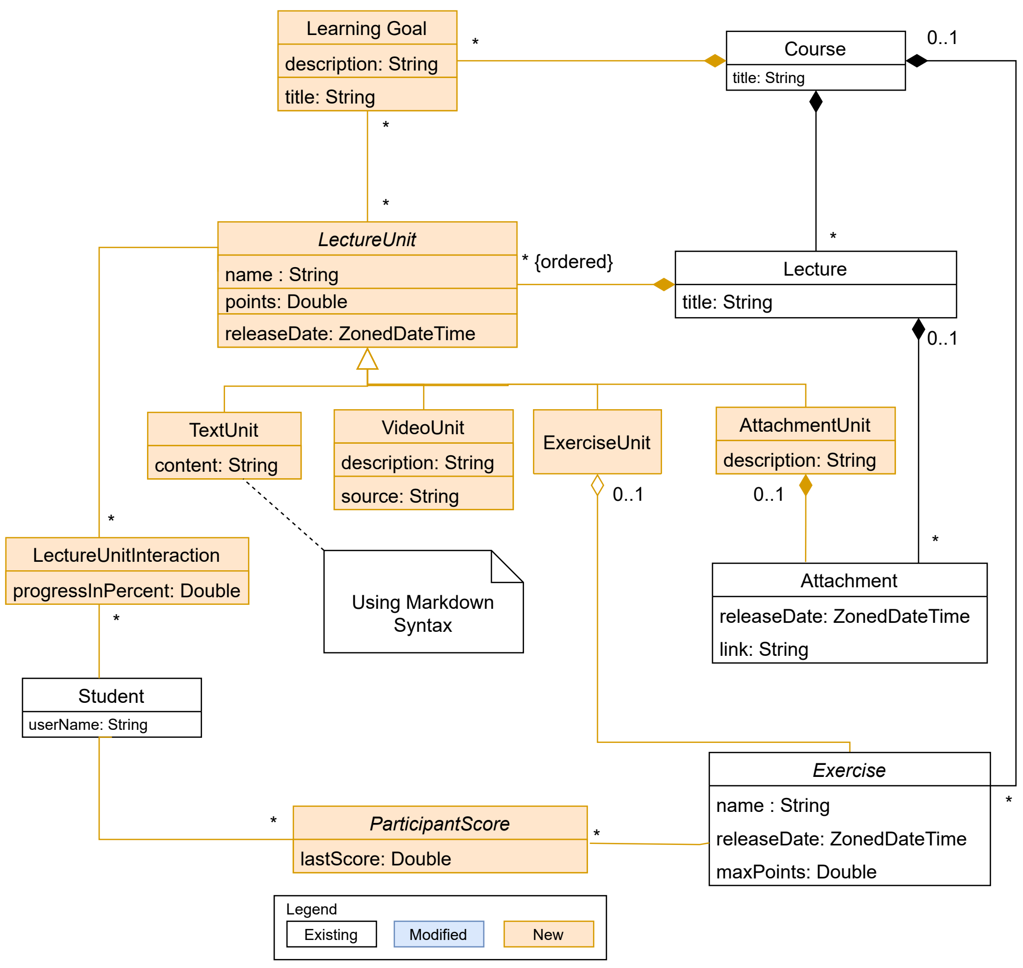


Figure 4.2: Overview of the changes to the solution domain entities in the proposed system[2]

For embedding notes in a lecture, we decided to use the existing Markdown[3] component in the Artemis client. This component offers the possibility to write text, format it using Markdown syntax, and include images. When a student opens the page of a lecture with embedded notes, the *TextUnit* is loaded from the database and sent to the client. The client then converts the content into HTML for display.

Instructors can use video units to embed any video (e.g., lecture recordings) on the lecture page. They have to upload the video to some video-sharing platform (e.g., YouTube) that supports embedded video players and copy the video's embeddable link into the video unit. This link is stored in

[3]https://daringfireball.net/projects/markdown/

the source attribute of the *VideoUnit* entity. When loading a lecture with a video, the client embeds the video directly on the lecture page.

Originally we had the idea to make *Attachment* and *Exercise* direct subclasses of *LectureUnit*. However, we rejected this idea, as it would have required great changes to the underlying database schema. Therefore, we decided to create the entities *ExerciseUnit* and *AttachmentUnit*. With the help of these new entities, the existing database schema of exercises and attachments can remain unchanged. Instructors select the course exercises they want to add to a lecture in the client. For those selected, an *ExerciseUnit* is created. The name, points, and release date are taken from the underlying exercise. The release date of an *AttachmentUnit* is taken from the release date attribute of the underlying *Attachment*. The link attribute stores the link to the uploaded file. Instructors are not required to use the new lecture format. Therefore, it is still possible to add one or more attachments directly to a lecture without using an attachment unit. This also saves us from creating a migration script to convert older lectures in the database to the new format.

