eleed
e-learning
& education

Issue se2023
2023

Design Principals for Building a Collaborative Exchange Platform for Auto-gradable Programming Exercises

Sebastian Serth
sebastian.serth@hpi.de

Karen von Schmieden
karen.schmieden@hpi.de

Mohamed Elhayany
mohamed.elhayany@hpi.de

Zuhra Sofyan
zuhra.sofyan@hpi.de

Christoph Meinel
christoph.meinel@hpi.de

Hasso Plattner Institute, University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

urn:nbn:de:0009-5-57791

Zusammenfassung

Das Aufkommen freier offener Onlinekurse, sogenannter Massive Open Online Courses (MOOCs), hat den Zugang zur Informatik erleichtert, indem Kurse zu verschiedenen Programmiersprachen und -technologien für jedermann über eine Internetverbindung zur Verfügung gestellt werden. Diese MOOCs haben gezeigt, dass die Bereitstellung notwendiger Tools für selbstreguliertes Lernen und formatives Feedback für Lernende von entscheidender Bedeutung für erfolgreiche Lernergebnisse ist. Ähnlich wie MOOC-Lernende profitieren auch Studierende an Universitäten sowie Schülerinnen und Schüler von interaktiven Übungen mit automatisiertem Feedback. Allerdings fehlt es Hochschullehrenden und vor allem Gymnasiallehrkräften oft an geeigneten Programmieraufgaben und den notwendigen Tools. Deshalb haben wir mit der Arbeit an CodeHarbor begonnen, wodurch Lehrkräften ein innovatives Tool zur Verfügung steht, mit dem sie automatisch-bewertbare Programmieraufgaben mit ihren Kolleg:innen teilen, bewerten und diskutieren können. In diesem Artikel beschreiben wir den Einsatz von Interviews und thematischen Analysen, mit denen wir die Herausforderungen und Chancen von Informatiklehrkräften beim Austausch digitaler Lehrmaterialien für ihren Unterricht untersuchten. Als Ergebnis haben wir zwölf Anwendungsfälle und drei Designprinzipien definiert, die bei der Entwicklung einer Plattform für den Austausch von Lehrmaterialien im Fach Informatik berücksichtigt werden sollten. Mit Werkzeugen, die in erster Linie für Informatiklehrkräfte entwickelt wurden und ihnen kollaboratives Arbeiten ermöglichen, sehen wir CodeHarbor als eine innovative Plattform, um den Informatikunterricht interaktiver und interessanter zu gestalten.

Stichwörter: e-learning; Benutzer:innenerfahrung; Programmierung; Code-Repository; Gemeinsame Nutzung; Übungen; Auto-Grader; MOOCs

Abstract

The emergence of Massive Open Online Courses (MOOCs) has made computer science more accessible by offering courses on various programming languages and technologies to anyone with an internet connection. These MOOCs have demonstrated that providing learners with the necessary tools for self-regulated learning and formative feedback remains crucial to ensuring successful learning outcomes. Similar to MOOC learners, university and high-school students also benefit from interactive exercises and automated feedback. However, university lecturers and especially high-school teachers often lack access to suitable programming tasks and the necessary tools. Therefore, we started working on CodeHarbor, an innovative tool designed explicitly for teachers to share, rate, and discuss auto-gradable programming exercises with their colleagues. In this article, we describe the use of interviews and thematic analyses to examine the problem space and opportunity areas experienced by computer science educators who engage in exchanging digital teaching materials for their classes. As a result, we defined twelve user stories and three design principles that should be considered when developing a platform for exchanging computer science teaching materials. With collaborative authoring tools designed primarily for computer science teachers, we envision CodeHarbor to become an important tool to make computer science education more interactive and enjoyable.

Keywords: e-learning; User Experience; Programming; Code Repository; Sharing; Exercises; Auto-Grader; MOOC

1. Introduction

The emergence of Massive Open Online Courses (MOOCs) has made programming education more accessible than ever before, offering courses on a range of programming languages and technologies to anyone with an internet connection. However, simply providing access to these courses is not enough to guarantee successful learning outcomes. It is essential to provide students with the tools they need to support self-regulated learning and receive formative feedback [Mal+19; Elh+22]. To tackle these challenges in a MOOC context, we developed *CodeOcean* [Sta+16]. It is a web-based platform designed to provide hands-on programming exercises for novices, currently used to introduce Python, object-oriented programming in Java, Ruby, and R to interested learners on openHPI [Mei+22], the MOOC platform of the Hasso Plattner Institute. As a so-called auto-grader, *CodeOcean* provides students with a sandboxed environment to solve hands-on programming exercises and receive automated feedback on their progress in a practical and effective manner [Sta+15].

In search of suitable resources for use in their programming classes, some high-school teachers discovered the openHPI courses on Java and Python featuring *CodeOcean* and used these directly with their students [STM17; Ser+21]. After interviewing those teachers on the use of *CodeOcean*, we found that they valued the tool's focus on providing feedback, hints, and support to students [Ser+19; STM21]: This feedback not only helps students understand where they made a mistake but also provides guidance on how to improve their programming skills. *CodeOcean*'s ability to provide scalable assessment and interoperability was also appreciated by teachers in surveys, as it allowed them to manage and assess large numbers of students efficiently.

However, as the use of online education continues to expand, teachers raised the desire to edit the existing exercises found in the openHPI programming courses for their students. This workflow was not yet covered in CodeOcean, and teachers were not able to exchange exercises with colleagues or design them collaboratively. Moreover, many high-school teachers raised concerns about defining new exercises with automated feedback due to their unfamiliarity with it, preventing them from leveraging the full potential of auto-graders for their students. Our vision was to create a platform that can address these shortcomings and better support teachers in their online teaching journey. This led to the development of CodeHarbor.

CodeHarbor is designed as an innovative tool that allows teachers to easily share, rate, and discuss auto-gradable programming exercises with colleagues and peers. Unlike *CodeOcean*, which was primarily designed for use by students in MOOCs, *CodeHarbor* is aimed at teachers who are looking for an efficient and effective way to create and share exercises with their peers [Ser+21]. Successfully establishing such a new platform in education calls for user-centered research and design approaches, specifically targeting the needs of educators to support them with their teaching preparation. Therefore, in this paper, we focused on the following two main research questions:

RQ1.	What are the key challenges and potential opportunities experienced by high-school teachers who engage in exchanging digital teaching materials for Computer Science classes?
RQ2.	What design principles should be considered when developing a platform for exchanging (coding) teaching materials in the German education system?

2 Background and Related Work

Our work contributes to research in the area of computer science education, supporting teachers to prepare engaging course content with interactive programming exercises providing automated feedback. Those contributions are based on a set of initial goals we aimed for (see Section 2.1), experiences from existing exchange formats and platforms (Section 2.2) as well as on other research studies on platform designs outlined in Section 2.3.

2.1 Initial Goals

We started working on *CodeHarbor* in late 2015 with a clear vision to support Computer Science educators in facilitating their teaching activities. Since the very beginning, our work is based on four distinct goals and the previous experience we gained creating, offering, and delivering various programming MOOCs for beginners or onsite courses for undergraduate students. These four goals are presented in the following:

Advance Computer Science Education. In Germany, where Hasso Plattner Institute is located, the sixteen federal states are responsible for financing the public education system [GG49] and, for example, define a syllabus as a guideline for all teachers in schools. However, there are major differences regarding computer science education: Some federal

states already included mandatory lessons for all students, while other states never included it or switched back to optional courses recently [KS16]. Those political decisions have several consequences, such as a continuation of the gender gap at universities and jobs [Has+19], or a shortcoming of teaching materials [Rot+23]. Therefore, many teachers across the states need to prepare their own content. With *CodeHarbor*, we aim to provide educators with an exchange platform to discover and exchange materials with each other.

Support Teachers in Creating Auto-gradable Exercises. Among the resources used by high-school teachers for their teaching activities are the openHPI courses on Java and Python. Teachers stated that the feedback and the other peer-assistance features included in *CodeOcean* allowed them to shift their role from an instructor to an individual tutor, focusing on those students struggling the most [Ser+19]. However, many teachers also raised concerns about creating the necessary test cases [Ser+19], which are necessary to provide automated feedback to students, thus lowering the usefulness of the tool. Hence, another goal we have with *CodeHarbor* is to assist teachers in defining test cases, allowing their students to benefit from next-step hints, and to encourage self-regulated and problem-based learning.

