FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

METIS: Multiplying Engagement Through Interacting Socially on the Artemis Learning Platform

Lorena Schlesinger

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

METIS: Multiplying Engagement Through Interacting Socially on the Artemis Learning Platform

METIS: Verstärkung des Engagements durch Soziale Interaktion auf der Artemis Lernplattform

Author:	Lorena Schlesinger
Supervisor:	Prof. Dr. Bernd Brügge
Advisor:	Prof. Dr. Stephan Krusche
Date:	15.11.2021

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

Munich, 15.11.2021 Lorena Schlesinger

Abstract

Artemis is an interactive learning platform that is evolving towards a powerful university course management system. Currently, Artemis only supports basic communication features enabling users to ask questions on specific exercises and lectures, which can be up- and down-voted. Instructors therefore complement Artemis with external communication platforms such as Slack, Zulip, and Rocket.Chat which allow more elaborate course discussions and information sharing. This causes redundancy and scattered information, which bears the risk of worse learning experience, increased moderation efforts, and decreased user engagement.

In this thesis, we address these problems by analyzing complementing communication platforms, other course management systems, and popular online class platforms. Based on the gained insights, we enhance Artemis' current discussion capabilities. We add features for course-wide announcements and duplication checks, as well as for pinning, archiving, moving, resolving, searching, filtering, tagging, and referencing posts. Moreover, by integrating emojis, we provide users with the ability to express reactions and engage with their peers. The new capabilities allow for fast and intuitive information retrieval, effective and efficient discussion moderation, and interactive conversations. They enhance Artemis' capabilities with regard to social interactions and information sharing to multiply the user engagement. Our new communication features are currently actively used by 1,700 students in 9 university courses conducted with Artemis.

Zusammenfassung

Artemis ist eine interaktive Lernplattform, die sich zu einem vielseitigen Kursmanagementsystem für den universitären Kontext entwickelt. Aktuell unterstützt Artemis grundlegende Kommunikationsfunktionen. Studierende können Fragen zu bestimmten Übungen und Vorlesungen stellen sowie für bestehende Fragen abstimmen. Um interaktive Diskussionen anzuregen und effizienten Informationsaustausch zu gewährleisten, wird Artemis durch externe Kommunikationsplattformen wie Slack, Zulip und Rocket.Chat ergänzt. Dies führt zu Redundanz und verstreuten Informationen und birgt das Risiko von schlechterer Lernerfahrung, erhöhtem Moderationsaufwand und geringerem Engagement der Studierenden in Diskussionen.

In dieser Arbeit entwickeln wir Lösungen für dieses Problem, indem wir ergänzende Kommunikationsplattformen, andere Kursmanagementsysteme und beliebte Online-Kursplattformen analysieren. Basierend auf den gewonnenen Erkenntnissen verbessern wir die aktuellen Diskussionsmöglichkeiten auf Artemis. Wir fügen Funktionen für kursweite Ankündigungen und einer Duplikationsprüfung hinzu und ermöglichen das Anpinnen, Archivieren, Verschieben, Auflösen, Suchen, Filtern, Markieren und Referenzieren von Beiträgen. Durch die Integration von Emojis bieten wir Studierenden außerdem die Möglichkeit, Reaktionen auszudrücken und auf einer non-verbalen Ebene mit Mitstudierenden in Kontakt zu treten. Die neuen Funktionen unterstützen schnelle und intuitive Informationsbeschaffung, effektive und effiziente Moderation und Betreuung der Diskussion sowie lebendige und interaktive Unterhaltungen. Sie erweitern die Möglichkeiten von Artemis im Hinblick auf soziale Interaktionen und den Austausch von Informationen mit dem Ziel, ein höheres Engagement der Studierenden auf der Lernplattform zu erreichen. Die neuen Kommunikationsfunktionen werden derzeit von 1.700 Studierenden in 9 Universitätskursen aktiv genutzt, die auf der Lernplattform Artemis abgehalten werden.

Contents

CMS	Course Management System
IDE	Integrated Development Environment
FAQ	Frequently Asked Questions
FR	Functional Requirement
HTTP	Hypertext Transfer Protocol
METIS	Multiplying Engagement Through Interacting Socially
MOOC	Massive Open Online Course
NLP	Natural Language Processing
NFR	Nonfunctional Requirement
Q&A	Question & Answer
UI	User Interface
UML	Unified Modeling Language

Chapter 1

Introduction

According to the annual report of Class Central[1], "one third of the learners that ever registered on a Massive Open Online Course (MOOC) platform joined in 2020". Courses in the field of computer science are particularly making use of online learning advances[2]. The engagement of students in online courses is key for achieving learning success, where engagement depends, among other things, on social interaction and building a community [RAB^{+}18, FSP14]. For instance, there is evidence that the number of students' posts, e.g., in a discussion forum, is positively correlated with their final grade [DRK20]. Similarly, prior research on MOOCs suggests that the existence of a discussion forum has "a positive effect on the probability to partially complete a course" [Ada13]. Specifically for programming courses, peer interaction has further been shown to be a significant factor for students' motivation as peers can mutually support each other and share collectively perceived struggles or successes [WRGW14].

Artemis is an automated assessment management system for interactive learning that has been designed for large university courses and online courses [KS18]. Since Artemis allows instructors to share course materials, conduct examinations, and provide immediate, automated feedback for programming exercises to students, it is evolving towards a versatile Course Management System (CMS) [Mee03]. Such systems also require capabilities to foster communication between students and instructors [WW07]. Artemis currently only implements a basic set of communication capabilities initially introduced by Meier [Mei19] and later enhanced by Gregurevic [Gre20]: The

[1]Dhawal Shah. By The Numbers: MOOCs in 2020. `https://www.classcentral.com/report/mooc-stats-2020/`, Nov 2020. Accessed: 2021-05-02

[2]Dhawal Shah. The Second Year of The MOOC: A Review of MOOC Stats and Trends in 2020. `https://www.classcentral.com/report/the-second-year-of-the-mooc/`, Dec 2020. Accessed: 2021-05-01

Question & Answer (Q&A) features enable users to ask questions on course material, up- and down-vote questions, and, as a moderator[3], approve answers given by students. Yet, discussion forums are "not inherently effective or engaging in themselves, [since] they are simply mechanisms" [CS18]. Hence, the mere existence of functionality to interact with peers without further incentives is often not sufficient [Dix10].

1.1 Problem & Motivation

Due to Artemis' limited communication capabilities, previous courses taught on Artemis almost always extensively relied on external communication platforms used in addition to Artemis [Mei19], such as Slack[4]. This bears additional costs and requires labor-intensive moderating to manually sync the communication activities between external platforms and Artemis Q&A. In order to advance Artemis' communication capabilities, two problem areas need to be investigated: (1) problems identified in the usage of Artemis Q&A, *and* (2) shortcomings of external communication platforms. In the following, we describe the identified problems that are addressed by this thesis. They are denoted by [P_i] to reference them throughout this document.

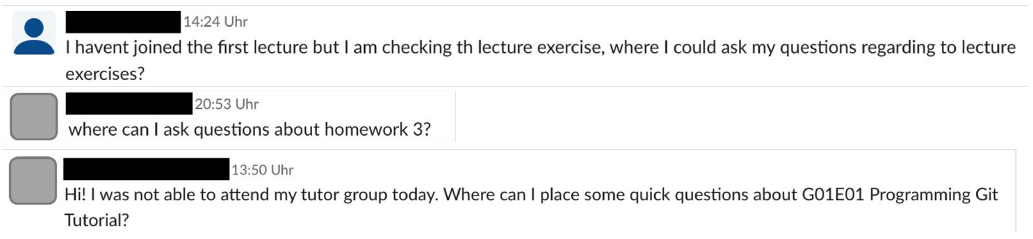
[P_1] **Insufficient topic organization lacking course-wide topics.** The current implementation of Artemis Q&A is limited with regard to organizing postings[5] into contexts: Students can only add postings to specific exercises or lectures, which implies that the posting is either associated to the exercise or lecture context, respectively. However, especially during the first weeks of a course, students have the need to ask organizational questions of course-wide relevance or require technical support that is not related to a certain exercise or lecture. During this phase, communication involves group forming and expectation setting, whereas it shifts towards socializing, experience sharing, and even celebration at the end of the course [PD16]. Due to the shortcoming that students cannot converse on course-wide topics, courses currently outsource these conversation types to other communication platforms such as Slack and create dedicated channels (e.g., *#announcements, #organizational, #tools*) to provide students with the required information.

[3]We collectively refer to users performing moderation tasks, such as student support or keeping the discussion content organized and up-to-date, as *moderators*. In the Artemis context, these particularly comprise of tutors and instructors.

[4]`https://slack.com/`, Accessed: 2021-10-20

[5]We collectively refer to posts and answer posts when we use the term *posting*. A *post* marks the start of a *discussion*, where subsequent postings are *answer posts*.

Without providing an equally suitable space for discussing course-wide topics on Artemis, instructors will refrain from using Artemis as the *all-in-one* solution for their course.



[redacted] 14:24 Uhr
I havent joined the first lecture but I am checking th lecture exercise, where I could ask my questions regarding to lecture exercises?

[redacted] 20:53 Uhr
where can I ask questions about homework 3?

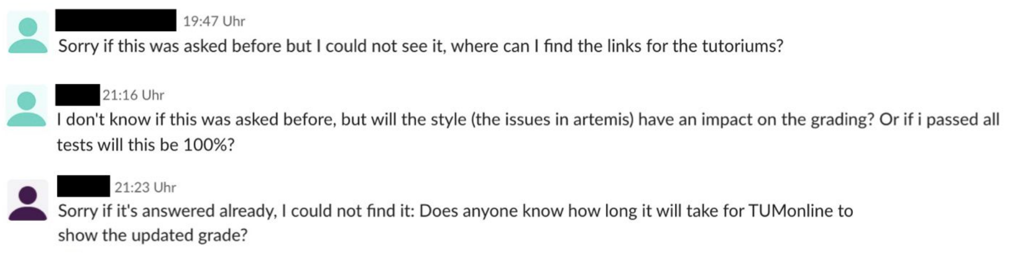
[redacted] 13:50 Uhr
Hi! I was not able to attend my tutor group today. Where can I place some quick questions about G01E01 Programming Git Tutorial?

Figure 1.1: Students' confusion about topic organization on Slack [P_1]

A related problem is a non-intuitive or overly fine-grained topic organization. For instance, if several channels with overlapping topics are used in parallel, this can lead to improperly placed postings and ultimately confusion of students [Hew18]. The real-world examples in Figure 1.1 showcase the occurrence of this problem in Slack. If questions are posted to the wrong discussion, channel, or, more generally, context, they might either be misunderstood or get lost. As a matter of fact, relevant information being lost in unstructured communication is one of the key challenges when designing course communication [Luc20].

[P_2] **Insufficient visibility and discoverability of posts.** When analyzing Slack channels used in addition to Artemis, the problem of information overload becomes visible: Figure 1.2 shows how students report, that it was hard for them to search for relevant information. Forum overload is the most dissatisfying characteristics of MOOC [Hew18] and leads to insufficient visibility of relevant information. Additionally, using the free version of Slack, the access to team messages is limited to the 10,000 most recent ones[6]. With course sizes of almost 2,000 students, discussion activity is only persisted over a time span of some weeks, depending on the message volume. As a consequence, visibility and discoverability clearly suffer. This entails follow-up problems such as redundancy, which is examined in [P_3]. In summary, the benefit of discussion forums is strongly bound to how content is searchable and discoverable to avoid redundancy, misinformation, and confusion which may ultimately lead to disengagement [DFCV15, GHG+20].

[6]`https://slack.com/intl/en-de/pricing`, Accessed: 2021-11-01

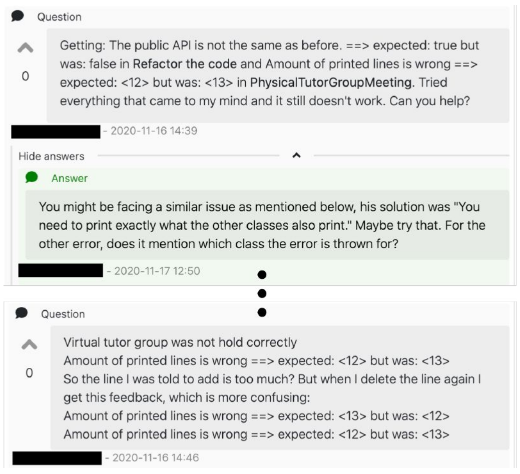

[redacted] 19:47 Uhr
Sorry if this was asked before but I could not see it, where can I find the links for the tutoriums?

[redacted] 21:16 Uhr
I don't know if this was asked before, but will the style (the issues in artemis) have an impact on the grading? Or if i passed all tests will this be 100%?

[redacted] 21:23 Uhr
Sorry if it's answered already, I could not find it: Does anyone know how long it will take for TUMonline to show the updated grade?

Figure 1.2: Students' difficulties to find information on Slack [P_2]

[P_3] **Content redundancy.** Prior research, such as work on Stack Overflow[7], suggests that the increase of duplicated questions is accompanied by a decline in information quality [SPD18, ZLXS15]. On Artemis, having redundant and duplicated questions will lead to unnecessary efforts for moderators, who have to read the post, identify and potentially mark it as duplication, and link related posts. In the worst case, duplicated questions are answered differently causing inconsistencies and misunderstandings. This problem occurs on both platforms, Artemis (see Figure 1.3a) and Slack (see Figure 1.3b) as well as between them (see Figure 1.3c). We refer to this as intra-platform redundancy [$P_{3.1}$] and inter-platform redundancy [$P_{3.2}$].


Question

Getting: The public API is not the same as before. ==> expected: true but was: false in **Refactor the code** and Amount of printed lines is wrong ==> expected: <12> but was: <13> in **PhysicalTutorGroupMeeting**. Tried everything that came to my mind and it still doesn't work. Can you help?

0

[redacted] - 2020-11-16 14:39

Hide answers

Answer

You might be facing a similar issue as mentioned below, his solution was "You need to print exactly what the other classes also print." Maybe try that. For the other error, does it mention which class the error is thrown for?

[redacted] - 2020-11-17 12:50

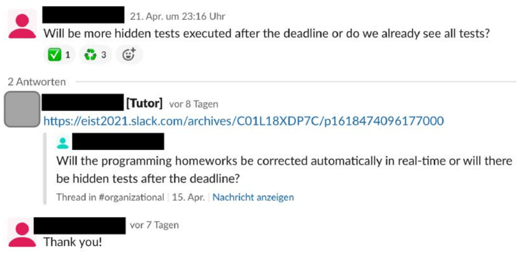
Question

Virtual tutor group was not hold correctly
Amount of printed lines is wrong ==> expected: <12> but was: <13>
So the line I was told to add is too much? But when I delete the line again I get this feedback, which is more confusing:
Amount of printed lines is wrong ==> expected: <13> but was: <12>
Amount of printed lines is wrong ==> expected: <12> but was: <13>

0

[redacted] - 2020-11-16 14:46

(a) Redundancy on Artemis

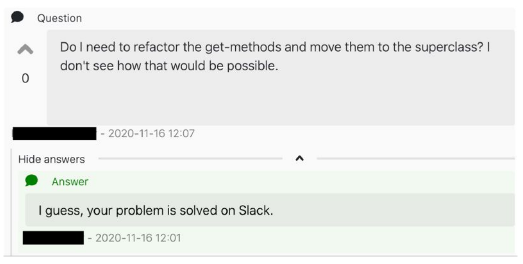

[redacted] 21. Apr. um 23:16 Uhr
Will be more hidden tests executed after the deadline or do we already see all tests?
✅ 1 ♻ 3

2 Antworten

[redacted] **[Tutor]** vor 8 Tagen
https://eist2021.slack.com/archives/C01L18XDP7C/p1618474096177000
[redacted]
Will the programming homeworks be corrected automatically in real-time or will there be hidden tests after the deadline?
Thread in #organizational | 15. Apr. | Nachricht anzeigen

[redacted] vor 7 Tagen
Thank you!

(b) Redundancy on Slack


Question

Do I need to refactor the get-methods and move them to the superclass? I don't see how that would be possible.

0

[redacted] - 2020-11-16 12:07

Hide answers

Answer

I guess, your problem is solved on Slack.

[redacted] - 2020-11-16 12:01

(c) Redundancy between platforms

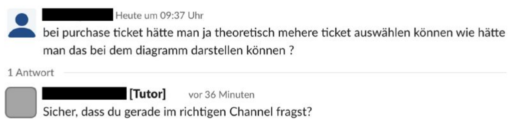
Figure 1.3: Intra-platform ([$P_{3.1}$]) and inter-platform ([$P_{3.2}$]) redundancy captured on Artemis and Slack

[7]`https://stackoverflow.com/`, Accessed: 2021-10-20

Furthermore, redundancy can occur due to private one-to-one conversations [$P_{3.3}$]: Questions are (repeatedly) answered in private instead of targeting everybody. Courses cannot benefit from shared knowledge in the community, if conversations are held in private.

[P_4] **High, text-based moderation efforts.** Moderating discussions on Artemis can be both, resource and time intensive. The effort is amplified by the overhead induced through redundant posts and platform co-existence (see [P_3]). Yet, moderation is currently *text-based* only, meaning that moderators have to *textually* describe links between posts or state if a post is resolved.

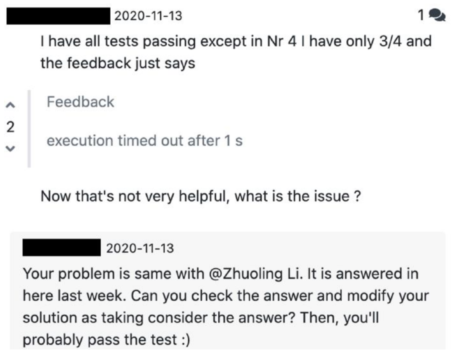
Figure 1.4 demonstrates two exemplary situations of text-based moderation: (1) Instead of simply moving the misplaced posting to where it belongs to, a moderator has to remind a student that the posting was added to the wrong channel (see Figure 1.4a). (2) Instead of simply referencing an existing answer to the student's question, a moderator verbosely explains that the problem was solved elsewhere (see Figure 1.4b).


[redacted] Heute um 09:37 Uhr
bei purchase ticket hätte man ja theoretisch mehere ticket auswählen können wie hätte man das bei dem diagramm darstellen können ?

1 Antwort

[redacted] [Tutor] vor 36 Minuten
Sicher, dass du gerade im richtigen Channel fragst?

(a) Text-based moderation on Slack


[redacted] 2020-11-13 1
I have all tests passing except in Nr 4 I have only 3/4 and the feedback just says

Feedback

execution timed out after 1 s

Now that's not very helpful, what is the issue ?

