Web-based learning in computer science: Insights into progress and problems of learners in MOOCs

Johannes Krugel and Peter Hubwieser

Technical University of Munich, School of Education, Munich, Germany

Abstract Web-based resources and massive open online courses (MOOCs) are promising forms of non-formal and technology-enhanced learning. Advantages are the flexibility regarding location and time and the possibilities for self-regulated learning. Furthermore, web technologies have considerably evolved over the past years, enabling complex interactive exercises and communication among the learners. However, online learning also has its challenges regarding, e.g., the motivation and low completion rates in MOOCs.
Following a design-based research methodology, we designed, developed, and evaluated a MOOC for the introduction of object-oriented programming. In three course runs, we collected extensive textual feedback from the participants which we analyzed inductive qualitative content analysis (QCA) by Mayring. We complement this with quantitative analyses regarding the performance of the learners in the course. The results give insights into the progress, preferences, and problems of learners in MOOCs. We furthermore used these results as a basis for adapting the course in the following iterations of our design-based research and observed a significant increase in the course completion rate.

1 Introduction

Web-based learning is a form of technology-enhanced learning with the advantage that the technical barriers are very low, for the learners as well as for the creators. Massive open online courses (MOOCs) combine several web-based learning activities in the form of a course. MOOCs became an educational buzzword in 2012 and have enjoyed wide media coverage in the popular press (Marshall, 2013). In contrast

to traditional ways of teaching, where the size of participants is restricted, MOOCs have to be easily scalable for large numbers of participants. They are usually free for everybody and can be used for formal as well as non-formal learning. According to the *e-learning hype curve*, MOOCs are already through the "Trough of Disillusionment" and nearly on the "slope of enlightenment" (Hicken, 2018). Our focus in this paper are MOOCs for the introduction of programming, especially object-oriented programming.

1.1 Our MOOC

Computer science (CS) education in school is varying strongly in many countries. In Germany, for example, the implementation of CS education at school is very diverse, unregulated and inconsistent in many states. In consequence, the prerequisite knowledge of freshmen at universities is very inhomogeneous (Hubwieser et al., 2015). As students cannot be expected to be present at university before lecturing starts, MOOCs (massive open online courses) seem to represent potential solutions to compensate or reduce these differences. This was the initial motivation to develop our MOOC called "LOOP: Learning Object-Oriented Programming".

The initial primary target group of the course are prospective students of science or engineering that are due to attend CS lessons in their first terms. However, since the course is available online and free for everybody, the target group now is a world-wide audience (speaking German) and with a more diverse background.

As learning to program is a substantial cognitive challenge (Hubwieser, 2008), MOOCs run in danger to overstrain the students, frustrating them already before their studies. To meet this challenge, we carefully designed LOOP, starting with a gentle introduction to computational thinking (Wing, 2006). The course is based on the *strictly objects first* approach (Gries, 2008), introducing the concepts *object*, *attribute*, and *method* just before *class* and before any programming activity. This helps to avoid excessive cognitive load following when it comes to actually write programs in an object-oriented programming language (in our course Java).

The course includes various interactive exercises to enable the learners to experiment with the presented concepts. Furthermore, we implemented programming exercises with constructive feedback for the learners using a web-based integrated development environment and additionally an automatic grading system.

A detailed description of the rationale behind the course design and the course curriculum was published in (Krugel & Hubwieser, 2018). The results of a pilot run of the course are presented in (Krugel & Hubwieser, 2017).

1.2 Research questions and research design

In this paper we examine the learners' perspectives to gain insights into the progress, problems, and preferences of learners in MOOCs. In particular, we aim to answer the following research questions:

1. What are learners' preferences in self-regulated introductory programming MOOCs?
2. What are the main challenges learners face in self-regulated introductory programming MOOCs?
3. What are the main reasons for not finishing introductory programming MOOCs?

The answers to those research questions are going to help to design self-learning courses and improve existing courses. Following a design-based research methodology, we start with a literature review, implement a prototypical course, assess the learning, and proceed in iterations (Design-Based Research Collective, 2003; Plomp, 2007).

We collect both quantitative and qualitative data, utilizing a mixed-methods approach for the data analysis; this approach makes it possible to gain in breadth and in depth understanding while balancing the weaknesses inherent in the use of each individual approach. To understand the learner's perspectives, we collected and analyzed feedback of the course participants. We applied an inductive qualitative content analysis to categorize the responses. In terms of learning progress, we pursue quantitative methods and conduced a hierarchical cluster analysis of participants' scores in the assignments. We use the results of the analysis to adapt the teaching and perform more iterations, observing the changes on the learners' side.

This paper is structured as follows. As a background, we describe the design-based research methodology, the foundations for the course design, and further related literature. Then we present the course design and its curriculum. The main part of this paper is the description of our data analysis methodology and the results. We conclude with a discussion of the limitations and an outlook.

2 Background and related work

In the following, we briefly describe the background of the research methodology, related MOOCs for introductory CS, research on MOOC design and drop-out behavior.

2.1 Design-based research (DBR)

DBR is an empirical approach where the application and investigation of theories is closely connected with the design and evaluation of learning interventions (Design-Based Research Collective, 2003). There is not a unique definition of the term but many authors agree that DBR encompasses a preliminary research, a prototyping phase, and an assessment phase (Plomp, 2007).

According to Reeves (2006), DBR consists of 4 phases: the analysis of a practical problem, the development of a solution, testing and refinement of the solution in practice, and a reflection to produce design principles, see Fig. 1. A main characteristic of DBR is the systematic and iterative cycle of design, exploration, and redesign (Design-Based Research Collective, 2003). The evaluation is often carried out using a mixed-methods approach (Anderson & Shattuck, 2012). Using mixed-methods for the evaluation makes it possible to gain in-breadth and in-depth insights while balancing the weaknesses inherent in the use of each individual approach.

DBR is nowadays widely used in different contexts of educational research, also for learning programming and algorithmic thinking (Geldreich et al., 2019; Papavlasopoulou et al., 2019).

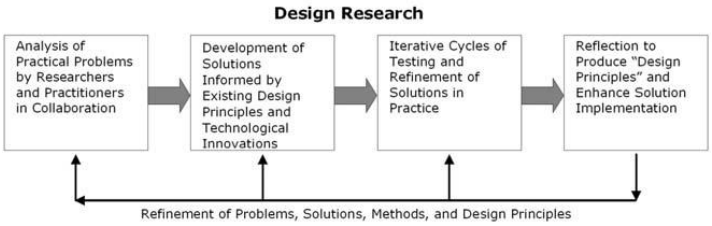

Fig. 1. Phases of design-based research (Reeves, 2006).

2.2 Research on MOOCs

There is an uncountable number of online courses for learning the basics of computer science. Some examples of MOOCs (massive open online courses) and SPOCs (small private online courses) that explicitly cover computational thinking or object-oriented programming (OOP) and were published in the scientific literature are described by Liyanagunawardena et al. (2014), Piccioni et al. (2014), Falkner et al. (2016), Alario-Hoyos et al. (2016), Kurhila and Vihavainen (2015), Vihavainen et al. (2012), Fitzpatrick et al. (2017) and Bajwa et al. (2019).

Several studies investigate the reasons of the learners for enrolling in MOOCs. Crues et al. (2018) analyzed the responses to open-ended questions in five MOOCs; they used methods from natural language processing (Latent Dirichlet Allocation) and identified a set of 26 “topics” as reasons to enroll in a MOOC.

Luik, Suviste et al. (2019) developed and tested an instrument to measure the motivational factors for enrolling in a programming MOOC.

Zheng et al. (2015) carried out interviews with participants of several MOOCs and analyzed them using grounded theory; among others, they investigated reasons for enrollment and how students learn in a MOOC.

