Evaluating Digital Worksheets with Interactive Programming Exercises for K-12 Education

Sebastian Serth
Hasso Plattner Institute
University of Potsdam
Potsdam, Brandenburg, Germany
sebastian.serth@student.hpi.de

Ralf Teusner
Hasso Plattner Institute
University of Potsdam
Potsdam, Brandenburg, Germany
ralf.teusner@hpi.de

Jan Renz
Hasso Plattner Institute
University of Potsdam
Potsdam, Brandenburg, Germany
jan.renz@hpi.de

Matthias Uflacker
Hasso Plattner Institute
University of Potsdam
Potsdam, Brandenburg, Germany
matthias.uflacker@hpi.de

***Abstract*—This Research Full Paper presents insights from digital worksheets with embedded interactive programming exercises tailored for high-school students new to programming. Computer science teachers often incorporate existing videos, quizzes, and practical programming exercises from Massive Open Online Courses (MOOCs). However, teachers' options to adapt the content to their specific needs are currently limited. Based on a qualitative survey with thirteen teachers, we developed a software prototype which allows teachers to create their own interactive worksheets consisting of texts, videos, quizzes, and practical programming exercises. Additionally, teachers can embed and further customize existing exercises from MOOCs. Further, we enable teachers to gain deeper insights by providing results from automated submission analysis, thus uncovering knowledge gaps and fostering content-driven in-class discussions. Our evaluation shows that the concept was well received by students and teachers alike: Teachers noticed the possibility of a shift in their role from a lecturing instructor to an individual tutor, as students are enabled to learn at their own pace and receive specific, direct feedback based on automated unit tests. Interactive worksheets, as an integrated part of digital education, thus foster informed teacher interventions as part of an individualized student learning process.**

Keywords—Digital Worksheets, K-12, Programming, Computer Science Education, Online Learning, School, MOOC

I. Introduction & Motivation

In K-12 education, teachers generally either present new topics to their students or practice and repeat previous subjects. For this, they have the choice between using an existing schoolbook with given content or creating their own worksheets. In the following, we concentrate on the worksheet scenario. When creating worksheets, teachers usually reuse existing content from schoolbooks or from other resources. Regardless of their sources, they are able to adapt the content to their needs when composing a new worksheet. This adaption is one of the biggest advantages teachers have: The freedom and possibility to tailor each part of the worksheet to fit their current situation in class to best support their students. Traditionally, these worksheets are distributed as paper copies by the teachers whenever needed. For teachers, this distribution form has some advantages as they know prior to the lesson that their material is copied and thus ready and that they do not need to rely on anything else (such as further technologies) to conduct their lesson.

Traditional, paper-based worksheets also have some general drawbacks when being compared to digital resources. Regardless of the environmental or financial costs, printed worksheets lack support for any form of interactivity or for the adaption of additional content to specific students. Many teachers strive for interactivity to foster active participation of their students. Thus, given the choice, today's teachers prefer using digital resources for their lessons if available. By doing so, they circumvent the missing interactivity and also get the benefit of links to other rich content (independent of the location, e.g., a web resource).

For computer science (CS) classes, especially those teaching programming skills, additional drawbacks arise. Similar to other classes, teachers prepare content to focus their students' attention on the most important parts. While a figure, e.g., a schematic representation of a heart in biology classes, can be easily annotated by students on a sheet of paper, it's difficult to work with programming source code on traditional worksheets, particularly if students must execute the code to observe the program behavior. If the code is distributed in paper form, students are required to manually typewrite the given code which is time-consuming and error-prone. Depending on the possible mistakes students make, it might even be difficult for teachers to find the bug in a chaotic "rewrite" of their own template. A second issue occurs when the teacher decides to discuss results or when collecting students' work. As every individual learner developed own solutions and probably saved it locally on the PC in front of them, the teacher has no direct access to the source code files. However, the teacher needs access for collecting the submissions and thus is required to rely on the assistance of the students to copy all files required to compile and run the program to a provided location. This increases the complexity for all parties involved and the time effort for the teacher when compared to collecting paper-based submissions. However, the work created by students is only available in a digital format and thus students cannot submit their work on a sheet of paper to the teacher. Based on the collection of these files, the teacher might also ask students to present their work in front of the class using a projector. When having no access to the source files, some teachers might have the possibility to share a student's desktop on their machine and thus circumvent the need for copying the files manually. However, this requires further software to be installed on the computers, compatible network architecture and is no viable solution for schools with a "bring-your-own-device" policy.