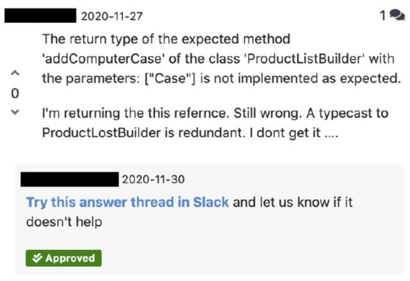
2

[redacted] 2020-11-13
Your problem is same with @Zhuoling Li. It is answered in here last week. Can you check the answer and modify your solution as taking consider the answer? Then, you'll probably pass the test :)

(b) Text-based moderation on Artemis

Figure 1.4: Text-based moderation effort [P_4]

Alongside, moderators not only spend time by answering questions, but also by browsing through discussions to determine if their support is needed. Currently, there is a mechanism to approve answers, where only moderators can mark answers as *approved* that they perceive as correct. However, there is no label that indicates if the problem that was raised in the author's original posting is actually *resolved*. Figure 1.5 exemplifies how moderators rely on textual feedback if the provided answer resolves the problem under discussion. Hence, Artemis Q&A misses to always involve the person that raised the problem in the post resolution process.


[redacted] 2020-11-27 1

The return type of the expected method 'addComputerCase' of the class 'ProductListBuilder' with the parameters: ["Case"] is not implemented as expected.

0

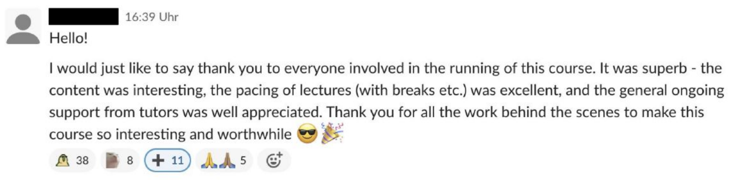
I'm returning the this refernce. Still wrong. A typecast to ProductLostBuilder is redundant. I dont get it ....

[redacted] 2020-11-30

Try this answer thread in Slack and let us know if it doesn't help

Figure 1.5: Text-based moderation to resolve problems on Artemis [P_4]

[P_5] **Low interaction and emotional involvement.** Artemis Q&A currently only supports social interaction to a limited degree: Agreement or support can be expressed through up-voting, whereas all other reactions have to be communicated via dedicated answer posts. According to research, online conversations without using emojis bear the risk of an emotional gap, making a dialog less vivid and nuanced [Hon20]. Figure 1.6 counteracts the described problem by demonstrating the emotional involvement of students in a course expressed by means of text-embedded emojis and emoji reactions.


[redacted] 16:39 Uhr

Hello!

I would just like to say thank you to everyone involved in the running of this course. It was superb - the content was interesting, the pacing of lectures (with breaks etc.) was excellent, and the general ongoing support from tutors was well appreciated. Thank you for all the work behind the scenes to make this course so interesting and worthwhile 😎🎉

Figure 1.6: Emotional involvement through emojis on Slack [P_5]

Furthermore, users can neither reference other postings nor users. As can be seen in Figure 1.4b this would also facilitate discussion moderation. Interactively contributing to conversations on Artemis is currently only possible when staying on the platform, notifications on certain topics or discussions that a user is interested to follow cannot be received outside of Artemis.

Motivation. The described problems [P_{1-5}] reveal several communication-related limitations and inefficiencies in Artemis. This motivates the development of more elaborate communication capabilities for Artemis that are aligned with existing research on effective communication in CMSs [DFCV15, GHG$^+$20, Luc20]. To address the existing limitations, this thesis proposes Multiplying Engagement Through Interacting Socially (METIS), a novel subsystem for the open source Artemis project[8].

[8] `https://github.com/ls1intum/Artemis`, Accessed: 2021-11-11

1.2 Objectives

After having outlined the prevalent problems due to Artemis' limited communication capabilities, we map the problems to objectives denoted by [O_j].

[O_1] **Effective and efficient information retrieval.** The effectiveness of a discussion forum depends on the information coverage and the filtering capabilities that users can utilize to find relevant content [DFCV15]. The information coverage includes all aspects of course communication, which particularly also includes course-wide, more general topics. Since the profusion of posts may impede or hinder finding relevant information [Hew18], we aim to adopt established content structuring and querying features. METIS should help students to find correct answers in one place, which has been identified as an effective means to prevent duplication [SPD18]. It is therefore a stated goal of METIS to optimize information retrieval in both aspects, effectiveness, i.e., users should find what they search for, as well as efficiency, i.e., they should not waste time and resources by doing so. This objective thus addresses [P_1], [P_2], and [P_3].

[O_2] **Effective and efficient discussion moderation.** We observe increasing numbers of computer science course enrollments on Artemis. As automated assessment management system for interactive learning, Artemis specifically tries to enhance the individual learning experience of students while keeping the efforts of instructors at a reasonable level [KS18]. This is also a paradigm for designing METIS: Students' assistance and moderation-related tasks should be pursued by personnel resources effectively and efficiently. Here, effectiveness implies that moderators are enabled to offer high quality support to students, by resolving their questions, as well as highlighting relevant and discarding irrelevant information. Efficiency means that moderators can complete these tasks as fast as possible and with low personnel effort. This specifically addresses [P_3] and [P_4].

[O_3] **Interactive conversations.** A recent study reports that teams with more "emoji engagement" are expected to be more effective, underlining the added value of emotionally nuanced conversation and nonverbal affirmation which forms team culture [AVWO20]. Reacting on postings with emojis enables a better expression of a user's affective state [ZIFK17]. Additionally, referencing postings, users, or other course content, such as specific sequences of a lecture or tasks in a programming exercise, facilitates information inter-connection. METIS aims to spur interactive conversations by providing features that add missing conversational aspects to Artemis, which specifically addresses [P_5].

1.3 Outline

At this point, we elucidated the problems inherent to the current state of course communication while using Artemis as interactive learning platform. We motivated the need for improvement and defined concrete objectives.

After we conduct the analysis of comparable communication platforms in Chapter 2, we derive enabling features during the requirement analysis in Chapter 3. In Chapter 4, we describe the system design for the communication capabilities that we introduce in this thesis. Chapter 5 describes implementation details, whereas we discuss decisions that were taken as well as evaluation aspects of METIS in Chapter 6. Finally, we summarize the contributions made by this thesis and close with concluding remarks and future work in Chapter 7.

Chapter 2

Comparison of State-of-the-Art

In this chapter, we assess the state-of-the-art by analyzing the communication capabilities of platforms from different domains. Therefore, we first introduce the studied platforms, second, compare their communication features, and third, summarize the insights. Upon this analysis, we will conduct the requirement elicitation in Chapter 3.

2.1 Compared Platforms

In the following, we first investigate communication platforms that we find to be used as complementary platforms alongside to Artemis courses at the Technical University of Munich. These are Slack, Zulip[1], RocketChat[2], and Moodle[3]. Second, Submitty[4] and Aurora[5] are considered, since they are open source university CMSs that have a similar scope as Artemis. Third, we analyze Stack Overflow as the leading global software development forum and knowledge sharing platform. Last, in order to get diverse insights for the system design developed within this thesis, we take into account the communication features offered by some of the most popular online course providers, namely Coursera[6], Udemy[7], edX[8], and FutureLearn[9].

[1]`https://zulip.com/`, Accessed: 2021-10-20
[2]`https://rocket.chat/`, Accessed: 2021-10-20
[3]`https://moodle.org/`, Accessed: 2021-10-20
[4]`https://submitty.org/`, Accessed: 2021-10-20
[5]`https://gitlab.iguw.tuwien.ac.at/aurora/aurora`, Accessed: 2021-10-20
[6]`https://coursera.org/`, Accessed: 2021-10-20
[7]`https://udemy.com/`, Accessed: 2021-10-20
[8]`https://edx.org/`, Accessed: 2021-10-20
[9]`https://futurelearn.com/`, Accessed: 2021-10-20

Communication Platforms

- **Slack**: Slack is a commercial communication platform offered by Slack Technologies, a Salesforce company. As of 2019 Slack had more than 10 million users per day[10].

- **Zulip**: Zulip is an open source communication platform maintained by Kandra Labs, who have initiated the project and are also offering cloud-based Zulip hosting. By 2021, Zulip is used every day by Fortune 500 companies, leading open source projects, and thousands of other organizations[11].

- **Rocket.Chat**: Rocket.Chat is an open source communication platform launched in 2015. Rocket.Chat Technologies further offers enterprise support for Rocket.Chat which is used by more than 12 million users across 150 countries as of 2021[12].

University Course Management Systems

- **Moodle**: Moodle is an open source learning management system that is used by roughly 300 million users worldwide as of 2021[13].

- **Submitty**: Submitty is an open source course management from the Rensselaer Center for Open Source Software (RCOS). In spring 2020, their software was used by more than 2,000 students and 242 professors, teaching assistants, and mentors[14].

- **Aurora**: Aurora is an open source learning platform developed and used at the Human Computer Interaction Group at the Institute of Visual Computing & Human-Centered Technology at the Vienna University of Technology [Luc20]. As of 2020, three lectures with around 600–800 participating students were conducted on that platform [Luc20].

[10]`https://slack.com/intl/de-de/blog/news/slack-has-10-million-daily-active-users`, Accessed: 2021-10-28
[11]`https://blog.zulip.com/2021/05/13/zulip-4-0-released`, Accessed: 2021-10-28
[12]`https://de.rocket.chat/enterprise`, Accessed: 2021-10-28
[13]`https://stats.moodle.org/`, Accessed: 2021-10-28
[14]`https://news.rpi.edu/content/2020/05/27/student-built-program-supports-thousands-during-remote-learning-experience`, Accessed: 2021-10-28

Forum

- **Stack Overvflow**: Stack Overflow is a discussion and Q&A website for programming content. As of 2021 Stack Overflow has more than 16 million registered users[15].

Online Course Platforms

- **Coursera**: Coursera is a MOOC provider founded at Stanford University in 2012. As of 2021 roughly 8,000 courses are offered on Coursera by partners and universities from more than 54 countries[16].
- **Udemy**: Udemy is a MOOC provider that has more than 44 million registered students and more than 183,000 courses offered in 75 languages[17].
- **edX**: edX is an open source MOOC platform that has more than 35 million users as of 2021[18].
- **FutureLearn**: FutureLearn is an online education provider based in the UK, that partners with around 175 top international universities and specialist organisations to offer online courses and degrees since 2013[19].

2.2 Feature Comparison

To describe the state-of-the-art, we divide the features of the compared platforms into five groups where each group is associated to one area of improvement. We approach (at least) one of the problems identified in Section 1.1 per group.

2.2.1 Topic Structuring and Organization

We checked the selected platforms for features to structurally separate organizational from content-related discussions. The results are summarized in Table 2.1. The communication platforms Slack, Zulip, and Rocket.Chat offer the possibility to create dedicated channels for certain topics, where

[15]`https://stackexchange.com/sites?view=list#users`, Accessed: 2021-10-28
[16]`https://www.coursera.org/about/partners`, Accessed: 2021-10-28
[17]`https://about.udemy.com/`, Accessed: 2021-10-28
[18]`https://www.edx.org/about-us`, Accessed: 2021-10-28
[19]`https://www.futurelearn.com/partners`, Accessed: 2021-10-28

one of them could also be the course organization. In Moodle, course administrators can create posts either in a general discussion forum or in an announcements forum, which allows to distinguish between bi-lateral discussion and uni-lateral information sharing of organizational character. While Submitty does not provide a separate space to discuss organization-related content, Aurora has dedicated discussion boards for general information and frequently asked questions. All MOOC providers, except for FutureLearn, also provide capabilities for course-wide announcements or separated organizational discussion topics.

	Topic Structuring and Organization
Platform	**Structural Separation of Organizational Topics**
Slack	Dedicated channels
Zulip	Dedicated channels
Rocket.Chat	Dedicated channels
Moodle	Announcements forum
Submitty	-
Aurora	✓
Stack Overflow	-
Coursera	Dedicated discussion forum
Udemy	Announcement section
edX	Discussion topics
FutureLearn	-

Table 2.1: Platform comparison – Features approaching [P_1]

2.2.2 Post Discoverability

In this section, we compare the selected platforms with regard to features that increase the discoverability of posts.

The first set of features addresses discoverability by providing means to filter and sort posts. Table 2.2 lists features including text search and offered sort and filter options. We find that besides Aurora and FutureLearn, all platforms offer a text search to the user, which can be seen as the most efficient way to query existing discussions for specific content. If sort options are provided, a user can always choose to sort the posts chronologically. In the case of edX, this refers to the most recent activity that is related to a post and not the timestamp of creation. Stack Overflow provides this sort option in addition to the creation date. Other platforms additionally offer to sort by topics (Moodle), number of answers or comments (Moodle, Submitty, Stack Overflow), and votes (Stack Overflow, Coursera, Udemy, edX, FutureLearn). Slack is the only pure communication platform that offers sorting, but only after querying or filtering the Slack workspace. The filter options depend on the features offered on the platform: If users can tag their posts, this can also be used as filter criterion (Submitty, Stack Overflow). The same applies to bookmarked or followed discussions (Zulip, Rocket.Chat, Udemy, FutureLearn). Communication platforms typically do not follow the Q&A concept. Hence, these platforms offer filtering in channels, whereas on most other platforms users can filter based on if a question was already answered (Stack Overflow, Coursera, Udemy, edX). On the Submitty platform users can filter posts that are not yet resolved.

	Post Searching, Filtering, and Sorting		
Platform	**Text Search**	**Sort Options**	**Filter Options**
Slack	✓	Relevance | Date	Channels | User | Date | Reaction Content Type | Message Type
Zulip	✓	-	Channel | Topic | Private | Own | Post ID | Bookmarked | Only New | Content Type | Message Type
Rocket.Chat	✓	-	Channel | Bookmarked
Moodle	✓	Date | Topic | Answers	-
Submitty	✓	Date | Answers	Tags | Problem State
Aurora	-	-	New
Stack Overflow	✓	Date | Activity | Votes | Not answered | Frequent	Not answered | Accepted answer | Tags
Coursera	✓	Date | Votes	Answered | Not answered
Udemy	✓	Date | Recommended | Votes	Own | Followed | Not answered
edX	✓	Activity | Votes	New | Not answered | Topics
FutureLearn	-	Date | Votes	Bookmarked | Own | Followed

Table 2.2: Platform comparison – Features approaching [P_2]

The second set of features adds metadata to posts to facilitate their discovery. As shown in Table 2.3, this includes post bookmarking, pinning, and tagging, as well as providing meta titles to posts. Bookmarking allows a certain user to mark a post of personally perceived relevance that should be accessible very fast, e.g., through filtering options. Pinning a post is only allowed for authorized users because it will lead to a change of the post visibility for *all* users, i.e., by listing it at the top of a view or in a dedicated section. We observe that on all platforms one of those two options is provided, indicating that users seek for features to highlight and mark content. Furthermore, to enhance visibility and discoverability, several platforms allow users to tag posts or add summarizing meta titles. The concept of titles is used on every platform except for Slack, Rocket.Chat, Aurora, and FutureLearn. For the communication platforms Slack and Rocket.Chat a possible explanation could be, that titles would visually and logically interrupt the reading flow of a chat-like dialog. Tagging a post is a crucial feature on Stack Overflow. Stack Overflow states that: "Tags are a means of connecting experts with questions they will be able to answer by sorting questions into specific, well-defined categories"[20]. Tags are also used for recommendations, duplication checks, and platform analysis [MMM+11, ZLXS15, SPD18]. Submitty forces users to label each discussion thread with one or more so-called categories which can be customized[21].

	Discoverable Post Metadata			
Platform	**Book-mark**	**Pin Thread / Post**	**Post Title**	**Tags**
Slack	✓	✓	-	-
Zulip	✓	✓	Thread topic	-
Rocket.Chat	✓	✓	-	-
Moodle	✓	-	✓	-
Submitty	✓	✓	✓	✓
Aurora	-	✓	-	-
Stack Overflow	✓	-	✓	✓
Coursera	-	✓	✓	-
Udemy	-	Dedicated section	✓	-
edX	-	✓	✓	-
FutureLearn	✓	-	-	-

Table 2.3: Platform comparison – Features approaching [P_2]

2.2.3 Content Redundancy Reduction

Features that reduce content redundancy are rarely integrated in the compared platforms as can be seen in Table 2.4. The communication platforms Slack, Zulip, and Rocket.Chat do not entail such features because they inherently do not care about duplicated content. The course management platform Submitty tries to restrict the number of duplicated posts by providing moderators the feature to merge discussion threads with similar content. Submitty also allows users to write posts as anonymous author, for which only the moderators are able to access the username. Thereby, questions including sensitive information can be posted publicly, but do not hurt rules with regard to data protection. This prevents redundant private conversations. Stack Overflow incorporates elaborate algorithms to detect duplication, on the one hand, and actively prevent the creation of question clones, on the other hand. During the creation of a question, the user is automatically suggested similar questions. Only if the user actively confirms that the identified, similar questions differ, they[22] can proceed with creating the new question. Although online course platforms have large numbers of posts due

[20]`https://stackoverflow.com/help/tagging`, Accessed: 2021-11-01

[21]`https://submitty.org/student/communication/forum`, Accessed: 2021-11-01

[22]We use the gender-neutral *singular-they* when referring to a person of either or unknown gender.

to the high number of participants in MOOCs, they seem to not explicitly prevent duplication. Among the compared platforms, only edX and Aurora make use of a Frequently Asked Questions (FAQ) section to answer the most common questions.

	Content Redundancy Reduction			
Platform	**Post Similarity Comparison**	**Merge Threads/ Posts**	**FAQ**	**Anonymous Posts**
Slack	-	-	-	-
Zulip	-	-	-	-
Rocket.Chat	-	-	-	-
Moodle	-	-	-	-
Submitty	-	✓	-	✓
Aurora	-	-	✓	-
Stack Overflow	✓	-	-	-
Coursera	-	-	-	-
Udemy	-	-	-	-
edX	-	-	✓	-
FutureLearn	-	-	-	-

Table 2.4: Platform comparison – Features approaching [P_3]

2.2.4 Moderation Effort Reduction

We investigated how the selected platforms attempt to reduce moderation effort and list the relevant features in Table 2.5. These features comprise of adding non-textual question state indication, that facilitate understanding if a question was already resolved and allow users to report posts that violate discussion rules. Since the online communication platforms Slack, Zulip, and Rocket.Chat serve the purpose of chatting, but not necessarily host forum-like discussions, they do not provide any dedicated features to indicate a non-textual question state indication. When being used in addition to Artemis, we observe that students as well as moderators use emoji reactions to indicate whether a question is resolved on Slack (e.g., the *thumbs-up* emoji).

Only on the platforms Submitty and Stack Overflow, users can state that the problem or question described in a post is resolved. If a Stack Overflow user accepts one of the given answers, they indicate, that the provided solution worked for them. Submitty offers a checkmark icon next to the post that can be activated to mark the associated question as resolved.