In general, MOOCs are known to have a rather high dropout (Delgado Kloos et al., 2014; Garcia et al., 2014; Piccioni et al., 2014) with completion rates usually in the range of 5 – 10%.

Zheng et al. (2015) also analyzed the interviews regarding the reasons for not completing a MOOC and identified 8 categories: *High Workload, Challenging Course Content, Lack of Time, Lack of Pressure, No Sense of Community or Awareness of Others, Social Influence, Lengthy Course Start-Up, Learning on Demand*.

In another study, Eriksson et al. (2017) carried out interviews with participants of two MOOCs; the main result regarding the drop-out in MOOCs is that "Time is the bottleneck".

To assess the learner's perspectives in MOOCs with a special focus on active learning, Topali et al. (2019) analyzed questionnaires, forum posts and logs of the learning platform; they discuss challenges of the learners and reasons for not completing the course.

Luik, Feklistova et al. (2019) investigate the connection of drop-out with demographic factors of the participants in three programming MOOCs. There are, furthermore, many approaches to predict the drop-out probability of the learners based on the data available in such online courses, see e.g., (Moreno-Marcos et al., 2020).

3 Course design

According to the DBR methodology by Reeves (2006), we started by analyzing the needs of the learners and developing a solution based on existing design principles (phase 1 and phase 2). The course consists of a series of short videos, followed by quizzes, interactive exercises, and a final exam. The communication of the learners and with the course team takes place in a discussion board. In the following, we describe those course elements, and present the details of the course design.

3.1 Videos

All topics of the course are presented in short videos with an average length of 5 minutes. The videos were produced based on the suggestions of Guo et al. (2014) and similar to the suggestions by Alonso-Ramos et al. (2016) published shortly after our recording.

Each of the 24 videos begins with a short advance organizer to help the learners focus on the relevant aspects. This is augmented with the talking head of the respective instructor (using chroma key compositing) facilitating the learners to establish a personal and emotional connection (Alonso-Ramos et al., 2016).

For the actual content of the videos, we decided to use a combination of slides and tablet drawing. The background of the video consists of presentation slides and the instructor uses a tablet to draw and develop additional aspects or to highlight important part of the slides. All slides are provided for download and we additionally added audio transcripts for the videos. By such video, audio, and textual representations, several senses are addressed simultaneously, making the content accessible to learners with different learning preferences or impairments.

3.2 Quizzes

After each video, the course contains quizzes as formative assessment. The main purpose is to provide the learners with direct and instant feedback on the learning progress. The quizzes use the standard assessment types offered by the MOOC platform, e. g., single-/multiple-choice questions, drop-down lists, drag-and-drop problems or text input problems. Depending on the answer, the learner gets a positive feedback or, otherwise, for example hints which previous parts of the course to repeat in more detail.

3.4 Interactive exercises

The videos introduce new concepts to the learners and the quizzes test the progress, which is, however, in general not sufficient to acquire practical competencies (Alario-Hoyos et al., 2016). Following a rather constructivist approach, we intend to let the learners experiment and interact with the concepts directly. Considering that, we include interactive exercises or programming task for all learning steps throughout the course. Special care was devoted to the selection and development of those interactive exercises to enable the learners to experiment and interact directly with the presented concepts. It can be a major obstacle for potential participants having to install special software (Liyanagunawardena et al., 2014; Piccioni et al., 2014), which is especially problematic in an online setting without a teacher who could help in person. We therefore decided to use only purely web-based tools. There are already many web-based tools for fostering computational thinking and learning OOP concepts available in the internet. We selected the in our opinion most suitable tools to support the intended learning goals. Where necessary we adapted or extended them to meet our needs. All tools are integrated seamlessly into the learning platform resulting in a smooth user experience.

3.5 Programming exercises

While in several introductory CS MOOCs the learners have to install an integrated development environment (IDE) for writing their first computer programs, we decided to rely on web-based tool also for this purpose (like (Piccioni et al., 2014)). We chose to use *Codeboard* (Estler & Nordio) (among several alternatives, (Derval et al., 2015; Skoric et al., 2016; Staubitz et al., 2016) because of the usability and seamless integration into the *edX* platform using the Learning Tools Interoperability (LTI) standard.

The programming assignments are graded automatically and the main purpose is to provide instant feedback to the learner. We therefore implemented tests for each assignment that make heavy use of the Java reflection functionality. While standard unit tests would fail with a compile error if, e.g., an attribute is missing or spelled differently. Reflection makes it possible to determine for a learner's submission if, e.g., all attributes and methods are defined with the correct names, types and parameters. Writing the tests requires more effort than for standard unit tests but can give more detailed feedback for the learners in case of mistakes.

Additionally we integrated the automatic grading and feedback system *JACK* (Striewe & Goedicke, 2013) using the external grader interface of the *edX* platform. Apart from static and dynamic tests, *JACK* also offers the generation and comparison of traces and visualization of object structures; however, we do not use this extended functionality yet.

3.6 Discussion board

The course also provides a discussion forum which is divided into several discussion boards. The communication among the learners and with the instructors is supposed to take place entirely in the discussion forum. Besides the default board for general and organizational issues, we created one separate discussion board for each course chapter (called chapter discussion board in the following). The idea is to organize the learners' discussions, such that it is easier for the learners keep an overview. In those discussion boards, the learners can ask questions and answer the questions of others. The course team also tracks the discussions and can intervene when problems cannot be solved timely by the learning community itself. It is also possible to vote for popular topics but we did not mention this possibility and nearly nobody used it.

Additionally we included separate discussion boards for six specific exercises in which the learners are supposed to upload their solutions to the forum (those boards are called exercise discussion boards in the following). In the first of those exercises, the learners are encouraged to introduce themselves to the community giving some

information about their age, where they life etc. In the exercise discussion boards, the learners are also prompted to give feedback to the contributions of others.

3.7 Final exam

The course contains a final exam at the end that covers topics of the whole course. The suggested time to work on the exam is 60 minutes; the time limit is, however, not ensured by technical limitations. The final exam as well as the graded exercises count for a final score. A minimal score of 50 % is necessary to successfully pass the course.

3.8 Course syllabus

Computational thinking (CT) as introduced by Wing (Wing, 2006) is a universal personal ability that can be used in many disciplines. Since the target group of our course comes from various different fields of study, we incorporated CT as integral part of the course. CT is on the one hand intended to facilitate learning programming and on the other hand a sustainable competency that can be used also outside of our course.

As pointed out in (Hubwieser, 2008), there is a fundamental didactical dilemma in teaching OOP: On the one hand, modern teaching approaches postulate to teach in a "real life" context (Cooper & Cunningham, 2010), i.e., to pose authentic problems to the learners. Therefore, it seems advisable to start with interesting, sufficiently complex tasks that convince the learners that the concepts they have to learn are helpful in their professional life. However, if we start with such problems, we might ask too much from the learners, because they will have to learn an enormous amount of new, partly very difficult concepts at once (Hubwieser, 2008).

Following a *strictly objects first* approach (Gries, 2008) and similar to the design of the school subject and an introductory university lecture, we solved this problem by distributing the learning objectives over the parts of the course that precede the "serious" programming part. This avoids to confront the learners with too many unknown concepts when they have to write their first program. Among others, we suggest to the learners to look at an object as a state machine (Hubwieser, 2008). In order to realize this in a learner-oriented way, the learners need to be able to understand a simulation program of a typical state machine, e.g., a traffic light system.

Concerning the choice of the examples, we set the emphasis on the relevance for the everyday life, which leads for instance to banking or domestic appliances.