Connect the ProFormA Community. While our approach originated from a MOOC context and slightly shifted towards assisting high-school teachers, the need of computer science educators to exchange auto-gradable programming exercises can also be addressed from a university background: Many study programs are aligned with the Computer Curricula jointly created by ACM and IEEE [CC220] and feature similarly structured content in introductory courses (i.e., CS1 and CS2 courses). Due to the larger number of students enrolled in a single university program compared to high-school classes, automatically assessing students' submissions is considered a key requirement [BKB22]. Having those tools for students, lecturers from different universities potentially using various auto-graders wished to be able to exchange tasks more easily without manually copying them. Therefore, in 2011, a group of researchers teamed up in the eCULT project to create *ProFormA*, an exchange format for programming exercises [Rei+19]. Their project focused on defining the XML-based standard but without a corresponding web-based exchange platform connecting the different users of the *ProFormA* standard. Therefore, we see *CodeHarbor* as a central repository for the *ProFormA* community, uniting their efforts.

Encouraging Vivid Discussions and Regular Contributions. In our opinion, building a vibrant community around programming exercises has two requirements that must be met: First, an initial set of users already sharing high-quality exercises needs to be present, making the platform attractive for new educators to join. Therefore, we want to include most exercises from the openHPI programming courses, already employed by some high-school teachers in their classes. Second, we want to promote *CodeHarbor* by encouraging educators to use the existing resources and share their own examples, also covering additional programming languages. By integrating with the German project entitled "digitale Vernetzungsinfrastruktur für die Bildung", more educators will be able to join the community on *CodeHarbor*, also making discussions more diverse and interesting. Along with new content either shared by users or by us, we want to keep the platform attractive and support long-term use.

With those four goals defined, we envision relieving teachers from laborious and repetitive tasks, allowing them to focus on their students. Ideally, teachers browse *CodeHarbor* when looking for inspiration for a new programming assignment, reuse or adapt an existing exercise, and mirror feedback received back to a community of like-minded educators. For the use with auto-graders such as *CodeOcean*, *CodeHarbor* features import and export capabilities with the standardized *ProFormA* format, reducing the manual setup of a new exercise.

2.2 Sharing Educational Resources Between Users and Platforms

The open and free accessibility of the *openHPI* platform provides opportunities to link its content with large repositories. As a result, various components of the content, including *CodeOcean* exercises as educational resources, can be made available for use by instructors seeking to facilitate the teaching of programming courses. This is especially relevant when utilizing the auto-grader features. To that end, we will discuss the available exchange platforms and data format options.

Exchange Platforms. In terms of educational resources in general, projects that collect and share educational resources include *MERLOT* [SB02], *oercommons* [Gal16], *4teachers* [4te23], and *X5GON* [Per+21]. All these examples work similarly, categorizing various types of educational resources into a single location to make it easier for educators to find appropriate content for their teaching activities.

In a narrower sense, some learning platforms, such as Learning Management Systems (LMS) or MOOCs, already have a resource bank concept that allows instructors to collect and share various materials. Usually, this includes both content and evaluation tools, such as quizzes and programming exercises. However, the main limitation of these resource banks is that most sharing can only be done within an instance, making it challenging to obtain the resource without direct access to the platform [STM17].

If we further focus on a specific use case, such as content sharing for programming content, particularly programming exercises, the options become even more limited. Although some projects mentioned above have a catalog of exercise content, those sharing platforms are not specifically designed for programming exercises, let alone to do auto-grading programming exercises. This necessitates a more intricate setup that encompasses not only the definition and sharing of files but also the capacity to execute the exercise for various existing programming languages, like what is offered by *CodeOcean*.

This is the motivation behind the creation of *CodeHarbor*. While *CodeHarbor* cannot execute the exercise in its repository, it has an export action that allows the exercise to be easily exported to another auto-grader system, such as *CodeOcean*. To do so, *CodeHarbor* requires a standard exchange format. Currently, the *ProFormA* format has been predominantly investigated, but other formats may also be explored for this purpose.

Exchange Formats. A standardized format is required for data exchange between platforms. The ability of the format to precisely define the exercise and apply it to the various programming language environments supported by the auto-grader tool itself is a consideration in selecting the appropriate exchange format to be used in the auto-grading programming exercise process.

In the context of exchanging data, a variety of structured data formats such as JSON [Bra17], YAML [BEI09], XML [Bra+08], and even HTML [HTM22] are commonly utilized due to their universal nature and broad applicability.

However, in the realm of educational resource exchange, metadata standards such as SCORM (Sharable Content Object Reference Model) and LOM (Learning Object Metadata) were developed to specifically cater to the unique needs of learning objects. Both standards serve the purpose of enabling the exchange of educational resources, but they differ in their approach. LOM is a metadata standard that describes the learning objects [LOM20], while SCORM is a technical standard that outlines the packaging, delivery, and tracking of the learning objects [ADL04].

Although both standards allow for the interoperability of interactive content packages such as quizzes and exercises sharing between different e-learning systems, due to their general nature, they are unsuitable to be used as standards in exchanging auto-grading programming exercises [STM17].

CodeHarbor uses the *ProFormA* standard to define programming exercise content to be exchanged between auto-grading tools. *ProFormA* is built using XML-defined structures based on the principle of separating the programming exercise's content from the auto-grader system's implementation details. This separation is achieved with metadata and a structured exercise description format. The *ProFormA* standard defines a set of metadata elements. It specifies a structured exercise description format that is divided into three parts, including the exercise statement, the expected output, and any additional resources, such as input files or helper functions, required for the exercise [Str+15].

The advantage of this separation is its flexibility, which allows this standard to define exercise content independent of the setup of different auto-grader tools, unlike other standards such as Programming Exercise Markup Language (PEML) and Programming Exercise Interoperability Language (PExIL).

PEML is designed as a standard to describe the content in structured plain-text form with keys and values. PExIL, on the other hand, is built on XML dialect that describes a Learning Object that contains a programming exercise life cycle. The main drawback of PEML is the high complexity of structuring descriptions in a markup language, which makes it heavily dependent on available parsers [ME23]. PExIL, on the other hand, suffers from a lack of flexibility in the case of more complex programming exercises, as the structuring of tasks only allows a single solution file [QL11]. In addition, both formats have fewer resources and tools available to support their use, so it will be more difficult to develop compared to *ProFormA* [Str+15; ME23].

The three example formats above demonstrate that the auto-grader tool provides numerous options for sharing exercise content. However, after considering the format's maturity and adoption [PB17], it was determined that the *ProFormA* format is the best choice to be used as a standard data exchange format in *CodeHarbor* [STM17; Ser+21].

2.3 User Experience Research Studies on Platform Design

On a methodological level, only a few authors documented their research on designing platforms based on qualitative or user experience research methods.

Preceding platform design, but establishing important touchpoints for user interfaces, Khambete and Athavankar (2010) used grounded theory in a User Experience Design case study to establish why and how users select touchpoints in their banking and telecommunication transactions.

Islind et al. examined their design of a digital healthcare platform by using boundary objects [Isl+19]. The authors applied a design ethnography approach and conducted a co-design phase with users consisting of a future-oriented workshop, role-playing sessions, and a workshop with concrete objects and prototypes. The authors argue that co-designing with boundary objects as design tools can be useful in heterogeneous user groups and complex settings.

Spagnoletti et al. developed a design theory for creating digital platforms that support online communities. In their approach, the author phrased an initial proposition based on Information Systems design literature, validated these propositions through a case study, and derived new validations and insights from several case studies. Ultimately, they phrase seven propositions that can advise fellow designers of digital platforms that support online communities [SRL15].

This related work shows that qualitative user experience design research methods are especially helpful when it comes to designing platforms that cater to new topical realms or target user groups and behaviors that are largely unknown or private. In comparison to quantitative methods, e.g., A/B Testing of functions, it enables designers to understand their users' needs and problems and conceptualize a platform that resonates with them from the get-go.