The online course platforms as well as Rocket.Chat provide users with the possibility to report a post, e.g., if it violates established communication or course rules. This also reduces the effort of moderators, who would need to manually check the plethora of postings for such violations.

	Moderation Effort Reduction	
Platform	**Non-Textual Question State**	**Report Post**
Slack	Emoji reaction	-
Zulip	-	-
Rocket.Chat	-	✓
Moodle	-	-
Submitty	Resolved icon	-
Aurora	-	-
Stack Overflow	Accepted answer	✓
Coursera	-	✓
Udemy	-	✓
edX	-	✓
FutureLearn	-	✓

Table 2.5: Platform comparison – Features approaching [P_4]

2.2.5 Interaction

As summarized in Table 2.6, interaction between platform users is a key requirement of all the compared platforms. On all platforms except for Moodle, Submitty, and Aurora, users can express (dis-)agreement or opinions on other posts by voting or reacting with emojis. The communication platforms integrate so-called reaction bars to select emojis as a reaction to some message. On the other hand, the Stack Overflow forum as well as the course management platform Aurora and the online course platforms rely on simple up- or down-voting mechanisms.

When it comes to referencing other posts, we can state that FutureLearn does not offer such a feature. Submitty and Aurora are not publicly available, but they do not advertise this feature. Yet, all other platforms included in the comparison allow referencing other posts. Directly referring to other users is a common feature on online communication platforms that therefore rely on the commonly used and intuitive pattern of prefixing the username with '@'. This in turn leads to a notification for the referenced user. On Stack Overflow, user referencing is only allowed in comments to prevent questions from being asked to specific users rather than the community. To inform involved users on updates on a certain post or thread, almost all platforms offer a feature to enable such notifications and follow an ongoing discussion.

2.3 Summary

The resulting groups of features per area of improvement as well as their popularity in platforms of different domains build the basis for the requirements elicitation and analysis that is carried out in the following Chapter 3. We draw the following conclusions from our state-of-the-art analysis:

	Interaction					
Platform	**Up-vote**	**Down-vote**	**Emoji Reactions**	**Single Post References**	**User References**	**Follow Discussion**
Slack	-	-	✓	✓	✓	✓
Zulip	-	-	✓	✓	✓	✓
Rocket.Chat	-	-	✓	✓	✓	✓
Moodle	-	-	-	✓	-	✓
Submitty	-	-	-	?	-	✓
Aurora	✓	✓	-	?	-	?
Stack Overflow	✓	✓	-	✓	In comments	✓
Coursera	✓	-	-	✓	-	✓
Udemy	✓	-	-	✓	-	✓
edX	✓	-	-	✓	-	✓
FutureLearn	✓	-	-	-	-	✓

Table 2.6: Platform comparison – Features approaching [P_5]

- It is important to create space for uni-lateral information sharing, e.g., for organizational matters or technical issues and separate this communication from content-related discussions.

- Features for querying existing content efficiently such as filters or text search are widespread. Users expect to be able to narrow down the vast amounts of content according to their needs.

- Contributions on platforms are not restricted to pure content. They include metadata such as titles, tags, or other forms of highlighting that increase discoverability.

- The more contributions a platforms contains, the more important are features that reduce or facilitate moderation effort. Platforms rely on *visual* instead of *textual* indication for the quality of an answer, the state of a question, or the role of a user.

- Content redundancy can be tackled in different ways but seems to be neglected at most parts. We find the duplication check that is employed on the Stack Overflow as the most promising attempt to prevent duplicated questions.

- Users are provided with mechanisms to express their agreement, disagreement, or other types of reactions on posted content. Emojis can be used for that purpose.

Chapter 3

Requirements Analysis

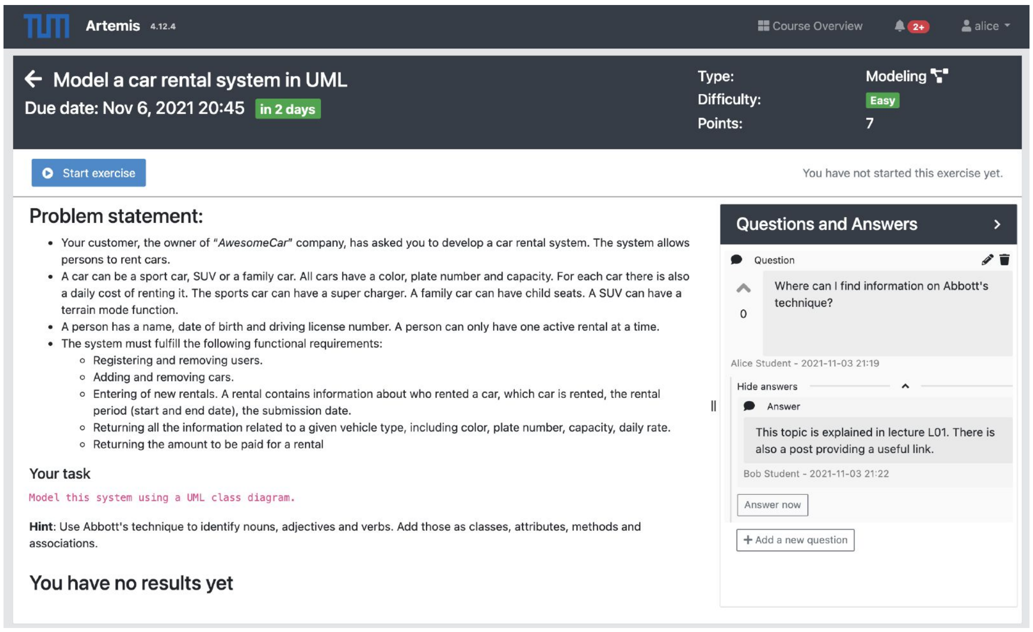
3.1 Overview

In this chapter, we analyze the current system 3.2 and describe the purpose of the proposed METIS subsystem 3.3. We elicit requirements that are further translated into a so-called system models which ensure that developers communicate their understanding of the system under construction to the users properly [BD09].

3.2 Current System

Artemis Q&A (at Artemis version 4.12.4) evolved through contributions by Meier [Mei19] and Gregurevic [Gre20]. It provides basic functionality for students to ask questions, and moderators or other students to provide answers to these questions. In the current system, the used terminology hence knows the two entities *Questions* and *Answers*. For our new implementation, i.e., for the METIS subsystem, we replaced the terminology by more general terms: A *Discussion* starts with a *Post*–which is not a question per se–that can be responded to through *Answer Posts*. However, for describing the current system (Artemis 4.12.4), we stick to the old terminology in this section.

For courses that activate the Artemis Q&A feature, each exercise and lecture page is integrated with a collapsible side panel listing the according *Questions and Answers*. Figure 3.1 captures the described discussion section for a modeling exercise. Within that panel, questions are sorted by upvotes and chronologically. Editing or deleting a question or answer is only permitted to authors and moderators. In Figure 3.1, the student *Alice* is only allowed to modify the question she posted. This is indicated by the icons at the top right corner. For adding or editing questions and answers, users are provided with an integrated markdown editor as exemplary shown in Figure 3.2.



Model a car rental system in UML
Due date: Nov 6, 2021 20:45 in 2 days

Type: Modeling
Difficulty: Easy
Points: 7

You have not started this exercise yet.

Problem statement:

- Your customer, the owner of "*AwesomeCar*" company, has asked you to develop a car rental system. The system allows persons to rent cars.
- A car can be a sport car, SUV or a family car. All cars have a color, plate number and capacity. For each car there is also a daily cost of renting it. The sports car can have a super charger. A family car can have child seats. A SUV can have a terrain mode function.
- A person has a name, date of birth and driving license number. A person can only have one active rental at a time.
- The system must fulfill the following functional requirements:
  - Registering and removing users.
  - Adding and removing cars.
  - Entering of new rentals. A rental contains information about who rented a car, which car is rented, the rental period (start and end date), the submission date.
  - Returning all the information related to a given vehicle type, including color, plate number, capacity, daily rate.
  - Returning the amount to be paid for a rental

Your task

`Model this system using a UML class diagram.`

Hint: Use Abbott's technique to identify nouns, adjectives and verbs. Add those as classes, attributes, methods and associations.

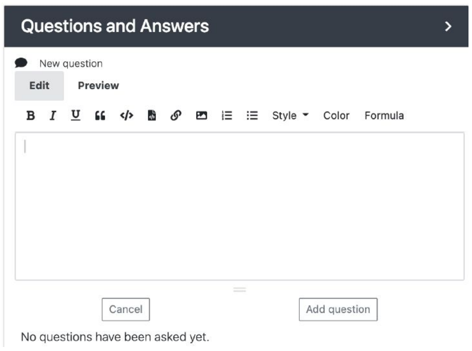
You have no results yet

Questions and Answers

Question
0
Where can I find information on Abbott's technique?
Alice Student - 2021-11-03 21:19

Answer
This topic is explained in lecture L01. There is also a post providing a useful link.
Bob Student - 2021-11-03 21:22

Figure 3.1: Exercise with one question on Artemis 4.12.4



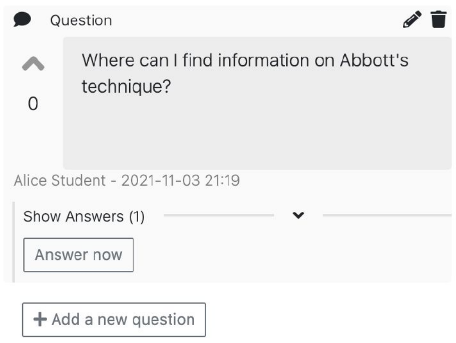
Questions and Answers

New question

No questions have been asked yet.

Figure 3.2: Integrated markdown editor to add a question on Artemis 4.12.4

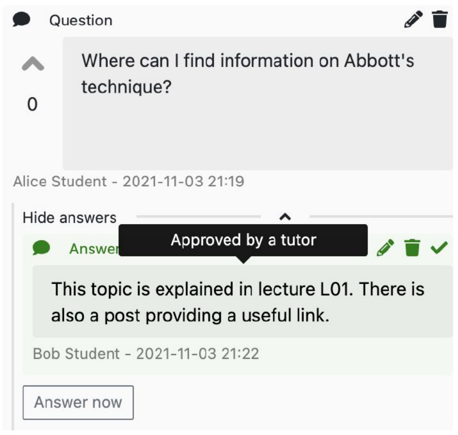
The most relevant of the basic discussion capabilities on Artemis 4.12.4 are depicted by Figure 3.3: Students can up- and down-vote questions to give them higher or lower priority by clicking on up or down arrows next to the questions where the number of resulting votes is displayed. The answers of a question can further be collapsed and expanded (see Figure 3.3a). Mod-



Where can I find information on Abbott's technique?

Alice Student - 2021-11-03 21:19

(a) Question with collapsed answer



Where can I find information on Abbott's technique?

Alice Student - 2021-11-03 21:19

This topic is explained in lecture L01. There is also a post providing a useful link.

Bob Student - 2021-11-03 21:22

(b) Approved answer

Figure 3.3: Interacting with questions and answers on Artemis 4.12.4

erators can approve given answers to mark them as correct. This is visually reflected by a green coloring of the according answer (see Figure 3.3b).

3.3 Proposed System

3.3.1 Functional Requirements

The open source character of the Artemis software project implies that users can actively request new features or adjustments for shortcomings of the current system. This ensures that Artemis' developers address changing needs as well as usability issues. During the requirements elicitation phase we not only analyzed the state-of-the-art in Chapter 2, but also specifically tried to collect requirements from Artemis users with different roles such as students, tutors, or instructors. Consequently, the Functional Requirements (FRs) that are not included in the tables provided in Section 5.1 originate from direct user input.

We divide the FRs into five areas of improvements that map to the problems identified in Section 1.1: *Topic Structure and Organization*, *Post Discoverability*, *Content Redundancy Reduction*, *Moderation Effort Reduction*, and *Interaction*.

Topic Structuring and Organization

The following FRs base on the feature comparison summarized in Section 2.2.1 and the analysis of external communication platforms used in parallel to

Artemis. These FRs aim to improve the currently insufficient topic organization which lacks course-wide topics ([P_1]).

FR1.1 **Post content of course-wide relevance**: Users are enabled to create postings of course-wide relevance, detached from specific exercises or lectures.

FR1.2 **Choose between predefined course-wide topics for a course post**: Users can select one of the predefined course-wide topics to create posts of course-wide relevance.

Post Discoverability

We further derive FRs that improve the visibility and discoverability of posts ([P_2]). State-of-the-art features are described in detail in Section 2.2.2.

FR2.1 **Summarize a post by title**: Post authors can summarize the content of their post by providing a title that enables other users to grasp the content of the post very quickly.

FR2.2 **Tag a post**: Post authors can tag their post to classify the content by creating new tags or reusing tags from existing posts. This enables other users to search for related posts by tag.

FR2.3 **Provide a course discussion overview listing all posts**: There is one view that serves as course-wide discussion overview and lists all posts within the course, regardless of their context, i.e., exercise, lecture, or course-wide.

FR2.4 **Navigate to a posting**: Users can navigate to a posting in any context.

FR2.5 **Search a post by text**: Users can search all posts within a course by providing a search term (i.e., text-based search) that refers to the title, content, or tag of a post.

FR2.6 **Search a post within a certain context**: Users can restrict their text-based search to a certain context.

FR2.7 **Filter posts by context**: Users can filter all posts within a course by selecting a certain context that they are interested in.

FR2.8 **Filter posts by answer state**: Users can filter posts by the answer state, that is, if there are any answers or not, or if there is a resolving answer among the provided ones.

FR2.9 **Filter posts to own posts only**: Users can filter posts to only see posts they have created.

FR2.10 **Filter posts to posts that the user answered or reacted on**: Users can filter posts by contribution state, i.e., if they authored an answer post or reacted to the discussion.

FR2.11 **Sort postings by date**: Users can sort postings by creation date.

FR2.12 **Sort posts by votes**: Users can sort posts by a popularity metric such as number of votes.

FR2.13 **Sort posts by answers**: Users can sort posts by answer count.

FR2.14 **Pin a post**: Moderators can pin posts in order to list them at the top in the according context i.e., give them a higher visibility.

FR2.15 **Archive a post**: Moderators can archive posts in order to list them in any view at the bottom of any rendered list of posts i.e., give them a lower visibility.

FR2.16 **Bookmark a post**: Users can bookmark posts that are of special interest for them and that should be found quickly.

Content Redundancy Reduction

The identified problem of redundant, duplicated content ([P_3]) that limits efficient and effective information retrieval is addressed by the FRs suggested in the following. Those base on the approaches used on other platforms as compared in Section 2.2.3.

FR3.1 **Conduct a similarity check during post creation**: During the creation of a post, the author should be automatically informed about the similar posts that already exist to avoid duplication of questions.

FR3.2 **Merge duplicated posts**: Moderators can merge posts that contribute to the same discussion.

FR3.3 **Provide a FAQ section**: Moderators can provide an FAQ section that anticipates common questions that are expected to arise during the course

FR3.4 **Write a post anonymously**: Students can publish posts anonymously instead of relying on private one-to-one conversations with moderators.

FR3.5 **Import postings of previous courses**: Instructors can import exercises, lectures, or courses together with their accompanying discussions, i.e., posts and answer posts.

Moderation Effort Reduction

The following FRs specifically aim at reducing moderation efforts ([P_4]). Results of the state-of-the-art analysis that concern platform moderation effort are presented in Section 2.2.4.

FR4.1 **Mark a post as resolved**: Post authors can mark their problem as resolved by marking the answer that solved the origin problem as correct, i.e., resolving.

FR4.2 **Change the context of a post**: Moderators can change the context of a post when the edit this post.

FR4.3 **Enable internal communication for moderators**: Moderators have a dedicated communication channel, where they can organize and communicate topics that are private within that authorized group.

FR4.4 **Report a posting**: Users can report a postings if the content violates certain communication rules.

Interaction

Within Section 2.2.5 of the state-of-the-art analysis, we investigate if other platforms have mechanisms in place that increase social engagement. The following FRs capture enhancements for user interaction and more emotionally involved communication ([P_5]).

FR5.1 **Use emojis within a posting**: Users can embed emojis within the post content they create to express a broader range of affective states.

FR5.2 **React on a posting with emojis**: Users can (emotionally) react on postings with emojis.

FR5.3 **Use emojis to up-vote a post**: Users can react with dedicated emojis to express agreement or consent with posts.

FR5.4 **Reference a user**: Users can reference other users that participate in the course.

FR5.5 **Reference a posting**: Users can reference other postings to create links between postings.

FR5.6 **Reference a course entity**: Users can reference course entities such as lecture units or certain exercise tasks inside postings.

FR5.7 **Reward a user for discussion contributions**: Users that have most communication activity and whose answers resolved posts are rewarded.

FR5.8 **Notify a user on discussion events outside Artemis**: Users can follow a discussion to stay informed about updates via notification mechanisms (e.g., email or push notification message).

3.3.2 Nonfunctional Requirements

As part of the requirements elicitation, we also define requirements that cover nonfunctional aspects of the system [BD09]. We make use of the established *FURPS+* model [JBR99] to list the identified Nonfunctional Requirements (NFRs) for the proposed system.

Usability

NFR1.1 **Understandability**: Users should be able to easily understand how to use the newly introduced features and how to improve their learning or moderation experience by making use of them.

NFR1.2 **Documentation**: Users should be able to access comprehensive documentation and tooltips that explain features in simple language.

NFR1.3 **Consistency**: The system should depend on readily available coloring and font schemes to not break with existing design patterns.

Reliability

NFR2 **Security**: The system should prevent malicious user input from causing harm or leaking data to unauthorized users. Besides, the new communication capabilities should not ease so-called Denial of Service (DoS) attacks.

Performance

NFR3.1 **Real-time Support**: The system should support real-time updates of the User Interface (UI) in case of newly added, deleted, or updated postings or reactions.

NFR3.2 **Response Time**: The system should immediately respond to a user interaction. This could be a loading spinner within the clicked button indicating that the client is waiting for the server response or, in case of a quick server response, immediately show the updated UI.

Supportability

NFR4 **Maintainability**: The current implementation of Artemis Q&A suffers duplication in client and server code which makes it difficult to consistently refactor code and fix bugs as they occur in multiple places. [NFR4] aims at modularized code design by following the *SOLID* principles [Mar14]. This reduces code clones, which imply manual syncing efforts, and eases code refactoring and bug fixing.

3.4 System Models

3.4.1 Scenarios

In this section, we provide two types of scenarios: visionary scenarios and demonstration scenarios. Each scenario represents an instance of a use case and describes a flow of events [BD09]. The visionary scenario prototypes an idea of the future system, whereas the demo scenarios, also called *as-is* scenario, describe the current situation and can be validated by users [BD09].