LOOP consists of the following five chapters:

1. Object-oriented modeling
   1.1. Objects
   1.2. Classes
   1.3. Methods and parameters
   1.4. Associations
   1.5. States of objects
2. Algorithms
   2.1. Concept of algorithm
   2.2. Structure of algorithms
3. Classes in programming languages
   3.1. Class definition
   3.2. Methods
   3.3. Creation of objects
4. Object-oriented programming
   4.1. Implementing algorithms
   4.2. Arrays
5. Associations and references
   5.1. Aggregation and references
   5.2. Managing references
   5.3. Communication of objects
   5.4. Sequence charts

A detailed description of the course design and syllabus was published in (Krugel & Hubwieser, 2018).

4 Methodology and results

We follow a design-based research methodology and in this work we focus on the assessment of our learning intervention (phase 3 and phase 4 according to Reeves (2006)). Collecting both quantitative and qualitative data in this case study, we utilize a mixed-methods approach for the data analysis to gain in-breadth and in-depth understanding while balancing the weaknesses inherent in the use of each individual approach. To understand the learner's perspectives, we collected and analyzed feedback of the course participants. For the analysis, we applied an inductive qualitative content analysis according to Mayring (2000). In terms of learning progress, we pursue quantitative methods and conduced a hierarchical cluster analysis of participants' scores in the assignments. This calculation is performed using the programming language R.

In the following, we first inform about the course implementation, our data collection and give some data on the course participants. We then describe the data analysis with its results. We performed three DBR cycles so far and describe the adaptations of the course, which were based on the results of our analyses.

4.1 Course implementation

We prepared the online course on the learning platform *edX*[1] and offered it three times in the summer holidays of 2016, 2017, and 2018. We will refer to a course run by its year in the following exposition.

Course run 2016 was offered as a small private online course (SPOC) on *edX Edge* and was announced internally at our university as a preparation course for CS basics. The course runs 2017 and 2018 were offered publicly as MOOCs on *edX*. For organizational reasons, 2017 actually consisted of two identical and directly consecutive course runs which we treat as one course run in the following. The course runs 2017 and 2018 were included in the global *edX* course catalog in the category *Computer sciences courses* and available world-wide. Our university announced the courses on its official Facebook page and informed all students in CS-related subjects about the course offerings.

The intended effort of the students is 5 hours per week. Everyone was free to participate in the courses without any formal or content-specific prerequisites. The course takes into account different learning preferences and impairments by providing the learning content as visual (videos, graphics), textual, and audio presentations. The only requirement was German language proficiency since the course was offered on German.

Participation was voluntary in all course runs and did not count towards a grade. In 2016, we issued informal certificates for successful participation (= obtaining at least 50% of the possible points in at least 12 of 16 course units). In 2017 and 2018, *edX* issued verified certificates for successful participation, which, however, had to be paid (49 $).

Each course run took five weeks (one week for each chapter) and the targeted workload of the learners was 5-10 hours per week. The communication among the learners and with the instructors took place entirely in the discussion forum. The main task of the instructor during the conduction of the course was to monitor the forum and to react accordingly, e.g., answer questions or fix problems with the grading system.

4.2 Data collection

We integrated an introductory online questionnaire into the course (called "course start survey" in the following), in which we asked the participants about their age, gender, major, and previous programming experience.

In a concluding questionnaire at the end of the course (called "course end survey" in the following), we asked for positive and negative textual feedback regarding the

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

course; it consists of two text field with the following questions (translated from German):

1. Positive aspects: *Which aspects of the course did you like?*
2. Negative aspects: *Which problems did you encounter during the course? Which aspects of the course did you not like? Which suggestions for improvements do you have?*

The course end survey also asked for the approximate weekly workload that the learners spent on the course.

To get an even more detailed picture in the course runs 2017 and 2018, we additionally requested the participants' perspectives already during the course. We therefore included the same questions 1 and 2 directly after each chapter, for organizational reasons with one combined text input field for both questions together (this is called "chapter feedback survey" in the following). We can therefore react more specifically regarding the current chapter and also earlier (even immediately during the course run). This chapter feedback also enables us to get responses by those who do not make it to the end of the course and to assess what they think before actually dropping out.

We are especially interested in why some learners successfully finish the course and others do not. We therefore wrote an individualized mail to every course participant who did not complete the course (course runs 2017 and 2018). In the mail, we simply asked for the reason for not finishing the course (this is called "drop-out survey" in the following, even though not everybody not finishing a course is necessarily an actual drop-out).

Further, all responses to the quizzes and other tasks, as well as all programs submitted by the participants during the online course were collected. From the postings in the discussion forum even more qualitative data was obtained.

4.3 Participants

The three course runs of LOOP attracted 87+2390+2520 = 4997 registrations. The following numbers are always the total of all course runs in which we collected the respective data if not stated otherwise.

For the course start survey we received 80+494+1039 = 1613 responses (female: 463, male: 992, diverse: 7, no answer: 151) with a diverse study background (numerous different majors, including Computer Science, Management, Engineering, Mathematics and many more). The participants come mainly from Germany, but in total from more than 70 countries. The average age was 23.5 and 315 participants were less than 20 years old. Regarding programming, 459 participants had no experience, 673 had basic knowledge, and 300 participants had already written a "bigger" program of at least 100 lines of code (no answer: 181 participants).

4.4 Learners' feedback

To assess the learners' perspectives, we analyzed all their free text responses. This reveals how the participants perceived and dealt with the online course. One goal of this analysis is to gain generalizable insights into the preferences and problems of learners in programming MOOCs. Another goal is to improve our course accordingly aiming at a higher satisfaction of the learners and a higher success rate.

Data analysis

We chose to analyze the learners' utterances of the course end survey, the chapter feedback and the drop-out survey using qualitative content analysis following the methodology of Mayring (2000). This allows to categorize the statements which helps to afterwards group and inspect similar statements. This will help to understand the learners' perspectives and to prioritize aspects of the course that can be improved. We are interested in the following three aspects which therefore guided the categorization:

1. Positive aspects of the course (course end survey and chapter feedback)
2. Negative aspects of the course (course end survey and chapter feedback)
3. Reasons for not finishing the course (drop-out survey)

We abstracted each learners' statement such that its wording is independent of our particular course: a learner wrote for example "*The drawing exercise took a lot of effort.*" and we abstracted as "*high effort*". We furthermore rephrased all abstractions as statements about the current state of the course: a learner wrote for example the suggestion "*It would be better to include more examples*" and we abstracted this as "*too little examples*". Since each learner's statement could consist of several sentences or even paragraphs, the rephrased versions could consist of multiple abstractions. We then inductively categorized all abstractions using several iterations over the data. In those iterations we introduced new categories, refined, and merged existing categories. We kept track of the category systems maintaining a coding manual with descriptions of the categories. Whenever the category system changed, we performed another iteration over the corresponding data until reaching a stable state.

The result of this analysis is a classification of the learners' free text responses of all course runs into the categories of the three supercategories while each response can fall into several categories.

Results for the course end survey

In the course end survey we received 13+102+191 = 306 answers. The learners reported their views about the videos and exercises, the level of difficulty, technical and organizational issues. They furthermore described their individual progress or

problems and proposed specific changes, among others. Due to the low number of responses in 2016, we did not include these responses in our further analysis.

Applying the inductive qualitative content analysis on the course end survey and the chapter feedback, we encountered 15 categories for the positive aspects and 45 categories for the negative aspects taking all three course runs together. The results for the course end survey are displayed in Table 1 and Table 2. The tables further contain for each category a description and the frequency within responses of the three course runs. Since this is the result from a *qualitative* analysis, we sorted the categories alphabetically and not by frequencies.

There are some categories listed with 0 occurrences. They stem from the chapter feedback because we used the same category system for the positive/negative aspects mentioned in the course end survey and the chapter feedback.

Over a third of the participants explicitly reported to enjoy the interactive exercises and many participants also liked the videos, the programming tasks, and the overall course structure and alignment.

