Increasing the Interactivity in Software Engineering MOOCs - A Case Study

Stephan Krusche
Technische Universität München
krusche@in.tum.de

Andreas Seitz
Technische Universität München
seitz@in.tum.de

Abstract

MOOCs differ from traditional university courses: instructors do not know the learners who have a diverse background and cannot talk to them in person due to the worldwide distribution. This has a decisive influence on the interactivity of teaching and the learning success in online courses. While typical online exercises such as multiple choice quizzes are interactive, they only stimulate basic cognitive skills and do not reflect software engineering working practices such as programming or testing. However, the application of knowledge in practical and realistic exercises is especially important in software engineering education.

In this paper, we present an approach to increase the interactivity in software engineering MOOCs. Our interactive learning approach focuses on a variety of practical and realistic exercises, such as analyzing, designing, modeling, programming, testing, and delivering software stimulating all cognitive skills. Semi-automatic feedback provides guidance and allows reflection on the learned theory. We applied this approach in the MOOC software engineering essentials SEECx on the edX platform. Since the beginning of the course, more than 15,000 learners from more than 160 countries have enrolled. We describe the design of the course and explain how its interactivity affects the learning success.

1. Introduction

Massive open online courses (MOOCs) offer new possibilities. Learners can participate in courses of interest with higher flexibility and are not bound to schedules, locations, or university costs. Universities can reach larger audiences outside of organizational boundaries [1]. With the help of the internet, education can be brought to countries with lower social and educational standards. MOOCs are becoming increasingly popular and more and more universities offer them in professional programs and micro masters.

However, it is not sufficient to replicate standard university lecture courses to design a MOOC [2]. It is not possible to promote an active learning process in online courses, only by broadcasting video recordings of lectures [1]. This poses the risk of students becoming passive and unmotivated. While embedding discussions and offering multiple choice quizzes can increase the interactivity, they typically do not stimulate higher cognitive skills [3, 4]. Most activities in existing online courses focus on lower cognitive skills, testing only the degree of understanding of the main concepts and forcing the learner to face recurrent mistakes [4].

Due to these limitations, simple software engineering MOOCs do not yet support the acquisition of higher cognitive skills such as applying, analyzing, evaluating and creating. However, software engineering is an activity that requires collaboration and practical application of knowledge [5, 6], in particular interaction and collaboration [7]. Video lectures, simple quizzes and reading material are not sufficient [8] because they do not reflect working practices such as programming, modeling and testing. The creation of new software depends on higher cognitive skills including application of knowledge, analysis, evaluation and creation.

Active learning engages course participants in the learning process by involving them into learning activities, e.g. into problem solving. It is used in more and more university courses [9], positively influences knowledge transfer and learners' performance, and leads to an improved learning experience [10]. However, instructors need to guide learners in these activities to facilitate learning and prevent misconceptions [11]. With several thousand learners in MOOCs, it is challenging to guide all learners.

In summary, MOOCs face the following three problems:

(P1) Lecture recordings are too static
(P2) Quizzes only stimulate basic cognitive skills
(P3) Guidance is challenging to scale

We present an interactive learning approach for software engineering MOOCs that addresses these

URI: https://hdl.handle.net/10125/60197
ISBN: 978-0-9981331-2-6

problems by using a variety of practical exercises. These interactive exercises go beyond multiple choice questions and stimulate cognitive skill acquisition on all levels. They provide individual guidance to learners through feedback and scale to a large number of students. The focus of the paper is **not** a discussion whether MOOCs can or will replace university courses.

The remainder of the paper is structured as follows: Section 2 describes active learning, Bloom's taxonomy for cognitive skills and existing definitions for interactivity as the foundations of this paper. Section 3 shows our interactive learning approach with different exercise types that stimulate all cognitive skills. In Section 4, we present a case study, in which we applied this approach in a software engineering online course. Section 5 discusses our learnings in this course and describes best practices for other instructors who want to adopt our approach. Section 6 presents related work and Section 7 concludes the paper.

2. Foundations

"MOOCs mostly replicated the standard lecture, an uninspiring teaching style but one with which the computer scientists were most familiar" [2].

2.1. Bloom's Taxonomy

Bloom developed a framework to classify expectations of what students should learn as the result of an instruction [12]. It serves as common language about learning goals. An example would be: "learners are able to describe the waterfall model". Bloom classified six major categories of cognitive processes ordered by their complexity from lowest to highest: knowledge, comprehension, application, analysis, synthesis and evaluation.

Constructive alignment proposes to align learning goals with activities and assessment. It was introduced by John Biggs [13] and is derived from constructivism and curriculum theory [14]. Biggs refers to the basic idea of constructivism that learners construct their own learning through learning activities, instead of passively receiving knowledge from the instructor. All components in the learning system - the learning goals, the learning activities, and the assessment tasks - are aligned to each other.