Visionary Scenarios

Scenario name	Targeted notification and smart moderator responsibility
Participating actor instances	***Alice***: Student in course *Patterns of Software Engineering* ***Bea***: Tutor, that previously answered questions covering the topic `JavaFX` in programming exercises ***Charlie***: Tutor, that previously answered questions related to exercises on architectural patterns ***Donald***: Tutor, that previously answered questions related to exercises on behavioral patterns
Flow of events	1. Alice is working on an exercise on the architectural pattern *Model View Controller* and is not able to run the project in her Integrated Development Environment (IDE) 2. Alice creates a post with the content “When running the project, I get the Error: `JavaFX` runtime components are missing, but are required to run the application” 3. Alice uses the tags ‘JavaFX’ and ‘Eclipse’ to indicate that her problem is related to the `JavaFX` framework and she is using the Eclipse IDE 4. Bea receives a mobile push notification about a new question targeting her area of expertise, as her answers to questions involving `JavaFX` always resolved the students’ problems 5. Charlie receives a mobile push notification about a new question targeting an exercise that he previously answered questions for 6. Bea clicks the provided link in her push notification and is navigated to Alice’s question 7. Bea answers the question with the hint that Alice should import the exercise as a Maven project in Eclipse 8. Alice is notified via mobile push notification that her question was answered and clicks the provided link to navigate to Bea’s reply 9. Alice follows Bea’s instructions and confirms that the described error is gone by marking Bea’s answer as resolving 10. Charlie logs into Artemis later to check the questions that Artemis recommended him to answer and sees that Alice’s problem is already resolved 11. Donald logs into Artemis later to check any questions that Artemis recommended him to answer based on his previous engagement in discussions; he does not see Alice’s post

Table 3.1: Visionary scenario 1

Demo Scenarios

Scenario name	Exercise specific question with reference
Participating actor instances	***Alice***: Student in course *Patterns of Software Engineering* ***Bea***: Tutor responsible for technical support ***Charlie***: Tutor responsible for *Template Method Pattern* exercise
Flow of events	1. Alice is working on a *Template Method Pattern* programming exercise and does not manage to achieve 100% with her implementation, even though she is sure that everything is implemented correctly and there has to be a general, technical issue 2. Alice creates a post with the content "Could someone from the tutors please check my code? I really do not think that I have a mistake but I am still missing 5 percent... Maybe something went wrong with the test?" and assigns it the course-wide topic 'Tech Support' 3. Alice uses the tag 'Tests' to indicate that there might be a problem with the automatic test runs 4. Bea reads this post when checking for new posts in her daily routine by filtering for unresolved posts in this context 5. Bea knows, that this is not a general, or course-wide problem but rather specifically related to the template method pattern exercise; she edits the post and changes the context to 'Template Method Exercise' 6. Charlie reads this post when checking for new posts in his daily routine by filtering for unresolved posts in this context 7. Charlie knows that other students faced similar problems in that exercise, which he solved along the discussion of the post with id 30; he simply uses the post reference ('#30') to answer Alice's post in a very short reply. 8. Alice is notified about the new reply to her post and checks the proposed solution on the exercise page the post was moved to by Bea 9. Alice marks the answer post from Charlie as resolving after having checked that all tests passed when following the steps described in the post with id 30

Table 3.2: Demo scenario 1

Scenario name	<u>Course organization without duplicated questions</u>
Participating actor instances	***Alice, Bob***: Students ***Charlie***: Tutor responsible for organizational issues ***Donna***: Instructor
Flow of events	1. Alice is interested in participating in a repetitorium for the software engineering course she took this semester 2. Alice visits the course discussion overview of that course and filters the posts to those addressing organizational questions by setting the context filter accordingly 3. Alice does not find any information on an upcoming repetitorium among the filter matches 4. Alice creates a post with the course-wide topic 'Organization' including a title 'Retake Repetitorium' 5. Bob, one day later, wants to ask if there is a repetitorium offered in that same course 6. Bob navigates to the course discussion overview and immediately starts to create a new post 7. Bob enters a post title including the term 'Repetitorium' 8. Bob is automatically provided a list of similar posts right below the title input field 9. Bob browses the suggested posts and finds the post created by Alice having a similar title 10. Bob expands the question content, which turns out to address the same issue 11. Bob navigates to the post by clicking the title and recognizes that there is no answer yet 12. Bob votes for question with a '+' emoji to express, that he is also interested in getting information on the repetitorium 13. Bob reacts with the 'nerd' emoji to make other people laugh reading this post 14. Charlie visits the course discussion overview, filters on organizational questions, and checks the option to only show unresolved posts as he wants to know, which students require organizational information 15. Charlie sees Alice's post at the top having one vote 16. Charlie starts the discussion and provides a short information on the preliminary plans for holding a repetitorium 17. Charlie marks his answer as resolving and thereby sets Alice's post to resolved 18. Donna realizes that students are interested in this topic as the retake exam approaches and decides to create an announcement including all relevant information 19. Alice, Bob and all other course students automatically receive an email that notifies them about the new announcement

Table 3.3: Demo scenario 2

Scenario name	Real-time updates of ongoing discussions on multiple clients
Participating actor instances	***Alice***: Student ***Donna***: Instructor
Flow of events	1. Donna uses filters to browse the course discussion overview to search for unresolved, organizational questions that require her input 2. Alice creates a post with the course-wide topic 'Organization' as she wants to know if there will be on-site tutorials during the semester 3. Donna's filtered view on course discussion posts is updated automatically and Alice's post pops up on Donna's screen 4. Donna answers Alice's question by creating an answer post 5. Alice, who filtered the list of course discussions to posts that she created, instantly sees that Donna has added an answer to her post 6. Alice marks Donna's answer post as resolving 7. Donna's filtered view is updated and does not display Alice's post anymore as it is now resolved

Table 3.4: Demo scenario 3

3.4.2 Use Case Model

In the following, we illustrate use cases implemented as part of this thesis. They describe the system's behavior from an actor's point of view [BD09]. Thereby, actors are defined as external entities which interact with the system [BD09]. We restrict the use case models to those use cases that describe new functionality derived from the formulated FRs. The actors involved in the following use cases are:

- **Instructor**: User, that is primarily responsible for a course
- **Tutor**: User, that operationally supports the instructor by providing support for students and participates in the discussions to share information and answer questions
- **Student**: User, that participates in a course and discussions to seek information and ask questions

Notably, use cases applicable for students are applicable for tutors and use cases applicable for tutors are also applicable for instructors. In order to stick to our convention, we use the term moderator, if referring to tutors *and* instructors.

The use cases shown in Figure 3.4 include FRs of three different areas of improvements. Moderators can change the display priority. If they want to change the default visibility, they can either chose to make a post more visible by pinning it to the top of a view or archiving it to always list it at the bottom of a view. Those actions specifically increase or decrease the post visibility if required. To allow more efficient discussion moderation, moderators can change the post context, that is the page at which a post is shown, namely a lecture or exercise page, or the course discussion overview. When moderators edit a post and select a different, more suitable post context, they effectively move the post. Moderators can mark their own but also answers of students as resolving. To other moderators and the post author, this indicates that the given answer is approved and thereby the original post is resolved. Finally, an instructor can post an announcement which is a post with the specified course-wide topic 'Announcement' in the course context. Per definition, announcements are posts of particular relevance and are therefore created with a high display priority as pinned posts.

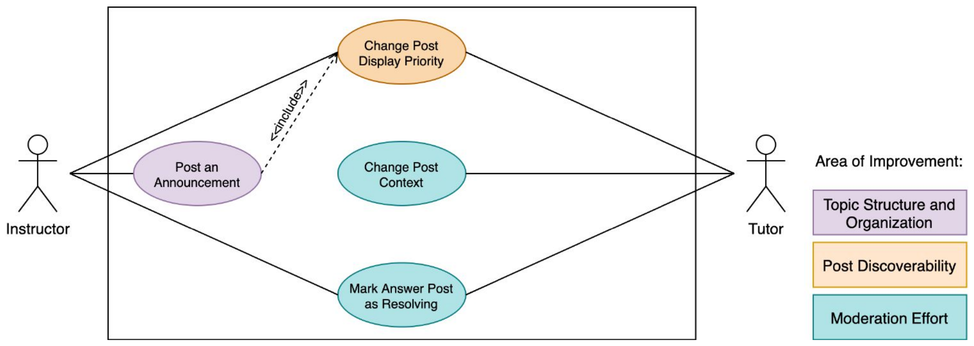


Figure 3.4: New use cases for instructors and tutors as UML use case diagram

Figure 3.5 shows use cases from a student's perspective. A student can search *all* existing postings in a course discussion overview through filters, sorting options, and text- and id-based search for faster information retrieval and better post discoverability. Regarding the topic structure, students can choose from course-wide topics 'Tech Support' or 'Organization' that allow having the course as context rather than a specific exercise or lecture. More specifically, in addition to existing exercise or lecture posts that have to be created on the exercise and lecture page, respectively, course-wide content can be posted on the course discussion level. Additionally, students can mark answer posts to their question as *resolving*, to indicate that the initial problem is resolved and no further support is required. To expand the possibilities to interact with peers, students can react on postings by choosing from the commonly known emoji palette. Furthermore, referencing postings allows users to communicate more interactively and spread discussions over multiple posts or even contexts without sacrificing structuredness and readability.

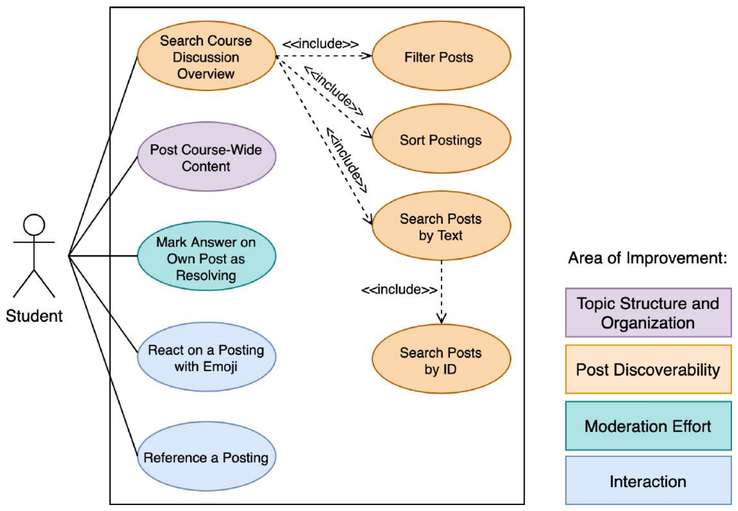


Figure 3.5: New use cases for students as UML use case diagram

3.4.3 Analysis Object Model

The analysis object model captures properties and relationships of individual concepts that are manipulated by the system [BD09]. The visualization given in Figure 3.6 represents and structures the main concepts visible to the user [BD09]. We briefly describe newly introduced objects and their relations in the following.

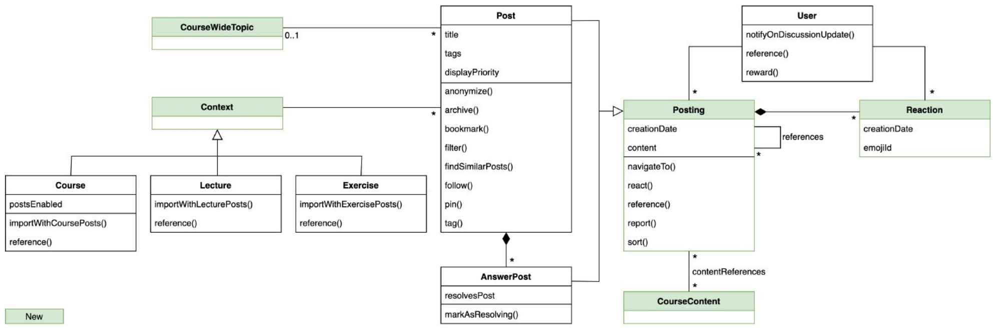


Figure 3.6: Analysis object model as UML class diagram

Every `Posting` is either a `Post`, which starts a discussion, or an `Answer Post` which is every `Posting` in response to a `Post`. These discussions have to be related to one of three possible contexts, that are `Exercise`, `Lecture`, or `Course`. `Posts` that are created in the `Course` context will cover exactly one of the provided `CourseWideTopics`, such as 'Organization'.

`Answer Posts` can be marked as the one resolving the problem stated in the original `Post`. A `Post`, as starting point of a discussion on any topic, exhibits some additional operations that facilitate the discoverability of such topics, i.e., through pinning (upgrade the priority with which a `Post` is displayed in its context) and archiving (downgrade the priority), bookmarking, and following. If `User`s follow a discussion, they will be notified on updates. In order find similar posts and ultimately decrease the amount of duplicated content, a `Post` is compared to other `Post`s.

Users can react to a `Posting` with multiple emoji-based `Reactions`, sort a list of `Posting`s by a certain criteria such as creation date, or report it to moderators if its content violates course discussion rules. Additionally, a `User` can navigate to a single-view of a `Posting`. Within the content of a `Posting`, (multiple) other `Posting`s can be referenced. `Posting`s can further refer to other types of `CourseContent` outside the context which might be a certain slide of the course material (e.g., "[Page 6 of the syllabus] states that..."), sequences of a lecture video (e.g., "The concept is well explained at [30:54 min of Introduction Lecture]..."), or a specific task in a programming exercise (e.g., "I fail to get 100% for [task 3 of Programming Exercise 1]...").

3.4.4 Dynamic Model

Since dynamic models focus on the behavior of the proposed system [BD09], this section presents a dynamic model of an essential use case introduced in this thesis: the de-duplication of posts. Figure 3.7 captures the event flow in an activity diagram. The actors in this process are a *Student* that has a question and therefore interacts with the *Artemis System* through the Artemis UI.

The event flow is initiated as the *Student* starts to create a post to formulate the question. Simultaneously, the *Artemis System* creates a list of posts that are similar to the one that is currently being created. The *Student* is provided with that list in the UI during the creation process. The *Student* can browse the previewed posts and decide whether one the suggested posts matches what they want to ask or not. If there is no matching post among the suggestions, the *Student* can proceed with creating a new post. The new post is added to the existing posts by the *Artemis System* as soon as the creation is finished. In case the student finds one of the suggested posts to be similar, they can navigate to the post to get more information. We further refer to that post as `SimilarPost`. If `SimilarPost` is not yet answered, the *Student* can up-vote `SimilarPost` to indicate their interest in it being answered. As the *Student* subscribes to the answers of `SimilarPost`, they will be automatically notified on updates. The process of creating a new post is not continued but ended with finding `SimilarPost` that is subscribed to. If `SimilarPost` has been marked as resolved, the *Student* will find the correct answer among the given answers and might or might not react on it. Again, the process of creating a new post is not continued but ended with finding the answer without posting a duplicate question. If `SimilarPost` is already answered but has not yet been marked as resolved, the *Student* can read the given answer(s). In case one of the given answers solves the *Student*'s problem, they can react on this answer. Otherwise, the *Student* may up-vote `SimilarPost` as if they do when there are no answers provided yet. According to the logic described in the activity diagram, a new post is published only if it does not duplicate an existing post in the course, regardless of whether the existing post has been replied to or not.

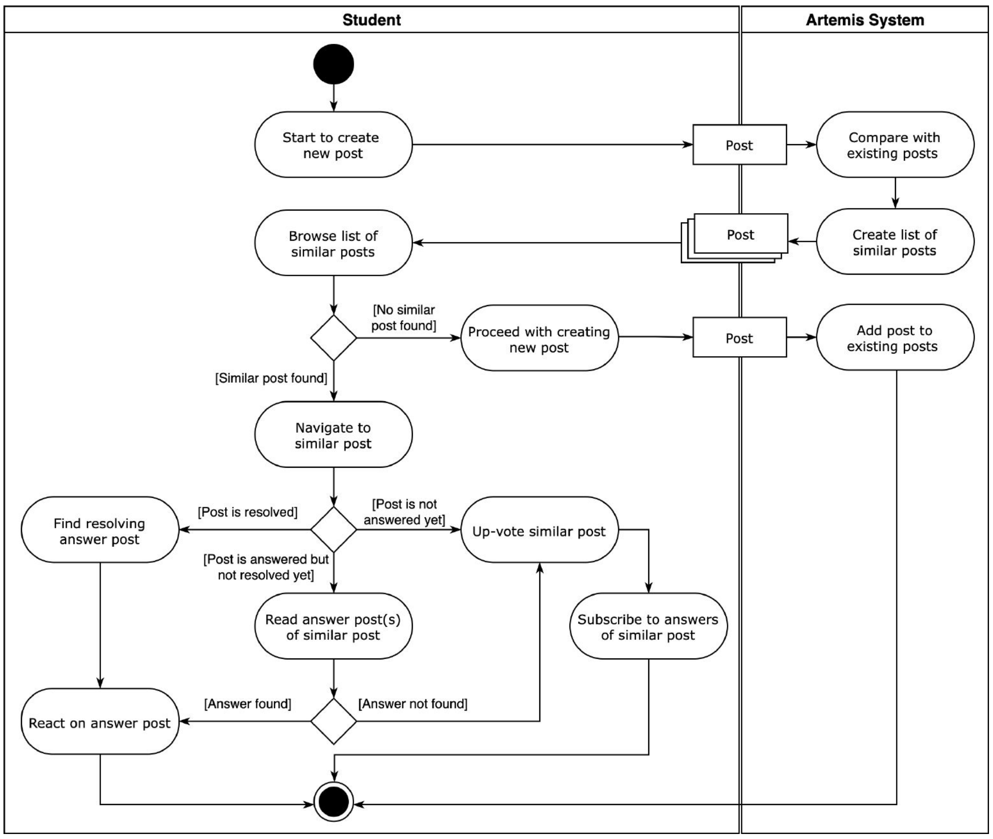

Student
Artemis System
Start to create new post
Post
Compare with existing posts
Browse list of similar posts
Post
Create list of similar posts
[No similar post found]
Proceed with creating new post
Post
Add post to existing posts
[Similar post found]
Navigate to similar post
Find resolving answer post
[Post is resolved]
[Post is not answered yet]
Up-vote similar post
[Post is answered but not resolved yet]
Read answer post(s) of similar post
Subscribe to answers of similar post
React on answer post
[Answer found]
[Answer not found]

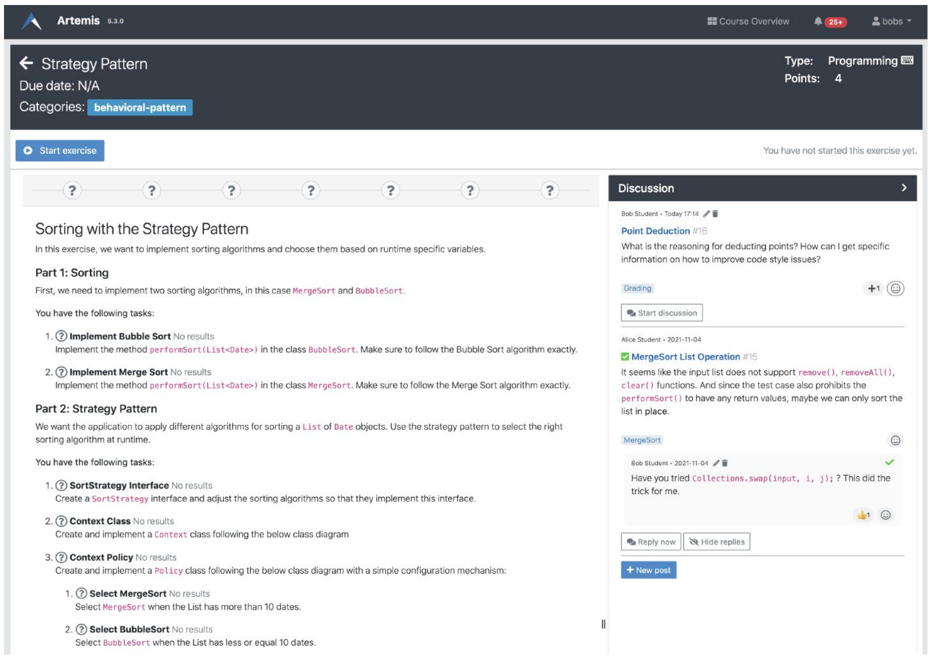
Figure 3.7: Dynamic model for post de-duplication as UML activity diagram

3.4.5 User Interface

During the analysis phase of the proposed METIS subsystem, we created mock-ups and click-prototypes to communicate on ideas and test them against the usability NFRs. Since the implemented features and new components are already integrated in the current Artemis version 5.3.0, we refrain from pasting mock-ups but instead share screenshots of the actual UI as it is currently actively used. For each of the contexts that a discussion can be associated with, we present how the FRs were translated into features of the novel METIS subsystem.