For the negative aspects, many learners reported that they had difficulties solving the programming tasks. Also the rise in difficulty was seen critically and many participants would have liked more comprehensive explanations of the concepts.

There are less categories for the positive aspects compared to the negative aspects. However, since we perform a *qualitative* analysis here, the comparison of numbers is not necessarily meaningful: For the positive aspects, the participants, e.g., usually just reported to like the videos, while for the negative aspects, they explained which aspect about the videos they did not like (which results in several categories). A quantitative comparison of those numbers can sometimes give a hint, but has to be interpreted carefully.

2017 (102)	2018 (191)	Positive aspect	Details
6,9%	3,7%	Automatic feedback	For quizzes and programming tasks
2,0%	0,0%	Contents	Selection of course contents
11,8%	9,4%	Course structure	Variability and alignment of the elements
5,9%	3,7%	Discussion board	Help by other learners and the course team
6,9%	4,2%	Examples	Real-world connection, clarity
31,4%	30,4%	Exercises	Interactivity
5,9%	10,5%	Explanations	Clarity, understandability
5,9%	3,1%	External tools	Integration of external tools into the course
1,0%	0,0%	Final exam	Tasks and alignment of the exam
4,9%	3,1%	Flexibility	Regarding time and location
3,9%	6,8%	Handouts	Preparation, layout, downloadable
11,8%	9,9%	Level of difficulty	Not too easy, not too hard
14,7%	15,7%	Programming tasks	Variability and alignment of the elements
6,9%	5,2%	Quizzes	Help by other learners and the course team
16,7%	18,8%	Videos	Video style, availability

Table 1: Categorization of negative aspects mentioned by the participants in the course end survey.

2017 (102)	2018 (191)	Negative aspect	Details
0.0%	2.6%	Automatic feedback	The feedback is not helpful
2.0%	0.5%	Connections unclear	Connections between the contents elements are not clear
2.9%	1.6%	Content missing	Some specific content is missing
3.9%	2.1%	Deadlines	Time-constraints of the course are too strict
1.0%	0.0%	Discussion board is confusing	Keeping an overview and finding relevant posts is difficult
2.9%	2.1%	Effort too high	Compared to learning outcome
4.9%	4.7%	Explanation unclear	
5.9%	1.6%	External tools	Complicated to use, technical problem with a tool
1.0%	0.0%	Language problems	German not as mother tongue
19.6%	14.1%	Level of difficulty increase	
0.0%	1.6%	Level of difficulty too high	
2.9%	0.0%	Level of difficulty too low	
2.0%	0.5%	Miscellaneous	
2.0%	0.5%	Motivational problem	
2.9%	5.8%	Obligation to post in forum	Prefer to learn alone without interaction
1.0%	1.0%	Overview handout is missing	Fact sheet, language reference etc.
22.5%	20.4%	Programming too difficult	E.g. difficulties debugging programming errors
1.0%	3.7%	Quizzes	Questions or answers unclear, too easy
2.0%	1.6%	Sample solution is missing	Missing or available too late
0.0%	1.0%	Specific task too difficult	Any specific task (except a programming task)
1.0%	1.6%	Syntax problems	
4.9%	5.2%	Task description unclear	
1.0%	0.5%	Teacher feedback is missing	For drawing or programming exercises
0.0%	6.8%	Technical problems	Regarding the learning platform, the videos etc.
11.8%	7.9%	Too concise	Prefer a more extensive explanation
7.8%	4.7%	Too few examples	
2.0%	2.6%	Too few exercises	
1.0%	2.6%	Too few hints	
0.0%	0.0%	Too few programming tasks	
3.9%	5.2%	Too theoretical	Prefer more practical explanations and tasks
7.8%	7.9%	Video style	Too serious, too little enthusiasm etc.
1.0%	0.5%	Videos too fast	
1.0%	1.6%	Videos too long	
0.0%	0.5%	Videos too short	
1.0%	0,0%	Videos too slow	
0.0%	0,0%	Videos volume too low	
0.0%	0,0%	Weekly exam is missing	Prefer to have additional exams in each chapter

Table 2: Categorization of negative aspects mentioned by the participants in the course end survey.

Results for the chapter feedback

In the two investigated course runs, we received 255+460 = 715 responses for the chapter feedback as a total their five chapters. The statements are more specific regarding particular aspects of the course sections than in the course end survey. We omit the detailed results for the positive aspects since they mainly reflect the results from the course end feedback and are not very interesting for adapting the

course. The results of the qualitative content analysis for the negative aspects are shown in Table 3.

Many participants reported that the effort compared to the learning outcome is too low in Chapter 1; they refer to two specific exercises where the learners are supposed to draw a graphic (the layout of their room and an object diagram). This is closely connected to the negative mentioning of embedded tools in Chapter 1.

In Chapter 2, the participants found the content to be too short and too theoretical in the course run 2017, but not in 2018 where we had added another practical exercise. The learners reported problems with an embedded tool in 2018, which was fixed by the authors of the external tool during the course run.

2017					**2018**					**Course run**
1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**Chapter**
108	69	39	28	11	171	147	77	44	21	**Responses**
0	0	0	0	0	0	0	0	0	0	Automatic feedback
1	2	0	0	0	0	0	0	0	0	Connections unclear
0	1	0	0	0	1	0	0	1	1	Content missing
3	0	0	0	0	0	0	0	1	0	Deadlines
2	0	0	0	0	1	0	0	0	0	Discussion board is confusing
12	0	0	0	0	24	2	0	2	1	Effort too high
1	0	2	2	0	3	2	3	4	1	Explanation unclear
17	0	0	0	0	13	23	1	0	1	External tools
0	0	0	0	0	2	0	0	0	0	Language problems
0	0	0	0	0	0	0	0	0	0	Level of difficulty increase
0	0	1	4	1	2	1	1	11	3	Level of difficulty too high
2	0	0	0	0	4	1	0	0	0	Level of difficulty too low
1	0	0	2	0	3	1	1	0	0	Miscellaneous
0	0	0	0	0	2	0	0	0	0	Motivational problem
0	0	0	0	0	1	0	0	0	2	Obligation to post in forum
2	0	0	0	0	3	0	2	0	1	Overview handout is missing
0	0	1	16	8	0	0	2	20	6	Programming too difficult
1	3	3	0	0	6	11	0	0	0	Quizzes
1	3	0	0	0	1	0	0	0	0	Sample solution is missing
0	1	0	0	0	2	12	0	0	0	Specific task too difficult
0	0	0	0	0	1	0	1	1	0	Syntax problems
7	0	9	1	0	5	0	4	0	0	Task description unclear
6	0	0	0	1	0	0	0	0	0	Teacher feedback is missing
6	0	1	1	0	9	8	12	0	0	Technical problems
2	6	0	0	0	5	3	2	1	0	Too concise
0	2	0	0	0	3	0	0	0	0	Too few examples
1	3	1	0	0	1	0	4	0	0	Too few exercises
0	2	0	0	0	0	2	1	0	0	Too few hints
3	1	0	0	1	0	14	1	0	0	Too few programming tasks
0	5	0	0	0	3	1	0	0	1	Too theoretical
0	1	0	0	0	16	1	0	1	0	Video style
0	1	0	0	0	2	0	0	0	0	Videos too fast
0	0	2	2	0	3	1	0	3	0	Videos too long
1	2	0	0	0	0	0	0	0	0	Videos too short
0	0	0	2	0	2	0	0	0	0	Videos too slow
0	0	0	0	0	1	0	0	0	0	Videos volume too low
3	1	0	0	0	1	0	0	0	0	Weekly exam is missing

Table 3: Categorization of negative aspects mentioned by the participants in the chapter feedback.