3 Research Approach and Method

In this paper, we describe a qualitative research study in the domain of User Experience Design. We apply an empirical study with a multi-method design, using qualitative research methods to establish principles for our design object [CC17]. As outlined in Section 1, the research questions we aim to answer are the following:

RQ1.	What are the key challenges and potential opportunities experienced by high-school teachers who engage in exchanging digital teaching materials for Computer Science classes?
RQ2.	What design principles should be considered when developing a platform for exchanging (coding) teaching materials in the German education system?

To validate and further specify the requirements for the *CodeHarbor* platform, a qualitative user research phase was conducted, starting in November 2021. The primary objective of the user research was to comprehend the work experience of potential *CodeHarbor* users

and identify their problems and needs. This data was then used to define possible requirements for the platform's design. Simultaneously, user feedback on an initial prototype helped in the rapid, iterative improvement of the basic idea.

For the user research, ten interviews were scheduled, each lasting between 60 and 90 minutes. The interviews were divided into two main parts: The "generative part" focused on the problems and needs of teachers when preparing programming tasks, while the "evaluative part" dealt with the presentation of a mockup prototype that allowed the interviewees to critique and provide initial feedback.

Generative User Research Part. The generative user research part aimed to identify potential requirements for the platform by gaining knowledge about teachers' problem space and areas that provide possibilities when preparing learning material for their classes. We used semi-structured interviews along with the participatory user research methods of user journeys and card ranking as described by Alsos and Dahl [AD09]: In each interview, the interviewees created a user journey in collaboration with us to depict the interviewees' experiences with creating programming tasks for their students, showcasing the highlights and pain points of their process. An example of this user journey is shown in Figure 1. Following this, the interviewees conducted a card ranking exercise: They ranked the features they deemed necessary for a task exchange platform based on their importance.

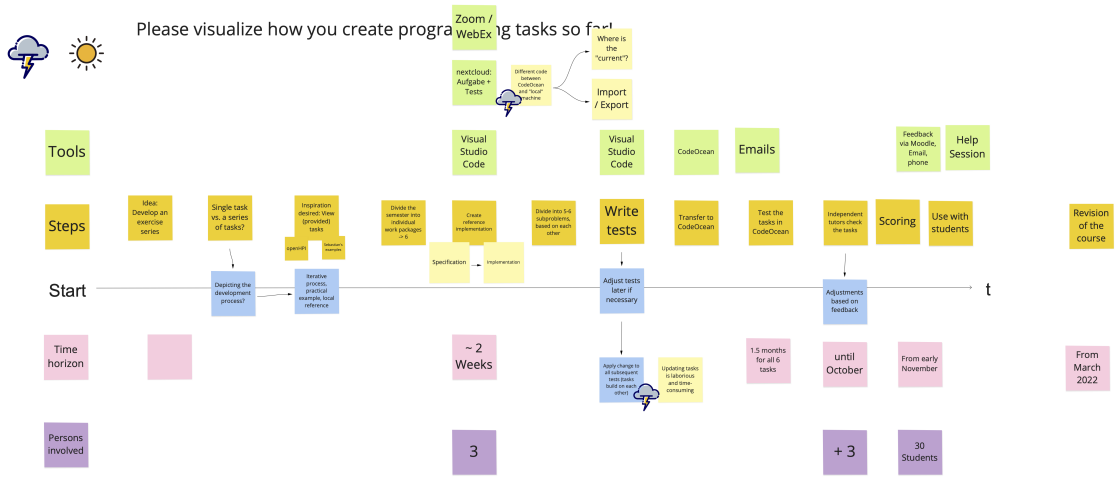

Fig. 1. An exemplary user journey as created during the second interview. The interviewee reflected on the various steps needed to create an exercise (in yellow) or update them (in blue), along with the different tools used (in green), the time horizon (pink) and the number of persons involved (purple).

Evaluative User Research Part. For the evaluative user research part, we prepared a wireframe prototype of the platform interface, showcasing the most important interaction points. Interviewees looked through the different wireframe slides and voiced their thoughts

out loud. During the interview, we documented these comments on the wireframe slides, both to emphasize the participatory aspect of the exercise and to create an overview for subsequent evaluation.

Interviewee Selection. Overall, we chose ten individuals from the three educational contexts of school, university, and MOOC platforms as interview partners (see Table 1). The aim of *CodeHarbor* is to facilitate programming task sharing among teachers from these diverse settings. However, the working realities of these contexts can vary significantly, and user research can help explore these environments and identify specific problems and needs. The interviewees had between five and 25 years of experience teaching computer science, with teaching formats aimed at both beginner and advanced students.

Interview Number	Context	Persona
1	MOOC	University students who are creating a MOOC for the first time. They are in exchange with experienced MOOC designers. They can communicate with each other confidently and have fun with their task.
2	Higher Education	A Lecturer in Information Systems at a University of Applied Sciences (UAS) who collaborates with colleagues and tutors to create and implement programming assignments for seminars.
3	School	A computer science teacher who works at a high-school. While there are other computer science teachers in the school, they do not have much interaction with them. Demanding courses like the advanced IT course are popular and “fought over” by all teachers.
4	Higher Education	UAS lecturer who offers programming courses for business students.
5	School	Computer science teacher at a school in Saxony, where computer science is taught continuously from 7th grade. Students are equipped with iPads. He is also involved in a project team that is designing the curriculum for the next ten years.
6	Higher Education	Professor who has years of experience in teaching and didactics.
7	Higher Education	Lecturer who writes programming assignments and helps develop systems for automatic assessment.
8	School	Teacher who has taught computer science for 18 years. He is a subject leader in computer science at his school and teaches mainly courses from 9th grade onwards.
9	MOOC	An “amateur” teaching team of students creating an online course for the first time.
10	Higher Education	A UAS instructor who creates programming assignments for business students in elective courses. In general, the subject matter is not particularly popular for his students.

Table 1. List of Interviewees.

Analysis. After conducting the ten interviews, we had a visual representation of the user journeys each interviewee experiences when creating programming exercises (see Sections 4.1 and 4.2). Further, we got a ranked list of desired functionalities an exercise repository as envisioned by the interview partners should have (Section 4.3). Finally, we received dedicated feedback on our existing prototype, as outlined in Section 4.4). Based on the interviewees’ user journeys, we conducted a thematic analysis [GMN12] of their workflows, with a focus on defining the problem space and areas of opportunity.

Additionally, we considered the users' rankings of platform features to further evaluate the urgency of the problem and opportunity areas. Subsequently, we also consolidated the comments collected on the prototype mockup slides and conducted another thematic analysis with these.

As a result of the thematic analyses, we identified 19 themes, which we translated into user stories. User Stories are a common tool in agile software development and can serve as a user-centered design principle for a design object such as the *CodeHarbor* platform [APB20]. We defined a total of 47 user stories, representing the needs and workflows of our ten interviewees. Following, we evaluated all user stories based on their importance, reusability by different stakeholders, and prerequisites in a method like software development prioritization techniques (e.g., as described by Wiegers [Wie99]) and then selected the 12 highest-ranked stories as the basis for defining Design Principles.

4 Results

In this part, we will present the results of our User Experience Research, which covers both the generative and evaluative research. Our results are clustered in the thematic analyses performed (Sections 4.1 and 4.2), the ranking of the desired functionality (Section 4.3) and feedback on the initial prototype (Section 4.4). Based on these results, we formulated twelve user stories, which we shortly summarized in Section 4.5.

4.1 Thematic Analysis: Problem Space

The thematic analysis of interviews revealed different challenges that users face when creating or sharing learning material online. We initially identified 15 themes, which interviewees described as their potential problem space.

- *Implementation:* Refers to problems regarding the implementation of tasks in the teacher's course context. Interviewee 1, a team of university students who design a MOOC for the first time, explained that *"difficult parts of the implementation of the auto-graded tasks were not clear or hidden before. We needed help with the implementation."*
- *Auto-graded Feedback:* Refers to the use of automated feedback to learners for coding tasks. Interviewee 10, a lecturer at a University of Applied Sciences, mentioned that *"with automatic evaluation, feedback also becomes smaller in a way — additional information would sometimes be helpful for the students because individual feedback is also an additional motivation."*
- *Adaptation During Runtime:* Refers to the ability of teachers to adjust or modify tasks during a running course, either because mistakes were pointed out or because they want to adapt tasks to learner progress.