2.2. Active Learning

Active learning led to improved learning experiences on different cognitive skill levels of Bloom's taxonomy in university courses. It emphasizes on developing skills through active participation and engagement in activities. It moves away from teacher-centered approaches, where teachers instruct and learners listen passively, to a more learner-centered approach, where learners play an active role. Bonwell and Eison define active learning as "anything that involves students in doing things and thinking about the things they are doing" [15]. It requires learners to regularly assess their own problem-solving skills and their understanding of the taught concepts [16]. Brophy and Good identify four main premises of active learning [17]:

1. Learners construct their own meanings
2. New learning builds on prior knowledge
3. Learning is enhanced by social interaction
4. Learning develops through 'authentic' tasks

Prince [10] and Michael [16] found support for all forms of examined active learning in their studies. They concluded that active learning improves learning outcomes compared to passive learning approaches.

2.3. Interactivity

When an instructor says 'I am trying to make my classes more interactive', the meaning of interactive seems clearly intuitive, however an agreed-upon definition of interactivity is hard to find. The term is used in the context of various fields, such as communication, advertising, websites, internet and education to name a few [18]. Since Rafaeli's statement "Interactivity is an underdefined concept" [19], a number of attempts have been made to define the concept of interactivity in its different contexts leading to the inconsistent use of the term [20].

The term interactivity is rooted in the term interaction. The Cambridge dictionary defines interaction as "an occasion when two or more people or things communicate with or react to each other". Steffensen differentiates between interaction and interactivity: "[...] interaction captures a relation of dependence between separable systems, interactivity explores their bidirectional coupling" [21].

Jones and Gerard propose that all social interaction is goal-oriented [22]. They distinguish four different types of interactions according to their influence on the interaction partners:

1. **Pseudo interaction:** A sequence of actions that follow predefined patterns. The actions of an involved participant are not intended to be interpreted by the other participant.
2. **Asymmetrical interaction:** One participant follows his or her intentions while another party reacts complementarily to the previous actions.

3. **Reactive interaction:** The involved parties do not interpret the intentions of the actions of the other party and react in an isolated form.
4. **Interdependent symmetrical interaction:** Aligning one's action to the own intentions while considering the intentions of the others in a reciprocal fashion.

Similarly, Rafaeli argues that interactivity is best defined by considering the degree of responsiveness [19]. He recognizes three levels of communication. Two-way (non-interactive), reactive (quasi-interactive) and fully interactive communication. For an interaction to be classified as two-way communication, messages must flow bilaterally. If the messages cohere with previous messages, the interaction is at least reactive or quasi-interactive. The third level, full interactivity, adds a reference to the content, nature or presence of earlier references. Rafaeli defines interactivity as "an expression of the extent that in a given series of communication exchanges, any third (or later) transmission (or message) is related to the degree to which previous exchanges referred to even earlier transmissions." Domagk, Schwartz and Plass [23] and Johnson et al. [24] identified two fundamental conditions common in interactivity research: (1) At least two participants interact with each other. (2) Actions of these participants are reciprocal[1] and responsive[2].

Yacci examined interactivity in the context of distance learning and computer-based teaching and identified major interactivity attributes [25]. Interactivity is a message loop, whose messages must be mutually coherent. Instructional interactivity occurs from the learner's perspective. Its outputs are content learning and affective benefits.

3. Interactive Learning Approach

Interactive learning combines theory and practice into interactive classes with multiple, small iterations of theory, example, exercise, solution and reflection [26]. It is based on active, computer based and experiential learning [27] and focuses on immediate feedback to provide guidance and improve the learning experience in large university classes. Hands-on activities in class increase motivation and engagement and allow continuous assessment.

Instructors teach and exercise small chunks of knowledge in short cycles. Learners reflect and increase their knowledge incrementally. This approach expects active participation and the use of computers. Instructors provide guidance to prevent misconceptions and to facilitate the learning process. Considering the existing definitions of interactivity in literature, we define interactivity in terms of MOOCs as follows:

> **Interactivity** means a reciprocal and responsive communication which is addressed in a context-sensitive way by the learning management system as a whole, so that learners can construct meaningful knowledge increments.

This definition focuses on automatic feedback using a learning management system, but also allows instructors, teaching assistants (TAs) and peer learners to respond to communication initiated by learners. The purpose of semi-automatic, context-sensitive feedback is guidance and reflection to prevent misconceptions. Communication in MOOCs can be initiated by instructors, e.g. when motivating students using course announcements or emails. It can also be initiated by learners when they ask questions or face problems.

Figure 1 shows the idea of continuous interactive learning[3] that we adapted from Scrum [28] and experiential learning [27]. The course syllabus consists of high-level learning goals that are typically structured into lectures giving them meaningful boundaries in the learning activities. Each lecture consists of more detailed learning goals. The instructor teaches each learning goal in a learning sprint, a cycle that starts with theory and examples. Learners then work on an exercise and receive immediate feedback building a second small cycle that allows them to iteratively improve their solution to the exercise. After the exercise, the instructor stimulates reflection so that students relate their experience in the exercise with the taught theory. This closes the cycle of the learning sprint and leads to a learning gain, which we call knowledge increment, with respect to the taught learning goal.

Examples and exercises are important elements and play a central role in the early phases of cognitive skill acquisition [30]. Carefully developed and integrated examples increase the learning outcome [31, 32]. Dynamic exercises with context-sensitive feedback solve P1 and enable a richer learning experience. Continuous interactive learning focuses on the application of knowledge in a variety of exercise types, e.g. programming and modeling exercises with instant feedback. This supports the cognitive skill acquisition [30] on all levels of Bloom's taxonomy shown in Figure 2 and solves P2. Multiple choice

[1] Reciprocal means that actions of one participant trigger responses from the other and lead to change in the first.