A possible solution is bringing the cloud concept to schools. One approach is the HPI Schul-Cloud (literally "School-Cloud") being developed to fit the German education system [1]. Currently, the offer, however, lacks specific support for CS classes.

Inspired by Massive Open Online Courses (MOOCs) [2], teachers wish to access tools used in e-learning environments for several reasons: First of all, teachers like the didactical approach and the methodology used in the courses. The course videos often use a different approach to explain abstract concepts than the one used in the schoolbook. In addition, online courses include interactive content, such as simple multiple-choice quizzes for self-assessment or interactive programming exercises. For the latter, the platforms usually provide a simplified web interface that is tailored for novices and offers additional support capabilities. For example, teachers appreciate that the code execution platforms used in MOOCs give direct feedback to students while working on the code. This direct feedback frees teachers from providing the same simple feedback over and over again and allows them to concentrate on students strongly struggling with the given task. In addition, these web-based code execution platforms are maintained by the respective platform provider eliminating administrative overhead for schools. If not using online tools, the PCs used in CS classes need to have the corresponding language tools (such as a compiler or interpreter) to be installed in conjunction with correctly configured developer tools. In comparison, everything that is required to participate in MOOCs is a modern web-browser and a stable internet connection.

Given the content available through MOOCs and the provided tools, some teachers currently embed resources from these e-learning tools in their classes. However, MOOC platforms are not tailored for this use case as they were built with a different design goal in mind. This results in two substantial drawbacks: (1) Teachers are unable to adapt the course material hosted online to their needs and the knowledge level of their students — the content is not editable and feels "static". In addition, (2) the MOOC platforms prevent teachers from getting insights into the learning progress of their students making teachers dependent on the feedback of their students.

To tackle the drawbacks and improve teachers' current situation, we address the following research questions in this paper:

RQ1. How can we enable teachers to reuse and adapt exercises (e.g., from MOOCs) and create their own interactive worksheets?

RQ2. Which tooling support do teachers need to help students struggling with given programming exercises?

RQ3. Which of the results can be transferred to the general MOOC context?

II. Related Work

This work is based on research in the context of interactive elements using ICT (information and communications technology) equipment in different school lessons and prior research on how to embed content from MOOCs in hybrid classes. Furthermore, we highlight different didactical approaches on how to use programming exercises in K-12 education. The following paragraphs give a short overview of the work this paper builds on.

A. Interactivity in Computer Science lessons

In 2013, Othman et al. stated that CS classes "contain numerous abstract concepts that cannot be easily explained using traditional educational methods" [3]. The authors compare different forms of online interactive content that can be used in addition to traditional learning methods, such as simulations, tutorials or quizzes. This is in line with the findings from Merchant et al. who describe that the use of games, simulations and virtual worlds improve the learning outcome gains in K-12 [4]. Existing MOOCs, especially those designed for a younger audience, make use of this strategy and tell a story that requires the help of the learner to solve a given problem [5]. Whenever it comes to interactive elements, Merchant et al. also highlight the importance of the individual experience. Compared to group interactivity, situations which actively require the involvement of every single learner enhance the student performance. Beauchamp and Kennewell further describe the advantages of using ICT as it supports the learner influence on content and methodology [6]. However, they also state the usage of those resources needs to be well embedded in the lesson to push the use of technology in the background while focusing on the desired content.