- *Updates:* Refers to potential updates of exercises and tests. Interviewee 2, a lecturer at a University for Applied Sciences working on programming exercises with his colleagues, remembered how *"tests are written and may have to be adjusted later, then these requirements must be adopted in all subsequent tests. This updating is very time-consuming."*
- *Assessment:* Refers to general aspects of assessing learner solutions.
- *Development Environment:* Refers to aspects of choosing or providing an Integrated Development Environment (IDE) for learners. Some exercises might work best in a specific, educational IDE, such as GreenFoot [Köl10].
- *Exchange:* Refers to the general practice of sharing or exchanging content between users on the platform.
- *Confidentiality:* Refers to problems regarding the process of safely logging in or accessing the platform and authenticating that only teachers are using the system. Interviewee 6, a professor at the University of Applied Sciences, wondered: *"How do we protect the system in a way that tasks do not end up publicly on the internet?"*.
- *Trust:* Refers to the establishment of user trust in the available tasks. This covers all aspects of an exercise, including the exercise description and test cases. According to interviewee 7, trust in the exercises can be increased by establishing a visible quality assurance process.
- *Quality Assurance:* Refers to problems of recognizing the degree to which the tasks on a platform are well-designed, complete, and effective.
- *Structure:* Refers to the organization and presentation of teaching material within the platform. For example, the resources could be structured thematically or by (the age of) the target group.
- *Contextualization:* Refers to the process of adapting teaching material to suit the specific context of courses — ranging from high-school classes to university courses. Interviewee 9, a student teaching team building their first online course, explained that: *"Many tasks found on the web did not fit our context — the research did not move our team forward in this way."*
- *Task Type & Difficulty:* Refers to problems of recognizing the different types of tasks or activities available on a platform, and possibly also the level of tasks. Interviewee 8, a high-school teacher, described that *"many tasks from internet sources are often at a university level and thus too difficult for my students."*
- *Task Status:* Refers to general status regarding programming tasks, such as the previous usage or an author categorization (draft, finished, practically used, etc.) *Natural language:* Refers to issues that come up regarding the use of human language, e.g., a situation described by Interviewee 8: *"Very young students may struggle when receiving error messages in English (for example from a compiler). This can be challenging for them."*

4.2 Thematic Analysis: Opportunity Areas

The thematic analysis of opportunities revealed different positive possibilities that users see when creating or sharing learning material online. In total, we identified 13 topics for the opportunity areas, nine of which were overlapping with the problem area: *Auto-graded Feedback, Updates, Exchange, Confidentiality, Trust, Quality Assurance, Structure, Contextualization,* and *Task Type & Difficulty*. Furthermore, we identified four additional opportunity areas. These are:

- *Inspiration:* Refers to the possibility of gaining inspiration from other tasks. Interviewee 2 explained that their last process of creating tasks started with *"examining available tasks on openHPI to get inspired"*.
- *Learning Goal-oriented Task Development:* Refers to the process of creating tasks or activities on the platform that are designed to help users achieve specific learning goals or objectives. Interviewee 1 described that their process of gathering programming tasks was to *"move from learning objectives to keywords to tasks. What do we want learners to solve in our tasks? A learning objective-oriented search function would be helpful."*
- *Collection:* Refers to the gathering or curating of tasks or resources on the platform in a collection, making it easier for users to access and use.
- *Storytelling:* Refers to the use of narratives or storytelling techniques within tasks, and the possibilities of collecting those on a platform as well [HSM23].

4.3 Ranking of the Desired Functionality

Participants identified and ranked functions that they deemed central to a task exchange platform, as shown exemplary in Figure 2. The ranking of elements is based on pain points identified during their user journey and helped us to prioritize the research results. Among the most requested topics were *Confidentiality, Quality Assurance, Inspiration, Exchange,* and *Structure*.

Fig. 2. A ranking of the desired features as done by Interviewee 2. The black line depicted on the left ranges from very important at the top to not so important at the bottom, allowing interviewees to indicate the most relevant aspects of a platform to share programming exercises. We grouped similar features in yellow, added quotes in orange, and took questions in blue.

4.4 Prototype Feedback

To analyze the prototype, we consolidated the comments on the prototype mockup slides, such as those shown in Figure 3, which we gained during the interviews. Based on the various clusters identified, we conducted a thematic analysis. The analysis revealed the five feedback areas of *general feedback, information about programming exercises, search & filter function, workflows & export,* and *collaboration*. These topical focus points helped us to prioritize the results gathered in the generative part of the research and eventually lead to the resulting design principles for the work on *CodeHarbor* (as presented in Section 5.2).

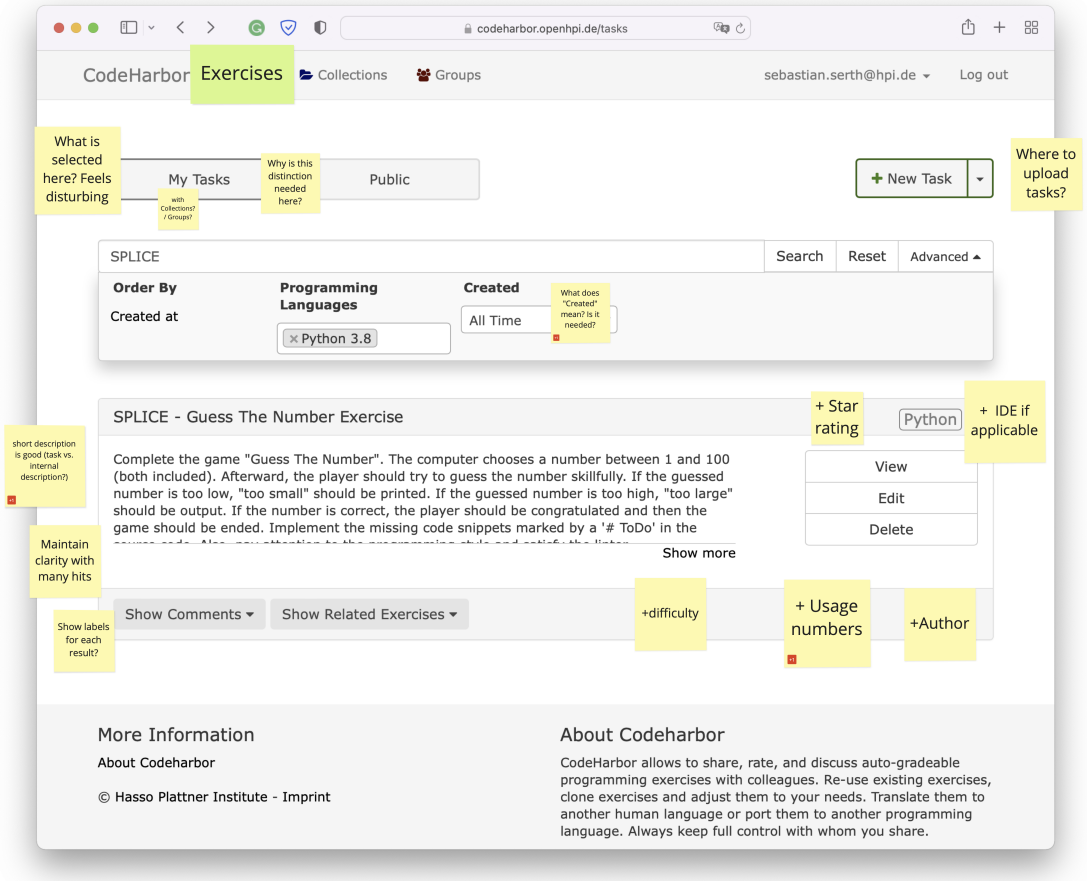

Fig. 3. A prototype of CodeHarbor with feedback, as collected during the second interview.