[2] Responsive means that actions and reactions are related and sustain the continuity of the interaction.

[3] We first integrated continuous interactive learning into a classroom course on games development [29] in 2016.

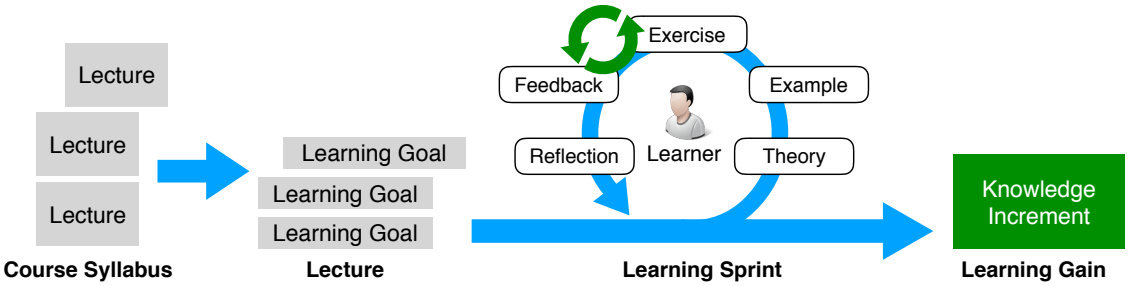

Figure 1. Continuous interactive learning embedded into a course consisting of lectures, each with a number of learning goals. Each learning goal is taught in a learning sprint through theory, example, exercise, feedback and reflection and leads to a new knowledge increment (adapted from Scrum [28] and experiential learning [27]).

quizzes focus on the first two levels, programming and modeling address the four higher and more complex levels[4]. The sample solution and the instant and context-sensitive feedback in the end of the cycle provide guidance. If feedback can be generated automatically (e.g. through test cases in programming exercises) or by other learners (e.g. through peer review in modeling exercises), it is scalable to a large number of learners and solves P3. This might require a higher effort for the creation of exercises, but reduces the effort during the conduction of the course.

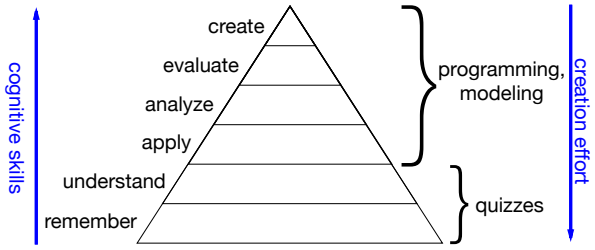

Figure 2. Mapping of exercises to cognitive skills

Depending on the type of the exercise, the learner's submission is automatically assessed or a manual review of the solution is carried out, involving other learners (peer review). The assessment leads to manual or automatic feedback which needs to be context-sensitive to be meaningful. Learners can use it to improve and submit another solution. Feedback motivates learners and allows them to reflect their learning progress.

We developed the concept of interactive instructions that visually explain the problem to be solved. Such instructions are dynamic and provide continuous and granular feedback with self-updating elements, e.g. tasks and UML diagrams with respect to the structure of the exercise. These elements respond to the interaction of learners by changing their color from red to green to indicate that the solution is correct as shown in Figure 3.

An interactive task is dynamically updated based on the learner's progress. It is associated with the assessment, e.g. a test or a peer review. An interactive task in a programming exercise is completed when all associated tests are passing. This association allows to refer the learner to the problem in the source code when the user clicks on the unfulfilled, red task. After completion, the task is displayed in green and ticked off.

An interactive diagram is dynamically created and updated based on the learner's progress. It consists of multiple elements, such as classes, attributes, or methods in a UML class diagram. A diagram element can be associated with an assessment and a source file. The implementation of a method is e.g. associated with its method name in the class diagram. Based on the test results, the color of this diagram element changes to green, if all associated tests succeed, or to red, if at least one test fails. Learners can immediately identify which parts of their exercise are correctly solved and which are still incorrect. In addition, the associated feedback includes context-sensitive information, why a test failed and refers to the theory learned in videos and handouts.

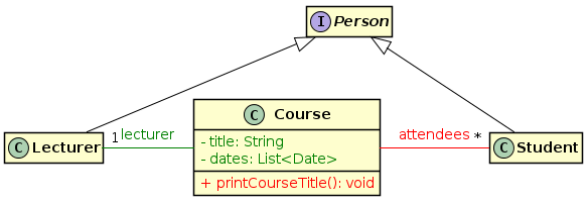

Figure 3. Interactive assignments (with UML diagrams) provide immediate feedback to learners about the correctness (red, green) of their solution

[4]While it might be possible to create multiple choice tests for higher cognitive skills, it is difficult and does not reflect software engineering working practices: software engineers do not answer multiple choice questions in their daily work when applying, analyzing, evaluating or creating something. Williams and Haladyna recommend to limit multiple choice tests to lower cognitive skills [3].