Context: Exercise Discussion Section

Figure 3.8 displays how a discussion is integrated with a problem statement of a programming exercise in the pre-existing side panel style. The listed posts have a summarizing title, and tags such as 'Grading' and 'MergeSort' ([FR2.1], [FR2.2]). As the first post was up-voted by adding the dedicated emoji reaction ([FR5.2]) it is listed at the top of the list. The second post has an answer that was marked as resolving the problem stated in the post ([FR4.1]).



Strategy Pattern
Due date: N/A
Categories: behavioral-pattern

Type: Programming
Points: 4

Start exercise

You have not started this exercise yet.

Sorting with the Strategy Pattern

In this exercise, we want to implement sorting algorithms and choose them based on runtime specific variables.

Part 1: Sorting

First, we need to implement two sorting algorithms, in this case `MergeSort` and `BubbleSort`.

You have the following tasks:

1. **Implement Bubble Sort** No results
   Implement the method `performSort(List<Date>)` in the class `BubbleSort`. Make sure to follow the Bubble Sort algorithm exactly.
2. **Implement Merge Sort** No results
   Implement the method `performSort(List<Date>)` in the class `MergeSort`. Make sure to follow the Merge Sort algorithm exactly.

Part 2: Strategy Pattern

We want the application to apply different algorithms for sorting a `List` of `Date` objects. Use the strategy pattern to select the right sorting algorithm at runtime.

You have the following tasks:

1. **SortStrategy Interface** No results
   Create a `SortStrategy` interface and adjust the sorting algorithms so that they implement this interface.
2. **Context Class** No results
   Create and implement a `Context` class following the below class diagram
3. **Context Policy** No results
   Create and implement a `Policy` class following the below class diagram with a simple configuration mechanism:
   1. **Select MergeSort** No results
      Select `MergeSort` when the List has more than 10 dates.
   2. **Select BubbleSort** No results
      Select `BubbleSort` when the List has less or equal 10 dates.

Discussion

Bob Student • Today 17:14

Point Deduction #16

What is the reasoning for deducting points? How can I get specific information on how to improve code style issues?

Grading

+1

Start discussion

Alice Student • 2021-11-04

✓ MergeSort List Operation #15

It seems like the input list does not support `remove()`, `removeAll()`, `clear()` functions. And since the test case also prohibits the `performSort()` to have any return values, maybe we can only sort the list in place.

MergeSort

Bob Student • 2021-11-04 ✓

Have you tried `Collections.swap(input, i, j);` ? This did the trick for me.

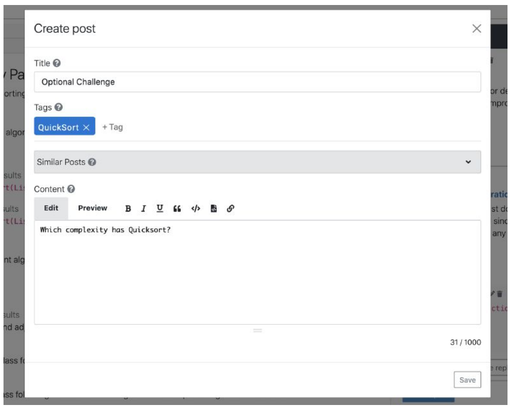
👍1

Reply now | Hide replies

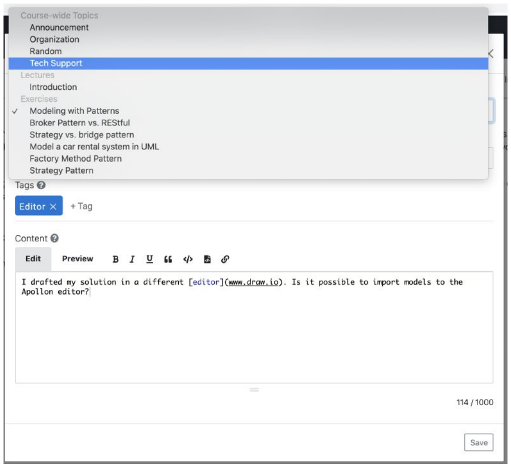
+ New post

Figure 3.8: Discussion section of an exercise page on Artemis 5.3.0

If a user clicks the button to add a new post, the modal containing a form to be filled is shown as depicted by Figure 3.9a. During the process of creating a new post, METIS determines and displays similar posts to prevent duplication ([FR3.1]). The post de-duplication is discussed in detail in Section 5.1. Similarly, a modal is opened when editing an existing posting. In contrast to students, moderators can not only change title, tags and content of an existing post, but they are also allowed to change the context ([FR4.2]) as shown in Figure 3.9b. This enables moderators to move posts between contexts.



(a) Creating a post on an exercise page



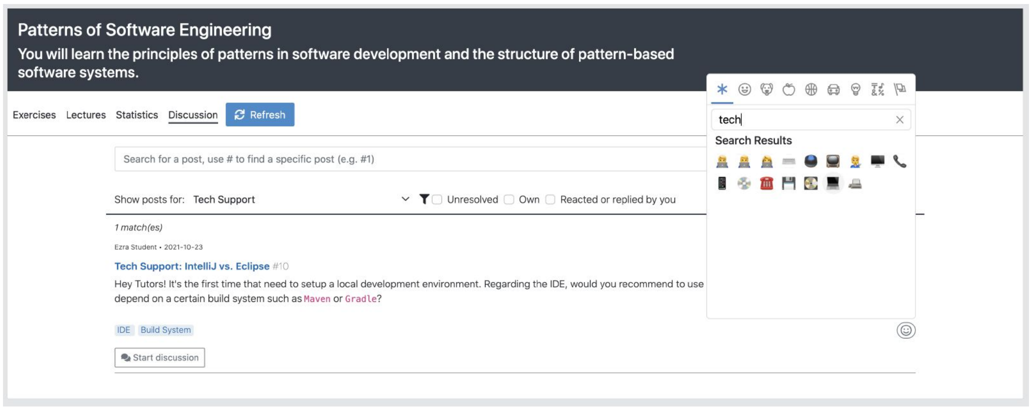
I drafted my solution in a different [editor](www.draw.io). Is it possible to import models to the Apollon editor?

(b) Editing a post as moderator

Figure 3.9: Creating and editing exercise posts on Artemis 5.3.0

Context: Course Discussion Overview

Artemis 5.3.0 exhibits a dedicated discussion overview to host conversations with course-wide topics but also includes all posts on exercises and lectures in this course. Hence, when users enter the tab 'Discussion' shown in the course discussion overview as captured by Figure 3.10, they are provided with a searchable and filterable list of *all* posts in the course ([FR2.3]).


Exercises
Lectures
Statistics
Discussion
Refresh
Search for a post, use # to find a specific post (e.g. #1)
Show posts for: Tech Support
Unresolved
Own
Reacted or replied by you
tech
Search Results
IDE
Build System
Start discussion

Patterns of Software Engineering
You will learn the principles of patterns in software development and the structure of pattern-based software systems.

1 match(es)

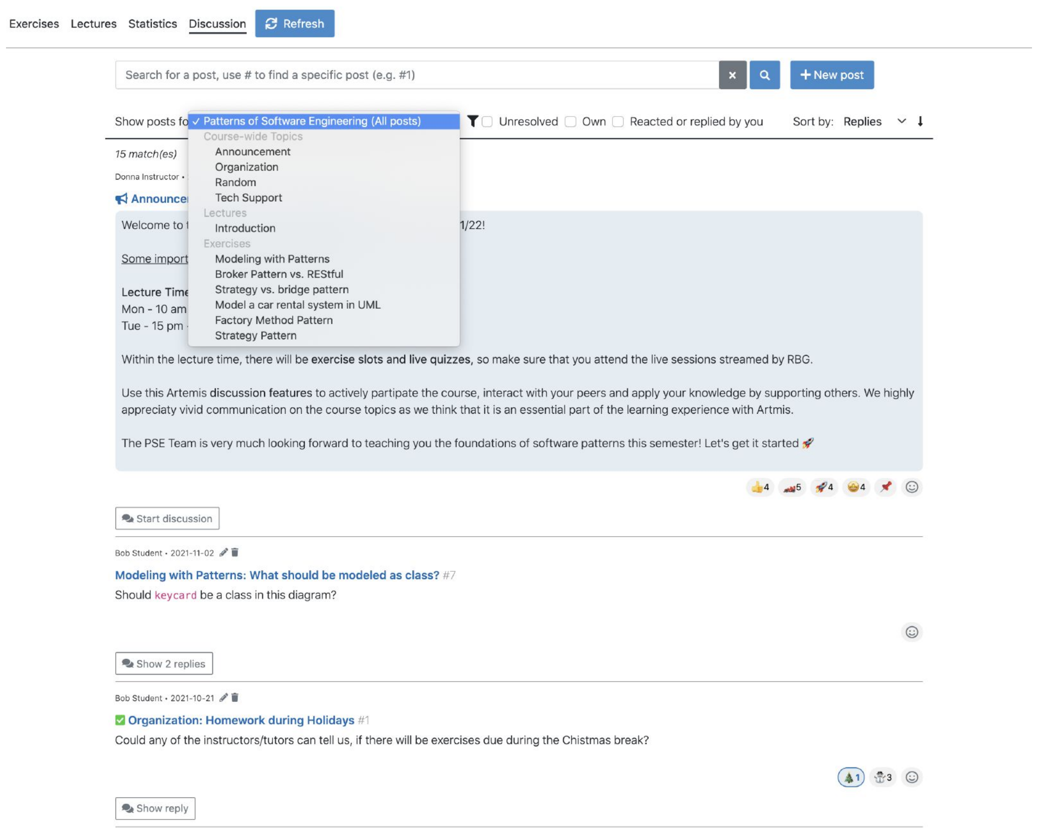
Ezra Student • 2021-10-23

Tech Support: IntelliJ vs. Eclipse #10

Hey Tutors! It's the first time that need to setup a local development environment. Regarding the IDE, would you recommend to use depend on a certain build system such as Maven or Gradle?

Figure 3.10: Reacting on post with course-wide topic on Artemis 5.3.0

The post that is shown in the screenshot addresses a technical issue that was raised by a student and is not associated to a certain exercise or lecture. For this purpose, the student used the course-wide topic 'Tech Support' ([FR1.1]). Users can react on postings by clicking the according button that toggles a component to select an emoji ([FR5.2]).



15 match(es)

Donna Instructor •

Announce

Welcome to 1/22!

Some import

Lecture Time
Mon - 10 am
Tue - 15 pm

Within the lecture time, there will be **exercise slots and live quizzes**, so make sure that you attend the live sessions streamed by RBG.

Use this Artemis **discussion features** to actively partipate the course, interact with your peers and apply your knowledge by supporting others. We highly appreciaty vivid communication on the course topics as we think that it is an essential part of the learning experience with Artmis.

The PSE Team is very much looking forward to teaching you the foundations of software patterns this semester! Let's get it started 🚀

👍4 5 🚀4 4 📌

Bob Student • 2021-11-02

Modeling with Patterns: What should be modeled as class? #7

Should `keycard` be a class in this diagram?

Bob Student • 2021-10-21

✅ **Organization: Homework during Holidays** #1

Could any of the instructors/tutors can tell us, if there will be exercises due during the Chistmas break?

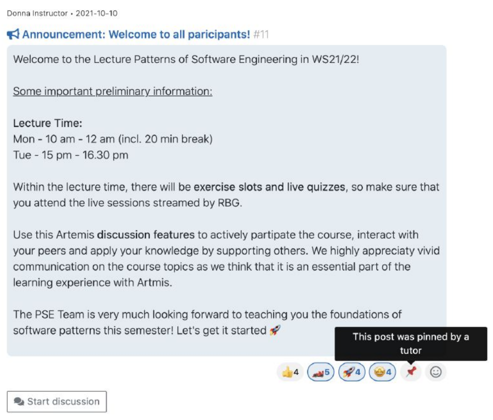
🎄1 ☃3

Figure 3.11: Filtering and sorting course discussion posts on Artemis 5.3.0

Figure 3.11 delineates other aspects of the course discussion overview: A search bar allows users to find posts exhibiting specific buzzwords ([FR2.5]). Users can select a specific context ([FR2.6], [FR2.7]) from the list of predefined, course-wide topics ([FR1.2]) as well as every course lecture and exercise to narrow down the displayed list of posts. Besides, filters that reflect other post criteria such as if a post is unresolved ([FR2.8]), was authored ([FR2.9]), answered or reacted by the current user ([FR2.10]), can be applied. To change the order of displayed posts, users can select a sort criterion ([FR2.11], [FR2.12], [FR2.13]). When browsing the posts in the course discussion overview, all post titles are prefixed with the post's context. For example, the second post in the previewed list in Figure 3.11 was created on the exercise page ‘Modeling with Patterns’ which is reflected by having the exercise name prepended to the post title. The context name is clickable and navigates the user to the exercise or lecture accordingly ([FR2.4]).

The posts included in the screenshot shown in Figure 3.11 also illustrate how emojis are used to express emotional reactions ([FR5.2]). The feature is strongly based on other implementations of so-called reaction bars, i.e., a horizontal list of emojis with according counts, as integrated in Slack for example.

Moreover, one of the course-wide topics that only an instructor is authorized to choose for a post is ‘Announcement’. Posts of these types have a bullhorn icon at the beginning of their header including the ‘Announcement’ label and the title. Furthermore, their content visually stands out by its colored background as can be seen in Figure 3.12a. By default, an announcement is pinned, meaning that it is shown at the top of the course discussion overview. Regarding a configurable display priority of *any* post, moderators are provided with the means to pin ([FR2.14]) or archive ([FR2.15]) posts. As shown in Figure 3.12b these options are represented by dedicated buttons with according emojis to respectively increase or decrease visibility, i.e., moving the post to the top or the bottom of a discussion list.



Donna Instructor • 2021-10-10

Announcement: Welcome to all paricipants! #11

Welcome to the Lecture Patterns of Software Engineering in WS21/22!

Some important preliminary information:

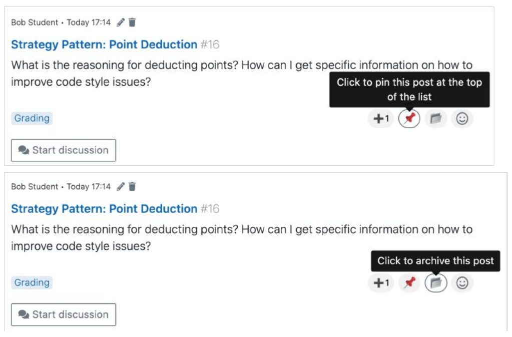
Lecture Time:
Mon - 10 am - 12 am (incl. 20 min break)
Tue - 15 pm - 16.30 pm

Within the lecture time, there will be **exercise slots and live quizzes**, so make sure that you attend the live sessions streamed by RBG.

Use this Artemis **discussion features** to actively partipate the course, interact with your peers and apply your knowledge by supporting others. We highly appreciaty vivid communication on the course topics as we think that it is an essential part of the learning experience with Artmis.

The PSE Team is very much looking forward to teaching you the foundations of software patterns this semester! Let's get it started 🚀

(a) Announcement



Bob Student • Today 17:14

Strategy Pattern: Point Deduction #16

What is the reasoning for deducting points? How can I get specific information on how to improve code style issues?

Click to pin this post at the top of the list
Grading
+1
Start discussion

Bob Student • Today 17:14

Strategy Pattern: Point Deduction #16

What is the reasoning for deducting points? How can I get specific information on how to improve code style issues?

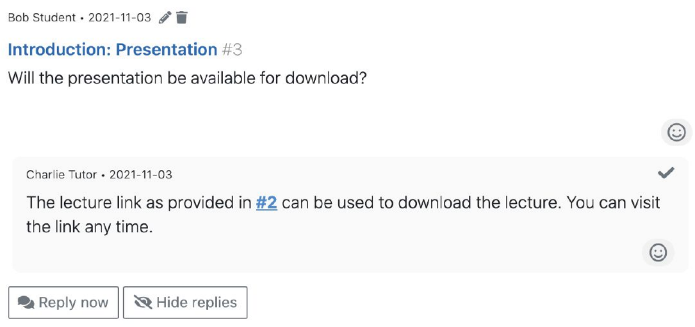
Click to archive this post
Grading
+1
Start discussion

(b) Pinning or archiving a post

Figure 3.12: Display priority of posts on Artemis 5.3.0

Context: Lecture Discussion Section

Posts can also target a certain lecture. The answer provided in Figure 3.13, uses the identifier of a lecture post encapsulated in a defined pattern ‘#4’ to reference another post ([FR5.5]). By clicking this reference, the user is navigated to the so-called single-post view as shown in Figure 3.14. Instead of listing all posts in the context, the referenced post and its expanded answers are displayed. The user can easily choose to see all posts within the lecture context by clicking on the arrow label 'Show all posts'.