4.5 Generated User Stories

Based on the thematic analysis and the importance of topics, we defined a total of 47 user stories and selected twelve by grouping and merging similar ones or generalizing aspects mentioned by different interviewees. We also placed an emphasis on the most pressing aspects to be addressed first, while the remaining aspects should be reconsidered in the

future. The selected user stories served as a basis for defining Design Principles based their importance and urgency for platform development. In Table 2, we present the twelve user stories that were advanced further.

Topics	User Story
Task Type & Difficulty	As a professor who uses programming assignments for students, I want task descriptions that include complexity and difficulty level, so that I can better assess whether assignments are appropriate for my students.
Confidentiality, Collection	As a professor who has already had bad experiences with assignment sharing platforms, I want to be able to regulate the visibility of my shared tasks myself, so that I can share certain tasks only in the circle of colleagues.
Confidentiality	As an instructor who has to use the class time frame of 45 minutes for programming assignments, I would like to have a platform that verifies identification as a teacher during sign-on, so that I can make sure that solutions are not used by students.
Inspiration	As a MOOC designer creating a course for the first time, I would like to browse through different assignments, so I can get inspiration for my own course.
Task Type & Difficulty	As a MOOC designer creating a course for the first time, I want to filter by different types of assignments and learning objectives, so that I can specifically search for suitable tasks.
Task Type & Difficulty	As an instructor who has to use the class time frame of 45 minutes for programming assignments, I want to filter tasks according to prior knowledge and learning goal, so that I can find tasks that fit better into the context of the students, and not always “only” math related tasks.
Exchange	As a MOOC designer who works collaboratively with colleagues on assignments, I would like to have an overview of tasks that makes it possible to visualize problems, to-dos, and changes made by others, so I can keep track and not overwrite work.
Implementation	As a MOOC designer who is using auto-graded feedback tasks for the first time, I would like to receive guidance on how to implement testing, so that I am prepared for the difficulties of implementation.
Development Environment, Task Type & Difficulty	As a teacher who has a clear preference for IDEs and programming languages in different grade levels, I would like to filter assignments by IDEs and programming languages, so that I can find appropriate material for my different grade levels.
Task Type & Difficulty, Quality, Task Status	As a teacher responsible for many courses, I want to be able to quickly see the quality and completeness of tasks (and other material), so that I do not spend time with long searches.
Trust, Confidentiality	As a professor who prepares and teaches many courses for students, I want a simple, open licensing system for tasks so that posting and using tasks can happen without worry.
Implementation	As an instructor developing new programming courses for students, I want to be able to easily export tasks into my learning environment, so that I do not experience disruptions between the assignment platform and the learning environment.

Table 2. Topic/User Stories.

5 Discussion

Based on our results presented in the previous Section 4, we reflect on the learnings, derive conclusions and answer our posed research questions. On the one hand, our study allowed us to learn more about the problem space and opportunity areas of our target group (RQ1, Section 5.1). On the other hand, we used that knowledge to refine design principles guiding our development efforts (RQ2, Section 5.2).

5.1 Problem Space and Opportunity Area

Although we consider ourselves to be quite close to one of *CodeHarbor's* target groups by regularly creating and offering (programming) MOOCs, we still wanted to talk to independent representatives working in schools, universities, or for other MOOCs. With a focus on high-school teachers as a new user group to auto-graders, we asked:

RQ1.	What are the key challenges and potential opportunities experienced by high-school teachers who engage in exchanging digital teaching materials for Computer Science classes?

Identified Problems. Our thematic analysis identified a total of 15 topics for the problem space that educators may face. Some of these themes, such as *structure, task type & difficulty*, or *task status*, relate to the presentation and organization of exercises on *CodeHarbor*. Teachers should be able to effortlessly navigate through the collection of tasks to quickly find the resources they are looking for. However, the majority of topics, such as *Implementation, Adaption During Runtime, Contextualization, Trust*, and *Quality Assurance*, are more related to the applicability and transferability of the exercises by teachers to their context: if reviewing and then modifying an exercise (e.g., to correct errors or to make minor adjustments to suit one's needs) takes too much time, teachers will stick to writing their own tasks rather than using tasks published by others.

During the interviews, another aspect regarding the *Auto-graded Feedback* and prerequisites became apparent among the interviewees. In most cases, university lecturers and MOOC instructors already have access to auto-graders and other collaboration tools with their colleagues. However, the situation of high-school teachers is usually more complex: most teachers do not have access to an auto-grader and some even do not have an established exchange of materials at their school. While we are seeking to provide teachers with access to our auto-grader *CodeOcean*, we envision that *CodeHarbor* will become one of the platforms to connect teachers with each other and allow for a subject-specific exchange.

Potential Opportunities. Based on the interviews conducted, we identified a total of 13 topics for the opportunity area, of which nine were overlapping with potential problems. It shows the potential to turn a problem into an opportunity by addressing it with a scalable solution in *CodeHarbor*. For example, one problem identified also raised as a potential opportunity is in the area of *Updates*, referring to the synchronization efforts required to keep various copies of the same exercise up-to-date. It applies to the same component being used in several contexts (such as two courses or a test that is used in multiple

exercises), but also describes refactorings that might happen later (e.g., in a template published for the use by learners and in a sample solution). Automating and assisting educators to spend less time on manual intervention and rather show relations between exercises or files is therefore important and can lead to a relief.

Further opportunities arise through the four specific topics introduced in Section 4.2. The majority of the topics listed there can be attributed to creative aspects of defining new exercises. For example, *Inspiration, Storytelling* and partially *Learning Goal-oriented Task Development* are beneficial to create own, engaging, and suitable tasks for learners and also help to overcome a burden to get started by browsing other examples. Therefore, it allows teachers to increase the use of (auto-gradable) programming exercises, which in turn will benefit learners.

5.2 Resulting Design Principles

For us, identifying the problem space and opportunity areas is only the first step. It allows us to gain a better understanding of the users' needs and thus answer the second research question:

RQ2.	What design principles should be considered when developing a platform for exchanging (coding) teaching materials in the German education system?

During our analysis, we identified three overarching design principles, touching several aspects of the user journey with *CodeHarbor*. These the design principles are presented in the following:

Search and Filter Function. The search and filter function was found to be of great importance to users during both the generative and evaluative stages of user research. Respondents found keyword searches to be intuitive but expressed a desire for more precise filtering options based on different categories. For example, a high-school teacher with clear specifications for IDEs and programming languages for different grade levels would prefer to use a filter mask based on these categories to quickly find appropriate material for each course. Consequently, the design phase uncovered the need to supplement existing filter options for users. Therefore, we are adding customized labels to exercises and the search, for example covering the topic, target learner group, or expected difficulty level.

In addition, users emphasized the importance of a filtered preview of assignments. They suggested that the preview should not be overwhelming in terms of the number or visual presentation of tasks. Nonetheless, several instructors expressed a desire to use the preview as a source of inspiration and to browse through a range of assignments to generate new ideas for their own courses. As we further develop our platform, our goal is to design the preview of assignments in such a way that instructors can quickly grasp the most important information about several assignments, without overwhelming them with too much detail.

Quality Assessment. During the design phase, user research focused on the crucial topic of quality assurance and completeness. Interviewees emphasized the need for a quick and easy way to determine the quality and completeness of an assignment, as an extended

search can be time-consuming and frustrating. For example, one UAS lecturer stated that creating new programming courses for students is already a massive task, and a lengthy search for materials would discourage them from using the platform altogether.

Completeness and quality have different dimensions, with completeness referring to the concern that some participants may provide incomplete tasks, without a sample solution or automatic tests, either out of ignorance or by mistake. Quality, on the other hand, refers to the assessment of the value of a task, considering factors such as originality and ease of comprehension. While a five-star rating scale presented in the mockup was visually appealing and reminiscent of digital shopping experiences, it raised questions for some users. For instance, they wanted to know precisely what the rating system entailed, how other users perceived quality, and whether they could rely on the ratings.

Overall, the user group has a pressing need for a quick and reliable overview of completeness and quality. Failure to address this demand could lead to the platform becoming unused. Therefore, we are planning to counteract from the very beginning with two approaches: First, educators should be motivated to provide as much information as possible to a task they share, for example by showing them a completeness indicator or further nudging them with gamification elements as described by Krath et al. [KSV21]. Second, we want to reward teachers who create well-rated content and encourage the community to report and address any deficiencies that are identified (for example, as done on technical Q&A platforms such as Stack Overflow [WCH20]).