4. Case Study

We describe the application of interactive learning for the design, creation, implementation and execution of the MOOC Software Engineering Essentials (SEECx) that we offer on edX[5]. Our goal was to make the course as interactive as possible. The MOOC was launched in May 2017 as instructor-paced course over 9 weeks. We repeated the course in October 2017. Since May 2018, the course is available as self-paced course. In all three instances, 15,276 students enrolled in total until now.

It is an intensive course with interactive exercises that go beyond the learning experience of existing software engineering MOOCs. It has the following learning goals: Learners get to know methods and techniques to develop software for different domains and platforms using agile techniques in the context of change. Starting from a problem statement, we teach the participants how to analyze requirements and transform them into models using textual analysis. They model multiple representations of the system consistently, understand and identify patterns. They map models to source code, integrate it into an app and deliver this app using build and release management techniques.

4.1. Course Structure

4 instructors and 7 TAs organized the course. It includes 8 sections (comparable to lectures) covering 8 major topics: project organization and management, software configuration management, object oriented programming, requirements analysis, system design, object design, testing, build and release management. All sections are decomposed into smaller topics and consists of 3 to 5 units, each covering a concrete learning goal. The whole course includes 34 units.

A unit includes a video in which the theory of the topic is taught and an example is shown, followed by a small exercise with feedback and a summary to reflect on the learned concepts. The duration of the videos ranges from 3 to 15 minutes (mean: 8.2 min). The videos are kept short in order to enable the learners to apply the newly acquired knowledge in practice in the exercises. In addition to slide-based lecture videos, we added short clips with animations in an explanation style and real world scenes into the video to make them more entertaining and rich in variety. Such videos make the thinking process visible and support cognitive apprenticeship [33]. After each unit, there is a quiz to assess whether the learners can remember and explain the learned concepts (level 1 and 2 in Bloom's taxonomy). Learners get immediate feedback on their response and test their newly acquired knowledge. Learners can try each quiz two times in the course, so even if they failed initially, they can have another look at the video and then score the full points in the assessment. This keeps the learners motivated.

Each section also includes programming and modeling exercises which focus on higher cognitive skills. They assess if learners can apply the previously obtained knowledge, analyze a problem, evaluate different solution strategies and create new solutions to given problems (level 3 - 6 in Bloom's taxonomy).

In order to pass the course, learners have to achieve at least 60 % of all available points (400). By participating in the interactive exercises, learners can earn up to 60 % of the total points (240), 30 points for each section. At the end of the course, students can participate in a final assessment which accounts the remaining 40 % of the total points (160).

4.2. Participation

In the following, we want to show how learners participated in the first instance (instructor-paced) of the course between May and July 2017. Figure 4 shows that in the beginning, our course had 786 active learners[6] and 620 learners who scored in at least one exercise (in section 1). In the last section 8, 47 % of the learners were still active and 15 % scored in exercises. Between section 1 and 3, there was a drop of 33 % of the active learners and 68 % of the learners who scored. We attribute this to the increased complexity of the exercises. In addition, multiple instructors of other software engineering courses initially participated in our course due to advertisement on typical software engineering mailing lists such as SE World and SIGCSE. They tried out some videos and exercises, but were not interested in completing the course. Towards the end of the course, the dropout rate decreased.

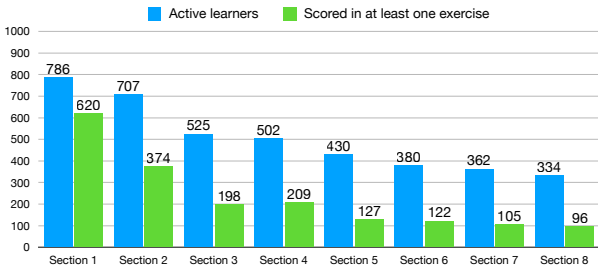

Figure 4. Number of active learners (blue) and learners who scored in at least one exercise (green) for each section in the first instructor-paced instance.

[5] www1.in.tum.de/seecx or www.edx.org/course/software-engineering-essentials

[6] Learners who visited at least one page in the course content.

4.3. Interactive Exercises

We use different types of exercises to make the course interactive and rich in variety following the learning goals. We used multiple choice, text input and drag & drop questions to support learning goals on level 1 and 2 of Bloom's taxonomy. In addition, we integrated interactive programming and modeling exercises.

We based all programming and modeling exercises on a common problem statement about the "University App", which we also used for examples in the videos. This allows the learners to recognize relationships between the topics (e.g. between the requirements and system design) and makes it easier to understand the context of the problem. In the following, we explain the different exercise types in more details.

4.3.1. Interactive Programming Exercises

We use an automated assessment system ArTEMiS for programming exercises based on version control and continuous integration [34]. Learners submit their exercise solution and receive immediate feedback through structural and behavioral tests. Learners can use this feedback to iteratively improve their solution. ArTEMiS automatically assesses the learners' submissions and provides context-sensitive feedback on their submissions.

The online editor of ArTEMiS includes assignments using interactive tasks and interactive diagrams. After each submission through the **Commit & Run Tests** button, the code of the learner is assessed. The result is shown immediately and the interactive tasks and diagrams are updated accordingly. In addition, learners can see detailed, individual feedback why their solution is wrong by clicking on the result. This helps to identify, which tasks the learners have already solved and which parts of their program does not work as expected.