Interactions of a student with a lecture unit are persisted using the new entity *LectureUnitInteraction*. This entity stores how far along a student is in working through a lecture unit.

A new entity is also introduced for *Learning Goals*. Learning goals are set by the instructor at the course level and can be linked to several lecture units. A student's progress in mastering a learning goal is calculated from their progress in completing related lecture units. For getting the progress of text units, video units, and attachment units the *LectureUnitInteraction* entity is used. For exercise units, the student's last achieved score in the underlying exercise is used. Students should have the ability to improve in learning goals, so we always consider the result of their last submission, even if they actually submitted their solution after the official deadline. This means, for example, that students can progress in learning goals by re-taking quizzes that they may have already completed. The last achieved score of a student in an exercise is stored in the new *ParticipantScore* entity.

Figure 4.3 offers a detailed view of the new *ParticipantScore* entity and its purpose. An *Exercise* in Artemis can either be an individual exercise or a team exercise. Teams are groups of students that work together on an exercise. Teams exist on the exercise level (relationship between *Team* and *Exercise* not shown in the figure for simplicity), meaning each exercise in the

[4] Attributes of *Student*, *Team*, *Exercise*, *Course*, *StudentParticipation*, *Submission*, and *Result* are fully or partially omitted for the sake of clarity. We also removed the relationship between *Team* and *Exercise* for clarity. The model also only considers course exercises for simplicity and not exam exercises.

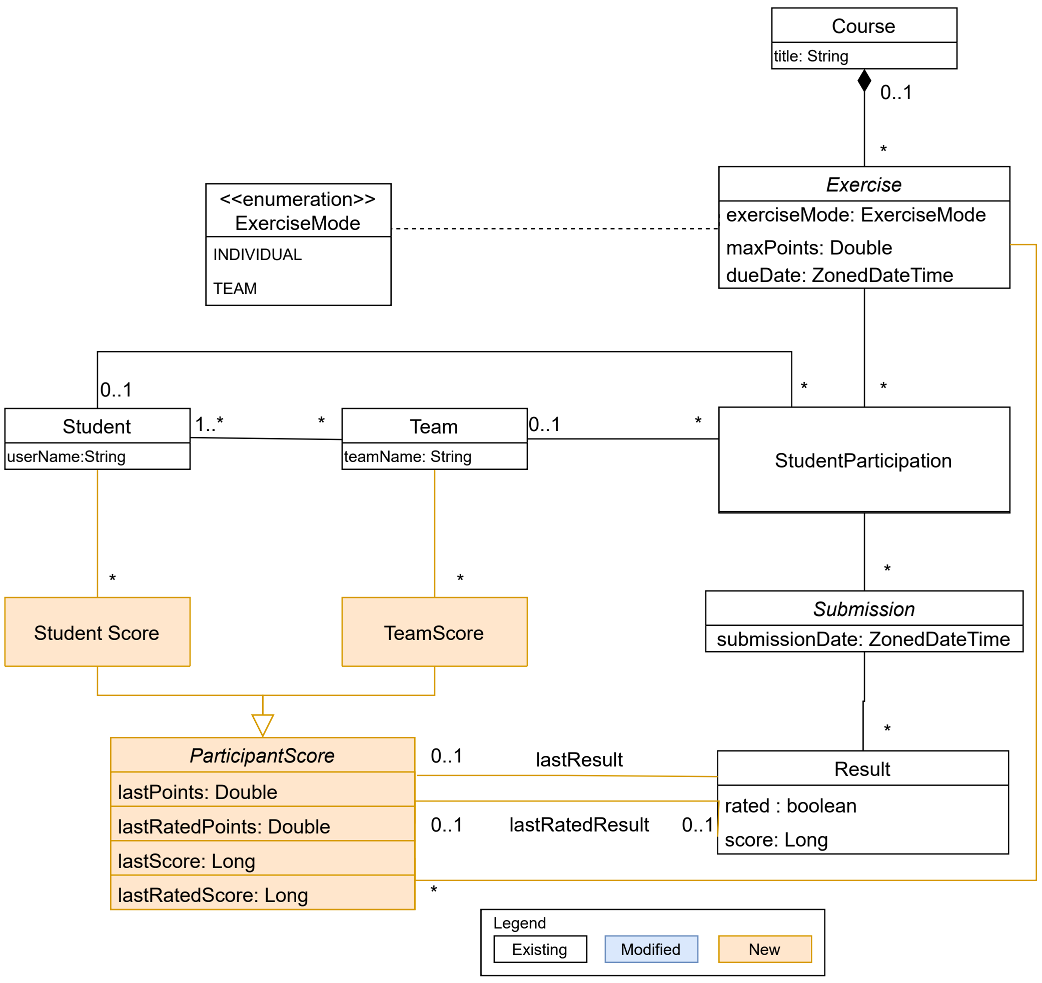


Figure 4.3: Detailed view on the *ParticipantScore* entity[4]