Repository. Several interviewees expressed their desire for the platform to function as a centralized repository for their ongoing work. For instance, a group of MOOC designers who collaborated on tasks wanted a visual representation that would enable them to keep track of problems, to-dos, and changes to tasks, thus avoiding overwriting each other's work as they have done in the past. Evolving *CodeHarbor* to a repository would significantly change the platform's focus and how educators can interact with it but would simultaneously allow users to further reduce the number of additional tools used.

One further idea was that *CodeHarbor* could function as a version control system, i.e., turning into a Git repository [Spi12]. Conversely, if the platform is developed into a (Git) repository, it could also solve the versioning issue for collaboratively posted or revised tasks. Therefore, extending the platform's capabilities to function as a repository and revising the export function are worth considering. We are currently planning to extend the content types teachers can manage on *CodeHarbor* beyond programming exercises, starting with more teaching resources. Further experiments and subsequent user studies to evaluate the transformation to a Git repository are the subject of future work.

6 Roadmap

From the very beginning, *CodeHarbor* was intended to address the needs of MOOC instructors, high-school teachers and university lecturers who wanted to create and share programming exercises more easily with another. Hence, the user-centered research described in this article is important and serves as a basis for the future development of the platform. The experience of teachers with repository-sharing platforms like *CodeHarbor* has shown that a tool's usefulness is determined by how widely it is used. Therefore, we recognize the importance of engaging with the community to promote the tool's adoption for

the sustainability of *CodeHarbor*. As a first step and to initially attract new educators, we will provide access to all programming exercises included in the openHPI programming courses.

Further, we want *CodeHarbor* to expand from a content sharing platform and add dedicated support for educators to add new exercises. Since we identified during the interviews that especially high-school teachers are not confident enough to write tests for their exercises, we are investigating how a suitable assistance might look like. On the one hand, we are currently working on a free MOOC planned for 2024 that should cover how automated tests work and how they are written. On the other hand, we are also looking into a test generator, intended to assist teachers in covering the most common test cases.

Another key objective on our roadmap is to explore ways to expand *CodeHarbor's* repository-sharing platform beyond the *ProFormA* format: Starting with support of IEEE LOM format as a metadata standard, we want to provide a tight integration of CodeHarbor with the German project "digitale Vernetzungsinfrastruktur für die Bildung" [Ini22]. Following, we will evaluate how we can incorporate emerging standards to ensure that *CodeHarbor* remains relevant and up-to-date. These efforts will be instrumental in transforming *CodeHarbor* into a more versatile and comprehensive tool that can be utilized internationally, allowing a broader range of educators to benefit from its capabilities.

6.1 Future Work

With the current project, we aim to improve *CodeHarbor's* functionality by targeting the twelve user stories defined in this article. As a result, the outcomes are expected to address these use cases, but might not yet cover all initial 47 user stories and advanced use cases. Also, some open questions outlined in the previous chapters still exist, for example how exactly the transformation of *CodeHarbor* into a repository will look like. In the future, we could also expand *CodeHarbor* into a comprehensive repository that goes beyond computer science and covers other fields around science, technology, engineering, and mathematics (STEM). Furthermore, numerous other types of interactive exercises still need to be addressed but can be included as *CodeHarbor* evolves.

We are also encouraging other researchers to contribute to *CodeHarbor's* development, the available integrations and to expand the repository with more teaching content, as these areas offer significant potential for improvement. We are convinced, that the integration of *CodeHarbor* with the German project "digitale Vernetzungsinfrastruktur für die Bildung" will provide more valuable opportunities for collaboration with other content providers, researchers, and educators, paving the way for more interconnected offerings and a better computer science education for future generations.

7 Conclusion

With the rise of Massive Open Online Courses (MOOCs), programming education has become more accessible and interactive, allowing an increasing number of learners to receive automated feedback [GC22]. Besides MOOC instructors, university lecturers and especially high-school teachers realized the advantages of auto-gradable programming

assignments, allowing them to focus on students struggling the most [Ser+19]. However, creating exercises with automated feedback is time-consuming and many high-school teachers do not have the confidence to create these [STM17; Ser+21]. Therefore, we started working on *CodeHarbor*, a platform that allows educators to exchange those auto-gradable programming exercises with each other.

In the paper at hand, we conducted a qualitative user research phase to comprehend the needs and problems of potential users for *CodeHarbor*. This research phase included ten in-depth interviews with a focus on the problems and needs of teachers when preparing programming tasks, as well as presenting a mockup prototype to receive feedback. The insights gained from this research helped to define the requirements for the platform's design and served to rapidly iterate on the desired functionality.

After conducting the interviews, a thematic analysis was performed to define the problem space and identify opportunity areas in the user journeys of the interviewees. Further, we used the rankings of platform features as indicated by the interviewees to evaluate the urgency of these areas. A second thematic analysis was conducted on the comments collected on the prototype mockup slides. The identified needs from both analyses were then translated into 47 user stories, from which we selected twelve to serve as a basis for defining design principles. These design principles prioritized ease of use, collaboration, and customization and will guide the development of the platform to ensure it meets the needs of the target group.

In conclusion, *CodeHarbor* aims to offer several key benefits to teachers and educators looking to enhance their programming education with interactive hands-on programming exercises. With CodeHarbor's ability to facilitate the exchange of programming exercises between different online education platforms, we enable teachers to easily discover and reuse exercises from their colleagues and a like-minded community, regardless of the specific auto-grader used. This collaborative approach to teaching allows educators to dedicate their time to prepare engaging teaching materials, avoids the replication of similar tasks across educators, and let them pick up new ideas by learning from one another.

As programming education continues to expand and become more widely adopted, there is a growing demand for accessible and user-friendly tools that are aimed at MOOC instructors as well as smaller groups of students and teachers in universities and high-schools. While *CodeOcean* was initially designed to serve a large number of participants with only a limited number of teachers, *CodeHarbor* was developed to address the need for a collaborative platform that allows teachers to easily share, rate and discuss auto-gradable programming exercises. With additional features, such as exercise author tools, and by integrating with relevant platforms and initiatives, *CodeHarbor* will become an essential tool for computer science educators to make their teaching more interactive and enjoyable.

Acknowledgement

The work in this study has been created during the CoHaP2 project, which is funded by the German Federal Ministry of Education and Research as part of the project "digitale Vernetzungsinfrastruktur für die Bildung". The initiative's goal is to "connect educational offerings and thus create access to a wider range of educational opportunities for every citizen achieved by common standards, formats and interoperable structures" [Ini22].

References

[4te23] 4teachers GmbH. 4teachers.de. 4teachers: Lehrproben, Unterrichtsentwürfe und Unterrichtsmaterial für Lehrer und Referendare! 2023. URL: https://www.4teachers.de/ (last check 2023-09-06)

[AD09] Alsos, Ole Andreas; Dahl, Yngve: The card ranking technique: Application and added value in comparative usability testing. 2009. https://folk.ntnu.no/oleanda/cardranking/cardranking.pdf (last check 2023-09-06)

[ADL04] Sharable Content Object Reference Model (SCORM®). Advanced Distributed Learning Initiative (ADL), 2004. URL: https://adlnet.gov/projects/scorm/ (last check 2023-09-06)

[APB20] Ananjeva, Alisa; Persson, John Stouby; Bruun, Anders: Integrating UX work with agile development through user stories: An action research study in a small software company. In: Journal of Systems and Software, 170, Dec. 2020, p. 110785. ISSN: 01641212. DOI: 10.1016/j.jss.2020.110785. (last check 2023-09-06) URL: https://linkinghub.elsevier.com/retrieve/pii/S0164121220301953 (last check 2023-09-06)

[BEI09] Ben-Kiki, Oren; Evans, Clark; Ingerson, Brian: YAML Ain't Markup Language (YAML) (tm) Version 1.2. Sept. 2009. URL: http://www.yaml.org/spec/1.2/spec.html (last check 2023-09-06)

[BKB22] Bernius, Jan Philip; Krusche, Stephan; Bruegge, Bernd: Machine learning based feedback on textual student answers in large courses. In: Computers and Education: Artificial Intelligence 3, 2022, p. 100081. ISSN: 2666920X. DOI: 10.1016/j.caeai.2022.100081 (last check 2023-09-06) URL: https://linkinghub.elsevier.com/retrieve/pii/S2666920X22000364 (last check 2023-09-06)

[Bra+08] Bray, Tim; Paoli, Jean; Sperberg-McQueen, C. M.; Maler, Eve; Yergeau, François: Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation, 2008. http://www.w3.org/TR/REC-xml/ (last check 2023-09-06)

[Bra17] Bray, Tim: The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259, Dec. 2017. DOI: 10.17487/RFC8259 (last check 2023-09-06) URL: https://www.rfc-editor.org/info/rfc8259 (last check 2023-09-06)

[CC17] Creswell, John W.; Plano Clark, Vicki L.: Designing and conducting mixed methods research. Sage publications, Los Angeles u.a, 2017.