Bob Student • 2021-11-03

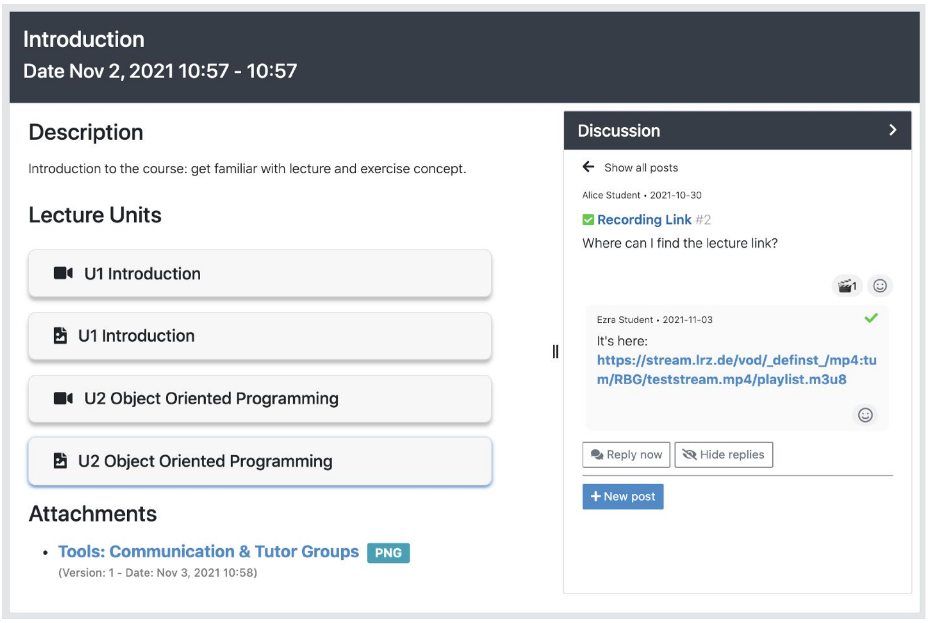
Introduction: Presentation #3

Will the presentation be available for download?

Charlie Tutor • 2021-11-03

The lecture link as provided in #2 can be used to download the lecture. You can visit the link any time.

Figure 3.13: Using a reference to another post on Artemis 5.3.0



Introduction
Date Nov 2, 2021 10:57 - 10:57

Description

Introduction to the course: get familiar with lecture and exercise concept.

Lecture Units

U1 Introduction
U1 Introduction
U2 Object Oriented Programming
U2 Object Oriented Programming

Attachments

- Tools: Communication & Tutor Groups PNG
  (Version: 1 - Date: Nov 3, 2021 10:58)

Discussion

Show all posts

Alice Student • 2021-10-30

Recording Link #2

Where can I find the lecture link?

Ezra Student • 2021-11-03

It's here:
https://stream.lrz.de/vod/_definst_/mp4:tum/RBG/teststream.mp4/playlist.m3u8

Figure 3.14: Single post view in a discussion section of a lecture page on Artemis 5.3.0

Chapter 4

System Design

4.1 Overview

System design involves transforming the analysis object, that is the artifact resulting from Chapter 3, into a system design model [BD09]. We first describe design goals that we take into account during designing the proposed METIS subsystem [BD09]. Second, we provide models of the decomposed system.

4.2 Design Goals

We derive the design goals from the NFRs introduced in Section 3.3.2. As the system design involves trade-offs with regard to the stated design goals, the following prioritization of NFRs guides the decisions.

First, we find the requirements summarized by *Usability* ([NFR1.1-1.3]) as most important. The newly added communication capabilities introduced by the METIS subsystem will only achieve the desired effects if we ensure high usability standards to motivate Artemis' users to actually make use of them.

Second, the *Performance* requirement *Real-time Support* ([NFR3.1]) needs to be taken into account. This requires extending the system design within the METIS subsystem by a websocket interface. Given that Artemis is continuously developing to be a full-fledged CMS, users will expect METIS to also adhere the standards of real-time updates that they are used to from other communication platforms.

Third, *Maintainability* ([NFR2]) should be increased when integrating the METIS subsystem with the existing Artemis system design. As we report in Chapter 7, not all FRs are completely implemented by this thesis. Thus, there are open goals as well as future work, which require METIS to be open for extensions and further enhancements.

Fourth, *Security* aspects influence design decisions. In contrast to other components, the METIS subsystem does not rely on sensitive data such as grades or personal user account information besides the username. Therefore, we regard [NFR2] as least important when designing the system, but still minimize the data transferred between client and server to achieve a higher level of data protection.

4.3 Subsystem Decomposition

At the top level, we identify two subsystems, the *Artemis Application Client* (see Figure 4.2) and the *Artemis Application Server* (see Figure 4.1). In both of the subsystems, we employ *Layering* as architectural style to impose a hierarchy onto subsystems within the *Artemis Application Client* and *Artemis Application Server*. Each of the layers provides higher-level services to the subsystem above by using lower-level services from the subsystem below it [BD09].

4.3.1 Client Decomposition

We decompose the METIS-related components of the *Artemis Application Client* subsystem into two layers.

The *Service Layer* encapsulates the components that interact with the *Artemis Application Server* subsystem via the provided Hypertext Transfer Protocol (HTTP) interface. Notably, the client-side `Metis Service` acts as a so-called facade [GHJV95] to the *Service Layer*: It provides `Page UI` components of the higher-level *User Interface Layer* with a unified interface. The `Metis Service` allows components to retrieve posts, reactions, or answer posts, while internally keeping track of the objects of the METIS subsystem and handling real-time updates via websockets. However, due to the complexity that needs to be managed by the `Metis Service`, it does not adhere to the single responsibility principle.

Within the *User Interface Layer*, we structure the components with regard to their atomicity to enable extensive reusability of the most atomic components. For instance, independently of displayed `Page UI` component, the discussion is composed of the same `Post` and `AnswerPost` components. Both components encapsulate the same sub-components, namely `Header`, `CreateEditModal`, and `Footer` that uses the `ReactionBar` component. To avoid duplication, we rely on class inheritance when implementing these com-

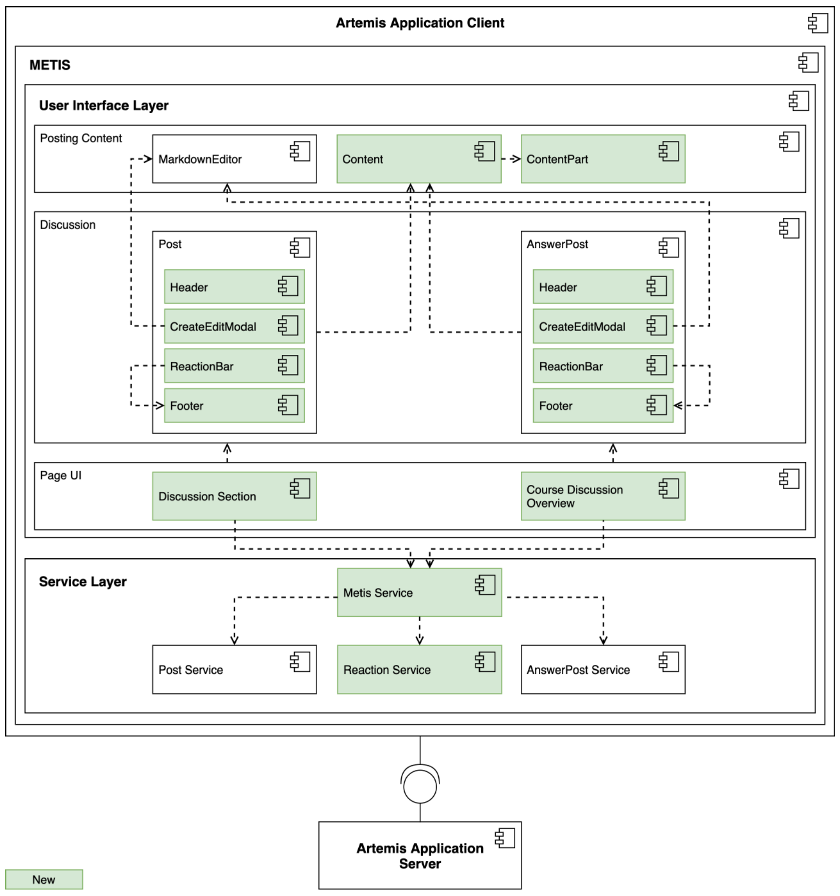


Figure 4.1: Client decomposition as UML component diagram

ponents. The `CreateEditModal` component uses `Posting Content` components to render post or answer contents. These components include the existing `MarkdownEditor` component, as well as the `Content` of a posting and `ContentParts` that further split the `Content` into components. Splitting the content into `ContentParts` allows us to embed metadata into the content, such as references to other posts.

4.3.2 Server Decomposition

Similar to the client, the *Artemis Application Server* follows a layered architecture, consisting of three layers.

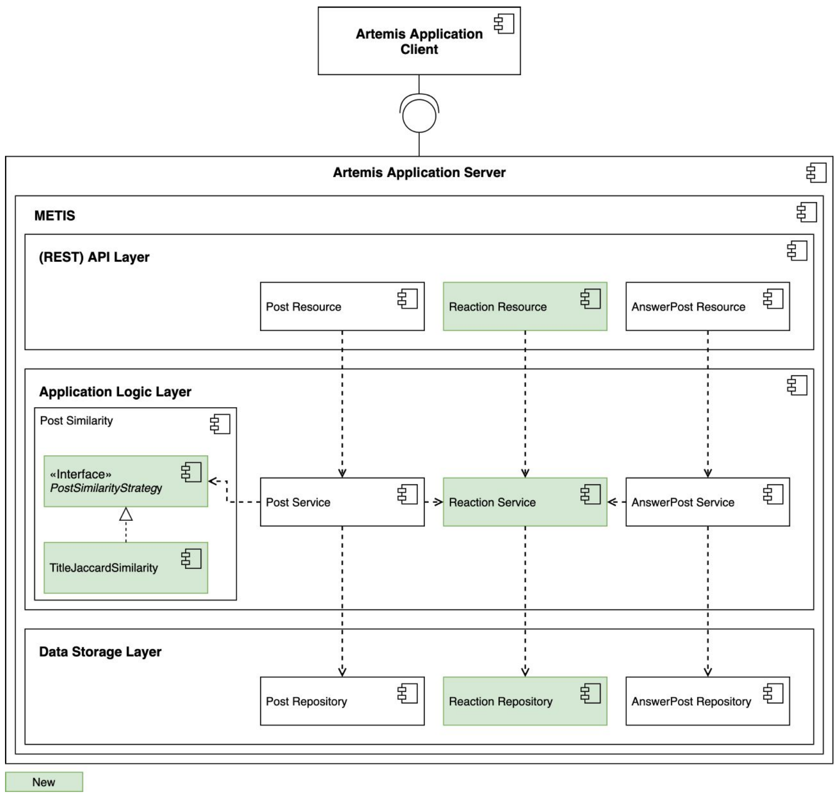


Figure 4.2: Server decomposition as UML component diagram

The *REST API Layer* provides the HTTP interface to the client. This layer consists of resources, where we added the `Reaction Resource` to the existing resources for post and answer post. The resources call the lower-level services, i.e., the new `Reaction Service`, the `AnswerPost Service`, and the `Post Service`. The latter uses the `Post Similarity` component that is added to enable duplication check by analyzing posts for their similarity. We explain the implementation of this component in detail within Section 5.1. Each service encapsulated in the *Application Logic Layer* further uses the according component of the *Data Storage Layer*. The enclosed components are the new `Reaction Repository`, the `AnswerPost Respository`, and the `Post Repository`. The *Data Storage Layer* is responsible for actually performing the requested operations on the database.

4.3.3 Client-Server Communication

Next to the HTTP-based communication interface between client and server, Artemis offers capabilities to implement websocket-based communication. Yet, the existing Artemis Q&A implementation does not use these capabilities for real-time updates on postings. With our novel implementation of the METIS subsystem, we introduce a websocket interface that is provided by the `Metis Service` on the client-side and used by `Post Service`, `Reaction Service`, and `AnswerPost Service` on the server-side as visualized in Figure 4.3.

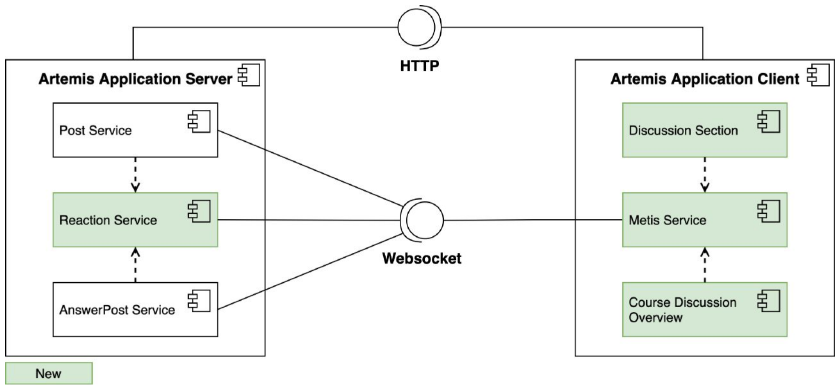


Figure 4.3: Client-server communication within the METIS subsystem

This enables users to subscribe to real-time updates on postings that are immediately reflected in the responsive UI of the `Page UI` components as aimed for in [NFR3.1] (see Section 3.3.2). Each time the client uses the HTTP interface provided by the server to perform some sort of *CRUD operation*[1], the respective service in the `Application Logic Layer` will, upon completion, emit a message via the provided websocket interface to notify all connected clients. The `Metis Service` on the client-side is subscribed to these messages and updates its state, i.e., the list of posts that it manages.

[1]Acronym describing the four basic operations of persistent storage, i.e., Create, Read, Update, and Delete.

The `Page UI` components that are subscribed to this list will then seamlessly update their views.

We instantiate the described communication process in Figure 4.4, to exemplify how the websocket interface is used to manage communication between *multiple* clients. A possible scenario of the instantiated system is given in Table 3.4: Alice, a *student*, and Donna, an *instructor*, act as clients that use the functionality provided by the METIS subsystem. Whenever one of the clients uses the HTTP interface the server-side service that handles the request will propagate the changes via the websocket interface to the `Metis Service` on *both* clients at the same time.

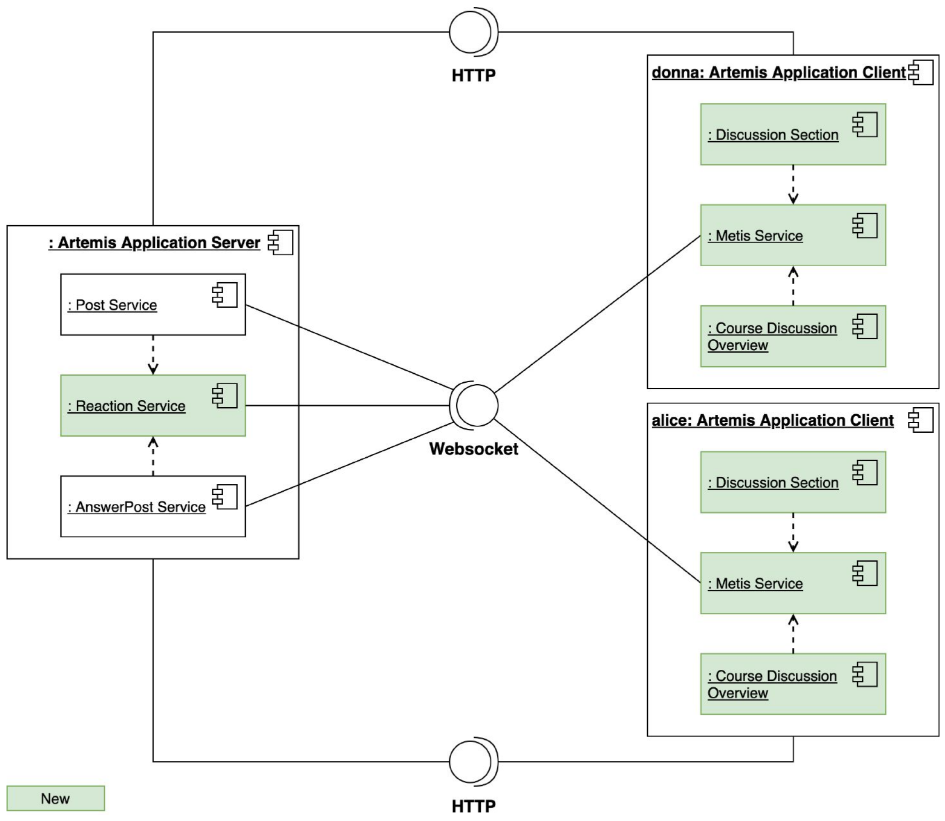

HTTP
donna: Artemis Application Client
: Discussion Section
: Metis Service
: Course Discussion Overview
: Artemis Application Server
: Post Service
: Reaction Service
: AnswerPost Service
Websocket
alice: Artemis Application Client
: Discussion Section
: Metis Service
: Course Discussion Overview
New
HTTP

Figure 4.4: Instantiation of client-server communication

4.4 Persistent Data Management

Within the development of the METIS subsystem, we introduced a new entity, `Reaction`. Next, we shortly describe how the instances of the entity are persisted in the existing relational `MySQL` database.

The entity and its relations to other classes have been introduced in Section 3.4.3 in the class diagram shown in Figure 3.6. As depicted in Figure 4.5, the properties of the `Reaction` entity are persisted in the columns `id`, `creation_date`, and `emoji_id` of the newly added database table `reaction`. The foreign keys that exist for the `reaction` table refer to *exactly* one `user` by its id and at most one `post` or `answer_post` by its id. However, a reaction must be associated to either one of them and cannot exist decoupled from any posting. This constraint is not integrated in the underlying database model, but we rather ensure this by adding server-side validation. With regard to the chosen column types, we stick to the pre-existing convention to use `BIGINT` for identifiers, and `DATETIME` for timestamps. The `emoji_id` column was initially restricted to 50 characters but later changed to `VARCHAR(255)` as the descriptive labels used as emoji identifiers in some cases exceed 50 characters.

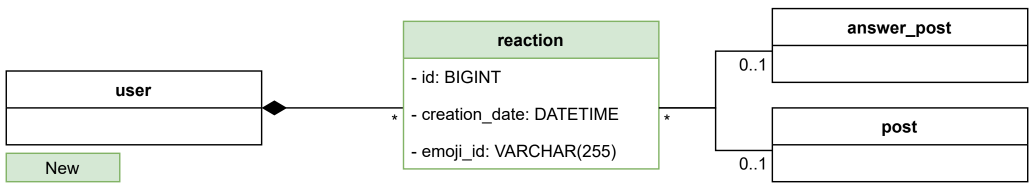


Figure 4.5: Persistent data management for *Reaction* entity as UML class diagram

Chapter 5

Object Design

As described in Chapter 4, integrating existing as well as new communication-related components into the proposed METIS subsystem leads to changes in the object design. This chapter explains essential implementation aspects on the object design level of two FRs, namely FR3.1 and FR4.1. More specifically, we illustrate how we design our post de-duplication implementation to be easily extensible and interchangeable, and re-design the existing answer approving process to adequately meet students' and moderators' needs of resolving posts.