The first programming exercises start in Chapter 3. However, the problems regarding programing tasks are mainly reported in Chapter 4. In this chapter, the tasks are more complex and obviously a hurdle for learners. This coincides with statements regarding a high difficulty, mainly also in Chapter 4.

Table 3 lists several more aspects mentioned by the learners, some of which were easy to change in the consecutive course runs, but also some that directly contradict each other.

Results for the drop-out survey

For the drop-out survey we sent out 1474+2270 = 3744 individualized e-mails to the participants that registered but did not successfully complete the course (course runs 2017 and 2018). We received 227+411 = 638 answers regarding the reason for that. The qualitative content analysis yielded 22 categories and the frequencies of the categories are displayed in Table 4.

Again we want to note that not everybody not finishing a course is necessarily an actual drop-out since there are several other possible reasons. By far the most prevalent reason mentioned was a lack of time (either due to side jobs, travel or other obligations). Several other participants apparently realized that they were already familiar with the contents and therefore discontinued the course. Further, both personal (illness, late registration, private reasons, etc.) as well as course-related reasons (rise in difficult, language problems, explanations, etc.) were mentioned. Technical problems hardly appear to be a reason to discontinue with the course.

2017	2018	Reason for not finishing	Details
227	411		
1	6	Effort too high	Especially regarding the learning outcome
3	1	Explanations unclear	
5	22	No further need for participation	E.g., due to changes of the study major
12	16	No internet connection	E.g., due to moving or travel
8	10	Illness	
9	14	Motivational problems	
4	13	Level of difficulty too high	
2	0	Level of difficulty too low	
16	26	Level of difficulty increase	
1	3	Only have a look	From the start no intention to learn actively
5	8	Private reasons	
4	12	Programming too difficult	
24	32	Contents already known	
0	6	Miscellaneous	
6	5	Registration too late	
6	4	Language problems	
3	11	Technical problems	E.g., with the learning platform
0	5	Forgotten	Registered but did not think of it afterwards
1	4	Video style	
141	223	Time constraints	Due to jobs, university courses etc.
1	3	Too concise	Explanations too concise to understand
0	4	Too theoretical	Especially regarding the learning outcome

Table 4: Categorization of reasons to not finish the course mentioned by the participants in the drop-out survey.

4.5 Adaptation of the course

Following the design-based research approach, we used the results to adapt the teaching, i.e. our online course. For the adaptation, we considered especially the negative aspects of the course end survey and chapter feedback, but also the positive aspects and the reasons for not finishing the course. The categorization helps to keep an overview, to identify and group similar issues, and also to notice contradicting views. The frequencies of the statements in the categories can also give hints which aspects are urgent for many learners and which statements are only isolated opinions of very few.

Some of the problems and suggestions mentioned in the learners' statements can be addressed easily while others would require big changes to the course structure. We applied some of the changes already during the course runs (i.e. in the case of technical problems or misleading task descriptions) while we changed bigger issues only after the course runs (i.e. introduce completely new exercises). In the following we describe the adaptations we made to the course.

In general, we clarified textual explanations, task descriptions and quizzes wherever reported. We did, however, not change any video recordings yet due to the costs involved.

Since the learners reported problems with an external drawing tool in Chapter 1 of run 2017, we provided alternatives in 2018 and allowed simply draw using paper and upload a photo of the result. We additionally clarified some task descriptions.

Because the learners found Chapter 2 too short and too theoretical, we added another practical exercise (A2.2.6) in 2017 and the feedback in 2018 shows that this is not seen as problem anymore. However, it turned out that the new exercises was seen as difficult, why we added some hints at critical steps after the course run 2018.

In Chapter 4, many learners had difficulties with the actual programming. We therefore added a smaller initial programming task (A4.1.5), provided more explanations and modified the existing programming tasks. Additionally, we included a page giving hints at how to best learn in a self-learning environment like a MOOC and pointing to supporting offers like the discussion board. This had some positive results, but still many learners struggle when having to program a non-trivial class with attributes and methods. After course run 2018 we therefore introduced an explanation of debugging basics, an overview of variable and array instantiation in Java, as well as a series of 20 step-wise hints for one programming exercise.

Most learners who did not finish the course mentioned time-constraints as the main reason. We therefore also changed the course organization slightly: In 2017, the chapters were released on Monday and due the following Monday. In 2018, the chapters were already released on the Friday before. This made it possible that each chapter is available on two weekends with a small overlap of the chapters.

We did, of course, not apply all changes proposed by the participants. So far we focused on changes that are not controversial among the learners and do not require to change the overall course structure.

4.6 Learners' communication

Unlike in the pilot course run 2016 (Krugel & Hubwieser, 2017), there were lively discussions in the discussion boards of course runs 2017 and 2018. At the end of the five-week courses, the discussion boards of course runs 2017 and 2018 contained 72+122 = 194 topics with 197+244 = 441 replies. The exercise-specific forums contained 1068+2442 = 3510 topics with 288+932 = 1220 replies in total. This is presumably partly due to our intensified aims at encouraging asking questions and helping other students. Another reason is certainly the substantially higher number of participants because discussions take place much easier above a certain critical mass.

During the course we observed the discussion board daily and reacted accordingly when needed. However, we did not systematically analyze the contents of the discussions yet.

4.7 Workload

In the course end survey, the learners reported their average weekly workload. The average was 3.8 hours per week in 2017 and 4.6 hours per week in 2018. This increase is presumably due to the additional exercises and material. Another reason might be that more learners reached the later and reportedly more time consuming chapters of the courses. Some participant's reported in the discussion forum and feedback texts to spend more than 15 hours per week on the course, especially when stuck in programming exercises.

4.8 Learners' performance

In principle, MOOCS are intended to attract large numbers of participants. In most cases, they don't offer any tutoring by human individuals. In consequence, there is no way of monitoring the atmosphere and the learning behavior of the participants by personal contact with teaching personal. Thus, it is seems essential to monitor the behavior of the learners by automatic means to get necessary feedback on particular difficult content and dropout rates or reasons. Due to the large scale of the courses, the behavior of some few individuals is not very relevant. Instead, we are interested in the learning process of larger groups that share certain behavior features, e.g. regarding scoring of certain tasks or dropout points, which seems to be a classical application for statistical cluster analysis.

First, we performed a cluster analysis of the pilot conduction of LOOP in 2016, for which we calculated the average of the achieved relative scores over each of the 16

course sections for each of the 187 participants. The results reflected in a quite instructive way, how the performance of large groups of participants developed over these sections and in which sections a substantial number of them gave up. For the next course runs in 2017 and 2018, the number of participants increased substantially, which allows us to perform cluster analysis on the level of singles tasks to get a much more detailed picture of the performance development.

For this purpose, we conducted a hierarchical cluster analysis on the individual scores of all tasks of the course. In a first step, we cleaned raw score matrices (one line per individual participant and one column by each task) by removing all lines that contained exclusively empty values ("not attempted"). In a second step, we normalized the resulting matrices to an interval scale (from 0.00 to 1.00) by replacing all "not attempted" values by 0 and by dividing all score values by the maximum score of the respective tasks (columns).

For the clustering, we regarded the columns of this matrix as dimensions. Thus, the set of all scores of each participant could be interpreted as the definition of a certain position in a multidimensional space. By this way, the pairwise distance between the positions of all participants could be calculated in a specific distance matrix. Looking for the best result, we tried two different distance metrics (Maximum and Euclidian). Finally, a hierarchical clustering was performed on this distance matrix, starting with one cluster per person and combining successively more and more persons to larger clusters according to their relative distance, applying several different clustering strategies (ward.D, Complete, Average and McQuitty, for details see (Everitt et al., 2001)). The calculation was performed in the statistical programming language R by applying the function *hclust*.