[CC220] CC2020 Task Force. Computing Curricula 2020: Paradigms for Global Computing Education. New York, NY, USA: ACM, Nov. 15, 2020. ISBN: 978-1-4503-9059-0. DOI: 10.1145/3467967 (last check 2023-09-06) URL: https://dl.acm.org/doi/book/10.1145/3467967 (last check 2023-09-06)

[Elh+22] Elhayany, Mohamed; Nair, Ranjiraj-Rajendran; Staubitz, Thomas; Meinel, Christoph: A Study about Future Prospects of JupyterHub in MOOCs. In: Proceedings of the Ninth ACM Conference on Learning @ Scale. L@S '22. New York City, NY, USA:

Association for Computing Machinery, 2022, pp. 275–279. ISBN: 9781450391580. DOI: 10.1145/3491140.3529537 (last check 2023-09-06) URL: https://doi.org/10.1145/3491140.3529537 (last check 2023-09-06)

[Gal16] Gallaway, Teri Oaks: OER Commons. In: The Charleston Advisor 17.4, Apr. 1, 2016, pp. 35–38. ISSN: 1525-4011. DOI: 10.5260/chara.17.4.35 (last check 2023-09-06) URL: http://www.ingentaconnect.com/content/10.5260/chara.17.4.35 (last check 2023-09-06)

[GC22] Gabbay, Hagit; Cohen, Anat: Investigating the effect of Automated Feedback on learning behavior in MOOCs for programming. In: Proceedings of the 15th International Conference on Educational Data Mining. Durham, United Kingdom. DOI: 10.5281/ZENODO.6853125 (last check 2023-09-06) URL: https://zenodo.org/record/6853125 (last check 2023-09-06)

[GG49] Basic Law for the Federal Republic of Germany. May 23, 1949. URL: https://www.gesetze-im-internet.de/englisch_gg/ (last check 2023-09-06)

[GMN12] Guest, Greg; MacQueen, Kathleen; Namey, Emily: Applied Thematic Analysis. SAGE Publications, Inc., Thousand Oaks, California, United States, 2012. ISBN: 978-1-4129-7167-6. DOI: 10.4135/9781483384436 (last check 2023-09-06) URL: https://methods.sagepub.com/book/applied-thematic-analysis (last check 2023-09-06)

[Has+19] Haselmeier, Kathrin; Humbert, Ludger; Killich, Klaus; Müller, Dorothee: Interesse an Informatik und Informatikselbstkonzept zu Beginn der Sekundarstufe I des Gymnasiums. In: Informatik für alle. Bonn, Germany: Gesellschaft für Informatik, 2019. ISBN: 978-3-88579-682-4. DOI: 10.18420/INFOS2019-B6 (last check 2023-09-06) URL: http://dl.gi.de/handle/20.500.12116/28969 (last check 2023-09-06)

[HSM23] Hagedorn, Christiane; Serth, Sebastian; Meinel, Christoph: The mysterious adventures of Detective Duke: How storified programming MOOCs support learners in achieving their learning goals. In: Gamage, Dilrukshi (Ed.): Frontiers in Education. A Paradigm Shift in Designing Education Technology for Online Learning: Opportunities and Challenges, 7, Jan. 11, 2023, p. 1016401. ISSN: 2504-284X. DOI: 10.3389/feduc.2022.1016401 (last check 2023-09-06) URL: https://www.frontiersin.org/articles/10.3389/feduc.2022.1016401/full (last check 2023-09-06)

[HTM22] HTML - Living Standard. Web Hypertext Application Technology Working Group (WHATWG), 2022. URL: https://html.spec.whatwg.org/ (last check 2023-09-06)

[Ini22] Initiative Digitale Bildung: National Education Platform. Feb. 2022. URL: https://www.oecd.org/education/ceri/National-Education-Platform-Germany.pdf (last check 2023-09-06)

[Isl+19] Islind, Anna Sigridur; Lindroth, Tomas; Lundin, Johan; Steineck, Gunnar: Co-designing a digital platform with boundary objects: bringing together heterogeneous users in healthcare. In: Health and Technology, 9, 2019, pp. 425–438 https://link.springer.com/article/10.1007/s12553-019-00332-5 (last check 2023-09-06)

[KA10] Khambete, Pramod; Athavankar, Uday: Grounded theory: An effective method for user experience design research. In: Design Thoughts 1.3, 2010, pp. 11–25 http://www.idc.iitb.ac.in/resources/dt-aug-2010/Grounded Theory.pdf (last check 2023-09-06)

[Köl10] Kölling, Michael: The Greenfoot Programming Environment. In: ACM Transactions on Computing Education, 10.4, Nov. 2010, pp. 1–21. issn: 1946-6226. DOI: 10.1145/1868358.1868361 (last check 2023-09-06) URL: https://dl.acm.org/doi/10.1145/1868358.1868361 (last check 2023-09-06)

[KS16] Tobias Kollmann, Tobias; Schmidt, Holger: Deutschland 4.0: wie die Digitale Transformation gelingt. OCLC: ocn932096511. Gabler, Wiesbaden, 2016. ISBN: 978-3-658-11981-2. URL: https://link.springer.com/book/10.1007/978-3-658-13145-6 (last check 2023-09-06)

[KSV21] Krath, Jeanine; Schürmann, Linda; Von Korflesch, Harald F.O.: Revealing the theoretical basis of gamification: A systematic review and analysis of theory in research on gamification, serious games and game-based learning. In: Computers in Human Behavior, 125, Dec. 2021, p. 106963. ISSN: 07475632. DOI: 10.1016/j.chb.2021.106963 (last check 2023-09-06) URL: https://linkinghub.elsevier.com/retrieve/pii/S0747563221002867 (last check 2023-09-06)

[LOM20] IEEE Standard for Learning Object Metadata. ISBN: 978-1-5044-7059-9. IEEE, 2020. DOI: 10.1109/IEEESTD.2020.9262118 URL: https://ieeexplore.ieee.org/document/9262118/ (last check 2023-09-06)

[Mal+19] Maldonado, Jorge; Alario-Hoyos, Carlos; Pérez-Sanagustín, Mar; Delgado-Kloos, Carlos; Alonso-Mencía, Maria; Estévez-Ayres, Iria: Self-Regulated Learning in MOOCs: Lessons Learned from a literature review. In: Educational Review, 71, Mar. 2019, pp. 1–27. DOI: 10.1080/00131911.2019.1566208 (last check 2023-09-06) https://www.tandfonline.com/doi/abs/10.1080/00131911.2019.1566208?journalCode=cedr20 (last check 2023-09-06)

[ME23] Mishra, Divyansh S.; Edwards, Stephen H.: The Programming Exercise Markup Language: Towards Reducing the Effort Needed to Use Automated Grading Tools. In: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. 2023, pp. 395–401. https://doi.org/10.1145/3545945.3569734 (last check 2023-09-06) https://dl.acm.org/doi/10.1145/3545945.3569734 (last check 2023-09-06)