course can have different teams. We use the generic term participant to refer to either a team of students or a single student. When a participant first takes part in an exercise, a *StudentParticipation* is created. For each solution submitted by the participant, a submission is created and connected to the participation. Each time the submission is checked for correctness, either manually by tutors or automatically, a *Result* is created. The score attribute represents the percentage of points the participant achieved. For example, a score of 50 in an exercise that gives 10 points means that the participant has achieved 5 points. The maximum points and bonus points a participant can achieve in an exercise are saved in the *Exercise* entity. For exercises, students usually have a certain amount of time to submit their solutions. The due date is stored in the *Exercise* entity. If a participant submits their solution before the due date of an exercise, the result is marked as rated; otherwise, it is marked as unrated.

In the exercise performance visualization of the learning analytics dashboard, Artemis displays for each exercise the points achieved by the logged-in student, the average points achieved in the course, and the maximum points achieved in the course. The system only considers rated results for this visualization as this means all students had the same amount of working time. Using the entities existing in the current system, Artemis has to traverse several links between entities to find the score a participant achieved in an exercise. Since such a query affects several tables, there is a significant performance hit. Therefore we have created the new abstract entity *ParticipantScore* with the concrete subclasses *StudentScore* and *TeamScore*. This entity is linked to a participant and an exercise. It is also linked to the last result and last rated result of the participant in the exercise. The system stores the corresponding score and achieved points in the entity. Queries involving the achieved score of a participant in an exercise are sped up using the new entity, as all the necessary information is stored in a single table in the database. This removes the need to traverse several links between entities. As a result, the exercise performance visualization can be loaded very quickly, even for large courses.

Chapter 5 explains in detail how the *ParticipantScore* entity is created and updated.

4.4 Access Control

In this section, we present the access control policy of the proposed system. Artemis has four roles a user can have in a course: Student, Tutor, Instructor, or Administrator. Whether a user is allowed to perform a certain action depends on their roles. Table 4.1 shows what roles are allowed to perform which action in the proposed system. The shorthand for yes is "y," and the shorthand for no is "n" in the table.

Every user in a course can access the learning goals, but only instructors and administrators can create, update or delete them. Students can check their own progress in learning goals. They can also check the average progress of the rest of the course for comparison. Only instructors and administrators can view the progress of each student.

The access rights for lecture units are almost identical to those for learning goals, except that lecture units have a release date, and students can only access lecture units that have already been released. Tutors, instructors, and administrators can access all lecture units of a lecture.

Participant scores are generated, updated, and deleted completely automatically (see Chapter 5). Therefore, no endpoints are offered for manual modification. Only instructors and administrators are allowed to query individual participant scores for control purposes. Every user of the course can see the performance visualization in the learning analytics dashboard.

	Student	**Tutor**	**Instructor**	**Admin**
Learning Goal				
Read	y	y	y	y
Create	n	n	y	y
Update	n	n	y	y
Delete	n	n	y	y
Query personal progress	y	y	y	y
Query average progress of course	y	y	y	y
Query individual progress of all students	n	n	y	y
Lecture Unit				
Read	y (only released)	y	y	y
Create	n	n	y	y
Update	n	n	y	y
Delete	n	n	y	y
Query personal progress	y	y	y	y
Query average progress of course	y	y	y	y
Query individual progress of all students	n	n	y	y
Participant Score				
Read	n	n	y	y
Create	n	n	n	n
Update	n	n	n	n
Delete	n	n	n	n
Query exercise statistics (personal, average and best performance)	y	y	y	y

Table 4.1: Access control matrix of the proposed system

Chapter 5

Object Design

This chapter focuses on specific implementation details; in particular, we discuss in Section 5.1 how the *ParticipantScore* entities are automatically created, updated, and deleted.

5.1 Participant Scores

As already described in Section 4.3, the *ParticipantScore* entity (with the concrete subclasses *StudentScore* and *TeamScore*) functions as a shortcut to quickly determine the achieved score (or points) of a participant (student or team) in an exercise. To achieve this, the participant score entity is linked to a participant and an exercise. It is also linked to the last result (rated or not) and the last rated result of the participant in the exercise. "Last" means that this is the most recently created result. A result is rated if the student has submitted the corresponding solution before the submission deadline. The achieved score and points are stored directly in the entity.