4.3.2. Modeling Exercises

The ability to understand and create models is an important learning goal for software engineers. Therefore, modeling is an essential part of our course. However, it is difficult to automatically correct models because there are different correct solutions. Modeling is a creative activity and we do not want to limit the creative thinking processes of students [35]. One learning goal of the course is that participants can review models given a set of quality criteria. Therefore, we used the concept of peer reviews consisting of the following steps: (1) upload response, (2) learn to assess responses, and (3) assess peers.

Learners first create a solution to a given problem and upload it, then they review sample solutions towards a given set of criteria to learn how to assess other solutions. Finally, they assess multiple other learners' solution, so that each model is evaluated by at least three reviewers (other learners). The final score is the average of three reviews. Learners receive valuable feedback about their models and can improve their modeling skills in the future. While edX's peer review system does not allow to improve the model according to the feedback, we are working on an interactive system that allows learners to iterate on their model solution.

While peer reviews lead to additional effort for learners, they stimulate the acquisition of higher cognitive skills: by assessing other solutions, students reflect about alternative solution approaches and evaluate if they are correct with respect to the given problem statement. This is particularly helpful, but it should be used carefully to not overload the learners. We include two peer review exercises in the course, one on creating low-fidelity mockups for the university app, and another one on creating an analysis object model.

4.3.3. Project Work

In addition, we also offer project work which allows the students to experience the full software engineering process from analysis over design to implementation, testing and delivery. A second problem statements allows learners to apply and transfer their knowledge to a different problem domain. The exercises in the project work focus on the upper two cognitive skill levels in Bloom's taxonomy where learners should create and evaluate a new solution to a problem. The project work starts in the fourth section and allows the learners to evaluate how their own decisions, e.g. in the requirements analysis, influence the system design and the implementation.

Examples of project work exercises are the analysis of the problem domain, the design of the software architecture, sketches of user interfaces up to the implementation, testing and delivery of a small app. We cannot assess such exercises automatically because we do not want to limit the creativity of learners. We motivated the learners to discuss their solutions with us and each other to get feedback on their solutions and the TAs provided timely feedback. Project work is optional for learners, they can pass the course without active participation. Nonetheless, we highly recommend to participate and give the learners the opportunity to deepen their knowledge and gain practical experience.

4.4. Communication

Guàrdia describes that designing a MOOC is to "[s]et up a space to foster social interaction and frequent contact between the learners." This results in our approach using a chat for instant and direct communication instead of discussion forums to further improve interactivity between course participants and instructors. Many existing MOOCs rely on discussion forums, which are limited in interactivity. We promote the exchange with and between learners. Both instructors and TAs can be approached directly in the chat, learners can provide feedback and ask for help.

We use Slack as instant messaging service because it has a lower entry barrier than discussion forums. Learners can get in touch with each other and write direct messages to instructors and TAs in case they need help. They ask questions more easily without having to pay attention to the exact wording and phrasing. We add repeating questions to a question and answer page. In total, 754 learners regularly used Slack to get in touch and already sent 14,282 messages. The #questions, #general and #feedback channels were the most important ones. In the #questions channel, learners asked question, in #feedback, they stated how to improve the learning material. Instructors and TAs answer questions within one working day to keep the interactivity high.

Clear communication of learning goals, expectations and deadlines is important. The course description clarifies what the learners can expect and what they have to accomplish to pass the course.

4.5. Survey Results

We evaluated our approach using two surveys, an entry and an exit survey. The entry survey covered the background and motivation of learners. 83 % of the participants are male, 17 % are female. The median learner age is 28 and most learners are between 20 and 40 years old. 3 % are pupils at school, 31 % are students in university or college, 51 % are employees in a company and 15 % are unemployed or searching for a job. 51 % have already participated in another software engineering course before. 86 % have already participated in another programming course before. Regarding motivation, 27 % need the certificate of the course, 96 % are interested in the topic, 84 % need to know the concepts taught in the course for their career and 66 % assume the course is fun.

The exit survey asked about the opinion on the interactivity of the course. 67 learners took part in both surveys and allowed us to compare their existing knowledge and motivation with their results.

We asked the learners how the different components of the course contributed to their learning. Figure 5 shows that all components of the contribute. Videos play an essential role in the transfer of knowledge, as they impart the theoretical content to learners. Learners indicate that the contribution of programming exercises is higher (82 %) to their learning than in quizzes (62 %). Modeling exercises also have a high contribution with 65 %. We attribute the smaller numbers of modeling exercises to the higher complexity that peer reviews entail.

We also evaluated the helpfulness of immediate feedback in programming exercises, as well as the usefulness of Slack. 83 % of the respondents agree that feedback in programming exercises is helpful (left diagram in Figure 6). 57 % of the participants agree to prefer Slack over traditional discussion forums (right diagram in Figure 6).

These results represent first anecdotal evidence. Further studies are required to evaluate the approach. Due to the small amount of participants in the exit survey, selection bias is a threat to validity. The participants opinion might not be generalizable. Nevertheless, the first survey results show that our approach improves the learning experience: there are ways to make MOOCs more interactive and to reduce the gap to interactive classroom courses.