B. Interactive Worksheets

While little research is available specifically for the use of interactive worksheets in CS classes, the concept has been tested in other areas. In 2000, Leslie-Pelecky showed that interactive worksheets in an introductory physics course increased the student/teacher interaction [7]. The main cause is the way how worksheets and embedded tasks are structured: They follow a common goal and, if designed appropriately, build on each other, starting with easier tasks. Usually, students work in the same order as the exercises appear on the worksheets. As soon as students start struggling, the teacher gets an impression which exercise is causing the students' problems and thus can intervene orally. Another advantage of interactive worksheets is described by Blayney and Freeman: Students are put in the position to learn at their own speed and pace [8]. This supports the individualization of the learning process, especially with an interactive worksheet that gives direct feedback to the learner. The researchers describe this as a possible mechanism to support students where they need the most help. With respect to the level of interactivity and feedback, Blayney and Freeman also concluded in a previous paper that better exercises than multiple-choice quizzes are needed and that it is important to provide frequent feedback to learners [9]. While multiple-choice quizzes are technically easy to verify, they ask only for a subset of the content learned and do not allow the students to demonstrate their understanding of more complex content such as approaches or methodologies.

C. Programming Exercises in K-12

For CS classes with a focus on programming education, an important part of the learning process is to apply the knowledge learned by writing source code (amongst other competencies, such as modeling). Besides simple read-eval-print loops (REPL), teachers wish that their students learn to code small programs. A good way to do so is described by Isomöttönen et al., who propose to provide students with scaffolded source code [10]. Besides that, scaffolding also helps students to make progress in a shorter period of time and abstracts more advanced concepts for novices. Other approaches are discussed by Hubwieser et al., as they raised many questions regarding the content and methodology of programming education [11]. In this paper,

we tackle some of these questions, especially those with respect to the tools and the environment required to support teachers in their teaching practice by reusing and customizing content, e.g., available from MOOCs.

D. Integrating MOOCs in classes

The majority of teachers who embedded MOOCs in their classes were satisfied with the results. This is due to the excellent content available and the use of other teaching approaches, according to research by Griffiths et al. in 2014 [12]. However, M. Israel showed that fitting existing courses that are not tailored for in-class usage is a huge challenge, for example regarding the engagement of students and the learning effectiveness [13]. A potential downside is described by Griffiths et al.: When comparing traditional face-to-face only lessons and hybrid lessons using MOOCs, student satisfaction is lower in hybrid lessons, while overall learning results remain on an equal level [12]. Counter measurements to prevent a decrease of satisfaction need to be taken by the teacher, for example by providing time within lessons to discuss the online content and to answer upcoming questions. Caulfield et al. have shown that students prefer this direct communication for questions initiated by them over asking for help in an online community as typically available in a MOOC [14]. Teachers also benefit from the in-class discussions as they allow them to get an understanding of the current progress and possible problems of their students. Generally, teachers have no direct way to access student activity in MOOCs [13] as MOOC platforms offer only two roles: MOOC instructors and participants. The role of a teacher or teaching assistant is currently not represented. Also, additional requirements imposed by school usage such as stretched course runtimes and awareness of school holidays are usually not represented.

E. Learning Analytics

Getting insights about the learning data is achieved with so-called learning analytics and is not only valuable for teachers to perform grading-related tasks but primarily of interest for them to improve their lessons and support students who need extra help [15], [16]. Teachers can use the information provided through the platform to prepare upcoming lessons, for example by including a repetition to tackle existing gaps in understanding of a topic. Learning Analytics, such as the individual learning path, the time spent, or points achieved also allow for predicting students' performance, as described by Williamson in 2016 [17]. Concerning programming exercises, Blikstein describes that the frequency of code runs and source code changes allows inferring the coding behavior, opening an additional window into students' cognition and approach [18]. Berland et al. describe that these results can either be used by a teacher (as described above) or to enable automated feedback to the learners [19]. Besides the increase in self-awareness of possible problems, Long and Siemens found out that showing personalized progress information is motivating for learners when compared to learning goals or peers [20]. Current research, for example as conducted by Rohloff et al., evaluates how to integrate learning objectives and corresponding feedback into MOOCs [21].