As hierarchical clustering is a local heuristic strategy, the results have to be inspected according their plausibility. For this purpose, we looked for plausible dendrograms that represented a proper hierarchy. We found that the Euclidean distance metrics produced the best results in combination with the ward.D algorithm. Fig. 2 shows an exemplary dendrogram for this combination. To find a suitable number of clusters, we inspected these dendrograms from the top down to a level where we found as many clusters as possible, but avoiding too small clusters with too small number of members. We found that the best height to cut would be at 4 branches, which suggests the following clusters c1, c2, c3, and c4 in course run 2017 and d1, d2, d3, and d4 in course run 2018 (with the numbers of members in parentheses):

- Course run 2017: c1 (104), c2 (207), c3 (235), and c4 (328)
- Course run 2018: d1 (513), d2 (196), d3 (472), and d4 (469)

Finally, we calculated the average performance over all course tasks for each of the 4 clusters. The results are displayed in Fig. 3 (for course run 2017) and Fig. 4 (for course run 2018). Please note that there is a small difference in the task lists between the runs of 2017 and 2018, because in 2018 two new exercises were added (A2.2.5 and A4.1.5). Exercise A2.2.5 (Maze) has been changed for a more flexible answer format and is now obviously more difficult. In addition, some other exercises might be easier due to improved explanations etc.

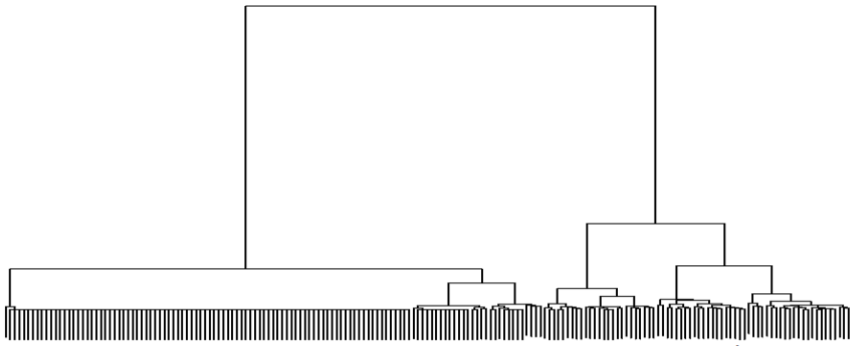

Fig. 2. Exemplary dendrogram, clustering by *Euclidean* distance with the *ward.D* algorithm.

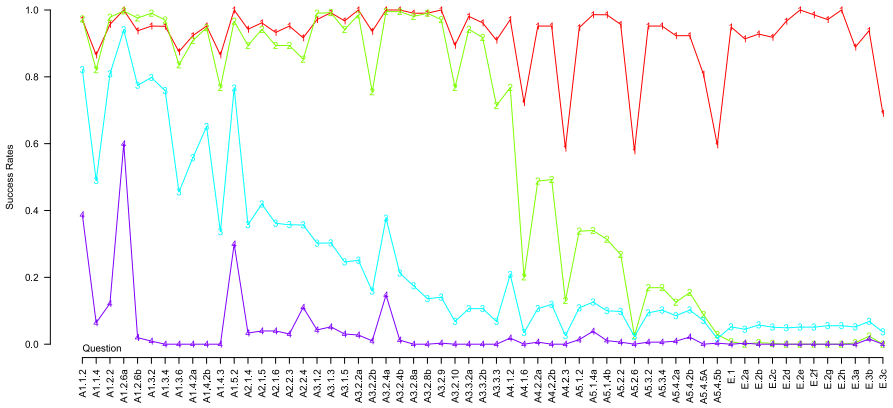

Fig. 3. Average performance of the learner clusters c1, c2, c3, and c4 in the course sections in 2017.

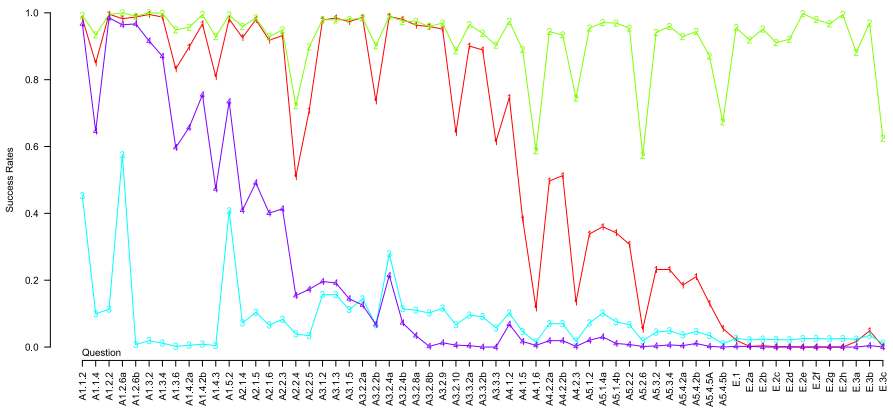

Fig. 4. Average performance of the learner clusters d1, d2, d3, and d4 in the course sections in 2018.

	2017		2018	
Name	**Cluster**	**Members**	**Cluster**	**Members**
High performers	c1	11.9%	d2	11.9%
Late droppers	c2	23.7%	d1	31.1%
Early droppers	c3	26.9%	d4	28.4%
Low performers	c4	37.5%	d3	28.6%

Table 5: Clusters of participants in the course runs 2017 and 2018.

Due to its design, our clustering reflects the performance and dropout behavior of four typical groups that represent many participants each. Unfortunately, the numbering of the clusters is set arbitrarily by the R packages. To support the comparison between the two runs, we introduce new names for the clusters:

1. The "high performers" kept comparably high scores over the whole course in both runs (c1 and d2).
2. The "late droppers" started with high scores, but dropped starting from section 4 (c2 and d1).
3. The performance of the "early droppers" dropped already during section 1 (c3 and d4).
4. The "low performers" reached high scores only in a few tasks (1.6.2a, 1.5.2, and 3.2.4a), but performed low in the rest (c4 and d3).

A comparison of the numbers of members of these clusters demonstrates that the percentage of high performers remained constant. The number of late droppers increased significantly, while the low performers were reduced. This might be interpreted as an overall improvement of LOOP.

In the course run 2017, 113 of the 2390 (4.7%.) registered participants successfully passed the course. In 2018, this ratio was up to 213 out of 2520 (8.5%). This is quite an encouraging result and indicates that the adaptations of the course meet the learners' preferences and help with the challenges.

5 Discussion

Using a mixed-methods approach and following a design-based research methodology we gained insights into the perspectives of learners in MOOCs, which also served as basis for adapting the course.

5.1 Lessons learned

1. What are learners' preferences in self-regulated introductory programming MOOCs?

Learners reported to like the very clear structure of a MOOC with well-aligned videos, quizzes, and exercises. Especially many learners mention to enjoy interactive exercises that let them interact with the concepts directly.

2. What are the main challenges learners face in self-regulated introductory programming MOOCs?

A major challenge for learners is to handle a high level of difficulty, especially when the difficulty increases during the course. This is also closely connected to motivational factors of the learners. If it is not possible to avoid such an increase, it seems to help the participants if the rise in difficulty is explicitly announced before and several offers for assistance are clearly pointed out.
Interestingly, some participants explicitly stated to dislike engaging in discussion boards and prefer to study on their own. This should be kept in mind when designing a course with collaborative elements, compulsory peer-feedback etc.

It was also confirmed to be important to keep the technical barriers very low when offering a course which is available worldwide for everybody. Especially novices cannot be expected to install additional software like an integrated development environment (IDE) or compilers when sitting alone at their computer. We therefore integrated only purely web-based tools and a few participants still mentioned some technical difficulties (even though no severe problems). In the recent course runs, several participants mentioned they would like to complete the course on their mobile phone or tablet computer. Even though the learning platform already provides apps for the most popular operating systems, some exercises involving external tools to not work satisfactorily.