5. Discussion

This section discusses both the approach and the experience we have gained in developing and carrying out a software engineering MOOC. We first discuss the learnings before we derive best practices that can be useful for other MOOC instructors.

We faced a trade-off between too detailed and superficial feedback. Too detailed feedback has the risk of including the actual solution which might prevent learning. However, if the feedback is too superficial, it does not help learners and demotivates them. Especially at the end of the course, when the exercises became more demanding, less learners participated. As the difficulty increases, so does the amount of time spent by the learners. Many learners are not willing to invest this extra time in solving difficult exercises.

We experienced that learners who have completed the course, look back positively on the varied exercises, even if they were sometimes more demanding. One learner stated: "The lectures include both practical and theoretical videos and explanatory test cases. The final exam, quizzes, peer review projects and programming

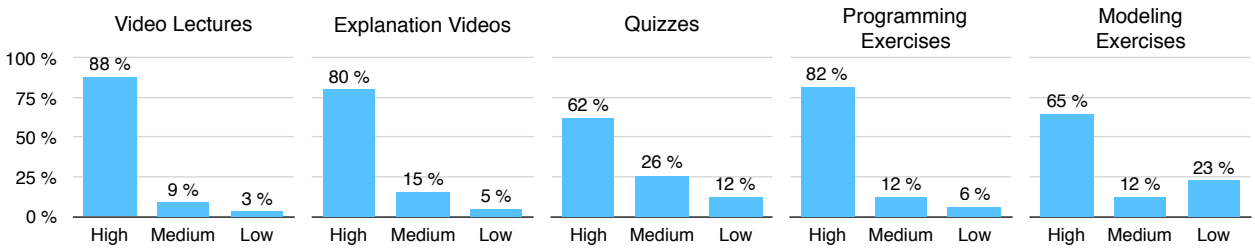

Figure 5. Answer distribution of learners about the contribution to their learning in the course.

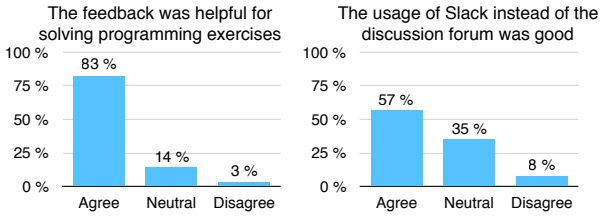

Figure 6. Helpfulness of feedback (left), Slack over discussion forums (right)

exercises are well thought and help in understanding and reviewing the core concepts of each week.“ This statement confirms the right mix of theory and practice contributing to learning success.

5.1. Learnings

We initially used a rather serious tone in the communication with the learners, but during the course we have moved further away from it. We approached the learners personally and asked them about their experiences via mail and Slack. This removes the barrier between instructors and learners and facilitates interactivity. We made clear why it is important for learners to participate in the interactive exercises. The typical MOOC learner is not accustomed to this multitude of varied exercises. Therefore, we communicate clearly from the beginning that our MOOC is an intensive course in which we expect active participation in exercises and discussions on Slack.

As a consequence of the learners' feedback, we have changed several aspects already during the first run of the MOOC. We extended the time for all exercises from one to two weeks to allow all participants to complete the exercises. This alleviated the stress factor, in particular for learners who worked in a full-time job and had families. In a further step, we increased the number of attempts for all quizzes from 1 to 2. This gives learners the opportunity to study the theory again after a wrong answer and motivates them. They can improve their knowledge and try the quiz a second time. It is important that learners receive feedback on their given answers, regardless of whether they are correct or incorrect. Only then, they can reflect on the theory again. We want learners to internalize the acquired knowledge. The feedback must be motivating and can include a sense of humor.

We also introduced a question and answers page and summary pages for each section. Many questions reached us multiple times via Slack, so we decided to collect the most frequently asked questions on a separate FAQ page. The section summary ensures that learners fully grasp the learning objectives of each section and connect the units with each other. This allows learners to reflect on the contents of the whole section again.

5.2. Best Practices

Make sure that all exercises are aligned with the learning goals in terms of constructive alignment. Double check the consistency of all exercises, especially with regard to difficulty and comprehensibility. Plan the learning goals before the production of the videos and adapt the exercises accordingly. Use the same working example throughout the course, ideally by using a problem statement that relates to the personal experience of the learners. Explain learners what they did wrong and why they did it wrong in the exercises using context-sensitive feedback. However, the feedback should not directly contain the solution, instead it should explain aspects of the theory related to the exercise. Learners have to come up with the correct solution themselves.

Be open to change and listen to the wishes and preferences of your learners. With small iteration cycles in interactive MOOCs, changes and wishes can be addressed easily. Small iteration cycles are only possible if the length of videos is limited. Address learners personally in videos to overcome the barriers in the beginning of the unit. Include animations and real world scenes in the videos and do not only rely on lecture style slides with too many text and bullet points. In programming exercises, write test cases to assess the behavior of the learners' solution and make sure the feedback in the assertions of the tests is understandable.

6. Related Work

In [36] and [37], we describe the application of interactive learning in the classroom. In the following, we focus on online courses.

Alario-Hoyos et al. describe their MOOC for introduction to programming with Java [4]. The authors state that they designed the course to enhance the learners' activity with learning contents. They found that traditional multiple choice quizzes only assess the two lower levels of Bloom's taxonomy. In addition to quizzes, they also rely on peer review and programming exercises, which they carried out with the help of the external tools *Blockly*, *Codeboard* and *Greenfoot*.