III. Approach

We conducted interviews with thirteen CS teachers in K-12 education, two principals and five students. In these interviews, we gathered insights on the approaches teachers use in typical lessons, the resources they use and the equipment that is available for them during (computer science) lessons. For the creation of worksheets, our interviewees highlighted the possibilities to (re-)use fitting content from websites and other resources. Besides identifying the aspects they like and dislike about their current situation, we also asked them to share some of their current worksheets with us. Finally, we distilled three main issues of traditional worksheets regardless of the distribution form:

(1) Lack of interactivity: Some use videos or online games, such as Pac-Man, to motivate students to work on a given task related to what they've experienced in the interactive content (e.g., a path-finding algorithm for ghosts in the context of Pac-Man). To allow students to work on the most challenging part, teachers then provide a framework with a few source code files. In this scenario, they distribute the code via a file server together with a PDF of the worksheet with embedded links to an instance of the online game or a video hosted online (e.g., on YouTube).

(2) Lack of adaptability: Others use content from MOOCs, e.g., the programming courses on Java and Python available on openHPI [22]. In these courses, an online code execution environment called *CodeOcean* is used to allow coding without the need to install further software [23]. While teachers like the approach used in these online courses, they wish to have the opportunity to customize exercises or to add additional references to the schoolbook available to the class. When it comes to changing the exercises, teachers want to adapt the exercise description and the scaffolded code, as well as the automated feedback the platform provides to learners. In their experience, students focus on achieving a maximum score while sometimes losing focus of the actual learning goals. Consequently, teachers might either accept exercises as they are or manually copy them and loose key features of the platforms, such as automated feedback.

(3) Lack of usability: Some other teachers remarked on the disadvantages of traditional submission systems based on uploading and downloading files by hand. They require too many manual steps to be practical for use in lessons.

Analyzing the shared worksheets, we further derived other needs not directly mentioned by the interviewees. For example, some teachers distributed their worksheets as editable text documents instead of PDFs as they included open questions that should be answered by the students in-line on the worksheets. These teachers need support to provide each student with an individual, editable copy of the digital worksheet. Others choose to put the content in learning management systems (LMS), such as a Moodle instance that also allows content creation and submission handling. With an LMS, open text questions might be asked together with simple multiple-choice quizzes allowing the automated generation of feedback for students. According to the teachers we interviewed, Moodle is seen as powerful, but also has a reputation for being too complicated for all parties involved. Concerning computer science classes, they still do not fit all the requirements teachers have, hence dedicated assistance is needed. By looking at the exercises included in the worksheets, we also learned that teachers sometimes wish to provide

a full working source code example allowing students to observe the execution flow or a programming pattern used. Presenting the code on PDFs or printed worksheets without the possibility to execute it gives the teacher control over the level of detail and granularity but requires some imagination from students to understand what the given code snippet does. Providing the full code as downloadable files enables students to execute it but reduces the influence a teacher has on the presentation of the relevant code in question as, for example, no code could be hidden.

All current solutions used by teachers in-class to distribute content and assignments have some drawbacks. Most of them are caused by the decoupled source code and other learning materials, resulting in the need for a context switch when working on a given topic. If some of those drawbacks are addressed, for example by using MOOCs and their integrated programming environments, teachers have to accept their decreased capabilities in editing the provided content and the lack of access to their students' submissions and progress. Consequently, there is currently no option that fully meets the needs of teachers.

Therefore, our approach is to build a novel kind of digital worksheets with embedded programming exercises to tackle these issues and provide a seamless experience for students and teachers. The prototype we built is based on the findings from the initial interviews described in this section as well as early and frequent feedback from teachers and students. The prototype was evaluated by teachers with their students in real lessons.