3. What are the main reasons for not finishing introductory programming MOOCs?

Time-wise flexibility seems to be very important to participants of online courses. For each chapter in the course run 2018, we allocated 10 days including two weekends in which the participants could work on the content at their own pace. However, the vast majority of participants that did not complete the course, still mentioned time constraints as the main reason. This confirms the main findings by Eriksson et al. (2017) and Topali et al. (2019).

Our qualitative analysis resulted in 22 categories of reasons for not finishing a MOOC. The interview study of Zheng et al. (2015) yielded 8 categories and the interview study of Eriksson et al. (2017) found 11 main factors (grouped into 4 themes). Most of our categories can be mapped to one of the categories of the pre-

vious studies and vice-versa. Table 6 shows a comparison of the three category systems. Our categorization is a bit more fine-granular and also more specific for programming MOOCs.

This work	**Zheng et al. (2015)**	**Eriksson et al. (2017)**
Effort too high	High workload	Utilitarian motivation
Explanations unclear		The learner's perception of the course design
No further need for participation	Learning on Demand	Utilitarian motivation
No internet connection		Internet access
Illness		External factors
Motivational problems	Lack of Pressure, No Sense of Community or Awareness of Others, Social Influence	Utilitarian motivation, Enjoyment motivation, Study techniques
Level of difficulty too high	Challenging Course Content	Perceived level of difficulty of the content
Level of difficulty too low		
Level of difficulty increase	Challenging Course Content	Perceived level of difficulty of the content
Only have a look	Learning on Demand	Utilitarian motivation
Private reasons		External factors
Programming too difficult	Challenging Course Content	Perceived level of difficulty of the content
Contents already known	Learning on Demand	Mismatch between expectations and actual content
Miscellaneous		
Registration too late	Lengthy Course Start-Up	
Language problems		English proficiency
Technical problems		
Forgotten	Lengthy Course Start-Up	
Video style		The learner's perception of the course design
Time constraints	Lack of Time	Lack of time
Too concise	Challenging Course Content	The learner's perception of the course design
Too theoretical	Challenging Course Content	The learner's perception of the course design

Table 6: Comparison of reasons for not finishing a MOOC.

5.2 Limitations

The learners' feedback of one MOOC is the main basis of our analysis. By its very nature, feedback is only self-reported and therefore not necessarily objective. For example when asking for the reasons to not finish the course, participants might attribute their drop-out rather to time constraints than to their inability to solve the tasks. Therefore it seems very important to follow a mixed-methods approach also in the future.

In the course runs, we did not study the effect of single changes to the course in isolation. This is organizationally not very easy but we also consider running controlled AB-tests in the future.

5.3 Outlook

The enhanced completion rate during the three course runs is presumably mostly due to our modifications on the course based on the learners' feedback. This underlines the necessity of a participatory approach to the ongoing development of online courses and teaching in general.

In the future we plan to analyze further data of our rich data collection. We are currently analyzing students' solutions of programming exercises of an on-campus university course as basis for the definition of competencies for OOP and the automatic generation of feedback (Krugel et al., 2020); we plan to extend this analysis to the programming solutions of our MOOC as well. Furthermore, we will take a closer look at the discussion board questions which can give insights into, e.g., misconceptions of the learners.

We have just added another chapter to the course: Chapter 6 introduces inheritance and polymorphism and thereby rounds off the course to cover the most important object-oriented concepts. Additionally, we develop and adapt the course further based on the learners' feedback and the quantitative data.

In the long term, we aim to use LOOP as a general tool to analyze learning processes in object-oriented programming. The online setting allows to perform experiments and analyses on a scale much larger than in a regular classroom course and, furthermore, poses new research questions (Settle et al., 2014).

References

Alario-Hoyos, C., Delgado Kloos, C., Estévez-Ayres, I., Fernández-Panadero, C., Blasco, J., Pastrana, S., Suárez-Tangil, G., & Villena-Román, J. (2016). Interactive activities: The key to learning programming with MOOCs. In *European Stakeholder Summit on experiences and best practices in and around MOOCs (EMOOCS'16).* Books on Demand.

Alonso-Ramos, M., Martin, S., Albert Maria, J., Morinigo, B., Rodriguez, M., Castro, M., & Assante, D. (2016). Computer science MOOCs: A methodology for the recording of videos. In *IEEE Global Engineering Education Conference (EDUCON'16)*.

Anderson, T., & Shattuck, J. (2012). Design-Based Research. *Educational Researcher*, *41* (1), 16–25. https://doi.org/10.3102/0013189X11428813

Bajwa, A., Hemberg, E., Bell, A., & O'Reilly, U.-M. (2019). Student Code Trajectories in an Introductory Programming MOOC. In Unknown (Ed.), *Proceedings of the Sixth (2019) ACM Conference on Learning @ Scale - L@S '19* (pp. 1–4). ACM Press. https://doi.org/10.1145/3330430.3333646

Cooper, S., & Cunningham, S. (2010). Teaching computer science in context. *ACM Inroads*, *1*, 5–8. https://doi.org/10.1145/1721933.1721934

Crues, R. W., Bosch, N., Anderson, C. J., Perry, M., Bhat, S., & Shaik, N. (2018). Who they are and what they want: Understanding the reasons for MOOC enrollment. In *Proceedings of the 11th International Conference on Educational Data Mining, EDM 2018, Buffalo, NY, USA, July 15-18, 2018*. http://educationaldatamining.org/files/conferences/EDM2018/papers/EDM2018_paper_121.pdf

Delgado Kloos, C., Munoz-Merino, P. J., Munoz-Organero, M., Alario-Hoyos, C., Perez-Sanagusti-n, M., Parada G., H. A., Ruiperez, J. A., & Luis Sanz, J. (2014). Experiences of running MOOCs and SPOCs at UC3M. In *IEEE Global Engineering Education Conference (EDUCON'14)*.

Derval, G., Gego, A., Reinbold, P., Frantzen, B., & van Roy, P. (2015). Automatic grading of programming exercises in a MOOC using the INGInious platform. In *European Stakeholder Summit on experiences and best practices in and around MOOCs (EMOOCS'15)*.

Design-Based Research Collective (2003). Design-Based Research: An Emerging Paradigm for Educational Inquiry. *Educational Researcher*, *32* (1), 5–8. https://doi.org/10.3102/0013189X032001005

Eriksson, T., Adawi, T., & Stöhr, C. (2017). "Time is the bottleneck": a qualitative study exploring why learners drop out of MOOCs. *Journal of Computing in Higher Education*, *29* (1), 133–146. https://doi.org/10.1007/s12528-016-9127-8

Estler, C., & Nordio, M. *Codeboard*. http://codeboard.io/

Everitt, B. S., Landau, S., & Leese, M. (2001). *Cluster analysis*. Arnold.

Falkner, K., Falkner, N., Szabo, C., & Vivian, R. (2016). Applying validated pedagogy to MOOCs. In *ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'16)* (pp. 326–331). ACM. https://doi.org/10.1145/2899415.2899429

Fitzpatrick, J. M., Lédeczi, Á., Narasimham, G., Lafferty, L., Labrie, R., Mielke, P. T., Kumar, A., & Brady, K. A. (2017). Lessons Learned in the Design and Delivery of an Introductory Programming MOOC. In M. E. Caspersen, S. H. Edwards, T. Barnes, & D. D. Garcia (Eds.), *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17* (pp. 219–224). ACM Press. https://doi.org/10.1145/3017680.3017730

Garcia, F., Diaz, G., Tawfik, M., Martin, S., Sancristobal, E., & Castro, M. (2014). A practice-based MOOC for learning electronics. In *IEEE Global Engineering Education Conference (EDUCON'14)*.