5.1 Reducing Content Redundancy

As stated in Section 1.2, one of the goals of this thesis is to decrease post duplication that will in turn allow more efficient and effective information retrieval. [FR3.1] captures the requirement for a post similarity check which happens during the post creation.

We approach this requirement in an iterative process: Instead of immediately investing a lot of development time in integrating a sophisticated algorithm to compare the similarity of posts, we use a straightforward baseline implementation to (1) provide a proof-of-concept and (2) collect user feedback early on.

To facilitate later addition of and easy replacement by more sophisticated algorithms, we make use of the strategy pattern [GHJV95]. The UML class diagram in Figure 5.1 depicts how the pattern is implemented. As intended by the strategy pattern, we provide an interface that defines a family of algorithms, the `PostSimilarityStrategy`. This way, the specific algorithm can vary independently from the `PostService` that uses it to perform a similarity check upon user request.

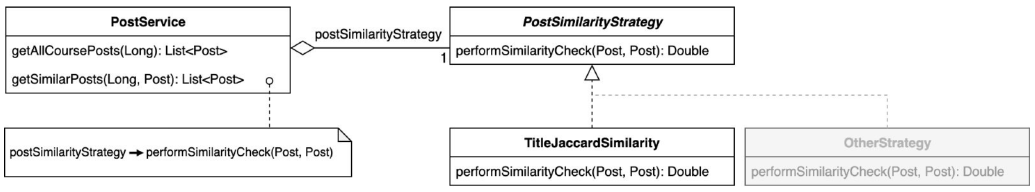
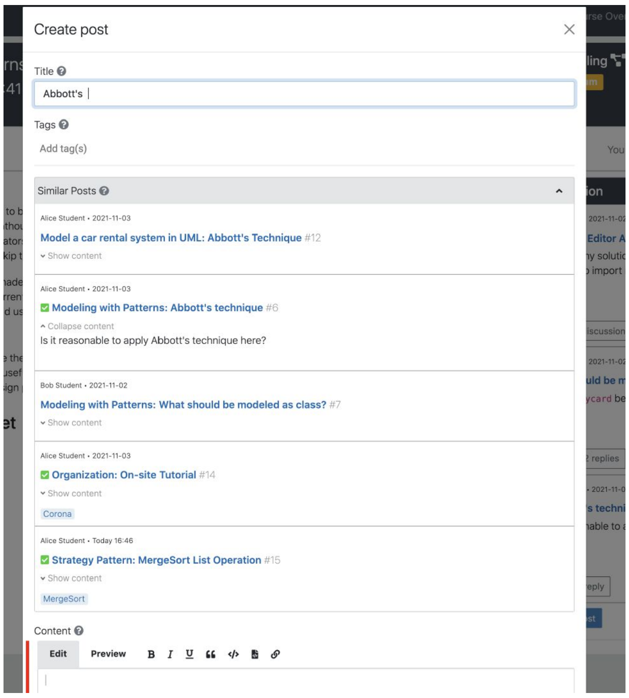

PostService
getAllCoursePosts(Long): List<Post>
getSimilarPosts(Long, Post): List<Post>
postSimilarityStrategy
1
PostSimilarityStrategy
performSimilarityCheck(Post, Post): Double
postSimilarityStrategy → performSimilarityCheck(Post, Post)
TitleJaccardSimilarity
performSimilarityCheck(Post, Post): Double
OtherStrategy
performSimilarityCheck(Post, Post): Double

Figure 5.1: Strategy pattern for determining post similarity

The first simple implementation of a `PostSimilarityStrategy` is the `TitleJaccardSimilarity`. The employed algorithm is based on the Jaccard Index[1]. When invoking the method to perform a similarity check for the post that a user wants to create, the `TitleJaccardSimilarity` strategy compares the extracted post title with the titles of all existing posts in the course. The computed similarity score takes into account how many words the compared titles have in common and ranges from 0 to 1, where 1 means exact title equality. The integration of more elaborate strategies is discussed in Section 6.2.3.



Create post

Title

Abbott's

Tags

Add tag(s)

Similar Posts

Alice Student • 2021-11-03
Model a car rental system in UML: Abbott's Technique #12
Show content

Alice Student • 2021-11-03
✓ Modeling with Patterns: Abbott's technique #6
Collapse content
Is it reasonable to apply Abbott's technique here?

Bob Student • 2021-11-02
Modeling with Patterns: What should be modeled as class? #7
Show content

Alice Student • 2021-11-03
✓ Organization: On-site Tutorial #14
Show content
Corona

Alice Student • Today 16:46
✓ Strategy Pattern: MergeSort List Operation #15
Show content
MergeSort

Content

Figure 5.2: Duplication check during post creation on Artemis 5.3.0

Besides the applied strategy, it is to discuss how to integrate the feature

[1]`https://commons.apache.org/proper/commons-text/apidocs/org/apache/commons/text/similarity/JaccardSimilarity.html`, Accessed: 2021-10-25

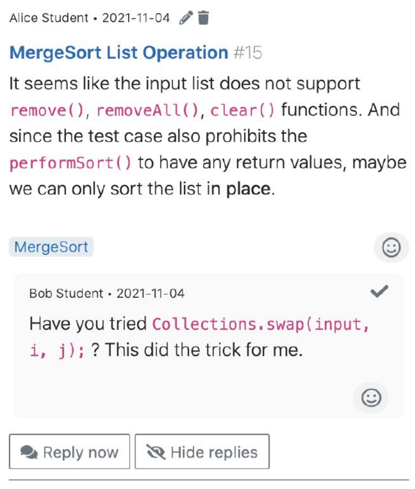
from a client-side perspective with good usability. Figure 5.2 captures the current approach: We add an expandable list including a fixed amount of similar posts right above the post content input field. This list appears when a user provides a title and can be browsed, whereby the content of a post is collapsed by default. The interaction with that list is illustrated in detail in Figure 3.7. However, the current approach is not yet a significant barrier to posting duplicate content, as it relies heavily on a user's cooperation. One could investigate, if the process of creating a post should be adapted in a way that makes it harder for users to add a post that probably contains a duplicated question.

5.2 Reducing Text-based Moderation Effort

As outlined in Section 1.2, we aim for efficient and effective discussion moderation by integrating features that tackle the high and text-based moderation effort. In the following, we describe parts of the implementation of the associated requirement [FR4.1].

The process of resolving a post is presented in Figure 5.3. It replaces the approval mechanism that did take into account if a moderator *approves* the theoretical correctness, but did not take into account, if the provided answer actually *resolves* the post author's problem. In the new process, the post author and moderators are able to mark answer posts as resolving. In the analysis model (see Section 3.4.3), we describe that answer posts can be marked as resolving to indicate that the post's matter is solved by that answer. We decided against persisting this post state, i.e., resolved or not resolved, in the database. Since a user should also be able to identify the answer that solved the problem, possibly among many given answers, we instead persist this information as additional property of the answer post entity. Hence, we rather evaluate the post state by checking the associated answer posts for this property. When re-designing the existing code, we replaced the property `approved` by `resolvesPost` in the `AnswerPost` class and renamed the existing column in the database table.

In the example from Figure 5.3, Alice, as the author of a post, is provided with a checkmark icon at the top right corner of every answer post (see Figure 5.3a). By clicking it, she confirms the correctness of the answer provided by Bob (see Figure 5.3b). This sets the Boolean property `resolvesPost` of Bob's answer post to `true`. The subsequent real-time update of the discussion component will also re-evaluate if any of the post's answer posts was marked as resolving. Since this is now indeed the case, Alice's post includes a green box which precedes the post title (see Figure 5.3c). If the answers of a post are collapsed, a user would immediately see if the post's issue is resolved or not.



Alice Student • 2021-11-04

MergeSort List Operation #15

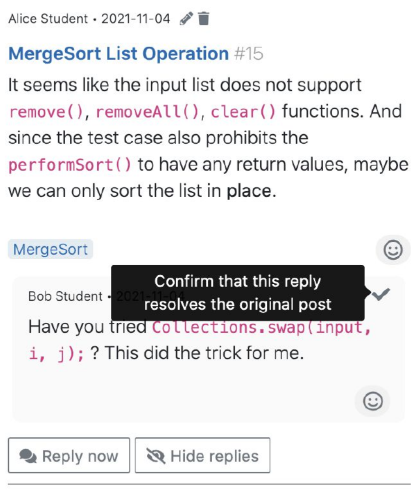
It seems like the input list does not support `remove()`, `removeAll()`, `clear()` functions. And since the test case also prohibits the `performSort()` to have any return values, maybe we can only sort the list in **place**.

MergeSort

Bob Student • 2021-11-04

Have you tried `Collections.swap(input, i, j);` ? This did the trick for me.

(a) Answered post



Alice Student • 2021-11-04

MergeSort List Operation #15

It seems like the input list does not support `remove()`, `removeAll()`, `clear()` functions. And since the test case also prohibits the `performSort()` to have any return values, maybe we can only sort the list in **place**.

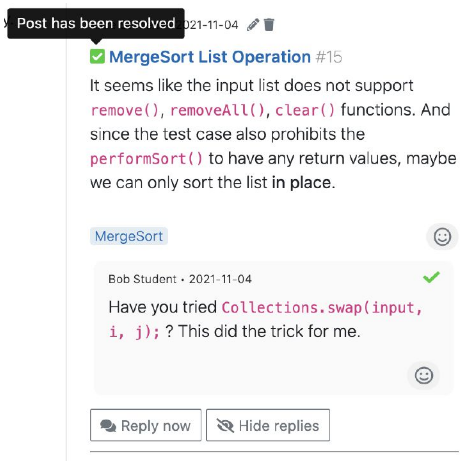
MergeSort

Bob Student •

Confirm that this reply resolves the original post

Have you tried `Collections.swap(input, i, j);` ? This did the trick for me.

(b) Marking answer post as resolving



Post has been resolved

021-11-04

MergeSort List Operation #15

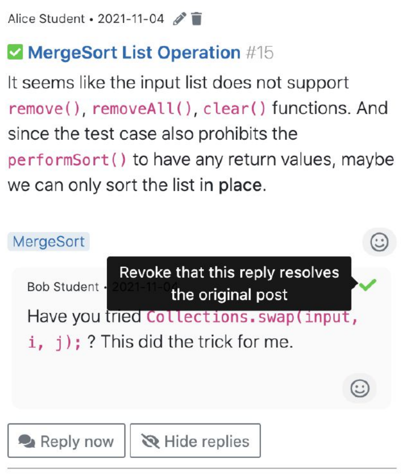
It seems like the input list does not support `remove()`, `removeAll()`, `clear()` functions. And since the test case also prohibits the `performSort()` to have any return values, maybe we can only sort the list in **place**.

MergeSort

Bob Student • 2021-11-04

Have you tried `Collections.swap(input, i, j);` ? This did the trick for me.

(c) Resolved Post



Alice Student • 2021-11-04

MergeSort List Operation #15

It seems like the input list does not support `remove()`, `removeAll()`, `clear()` functions. And since the test case also prohibits the `performSort()` to have any return values, maybe we can only sort the list in **place**.

MergeSort

Bob Student •

Revoke that this reply resolves the original post

Have you tried `Collections.swap(input, i, j);` ? This did the trick for me.

(d) Revoking the resolution

Figure 5.3: Process of resolving a post on Artemis 5.3.0

Notably, the implementation currently allows that Alice or a moderator could revoke this action (see Figure 5.3d). A possible future extension could be to only allow revoking, if either a reasonable explanation for this is given by the user or a different answer post is marked as resolving.

Chapter 6

Discussion

As part of this chapter, we first discuss how the communication capabilities introduced and enhanced by this thesis can be evaluated during real-world operation of Artemis. Second, we reflect on design and implementation decisions that were taken throughout the course of this thesis and their implied limitations.

6.1 Future Evaluation

The implementation period spanned over six months, starting from May 2021. Hence, for the first time, in winter semester 2021/2022, 9 courses with more than 1,700 participating students are actively using the new communication features. To measure how the introduced changes affect the engagement of students and quality of information sharing as well as moderation effort, an empirical study has to be conducted by the end of the semester. The following quantitative evaluation metrics can easily be retrieved through database queries:

- Number of courses that previously relied on additional communication platforms, but switched to using Artemis as all-in-one solution for course management
- Number of postings
- Number of comments on postings indicating that the post is a duplicate
- Number of postings with course-wide topics
- Distribution of emojis used to react on postings

Qualitative evaluation aspects can be gathered through interviews and questionnaires:

- Do moderators use the new course discussion overview to manage discussions and determine where their input is required?
- Are students able to easily and quickly find the information they are looking for?
- Do moderators save time by using features such as resolving, moving, tagging, filtering posts?

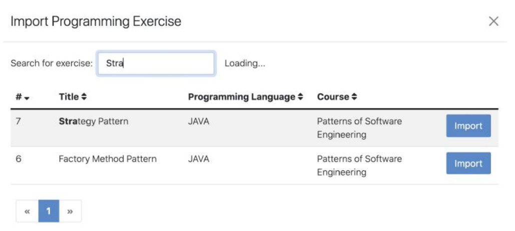
By analyzing the proposed quantitative and qualitative evaluation aspects, Artemis' existing communication capabilities can be further improved and development efforts steered towards yet more sophisticated communication features.

6.2 Design and Implementation Reflection

Throughout the development of this thesis, several design and implementation decisions had to be taken. Below, we critically reflect on three decisions and their implied limitations.

6.2.1 Search Bar

In Section 3.3.2, we derived consistency as a NFR to foster usability of METIS. Yet, in Artemis' existing UI components, we find different variants of search bars—for instance, in the UI to import an exercise (see Figure 6.1a) or when searching for a user (see Figure 6.1b). However, to enable convenient querying and filtering of posts in a course, we require an even more elaborate search bar component, as depicted in Figure 6.2. Since this introduces the third variant of a search bar component, the usability paradigm of consist design is even more violated than before. Hence, ultimately there should only be one search bar component that can be parameterized and adjusted depending on the complexity of the use case of the search bar. Based on the readily existing search bars, a unified search bar component should include (1) similar buttons and labels, (2) similar means to actually trigger the search (i.e., clicking a button versus hitting the enter key versus no active triggering required), (3) optionally visible filter options, and (4) a consistent search results list view, irrespective of the listed entities.

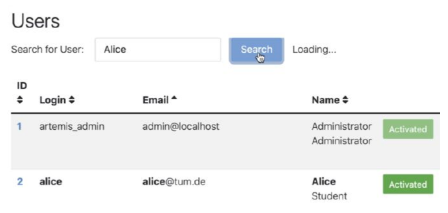

Import Programming Exercise

Search for exercise: Stra Loading...

#	Title	Programming Language	Course	
7	Strategy Pattern	JAVA	Patterns of Software Engineering	Import
6	Factory Method Pattern	JAVA	Patterns of Software Engineering	Import

« 1 »

(a) Exercise search

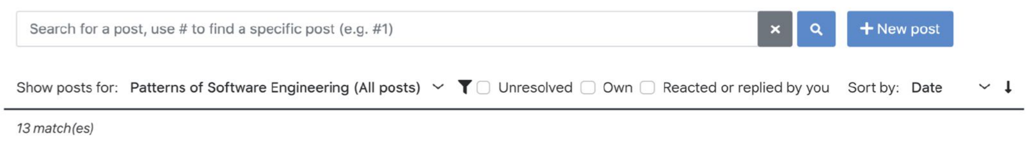

Users

Search for User: Alice Search Loading...

ID	Login	Email	Name	
1	artemis_admin	admin@localhost	Administrator Administrator	Activated
2	alice	alice@tum.de	Alice Student	Activated

(b) User search

Figure 6.1: Different implementations of search bars on Artemis 5.3.0


Search for a post, use # to find a specific post (e.g. #1)

+ New post

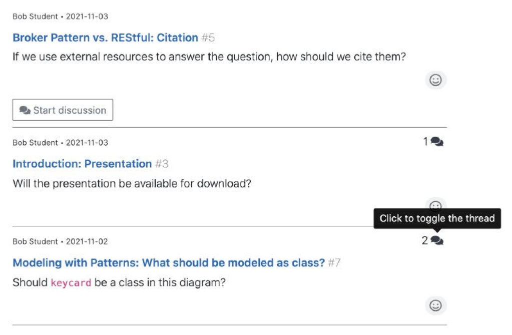
Show posts for: Patterns of Software Engineering (All posts) Unresolved Own Reacted or replied by you Sort by: Date

13 match(es)

Figure 6.2: Search bar at the course discussion overview on Artemis 5.3.0

6.2.2 Browsing Discussions

As described in Section 3.4.3, answer posts cannot exist independently from posts. A post therefore states the beginning of a discussion that can have several answer posts. When designing the UI for browsing these discussions in a certain context, i.e., on a lecture, exercise, or course discussion overview, the requirements are that a user should (1) immediately see how many answers exist for each post, (2) read the existing answers before adding a new one, and (3) be able to switch between collapsed and expanded answers.


Bob Student • 2021-11-03
Broker Pattern vs. RESTful: Citation #5
If we use external resources to answer the question, how should we cite them?

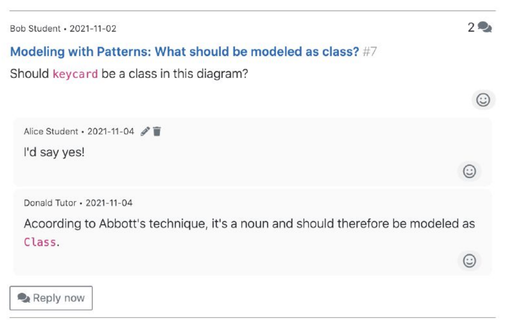
Start discussion

Bob Student • 2021-11-03 1
Introduction: Presentation #3
Will the presentation be available for download?

Click to toggle the thread

Bob Student • 2021-11-02 2
Modeling with Patterns: What should be modeled as class? #7
Should `keycard` be a class in this diagram?

(a) List of posts with collapsed answers


Bob Student • 2021-11-02 2
Modeling with Patterns: What should be modeled as class? #7
Should `keycard` be a class in this diagram?

Alice Student • 2021-11-04
I'd say yes!