Daun et al. integrate conceptional modeling into their MOOC [38] by using ambiguous exercises and sketching multiple solutions in brief whiteboard-style videos. They state that it enables the students to assess their own solutions and fulfills their educational needs. In contrast, we use peer reviews to allow the students to receive feedback on their model solutions.

Kloos et al. use MOOCs as out-of-class-activities in addition to normal interactive classes following the inverted classroom approach [39]. They argue that more time can then be spent in interactive participation and on-site interaction leading to more effective learning. This confirms that MOOCs need to become more interactive in order to achieve a better learning experience for learners.

Krugel et al. describe their experiences on designing an interactive MOOC about object-oriented programming [40]. In addition to traditional quizzes, the authors rely on interactive programming exercises in order to put the theoretical knowledge into practice. They use various external tools, such as *SVG-edit*, *trinket*, *Java-Tutor*, and *Codeboard*. Kolas et al. introduce interactive modules [41], which are either videos or presentations to motivate and activate learners. In contrast to our approach, the authors do not focus on exercises.

Gruenewald et al. focus on the challenge of conducting interactive programming lectures as MOOCs [42]. They integrate active experimentation and relate to concrete experience. Existing literature shows that interactivity plays a role in online courses. Nevertheless, as far as we know, no one has yet defined what interactivity means in the context of MOOCs.

7. Conclusion

MOOCs can complement university courses and provide education to places that would otherwise have no access. However, they also face challenges in terms of interactivity, the stimulation of all cognitive skills and the provision of context-sensitive guidance to a large number of learners. We introduced an interactivity model for MOOCs that addresses these challenges. It includes a variety of practical exercises, in particular programming and modeling, which are typical learning goals in software engineering.

Learners can participate multiple times in exercises and learn from their failures and the context-sensitive feedback. We found first evidence that this improves the learning success. The different exercise types, the division into small learning sprints, direct communication and immediate feedback increase the interactivity and improve the learning experience.

In the future, we want to integrate semi-automatic assessment of modeling exercises using machine learning. It allows multiple solutions to be assessed as correct and does not limit the creative thinking of students. Using this approach, we can integrate more modeling exercises. In addition, we plan to integrate code reviews as described in [43]. It is important that students do not only learn to write correct programs, the code also needs to be understandable.

References

[1] T. Daradoumis, R. Bassi, F. Xhafa, and S. Caballé, "A review on massive e-learning (MOOC) design, delivery and assessment," in *8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 208–213, IEEE, 2013.

[2] R. Ubell, "How the pioneers of the mooc got it wrong," *IEEE Spectrum*, 2017.

[3] R. Williams and T. Haladyna, "Logical operations for generating intended questions (logiq): A typology for higher level test items," *A technology for test-item writing*, pp. 161–186, 1982.

[4] C. Alario-Hoyos, C. Kloos, I. Estévez-Ayres, C. Fernández-Panadero, J. Blasco, S. Pastrana, and J. Villena-Román, "Interactive activities: the key to learning programming with MOOCs," *Proceedings of the European Stakeholder Summit on Experiences and Best Practices in and Around MOOCs*, 2016.

[5] T. Connolly, M. Stansfield, and T. Hainey, "An application of games-based learning within software engineering," *British Journal of Educational Technology*, vol. 38, no. 3, pp. 416–428, 2007.

[6] D. Shaffer, "Pedagogical praxis: The professions as models for postindustrial education," *Teachers College Record*, vol. 106, no. 7, pp. 1401–1421, 2004.

[7] J. Whitehead, "Collaboration in software engineering: A roadmap," *FOSE*, vol. 7, no. 2007, pp. 214–225, 2007.

[8] T. Staubitz *et al.*, "Towards Practical Programming Exercises and Automated Assessment in Massive Open Online Courses," in *International Conference on Teaching, Assessment, and Learning for Engineering*, pp. 23–30, 2015.

[9] S. Freeman, S. Eddy, M. McDonough, M. Smith, N. Okoroafor, H. Jordt, and M. Wenderoth, "Active

learning increases student performance in science, engineering, and mathematics," *Proceedings of the National Academy of Sciences*, vol. 111, no. 23, pp. 8410–8415, 2014.

[10] M. Prince, "Does active learning work? a review of the research," *Journal of Engineering Education*, vol. 93, no. 4, pp. 223–231, 2004.

[11] P. Kirschner, J. Sweller, and R. Clark, "Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching," *Educational psychologist*, vol. 41, no. 2, pp. 75–86, 2006.

[12] B. Bloom, M. Engelhart, E. Furst, W. Hill, and D. Krathwohl, "Taxonomy of educational objectives: The classification of educational goals," 1956.

[13] J. Biggs, "Aligning teaching and assessing to course objectives," *Teaching and learning in higher education: New trends and innovations*, vol. 2, pp. 13–17, 2003.

[14] F. Marton and S. Booth, *Learning and awareness*. Psychology, 1997.

[15] C. Bonwell and J. Eison, *Active Learning: Creating Excitement in the Classroom*. ASHE-ERIC Higher Education Reports, 1991.