Geldreich, K., Simon, A., & Hubwieser, P. (2019). Design-Based Research als Ansatz zur Einführung von Algorithmik und Programmierung an bayerischen Grundschulen. *MedienPädagogik: Zeitschrift Für Theorie Und Praxis Der Medienbildung*, *33* (Medienpäda). https://doi.org/10.21240/mpaed/33/2019.02.15.X

Gries, D. (2008). A principled approach to teaching OO first. *ACM SIGCSE Bulletin*, *40* (1), 31. https://doi.org/10.1145/1352322.1352149

Guo, P. J., Kim, J., & Rubin, R. (2014). How video production affects student engagement. In M. Sahami, A. Fox, M. A. Hearst, & M. T. H. Chi (Eds.), *1st ACM Conference on Learning@Scale (L@S'14)* (pp. 41–50). ACM. https://doi.org/10.1145/2556325.2566239

Hicken, A. (2018). *2019 eLearning Hype Curve Predictions*. https://webcourseworks.com/elearning-predictions-hype-curve/

Hubwieser, P. (2008). Analysis of learning objectives in object oriented programming. In R. T. Mittermeir & M. M. Syslo (Eds.), *Lecture notes in computer science, Informatics Education - Supporting Computational Thinking, 3rd International Conference on Informatics in Secondary Schools - Evolution and Perspectives (ISSEP'08)* (pp. 142–150). Springer. https://doi.org/10.1007/978-3-540-69924-8_13

Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., Pal, Y., Jackova, J., & Jasute, E. (2015). A global snapshot of computer science education in K-12 schools. In *ITiCSE Working Group Reports* (pp. 65–83). ACM. https://doi.org/10.1145/2858796.2858799

Krugel, J., & Hubwieser, P. (2017). Computational thinking as springboard for learning object-oriented programming in an interactive MOOC. In *2017 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1709–1712). IEEE. https://doi.org/10.1109/EDUCON.2017.7943079

Krugel, J., & Hubwieser, P. (2018). Strictly Objects First: A Multipurpose Course on Computational Thinking. In M. S. Khine (Ed.), *Computational Thinking in the STEM Disciplines* (Vol. 49, pp. 73–98). Springer International Publishing. https://doi.org/10.1007/978-3-319-93566-9_5

Krugel, J., Hubwieser, P., Goedicke, M., Striewe, M., Talbot, M., Olbricht, C., Schypula, M., & Zettler, S. (2020). Automated Measurement of Competencies and Generation of Feedback in Object-Oriented Programming Courses (preprint). In *2020 IEEE Global Engineering Education Conference (EDUCON).* IEEE.

Kurhila, J., & Vihavainen, A. (2015). A purposeful MOOC to alleviate insufficient CS education in Finnish schools. *ACM Transactions on Computing Education*, *15* (2), 1–18. https://doi.org/10.1145/2716314

Liyanagunawardena, T. R., Lundqvist, K. O., Micallef, L., & Williams, S. A. (2014). Teaching programming to beginners in a massive open online course. In *Building Communities of Open Practice (OER'14)*.

Luik, P., Feklistova, L., Lepp, M., Tõnisson, E., Suviste, R., Gaiduk, M., Säde, M., & Palts, T. (2019). Participants and completers in programming MOOCs. *Education and Information Technologies*, *24* (6), 3689–3706. https://doi.org/10.1007/s10639-019-09954-8

Luik, P., Suviste, R., Lepp, M., Palts, T., Tõnisson, E., Säde, M., & Papli, K. (2019). What motivates enrolment in programming MOOCs? *British Journal of Educational Technology*, *50* (1), 153–165. https://doi.org/10.1111/bjet.12600

Mayring, P. (2000). Qulitative Content Analysis. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, *1* (2), Article 20. http://nbn-resolving.de/urn:nbn:de:0114-fqs0002204

Moreno-Marcos, P. M., Muñoz-Merino, P. J., Maldonado-Mahauad, J., Pérez-Sanagustín, M., Alario-Hoyos, C., & Delgado Kloos, C. (2020). Temporal analysis for dropout prediction using self-regulated learning strategies in self-paced MOOCs. *Computers & Education*, *145*, 103728. https://doi.org/10.1016/j.compedu.2019.103728

Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2019). Exploring children's learning experience in constructionism-based coding activities through design-based research. *Computers in Human Behavior*, *99*, 415–427. https://doi.org/10.1016/j.chb.2019.01.008

Piccioni, M., Estler, C., & Meyer, B. (2014). SPOC-supported introduction to programming. In Å. Cajander, M. Daniels, T. Clear, & A. Pears (Chairs), *Innovation & Technology in Computer Science Education (ITiCSE'14),* Uppsala, Sweden.

Plomp, T. (2007). Educational Design Research: An Introduction. *An Introduction to Educational Design Research*, 9–35.

Reeves, T. C. (2006). Design research from a technology perspective. In J. van den Akker (Ed.), *Educational design research* (pp. 52–66). Routledge.

Settle, A., Vihavainen, A., & Miller, C. S. (2014). Research directions for teaching programming online. In *Proceedings of the International Conference on Frontiers in Education Computer Science and Computer Engineering (FECS'14)*.

Skoric, I., Pein, B., & Orehovacki, T. (2016). Selecting the most appropriate web IDE for learning programming using AHP. In *39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO'16)* (pp. 877–882). IEEE. https://doi.org/10.1109/MIPRO.2016.7522263

Staubitz, T., Klement, H., Teusner, R., Renz, J., & Meinel, C. (2016). CodeOcean - A versatile platform for practical programming excercises in online environments. In *IEEE Global Engineering Education Conference (EDUCON'16)*.

Striewe, M., & Goedicke, M. (2013). JACK revisited: Scaling up in multiple dimensions. In *Lecture notes in computer science, 8th European Conference, on Technology Enhanced Learning (EC-TEL'13): Scaling up Learning for Sustained Impact* (pp. 635–636). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-40814-4_88

Topali, P., Ortega-Arranz, A., Er, E., Martínez-Monés, A., Villagrá-Sobrino, S. L., & Dimitriadis, Y. (2019). Exploring the Problems Experienced by Learners in a MOOC Implementing Active Learning Pedagogies. In M. Calise, C. Delgado Kloos, J. Reich, J. A. Ruiperez-Valiente, & M. Wirsing (Eds.), *Lecture notes in computer science. Digital Education: At the MOOC Crossroads Where the Interests of Academia and Business Converge* (Vol. 11475, pp. 81–90). Springer International Publishing. https://doi.org/10.1007/978-3-030-19875-6_10

Vihavainen, A., Luukkainen, M., & Kurhila, J. (2012). Multi-faceted support for MOOC in programming. In R. Connolly (Ed.), *ACM Digital Library, Proceedings of the 13th Annual Conference on Information Technology Education* (p. 171). ACM. https://doi.org/10.1145/2380552.2380603

Wing, J. (2006). Computational Thinking. *Communications of the ACM*, *49* (3), 33–35. https://doi.org/10.1145/1118178.1118215

Zheng, S., Rosson, M. B., Shih, P. C., & Carroll, J. M. (2015). Understanding Student Motivation, Behaviors and Perceptions in MOOCs. In D. Cosley, A. Forte, L. Ciolfi, & D. McDonald (Eds.), *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15* (pp. 1882–1895). ACM Press. https://doi.org/10.1145/2675133.2675217

Acknowledgements

This work was partially supported by the Dr.-Ing. Leonhard Lorenz-Stiftung (Grant No. 949/17). We thank Alexandra Funke and Marc Berges for developing parts of the course. We would furthermore like to thank all course participants for taking the time to provide their valuable feedback and the reviewers for the comments that helped to improve the exposition.