IV. Concept

Our vision is that teachers are enabled to mix existing editable content in worksheets with their own additions. While it is easy for teachers to adapt a text on a worksheet (worst case by copying it manually and subsequently changing it), it gets trickier with other content types, regardless of the editor used. Videos can be downloaded and embedded on other sites but editing requires manual labor and some familiarity with video editing tools. In fact, none of the teachers we interviewed was interested in editing a video. Either, they embed the full video as it is, or they do not use it. Some thought about recording their own videos but did not find enough time to prepare those. Therefore, we concluded the need to support embedding videos from MOOCs and also videos from common video hosting platforms, such as YouTube. For quizzes, two major use cases were distinguished: Either (1) self-tests for the learner or (2) graded or ungraded assignments with results being checked by the teacher.

Major challenges for teachers are the embedding and customization of programming exercises from MOOCs. For that, we decided to show exercises on the same page as other elements to reduce context switches and to fit our worksheet analogy. Offering a free combination of learning materials, it is possible to replace other forms of worksheets whether analog, PDF- or text-based. For each content type, a plugin is provided. The default set of plugins offered by our worksheet editor include those required to display images, embed videos or create multiple-choice quizzes.

Besides providing a tool for creating and delivering content, our concept includes a deep integration of learning analytics. As mentioned in the previous section, this data is usually not available to teachers even though they are likely to benefit from a view on prepared data. We designed the architecture of our digital worksheets to support learning analytics per plugin instance (meaning "per exercise" for programming exercises) and a high-level overview of the whole worksheet. The teacher's view will show a short excerpt of key metrics below each part of the worksheet to give teachers a quick overview of the current status of their students. Further, a detailed analysis per programming exercise should also include an overview of time spent by students to solve an exercise, exceptions, and errors raised during the work and which questions in form of help requests were posted in the context of the given assignment.

In addition to a suitable content editor, teachers also need direct access to a code execution platform, in our case *CodeOcean*. Up to now, *CodeOcean* was designed with MOOCs in mind, resulting in a simple authentication model of administrators drafting exercises or managing the platform and students using the system to solve exercises without any backend access. In *CodeOcean*, each exercise comprises one or more files visible or editable by the user, which already contain some scaffolding, and hidden unit tests to check the correctness of the student's submission. The result of a test-run is then used to provide automated feedback to students and to direct them to the mistakes they made. For the desired use in class, this is not sufficient as it completely omits the teacher role. Therefore, our concept implies that *CodeOcean* is extended to support a third role, called teacher. It allows teachers to (1) create new exercises that (technically) do not differ from existing exercises created by MOOC instructors and (2) browse a list of public exercises and clone those if desired. Given these changes, *CodeOcean* can become the starting point for teachers to build a repository of exercises. These might then be shared using *CodeHarbor*, a platform designed to find, share and improve exercises which are fully compatible with *CodeOcean* [24]. For learners belonging to a class the teacher educates, the teacher is enabled to view learning analytics and access student submissions on demand.

During our initial interviews, we found that providing the same tools that MOOC instructors use to create exercises including unit tests for automated learner feedback is not sufficient and suitable for teachers. Many teachers we consulted either did not know about unit tests at all or admitted they are unable to write their own tests. As a result, none of our interviewees is willing to invest the time and effort to create tests for their own exercises. On the other side, teachers described the feedback given to learners on the basis of the tests as one of the biggest advantages *CodeOcean* offers when compared to other solutions.

The proposed concept aims to enable teachers to create worksheets with customized content from MOOCs enriched by own exercises and other references (see Fig. 1 for an example). The documents are stored in the context of the HPI Schul-Cloud. In order to further provide fine-granular control for teachers during lessons, content in the worksheet can be unlocked or even edited live without the need to rely on third-parties (e.g., MOOC providers). When adding an interactive programming exercise to a worksheet, teachers should also have the possibility to specify embedding options. For example, teachers wished to disable the automated feedback via unit tests or to hide the score achieved by students, e.g., for an exam. The worksheet editor needs to support these restrictions and enforces them (if applied by the teacher) when students access embedded exercises.