There are different scenarios where Artemis creates multiple results for a participant and an exercise. For example, each time a participant submits a solution to a programming exercise, the code is automatically checked using corresponding test cases. Depending on the number of test cases passed, a score is then calculated and saved in a new result. Later, tutors manually check and evaluate the submissions, whereby a new result is created again. Although several results exist in the database, in most cases, Artemis is only interested in the last rated result, as this is the official score achieved in an exercise. However, for calculating the progress of a student in an exercise unit, we consider the last result, rated or not, as students should have the ability to improve in connected learning goals even though the official deadline has passed. Therefore both the last result and the last rated result are of interest.

To keep the participant score table correct, the table must be updated when creating, updating, and deleting results. For example, if a new result is created for a student and an exercise, the corresponding participant score entity must also be updated. We wanted to keep the implementation as maintenance-free as possible. This means that not every current or future spot in the codebase that affects results should have to be touched to keep the table updated. Therefore, we decided to implement a so-called entity listener.

Entity listeners [1] are a concept specified in the Jakarta Persistence API (formerly Java Persistence API) [2] and implemented by Hibernate [3]. Hibernate is the object-relational mapping tool used by Artemis.

An entity listener allows the developer to define callback methods for life cycle events of a particular entity. For updating the participant scores, we are using an entity listener for the *Result* entity. Whenever a new result is created, updated, or deleted, the entity listener's callback methods are executed. These callback methods are then creating, updating, or deleting the corresponding *ParticipantScore* entities.

Figure 5.1 shows the logic that the entity listener executes after a result entity is created (post-persist) or updated (post-update) in the application logic. This would be the case, for example, when a tutor corrects a student's submission or an instructor manually changes a student's score. For simplicity, the figure ignores the distinction between rated and unrated results. First, the system determines which exercise and participant the result concerns. Then it checks if there is already a participant score for this particular exercise and participant. If none exists, a new participant score entity is created, and the result is set as the last result. If there is already a participant score, it is checked if the result is equal (update case) to or newer (create case) than the result currently set as the last one. If neither is the case, the system can ignore the result; otherwise, the system updates or replaces the participant score entity's last result and corresponding score / point attributes. As soon as the callback function is completed, the application logic continues.

[1] `https://docs.jboss.org/hibernate/stable/entitymanager/reference/en/html/listeners.html`