[Mei+22] Meinel, Christoph; Willems, Christian; Staubitz, Thomas; Sauer, Dominic; Hagedorn, Christiane: openHPI: 10 Years of MOOCs at the Hasso Plattner Institute: 10 Jahre MOOCs am Hasso-Plattner-Institut. Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam 148. ISSN: 1613-5652. Universitätsverlag Potsdam, 2022. DOI: 10.25932/PUBLISHUP-56020 (last check 2023-09-06) URL: https://publishup.uni-potsdam.de/56020 (last check 2023-09-06)

[PB17] Priss, Uta; Borm, Karin: An Editor for the ProFormA Format for Exchanging Programming Exercises. In: Strickroth, Sven; Müller, Oliver; Striewe, Michael (Eds): Proceedings of the Third Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2017), Potsdam, Germany, October 5-6, 2017, Vol. 2015. CEUR Workshop Proceedings. CEUR-WS.org, 2017. URL: https://ceur-ws.org/Vol-2015/ABP2017_paper_10.pdf (last check 2023-09-06)

[Per+21] Perez-Ortiz, Maria; Novak, Erik; Bulathwela, Sahan; Shawe-Taylor, John: An AI-based Learning Companion Promoting Lifelong Learning Opportunities for All. Nov. 16, 2021. arXiv: 2112.01242[cs]. URL: http://arxiv.org/abs/2112.01242 (last check 2023-09-06)

[QL11] Queirós, Ricardo; Leal, José Paulo: Pexil: Programming exercises interoperability language. In: Conferência Nacional XATA: XML, aplicações e tecnologias associadas, 9. ª. ESEIG. 2011, pp. 37–48.

[Rei+19] Reiser, Paul; Borm, Karin; Feldschnieders, Dominik; Garmann, Robert ; Ludwig, Elmar; Müller, Oliver; Priss, Uta: ProFormA 2.0 – ein Austauschformat für automatisiert bewertete Programmieraufgaben und für deren Einreichungen und Feedback. In: Proceedings of the Fourth Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2019). Essen, Germany: Gesellschaft für Informatik e.V., 2019. DOI: 10.18420/ABP2019-6 (last check 2023-09-06) URL: http://dl.gi.de/handle/20.500.12116/27945 (last check 2023-09-06)

[Rot+23] Roth, Jürgen; Baum, Michael; Eilerts, Katja; Hornung, Gabriele; Trefzger, Thomas (Eds): Die Zukunft des MINT-Lernens – Band 2: Digitale Tools und Methoden für das Lehren und Lernen. Springer, Berlin, Heidelberg, 2023. ISBN: 978-3-662-66132-1. DOI: 10.1007/978-3-662-66133-8 (last check 2023-09-06) URL: https://link.springer.com/10.1007/978-3-662-66133-8 (last check 2023-09-06)

[SB02] Schell, George P.; Burns, Max: Merlot : A Repository of e-Learning Objects for Higher Education.” In: e-Service Journal, 1.2, 2002, pp. 53–64. ISSN: 1528-8234. DOI: 10.1353/esj.2002.0004 (last check 2023-09-06) URL: https://muse.jhu.edu/content/crossref/journals/eservice_journal/v001/1.2schell.html (last check 2023-09-06)

[Ser+19] Serth, Sebastian; Teusner, Ralf; Renz, Jan; Uflacker, Matthias: Evaluating Digital Worksheets with Interactive Programming Exercises for K-12 Education. In: 2019 IEEE Frontiers in Education Conference (FIE). Cincinnati, OH, USA: IEEE, Oct. 16, 2019, pp. 1–9. ISBN: 978-1-72811-746-1. DOI: 10.1109/FIE43999.2019.9028680 (last check 2023-09-06) URL: https://ieeexplore.ieee.org/document/9028680 (last check 2023-09-06)

[Ser+21] Serth, Sebastian; Staubitz, Thomas; Teusner, Ralf; Meinel, Christoph: CodeOcean and CodeHarbor: Auto-Grader and Code Repository.” In: SPLICE 2021 workshop CS Education Infrastructure for All III: From Ideas to Practice. 52nd ACM Technical Symposium on Computer Science Education. Virtual Event, Mar. 15, 2021, p. 5. URL: https://cssplice.github.io/SIGCSE21/proc/SPLICE2021_SIGCSE_paper_13.pdf (last check 2023-09-06)

[Spi12] Spinellis, Diomidis: Git. In: IEEE Software 29.3 (May 2012), pp. 100–101. ISSN: 0740-7459. DOI: 10.1109/MS.2012.61 (last check 2023-09-06) URL: http://ieeexplore.ieee.org/document/6188603/ (last check 2023-09-06)

[SRL15] Spagnoletti, Paolo; Resca, Andrea; Lee, Gwanhoo: A design theory for digital platforms supporting online communities: a multiple case study. In: Journal of Information technology 30.4, 2015, pp. 364–380. https://doi.org/10.1057/jit.2014.37 https://journals.sagepub.com/doi/abs/10.1057/jit.2014.37 (last check 2023-09-06)

[Sta+15] Staubitz, Thomas; Klement, Hauke; Renz, Jan; Teusner, Ralf; Meinel, Christoph: Towards practical programming exercises and automated assessment in Massive Open Online Courses.” In: IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE). 2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE). Zhuhai, China: IEEE, Dec. 2015, pp. 23–30. ISBN: 978-1-4673-9226-6. DOI: 10.1109/TALE.2015.7386010 (last check 2023-09-06) URL: http://ieeexplore.ieee.org/document/7386010/ (last check 2023-09-06)

[Sta+16] Staubitz, Thomas; Klement, Hauke; Teusner, Ralf; Renz, Jan; Meinel, Christoph: CodeOcean - A versatile platform for practical programming excercises in online environments. In: 2016 IEEE Global Engineering Education Conference (EDUCON). Abu Dhabi: IEEE, Apr. 2016, pp. 314–323. ISBN: 978-1-4673-8633-3. DOI: 10.1109/EDUCON.2016.7474573 (last check 2023-09-06) URL: http://ieeexplore.ieee.org/document/7474573/ (last check 2023-09-06)

[STM17] Staubitz, Thomas; Teusner, Ralf; Meinel, Christoph: Towards a repository for open auto-gradable programming exercises. In: 2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE). Hong Kong: IEEE, Dec. 2017, pp. 66–73. ISBN: 978-1-5386-0900-2. DOI: 10.1109/TALE.2017.8252306 (last check 2023-09-06) URL: http://ieeexplore.ieee.org/document/8252306/ (last check 2023-09-06)

[STM21] Serth, Sebastian; Teusner, Ralf; Meinel, Christoph: Impact of Contextual Tips for Auto-Gradable Programming Exercises in MOOCs. In: Proceedings of the Eighth ACM Conference on Learning @ Scale. L@S '21: Eighth (2021) ACM Conference on Learning @ Scale. Virtual Event, Germany: ACM, June 8, 2021, pp. 307–310. ISBN: 978-1-4503-8215-1. DOI: 10.1145/3430895.3460166 (last check 2023-09-06) URL: https://dl.acm.org/doi/10.1145/3430895.3460166 (last check 2023-09-06)

[Str+15] Strickroth, Sven; Striewe, Michael; Müller, Oliver; Priss, Uta; Becker, Sebastian; Rod, Oliver; Garmann, Robert; Bott, Oliver J.; Pinkwart, Niels: ProFormA: An XML-based exchange format for programming tasks. In: eleed 11, 2015. urn:nbn:de:0009-5-41389 (last check 2023-09-06) https://www.eleed.de/archive/11/4138 (last check 2023-09-06)

[WCH20] Wang, Shaowei; Chen, Tse-Hsun; Hassan, Ahmed E.: How Do Users Revise Answers on Technical Q&A Websites? A Case Study on Stack Overflow. In: IEEE Transactions on Software Engineering 46.9, Sept. 1, 2020, pp. 1024–1038. ISSN: 0098-5589, 1939-3520, 2326-3881. DOI: 10.1109/TSE.2018.2874470 (last check 2023-09-06) URL: https://ieeexplore.ieee.org/document/8485395/ (last check 2023-09-06)

[Wie99] Wiegers, Karl E.: First Things First: Prioritizing Requirements. In: Software Development, 1999. URL: https://api.semanticscholar.org/CorpusID:166830711 (last check 2023-09-06)