V. Implementation

Technically, four main components are involved to provide the functionality of our concept with embedded programming exercises: (1) The HPI Schul-Cloud platform, (2) a content editor called *edtr.io*, (3) the programming execution environment *CodeOcean*, and (4) a Learning Record Store (LRS). In our prototype, the HPI Schul-Cloud acts as the identity provider, manages classes consisting of students and teachers and grants access to course content, such as digital worksheets. By opening a worksheet, the identity of each user together with the role (student/teacher) is passed to the worksheet editor to customize the view being rendered for this user. It is also required to save a user's progress to continue working later and to store submissions, which might be collected and inspected by a teacher.

edtr.io persists all user progress together with documents in a relational database accessed through a GraphQL backend. This architecture allows to securely restrict the information that is visible for users and to implement live updates through a WebSocket connection. This is especially useful to provide teachers with control over the content available to students. It allows teachers to hide or unhide specific information throughout the lesson. The change will instantly appear on all connected devices — an advantage teachers never had before. In addition, the technology stack eliminates the need for a user to explicitly save progress, thus minimizing the danger of accidental data loss.

The *edtr.io* backend stores the current user progress and submissions of some native plugins, such as answers given for multiple-choice questions in a dedicated database. In addition, selected learning data is sent to Learning Locker[1], an open-source LRS deployed in the HPI Schul-Cloud, which accepts and processes learning activity data defined using the Experience API (xAPI) standard[2]. Each activity is described with a statement consisting of up to four elements. Statements follow a simple "actor – verb – activity" structure and may contain additional attributes. For example, a valid statement is: John (actor) passed (verb) test A (activity) with 65% (additional attributes). Based on xAPI data, the LRS creates custom views and graphs for a suitable visualization.

The open plugin architecture of *edtr.io* allows teachers to embed a wide range of different content. Some plugins, such as the video player, are user-agnostic and only differ in their appearance based on the role of a user, for example, to allow editing textual content. Others, such as a multiple-choice plugin, require knowledge about the current user to associate answers with the corresponding user. In our concept, it is the responsibility of each plugin to display the content or to process integrated learning analytics, as domain knowledge is helpful to create customized views providing a meaningful abstraction of the data.

Therefore, each plugin is enabled to publish learning analytic data that is either directly handled by *edtr.io* and also processed directly (e.g., to provide instant feedback) or stored for future reference in the Learning Locker. The multiple-choice plugin is an example of a plugin completely integrated into *edtr.io* using the same GraphQL backend to store content and user data. For external tools or more complicated use cases, building *edtr.io* plugins is not feasible. Many third-party web apps designed for educational use cases offer an integration via the Learning Tools Interoperability (LTI) standard[3]. This allows a learning tool to be linked from an embedding site with some parameters, e.g., to specify an exercise and hand over selected user information. Both, the HPI Schul-Cloud and *edtr.io* integrate the LTI standard as a so-called "tool consumer" to allow starting learning apps. Furthermore, *CodeOcean* provides an LTI interface as a "tool provider" to offer direct access to programming exercises. LTI uses standard web technologies so that tools can be accessed through a link or be directly embedded via an iFrame. The parameters used when launching a tool do not only describe the user or the exercise but can also specify an endpoint for syncing a score between the tool provider and tool consumer or define application-dependent attributes. *CodeOcean* uses custom parameters to realize feature restrictions and enforce restrictions applied by the teacher. In total, we added support for eleven different restrictions allowing teachers to customize the integration of *CodeOcean* exercises. For example, teachers can disable automated feedback or include a read-only exercise for a demonstration of a concept. While these parameters are part of the request sent through the learner's web browser, all LTI messages are signed and verified to prevent any tampering of the data.