[2] `https://jakarta.ee/specifications/persistence/`

[3] `https://hibernate.org/`

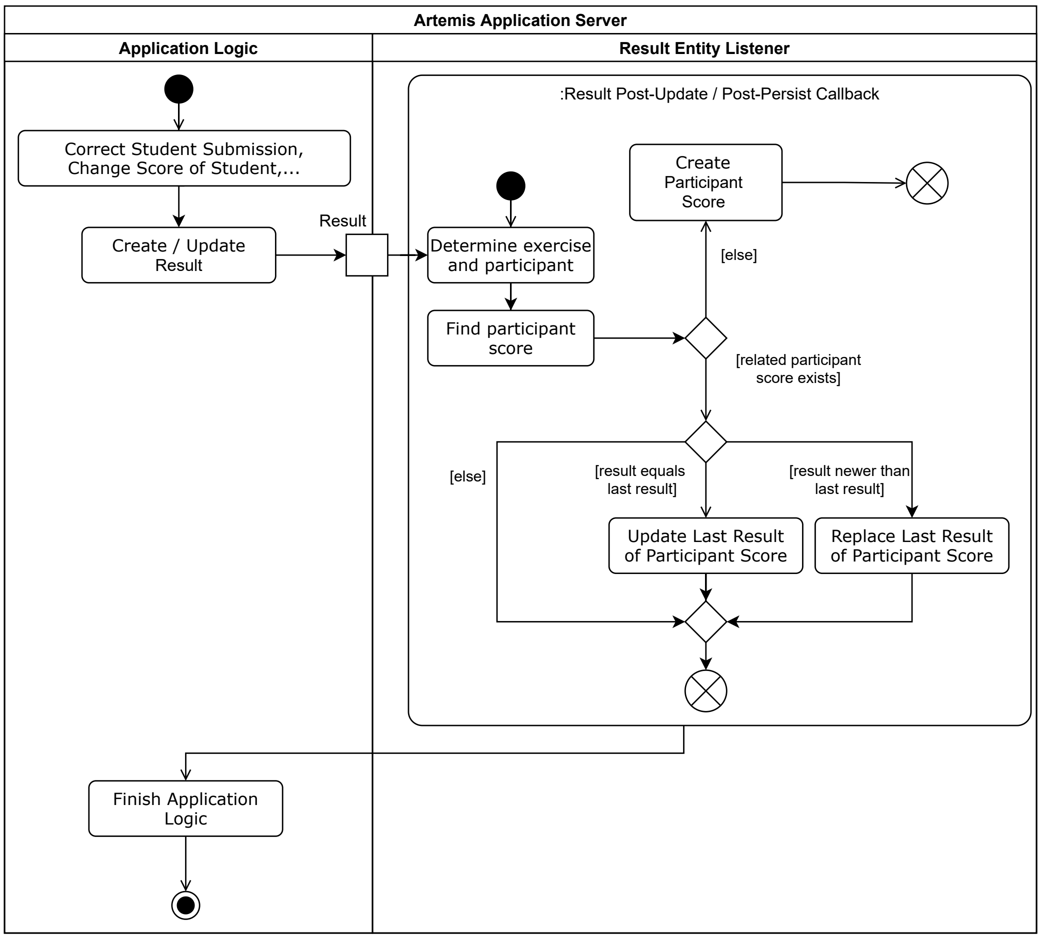

Artemis Application Server
Application Logic
Result Entity Listener
Correct Student Submission, Change Score of Student,...
Create / Update Result
Result
:Result Post-Update / Post-Persist Callback
Determine exercise and participant
Find participant score
Create Participant Score
[else]
[related participant score exists]
[else]
[result equals last result]
[result newer than last result]
Update Last Result of Participant Score
Replace Last Result of Participant Score
Finish Application Logic

Figure 5.1: Result Post-Update / Post-Persist Callback Logic

Figure 5.2 shows the logic that the entity listener executes before a result entity is deleted (pre-remove) in the application logic. This would be the case, for example, when an instructor deletes a student's submission or result. For simplicity, the figure ignores again the distinction between rated and unrated results. First, it is checked if there exists a participant score that is connected to that result. If none exists, the result is deleted, and the business logic continues. If there is a participant score related to the result, it is checked if a penultimate result can be used as a replacement. If no replacement exists, the participant score is deleted. If one exists, the last result of the participant score is replaced with the penultimate result. In either case, the result is deleted, and the application logic continues.