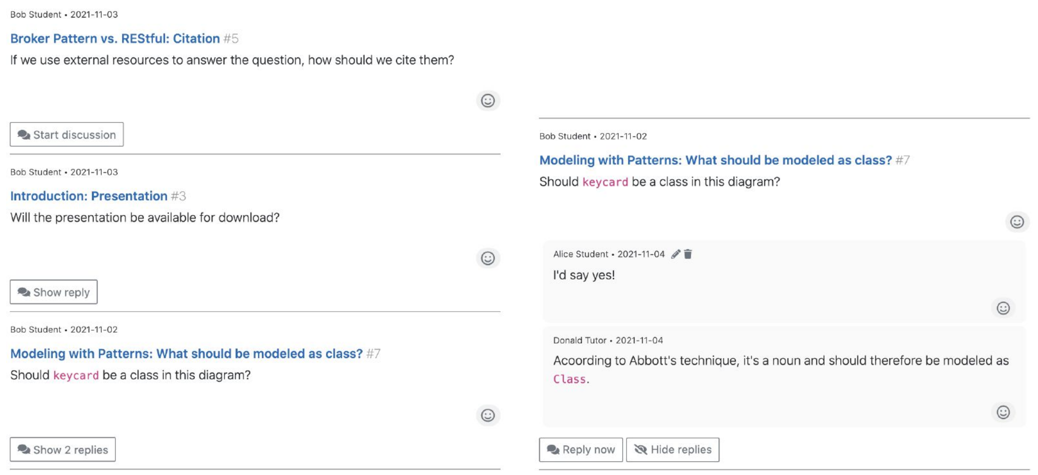
Donald Tutor • 2021-11-04
According to Abbott's technique, it's a noun and should therefore be modeled as `Class`.

Reply now

(b) Post with expanded answers

Figure 6.3: List of posts with collapsed and expanded answers (intermediate version)

Figure 6.3 shows an intermediate whereas Figure 6.4 shows a revised, new version of the UI that we implemented throughout the course of this


Bob Student • 2021-11-03
Broker Pattern vs. RESTful: Citation #5
If we use external resources to answer the question, how should we cite them?
Start discussion

Bob Student • 2021-11-03
Introduction: Presentation #3
Will the presentation be available for download?
Show reply

Bob Student • 2021-11-02
Modeling with Patterns: What should be modeled as class? #7
Should `keycard` be a class in this diagram?
Show 2 replies

Bob Student • 2021-11-02
Modeling with Patterns: What should be modeled as class? #7
Should `keycard` be a class in this diagram?

Alice Student • 2021-11-04
I'd say yes!

Donald Tutor • 2021-11-04
According to Abbott's technique, it's a noun and should therefore be modeled as `Class`.

Reply now
Hide replies

(a) List of posts with collapsed answers **(b)** Post with expanded answers

Figure 6.4: List of posts with collapsed and expanded answers on Artemis 5.3.0

work. Both versions depict the same list of posts (i.e., discussions), having zero, one, and two answers. In the intermediate version (see Figure 6.3a and Figure 6.3b), the button to start a discussion is only visible when there are no answers yet. If there are answers to a post, the user has to click on the icon at the top right to collapse and expand the answers. The number next to the icon represents the number of answers. For this version, we received the feedback that the design is misleading as users concluded that there were no answers when not having a button displayed below the content. Presumably, the icon's function was not intuitively clear to the users, as they expected a button for adding an answer and to expand or collapse existing answers.

Therefore, we implemented a new version (see Figure 6.4a and Figure 6.4b) that exactly reflects these expectations. We received the feedback that this new implementation is more consistent with other parts of Artemis and thus more intuitive to use.

6.2.3 NLP for Post De-Duplication

To compare posts regarding their similarity and perform post de-duplication, prior research conducted at Stack Overflow suggests to use more elaborate strategies that rely on algorithms from the field of Natural Language Processing (NLP) [ZLXS15,SPD18]. However, while these algorithms have been shown to work well in these contexts, there are challenges and trade-offs: First, as Artemis supports English and German language, classic NLP pre-processing steps such as stop word removal or stemming require language-specific implementations [KG14]. Second, there is a trade-off between how fast an algorithm should perform the similarity check and how accurate it should be. More precisely, one has to decide if it is more important to users to be provided with identified similar posts immediately or that the provided list of suggested similar posts is highly accurate with few so-called false positives.

The current implementation described in Section 5.1 prioritizes efficiency over accuracy of results, which keeps required computational resources and response time at a minimum and avoids language-specific implementation efforts.

Chapter 7

Summary

To summarize, we report on the status of derived requirements, outline visionary ideas for further enhancing and making use Artemis' communication features introduced by this thesis, and finally conclude the contribution of this thesis.

7.1 Status of Requirements

We outline the state of the requirements elicited in Chapter 3 in Table 7.1. Therefore, we group requirements identified in Section 3.3.1 by the identified problems that the thesis aims to solve and the objectives that were described in Chapter 1. This way, the status of the requirements also indicates which of the identified problems is solved and which objective is met to which degree. The status symbols can be interpreted as follows:

● Requirement fully addressed by new feature(s)

◐ Requirement partially addressed by new feature(s)

○ Requirement not yet addressed

FRs that are not yet addressed should be regarded as open goals. In the field of post discoverability this includes the bookmarking of posts ([FR2.16]). As single posts are already referable by a unique Uniform Resource Locator (URL) including the post identifier, the bookmarking feature can be built on top of that. Moreover, we integrated post navigation but do not support navigation to specific answer posts yet, which makes [FR2.16] only partially met. Same holds true for [FR2.16] where a user can only customize the sorting of posts but not answer posts.

Problem	**Requirement**	**Objective**	**Status**
[P_1] Insufficient topic organization lacking course-wide topics	[FR1.1] Post content of course-wide relevance	[O_1]	●
	[FR1.2] Choose between predefined course-wide topics for a course post	[O_1]	●
[P_2] Insufficient visibility and discoverability of posts	[FR2.1] Summarize a post by title	[O_1]	●
	[FR2.2] Tag a post	[O_1]	●
	[FR2.3] Provide a course discussion overview listing all posts	[O_1]	●
	[FR2.4] Navigate to a posting	[O_1]	◐
	[FR2.5] Search a post by text	[O_1]	●
	[FR2.6] Search a post within a certain context	[O_1]	●
	[FR2.7] Filter posts by context	[O_1]	●
	[FR2.8] Filter posts by answer state	[O_2]	●
	[FR2.9] Filter posts to own posts only	[O_1]	●
	[FR2.10] Filter posts to posts that the user answered or reacted on	[O_1]	●
	[FR2.11] Sort postings by date	[O_1]	◐
	[FR2.12] Sort posts by votes	[O_1]	●
	[FR2.13] Sort posts by answers	[O_1]	●
	[FR2.14] Pin a post	[O_1]	●
	[FR2.15] Archive a post	[O_2]	●
	[FR2.16] Bookmark a post	[O_1]	○
[P_3] Content redundancy	[FR3.1] Conduct a similarity check during post creation	[O_2]	●
	[FR3.2] Merge duplicated posts	[O_2]	○
	[FR3.3] Provide a FAQ section	[O_1]	○
	[FR3.4] Write a post anonymously	[O_2]	○
	[FR3.5] Import postings of previous courses	[O_2]	○
[P_4] High, text-based moderation efforts	[FR4.1] Mark a post as resolved	[O_2]	●
	[FR4.2] Change the context of a post	[O_2]	●
	[FR4.3] Enable internal communication for moderators	[O_2]	○
	[FR4.4] Report a posting	[O_2]	○
[P_5] Low interaction and emotional involvement	[FR5.1] Use emojis within a posting	[O_3]	◐
	[FR5.2] React on a posting with emojis	[O_3]	●
	[FR5.3] Use emojis to up-vote a post	[O_3]	●
	[FR5.4] Reference a user	[O_3]	○
	[FR5.5] Reference a posting	[O_3]	◐
	[FR5.6] Reference a course entity	[O_3]	○
	[FR5.7] Reward a user for discussion contributions	[O_3]	○
	[FR5.8] Notify a user on discussion events outside Artemis	[O_3]	●

Table 7.1: Summary on FRs and their status

With regard to post de-duplication, the current implementation misses a dedicated FAQ section ([FR3.3]), functionality to import postings ([FR3.5]), as well as a features to merge duplicated posts ([FR3.2]) or contribute content anonymously ([FR3.4]).

Additionally, one could further facilitate discussion moderation by providing the means for *internal* communication among moderators that is not meant to be disclosed to students ([FR4.3]). Adding a course-wide topic that is only available and visible for authorized users would enable this type of conversation. Another open goal regarding improved moderation is that Artemis could follow the example of online class platforms and allow users to report posts ([FR4.4]). This could in turn contribute to a discussion space where users adhere to established course rules or a code of conduct.

Regarding the FRs that address the problem of low interaction capabilities, referencing users ([FR5.4]) remains an open goal. So does the feature for referencing other course entities ([FR5.6]) such as lectures, exercises, or, even more fine grained, certain lecture slides, sequences in lecture videos, or tasks in problem statements of exercises. Besides, expanding the use of emojis as part of a posting's content ([FR5.1]) is not yet fully addressed, but the newly integrated component for selecting emojis as post reaction could be extended for use inside post content. We further elaborate on the open goal of rewarding users in the following Section 7.2.

The requirement of notifying users on discussion events outside Artemis ([FR5.8]) was achieved in cooperation with another Artemis developer, who specifically enhanced Artemis' notification enhancements [Mal21]. Our joint contributions allow that students are notified on new posts, answer posts to own posts, and announcements via automatically generated emails containing a preview of the post as well as a link.

7.2 Future Work

There are several aspects that we perceive as worthwhile starting points for future work.

First, future work could enhance existing features such as adding more complex and elaborate post comparison strategies to avoid duplication as described in Section 5.1. To increase the accuracy with which duplicated questions are identified and presented to users that are in the process of creating a new post, one could integrate a strategy that employs machine learning algorithms. Those could rely on topic modeling instead of searching for exact text similarity. Such an approach is used on Stack Overflow, where a sophisticated algorithm called `DupPredictor`[1] was developed. The research on duplication prevention on Stack Overflow provides meaningful insights on which attributes of posts and which machine learning algorithms work best [SPD18]. Advancing the integrated duplication check is crucial to avoid that users ignore the proposed list of similar posts as the results to not match their expectations.

Second, the new communication features pave the way for many further features and advances. Some of them are outlined in the following. Future research could investigate to which degree sentiment analysis on the posted content can be harnessed for course evaluation purposes. When enriching such data with the analysis of reactions used, this might be used to evaluate a course more accurately than asking for feedback in pre-defined questionnaires that suffer participation bias and have an inherent delay between the actual learning experience and reporting about it. An associated approach is to analyze courses using social network analysis techniques [RTZ11]. Mining the course discussions with regard to topics and interactions can generate valuable insights regarding which topics stimulate students' involvement. Additionally, research on peer interaction in discussion forums suggests to incentivize and reward participation in discussions by introducing a reputation system – similar to Stack Overflow – that can help minimizing moderation efforts, and increase involvement and motivation [HS14, ZIFK17]. Statistics over discussion participation could also be used for grading. The final grade would then not only be based on a snapshot performance during an exam, but also on communicative skills. Finally, future work could also investigate if machine learning could be applied to automatically assign moderators to posts based on their expertise or to detect urgency of posts [AJ19], which could ultimately improve moderation and learning experience for students.

Last, the METIS subsystem introduced in this thesis has been designed to be largely decoupled of other Artemis parts on both, client- and server-side. Hence, METIS is well-suited for a potential migration into a more distributed, microservice-oriented architecture. In fact, METIS could easily be extracted from the Artemis monolith and deployed as a standalone Web service with corresponding micro-frontend.

7.3 Conclusion

Artemis is an interactive learning platform that is becoming a powerful CMS in the university context. However, Artemis currently lacks features that enable elaborate discussions and course communication, making instructors

[1]`https://github.com/muldon/dupPredictorRep`, Accessed: 2021-11-08

hesitant to rely on Artemis as the sole communication platform for large courses. In this thesis, we propose Multiplying Engagement Through Interacting Socially (METIS), a novel Artemis subsystem addressing the existing limitations. We thereby contribute to efficient and effective information sharing as well as to more engaged and interactive discussions in a variety of ways: The problem of insufficient topic organization and the lack of course-wide topics is well-addressed by the integration of the course discussion overview, which allows to post course-wide content under specified topics. Additionally, we develop various features that improve the restricted discoverability and visibility of posts on Artemis. A straightforward approach for de-duplicating posts by employing automatic similarity checks will decrease the content redundancy. Moreover, we reduce the previously high, text-based moderation efforts by useful features such as resolving a post or moving it to another context. Lastly, engaging features such as an emoji reaction bar, in-app and email notifications, and the ability to reference other posts aim to increase user interaction and emotional involvement. Thus, METIS motivates intra-platform user interaction to increase students' engagement and ultimately learning success. Our new communication features are currently actively used by 1,700 students in 9 university courses conducted with Artemis. We consider the implementation of the METIS subsystem including the introduced features as an important step towards Artemis being used by instructors as the all-in-one solution for their course management.

List of Figures

List of Tables

Bibliography

[Ada13] Panagiotis Adamopoulos. What makes a great MOOC? An interdisciplinary analysis of student retention in online courses. In *Proceedings of the 43th International Conference on Information Systems*, 2013.

[AJ19] Omaima Almatrafi and Aditya Johri. Systematic Review of Discussion Forums in Massive Open Online Courses (MOOCs). *IEEE Transactions on Learning Technologies*, 12(3):413–428, 2019.

[AVWO20] Lauren Adolphe, Georgia D Van de Zande, David Wallace, and Alison Olechoswski. Analysis of Virtual Communication within Engineering Design Teams and its Impact on Team Effectiveness. In *Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 8, 2020.

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

[CS18] Alicia Cundell and Emily Sheepy. Student Perceptions of the Most Effective and Engaging Online Learning Activities in a Blended Graduate Seminar. *Online Learning Journal*, 22(3):87–102, 2018.

[DFCV15] Damiano Distante, Alejandro Fernandez, Luigi Cerulo, and Aaron Visaggio. Enhancing Online Discussion Forums with Topic-Driven Content Search and Assisted Posting. *Communications in Computer and Information Science*, 553:161–180, 2015.

[Dix10] Marcia D Dixson. Creating effective student engagement in online courses: What do students find engaging? *Journal of the Scholarship of Teaching and Learning*, 10(2):1–13, 2010.

[DRK20] Urvashi Desai, Vijayalakshmi Ramasamy, and James D. Kiper. A Study on Student Performance Evaluation using Discussion Board Networks. In *Annual Conference on Innovation and Technology in Computer Science Education*, pages 500–506, 2020.

[FSP14] Carrie Furrer, Ellen Skinner, and Jennifer Pitzer. The Influence of Teacher and Peer Relationships on Students' Classroom Engagement and Everyday Resilience. In *Engaging youth in schools: Empirically-based models to guide future innovations, National Society for the Study of Education Yearbook*, pages 101–123. Teachers College Record, 2014.

[GHG$^+$20] Shay A. Geller, Nicholas Hoernle, Kobi Gal, Avi Segal, Amy X. Zhang, David Karger, Marc T. Facciotti, and Michele Igo. #Confused and beyond: Detecting confusion in course forums using students' hashtags. In *Proceedings of the 10th International Conference on Learning Analytics & Knowledge*, pages 589–594, 2020.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1st edition, 1995.

[Gre20] Filip Gregurevic. *Improving Questions and Answers in Artemis.* Master's thesis, Technical University of Munich, 2020.

[Hew18] Khe Foon Hew. Unpacking the Strategies of Ten Highly Rated MOOCs: Implications for Engaging Students in Large Online Courses. *Teachers College Record*, 120(1):1–40, 2018.

[Hon20] Nicholas Yeung Ming Hon. The Game of Communication: An analysis of The Emoji Landscape in Social Media. *Journal of Physics: Conference Series*, 1684(1), 2020.

[HS14] Kerry Hart and Anita Sarma. Perceptions of Answer Quality in an Online Technical Question and Answer Forum. In *Proceedings of the 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 103–106, 2014.

[JBR99] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process.* Addison-Wesley, Reading, MA, 1999.

[KG14] Subbu Kannan and Vairaprakash Gurusamy. Preprocessing Techniques for Text Mining. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2014.

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

[Luc20] Naemi Luckner. *Enabling Peer Review in Large University Courses.* Dissertation, TU Wien, 2020.

[Mal21] Alexander Malyuk. *Run Time Notifications in Dynamically Changing Systems.* Master's thesis, Technical University of Munich, 2021.

[Mar14] Robert Cecil Martin. *Agile Software Development: Principles, Patterns, and Practices.* Pearson Education, 1st edition, 2014.

[Mee03] John Meerts. Course Management Systems (CMS). Technical report, Wesleyan University, 2003.

[Mei19] Maximilian Meier. *Improvement of the usability of the course structure in ArTEMiS.* Master's thesis, Technical University of Munich, 2019.

[MMM$^+$11] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design Lessons from the Fastest Q&A Site in the West. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 2857–2866, 2011.

[PD16] Oleksandra Poquet and Shane Dawson. Untangling MOOC Learner Networks. In *Proceedings of the 6th International Conference on Learning Analytics & Knowledge*, pages 208–212, 2016.

[RAB$^+$18] Petrea Redmond, Lindy Anne Abawi, Alice Brown, Robyn Henderson, and Amanda Heffernan. An online engagement framework for higher education. *Online Learning Journal*, 22(1):183–204, 2018.

[RTZ11] K. Reihaneh Rabbany, Mansoureh Takaffoli, and Osmar R. Zäiane. Analyzing Participation of Students in Online Courses Using Social Network Analysis Techniques. In *Proceedings of the 4th International Conference on Educational Data Mining*, pages 21–30, 2011.

[SPD18] Rodrigo F.G. Silva, Klérisson Paixão, and Marcelo De Almeida Maia. Duplicate Question Detection in Stack Overflow: A Reproducibility Study. In *Proceedings of the 25th International Conference on Software Analysis, Evolution and Reengineering*, pages 572–581, 2018.

[WRGW14] Joe Warren, Scott Rixner, John Greiner, and Stephen Wong. Facilitating human interaction in an online programming course. In *Proceedings of the 45th Technical Symposium on Computer Science Education*, pages 665–670, 2014.

[WW07] William R. Watson and Sunnie Lee Watson. An argument for clarity: What are learning management systems, what are they not, and what should they become? *TechTrends*, 51(2):28–34, 2007.

[ZIFK17] Amy X. Zhang, Michele Igo, Marc Facciotti, and David Karger. Using student annotated hashtags and emojis to collect nuanced affective states. In *Proceedings of the 4th Conference on Learning at Scale*, pages 319–322, 2017.

[ZLXS15] Yun Zhang, David Lo, Xin Xia, and Jian Ling Sun. Multi-Factor Duplicate Question Detection in Stack Overflow. *Journal of Computer Science and Technology*, 30(5):981–997, 2015.