[16] J. Michael, "Where's the evidence that active learning works?," *Advances in physiology education*, vol. 30, no. 4, pp. 159–167, 2006.

[17] T. Good and J. Brophy, *Looking in classrooms*. Harper & Row, 1987.

[18] P. Lowry, N. Romano, J. Jenkins, and R. Guthrie, "The CMC interactivity model: How interactivity enhances communication quality and process satisfaction in lean-media groups," *Journal of Management Information Systems*, vol. 26, no. 1, pp. 155–196, 2009.

[19] S. Rafaeli, "From new media to communication," *Sage annual review of communication research: Advancing communication science*, vol. 16, pp. 110–134, 1988.

[20] R. Mayer, *The Cambridge handbook of multimedia learning*. Cambridge university press, 2005.

[21] S. Steffensen, "Human interactivity: problem-solving, solution-probing and verbal patterns in the wild," in *Cognition beyond the brain*, pp. 195–221, 2013.

[22] E. Jones and H. Gerard, "Foundations of social psychology," 1967.

[23] S. Domagk, R. Schwartz, and J. Plass, "Interactivity in multimedia learning: An integrated model," *Computers in Human Behavior*, vol. 26, no. 5, 2010.

[24] G. Johnson, G. Bruner, and A. Kumar, "Interactivity and its facets revisited: Theory and empirical test," *Journal of Advertising*, vol. 35, no. 4, 2006.

[25] M. Yacci, "Interactivity demystified: A structural definition for distance education and intelligent CBT," *Educational Technology*, vol. 40, no. 4, pp. 5–16, 2000.

[26] S. Krusche, A. Seitz, J. Börstler, and B. Bruegge, "Interactive learning: Increasing student participation through shorter exercise cycles," in *Proceedings of the 19th Australasian Computing Education Conference*, pp. 17–26, ACM, 2017.

[27] D. Kolb, *Experiential learning: Experience as the source of learning and development*, vol. 1. Prentice Hall, 1984.

[28] K. Schwaber, "Scrum development process," in *Proceedings of the OOPSLA Workshop on Business Object Design and Information*, 1995.

[29] S. Krusche, B. Reichart, P. Tolstoi, and B. Bruegge, "Experiences from an experiential learning course on games development," in *Proceedings of the 47th SIGCSE*, pp. 582–587, 2016.

[30] K. VanLehn, "Cognitive skill acquisition," *Annual Review of Psychology*, vol. 47, pp. 513–539, 1996.

[31] J. Sweller and G. Cooper, "The use of worked examples as a substitute for problem solving in learning algebra," *Cognition and Instruction*, vol. 2, no. 1, pp. 59–89, 1985.

[32] J. Trafton and B. Reiser, "Studying examples and solving problems: Contributions to skill acquisition," tech. rep., Naval HCI Research Lab, 1993.

[33] A. Collins, J. Brown, and A. Holum, "Cognitive apprenticeship: Making thinking visible," *American educator*, 1991.

[34] S. Krusche and A. Seitz, "ArTEMiS - An Automatic Assessment Management System for Interactive Learning," in *Proceedings of the 49th SIGCSE*, ACM, 2018.

[35] S. Krusche, B. Brügge, I. Camilleri, K. Krinkin, A. Seitz, and C. Wöbker, "Chaordic learning: A case study," in *Proceedings of the 39th ICSE*, pp. 87–96, 2017.

[36] A. Seitz and B. Bruegge, "Teaching pattern-based development," in *Proceedings of the 1st Workshop on Innovative Software Engineering Education*, pp. 20–23, 2018.

[37] S. Krusche, N. von Frankenberg, and S. Afifi, "Experiences of a software engineering course based on interactive learning.," in *Proceedings of the 15th SEUH Workshop*, pp. 32–40, 2017.

[38] M. Daun, J. Brings, P. Obe, K. Pohl, S. Moser, H. Schumacher, and M. Rieß, "Teaching conceptual modeling in online courses: Coping with the need for individual feedback to modeling exercises," in *30th Conference on Software Engineering Education and Training*, pp. 134–143, 2017.

[39] C. Kloos, C. Alario-Hoyos, I. Estévez-Ayres, P. Muñoz-Merino, M. Ibáñez, and R. Crespo-García, "Boosting interaction with educational technology," in *Global Engineering Education Conference*, pp. 1763–1767, April 2017.

[40] J. Krugel and P. Hubwieser, "Computational thinking as springboard for learning object-oriented programming in an interactive mooc," in *IEEE Global Engineering Education Conference*, pp. 1709–1712, 2017.

[41] L. Kolås, H. Nordseth, and J. Hoem, "Interactive modules in a mooc," in *15th International Conference on Information Technology Based Higher Education and Training*, pp. 1–8, Sept 2016.

[42] F. Grünewald, C. Meinel, M. Totschnig, and C. Willems, "Designing MOOCs for the Support of Multiple Learning Styles," in *European Conference on Technology Enhanced Learning*, pp. 371–382, 2013.

[43] S. Krusche, M. Berisha, and B. Bruegge, "Teaching code review management using branch based workflows," in *Proceedings of the 38th ICSE*, pp. 384–393, 2016.