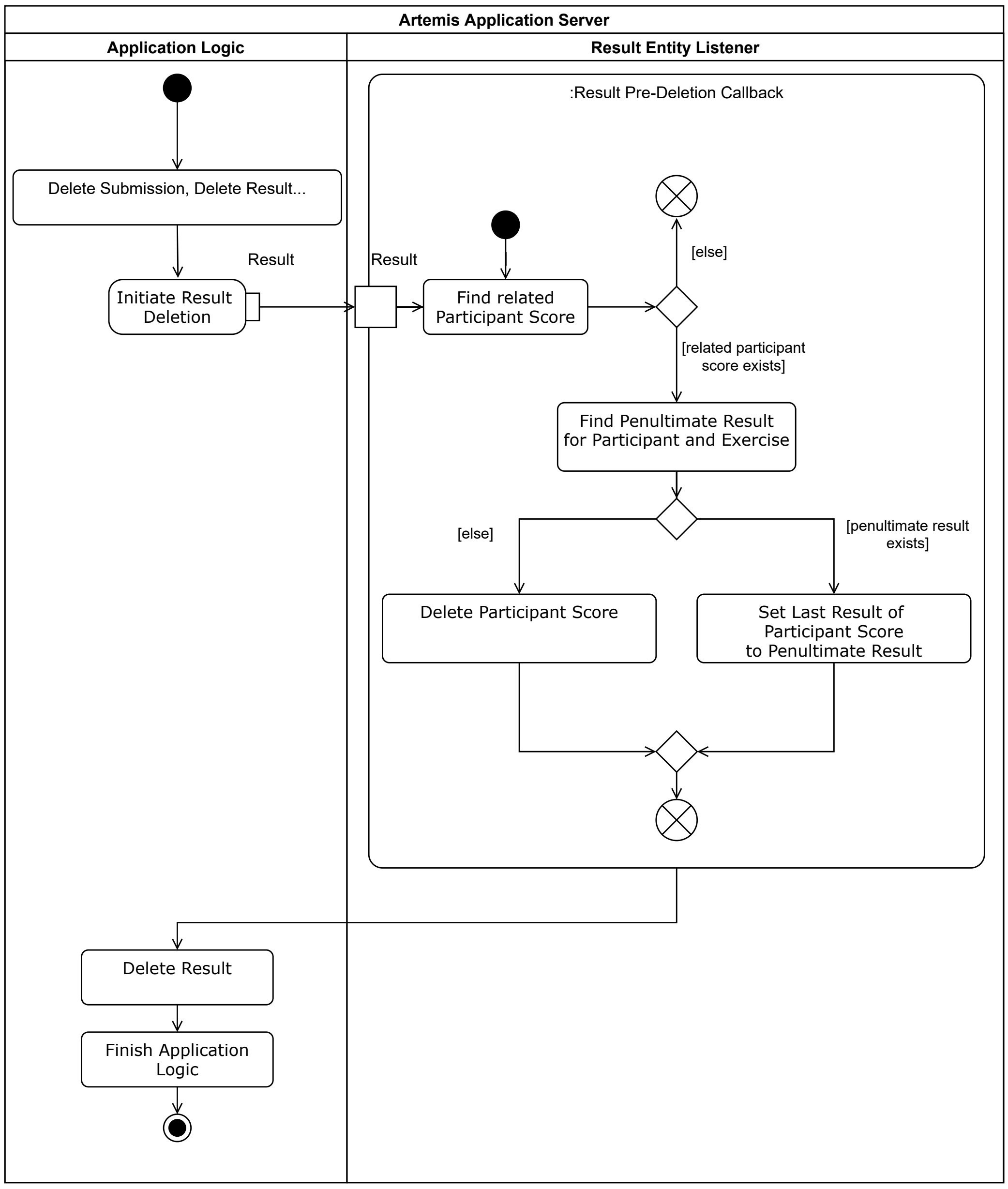

Artemis Application Server
Application Logic
Result Entity Listener
:Result Pre-Deletion Callback
Delete Submission, Delete Result...
Result
Initiate Result Deletion
Result
Find related Participant Score
[else]
[related participant score exists]
Find Penultimate Result for Participant and Exercise
[else]
[penultimate result exists]
Delete Participant Score
Set Last Result of Participant Score to Penultimate Result
Delete Result
Finish Application Logic

Figure 5.2: Result Pre-Remove Callback Logic

Chapter 6

Summary

This chapter summarizes this work and presents the status quo. We discuss realized and open goals in Section 6.1. We summarize in Section 6.2 the importance of this work for Artemis. Finally, in Section 6.3, we present suggestions for further work on learning analytics in Artemis.

6.1 Status

In this section, we compare the requirements we described in Chapter 3 with the system's current status. The following notation is used to highlight the implementation status of the requirements:

● **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.

Table 6.1 gives an overview of the implementation status for the functional requirements. Table 6.2 shows the implementation status of the non-functional requirements.

Functional Requirement	**Status**
Lecture Redesign	
FR1.1 Provide lecture recordings to students	●
FR1.2 Provide lecture notes to students	●
FR1.3 Provide files to students	●
FR1.4 Provide related exercises to students	●
FR1.5 Track lecture completion progress of students	○
FR1.6 Set release date	●
Learning Goals	
FR2.1 Define learning goals	●
FR2.2 Progress in learning goals	◐
FR2.3 Define learning paths	○
Learning Analytics Dashboard	
FR3.1 View development of own exercise performance	●
FR3.2 View course exercise performance development	●
FR3.3 View own learning goal progress	●
FR3.4 View course learning goal progress	●

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

Non-functional Requirement	**Status**
Lecture Redesign	
NFR1.1 Usability(Efficiency)	●
NFR1.2 Reliability(Robustness)	●
NFR1.3 Supportability(Adaptability)	●
NFR1.4 Supportability(Learnability)	●
NFR1.5 Performance(Response Time)	●
Learning Goals	
NFR2.1 Usability(Efficiency)	●
NFR2.2 Usability(Efficiency)	●
NFR2.3 Usability(Learnability)	◐
Learning Analytics Dashboard	
NFR3.1 Performance(Response Time)	●
NFR3.2 Usability(Learnability)	●

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

6.1.1 Realized Goals

In total, 10 of the 13 functional requirements are fully implemented.

Instructors can now attach videos (FR 1.1), notes (FR 1.2), files (FR 1.3), and exercises (FR 1.4) to lectures. We also managed to meet the non-functional requirements: The lecture content is all presented on the lecture page (NFR 1.1) in a similar design (NFR 1.4) in the instructor's order. Even a lecture page with a lot of content loads quickly (NFR 1.5). When creating lecture notes, the writing progress is saved so that even longer notes can be written without the risk of data loss (NFR 1.2). New content types can be simply defined as subclasses of the abstract entity *LectureUnit* (NFR 1.5) (see Section 4.3). The new lecture format was already used in the winter semester 2020 / 2021 course *"Patterns in Software Engineering"* at the Technical University of Munich. Due to the worldwide coronavirus pandemic, the lecture had to take place without face-to-face teaching. All 13 lectures of the course used the new lecture format to provide students with learning material (including lecture recordings).

The instructor of the *"Patterns in Software Engineering"* course also defined 5 learning goals (NFR 2.1) and linked them to exercises. Students can view their learning goal progress (FR 2.2) in the learning analytics dashboard (FR 3.3). Students and instructors can also view how far the course is on average in mastering the learning goals (FR 3.4). However, we only partially succeeded in implementing the learning goal progress feature (FR 2.2), as currently, only a student's performance in connected exercises plays a role. No progress tracking is currently implemented for the remaining types of lecture content. We were able to meet the usability requirements successfully. An instructor can easily create a learning goal (NFR 2.1) and link it to lecture content (NFR 2.2). The content just has to be marked in a table, and it will be linked to the learning goal. Good learnability (NFR 2.3) could only be partially achieved. The meaning and calculation logic of the learning goal progress is not obvious. An explanatory text is shown to the user.

The learning analytics dashboard is implemented successfully (FR 3.1-FR 3.4) in functional and non-functional terms (NFR3.1-NFR 3.2). Thanks to the new *ParticipantScore* entity (see Section 4.3 and Chapter 5), the system is now able to calculate the necessary exercise performance statistics very efficiently in order to make them available in a visualization on the learning analytics dashboard.

6.1.2 Open Goals

We failed in meeting two functional requirements. The system cannot currently track a student working through all lecture content types (FR 1.5). Measuring how much of a text a student has read or how much of a video he or she has actually watched with attention is not trivial to implement and was not feasible within the scope of this thesis. Therefore, as described above, only the linked exercises' performance is currently used to calculate the learning goal progress. Due to time constraints, we could also not implement the linking of learning goals to learning paths (FR2.3). Consequently, recommended learning paths currently have to be communicated to the student indirectly, e.g., by the instructor in the lecture.

6.2 Conclusion

With this work, we have added important learning analytics features to Artemis, laying the groundwork for further enhancements in this area. Previously, students lacked the context to judge their performance in the course. Now they can see in the learning analytics dashboard how they compare with the rest of the course. The first step towards competency-based learning was taken with the introduction of learning goals. Students can now clearly identify which core competencies they should master in the course. Finally, the concept of a lecture was made more dynamic. Before, a lecture was just a place in Artemis where the instructor attached the lecture slides. Now a lecture is the central place in Artemis where all course material is collected. Since recordings, notes, files, and exercises can now be embedded on a lecture page, a larger part of the learning process is available for further learning analytics steps.

6.3 Future Work

In addition to the unrealized goals described above, there are other opportunities to expand Artemis' learning analytics capabilities:

Adaptive Learning: Dynamically adapting exercise difficulty

Adaptive learning means that a student's interaction with learning content determines in part the material that is subsequently recommended to him or her by the system. The goal of this process is to create a personalized learning experience [Ker16]. In the context of Artemis, a dynamic adaptation of the exercise difficulty level to the student's abilities would be useful. Typical for introductory courses in computer science is the sudden increase in the difficulty of programming exercises. This causes problems, especially for students without much previous programming experience. If Artemis adapts the pace of the increase in difficulty to the respective student, i.e., recommends easier exercises to weaker students until they are ready for more difficult exercises, frustration can be prevented. Exercises in Artemis already have a difficulty attribute, which could be used in the implementation. A new exercise grouping entity is needed to indicate that the exercises cover the same topic and differ in difficulty. The student can then either choose exercises from this group freely or be guided by the recommendation logic. A simple implementation could be that an easy exercise is randomly presented to the student until they achieve 80 % of the points. Then the remaining easy exercises are skipped, and a moderately difficult exercise is recommended at random. This continues until the student has achieved 80 % in an exercise of the hardest level.

Manual Learning Path Construction

Due to time constraints, it was not possible to implement the linking of several learning goals to learning paths within this thesis's scope. For the development of learning paths, instructors should be able to define learning goals as prerequisites for other learning goals. The student can then determine how to move from their current level of knowledge to a desired level of knowledge. In addition, other types of links seem useful. For example, linking learning goals that are conceptually related but do not have a clear prerequisite relationship. A subdivision of learning goals into smaller subordinate learning goals by means of a parent-child relationship would allow for finer-grained modeling capabilities. A visualization technique for the client would be a graph with directed (prerequisite links) and undirected (conceptual links) edges. If the user clicks on a learning goal in the graph, the subordinate learning objectives goals appear.

Automated Learning Path Construction

It would be interesting if Artemis users could create their own learning paths by indicating which learning goals they have already mastered and which ones they would like to master. Artemis could then generate an individual learning path to bridge the gap between initial knowledge and desired knowledge. This could even be done across courses. For example, imagine the scenario that someone with no programming experience wants to learn how to build a website. The system would then generate an individual learning path for this user, containing material from various courses such as an introduction to programming course and a web development course. In addition to the linking of learning goals to learning paths (see above), the implementation also requires the possibility of defining entire courses as prerequisites for other courses and the linking of learning goals across courses. When the system tries to generate a path, either a direct link between learning goals can be used or, if not available, a link between courses.

Semester Long Projects

Programming courses in Artemis currently contain many relatively small weekly programming assignments for students to complete. It would help competency-based learning if Artemis allowed a course to be structured around a semester-long programming project. Students are given access to a git repository at the beginning of the course, where they store the project's progress. Snapshots of the project are taken at set times and assessed by tutors. An example of such a project would be the development of a compiler. This course design has many advantages: Students tackle a challenging development task where each step builds on the previous one. Along the way, students grapple with large-scale development problems and integrate knowledge from the entire course.

List of Figures

List of Tables

Bibliography

[Akp20] Ben Akpan. Mastery learning—benjamin bloom. In *Science Education in Theory and Practice*, pages 149–162. Springer, 2020.

[BD09] Bernd Bruegge and Allen H Dutoit. *Object Oriented Software Engineering Using UML, Patterns, and Java.* Prentice Hall, 2009.

[Blo68] Benjamin S. Bloom. Learning for mastery. instruction and curriculum. regional education laboratory for the carolina and virginia, topical papers and reprints, number 1. *Evaluation comment*, 1(2):n2, 1968.

[BT11] J. Biggs and C. Tang. *Teaching For Quality Learning At University.* SRHE and Open University Press Imprint. McGraw-Hill Education, 2011.

[CDB15] Linda Corrin and Paula De Barba. How do students interpret feedback delivered via dashboards? In *Proceedings of the fifth international conference on learning analytics and knowledge*, pages 430–431, 2015.

[CJB05] Hamish Coates, Richard James, and Gabrielle Baldwin. A Critical Examination Of The Effects Of Learning Management Systems On University Teaching And Learning. *Tertiary Education and Management*, 11(1):19–36, 2005.

[Dor20] Shayan Doroudi. Mastery learning heuristics and their hidden models. In *International Conference on Artificial Intelligence in Education*, pages 86–91. Springer, 2020.

[GKR14] Philip J Guo, Juho Kim, and Rob Rubin. How video production affects student engagement: An empirical study of mooc videos. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 41–50, 2014.

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

[Kan14] Peter Kandlbinder. Constructive alignment in university teaching. *HERDSA News*, 36(3):5–6, 2014.

[Ker16] Philip Kerr. Adaptive learning. *Elt Journal*, 70(1):88–93, 2016.

[KJP16] Jeonghyun Kim, Il-Hyun Jo, and Yeonjeong Park. Effects of learning analytics dashboard: analyzing the relations among dashboard utilization, satisfaction, and learning achievement. *Asia Pacific Education Review*, 17(1):13–24, 2016.

[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*, SIGCSE '18, page 284–289, New York, NY, USA, 2018. Association for Computing Machinery.

[KSBB17] Stephan Krusche, Andreas Seitz, Jürgen Börstler, and Bernd Bruegge. Interactive learning: Increasing student participation through shorter exercise cycles. In *Proceedings of the Nineteenth Australasian Computing Education Conference*, pages 17–26, 2017.

[KTKK12] Carola Kruse, Thanh-Thu Phan Tan, Arne Koesling, and Marc Krüger. Strategies of lms implementation at german universities. In *Virtual Learning Environments: Concepts, Methodologies, Tools and Applications*, pages 522–541. IGI Global, 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.

[LGHB14] Tomasz D Loboda, Julio Guerra, Roya Hosseini, and Peter Brusilovsky. Mastery grids: An open source social educational progress visualization. In *European conference on technology enhanced learning*, pages 235–248. Springer, 2014.

[May05] Richard E Mayer. Cognitive theory of multimedia learning. *The Cambridge handbook of multimedia learning*, 41:31–48, 2005.

[Pin00] Paul R. Pintrich. Multiple Goals, Multiple Pathways: The Role of Goal Orientation in Learning and Achievement. *Journal of Educational Psychology*, 92(3):544–555, 2000.

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

[SRTV$^+$16] Beat A. Schwendimann, Maria Jesus Rodriguez-Triana, Andrii Vozniuk, Luis P. Prieto, Mina Shirvani Boroujeni, Adrian Holzer, Denis Gillet, and Pierre Dillenbourg. Perceiving Learning at a Glance: A Systematic Literature Review of Learning Dashboard Research. *IEEE Transactions on Learning Technologies*, 10(1):30–41, 2016.

[TYKJ16] Jennifer Pei-Ling Tan, Simon Yang, Elizabeth Koh, and Christin Jonathan. Fostering 21st century literacies through a collaborative critical reading and learning analytics environment: user-perceived benefits and problematics. In *Proceedings of the sixth international conference on learning analytics & knowledge*, pages 430–434, 2016.

[Wis14] Alyssa Friend Wise. Designing pedagogical interventions to support student use of learning analytics. In *Proceedings of the fourth international conference on learning analytics and knowledge*, pages 203–211, 2014.