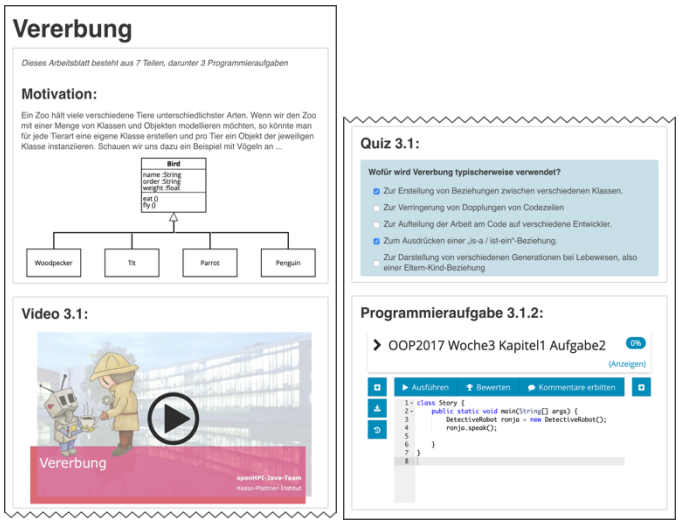

Fig. 1. Exemplary worksheet in German on the topic "inheritance" as used in our study, consisting of text, an image, a video, a multiple choice quiz, and a programming exercise.

In the context of school education in Germany, privacy and legal considerations have to be a major part of the overall product design. Therefore, all external plugins within a digital worksheet only get limited access to user data if at all. *CodeOcean* needs to identify returning learners to enable continuing previous work. Therefore, the HPI Schul-Cloud is configured to create a pseudonymous ID for each user and service, which is the only person-related information *CodeOcean* requires. Additionally, to represent the student-teacher relationship on the programming platform, each tool launch via LTI also includes a context ID and the role of each user (student/teacher). The context ID must be a unique identifier of a school class and grants teachers access to analytics about their students exclusively.

[1] https://learninglocker.net

[2] https://xapi.com

[3] http://www.imsglobal.org/activity/learning-tools-interoperability

Based on the unique course ID, *CodeOcean* builds so-called study groups with all learners of the given class being members. Depending on the context a student uses to access a *CodeOcean* exercise, the progress is associated with the study group and thus only visible to the teacher if the study groups match. This allows a student to access and implement exercises through a MOOC or another study group without exposing that data to an unrelated teacher in this constellation.

Authorized teachers can access the learning analytics through the LRS offered by the HPI Schul-Cloud. Additionally, they get more detailed information about their students in *CodeOcean* via a live dashboard summarizing activity per exercise (see Fig. 2). It shows the time spent by the students in correlation with the points achieved for a given exercise. In addition, the dashboard also shows live questions asked by the students while working on the assignment. The teacher can access the dashboard either as a standalone page within *CodeOcean* or as an embedded view within the worksheet. Further, the dashboard will allow a comparison with MOOC learners or will show a list of exceptions that occurred. These data are already available; for example, exceptions are extracted from the command line output and displayed separately to help students to identify the mistakes they made.

VI. Evaluation

To ensure that our concept and prototype fit the needs of users, we regularly asked for feedback within 27 interviews in total. We did not only use those contacts for the initial need finding but also to clarify questions that arose during further development. The teachers we interviewed use different tools and approaches in their lessons and have various backgrounds. While some use the HPI Schul-Cloud or an LMS regularly, others do not want to work with these tools or are unable to use them due to organizational barriers. Some have used MOOCs with their classes before, while others have not done so yet. We also presented numerous versions of our prototype to teachers and students to incorporate their feedback as early as possible.

Measuring effects, it is important to consider how students and teachers evaluate our approach and to understand how digital worksheets with interactive programming exercises change the in-class situation. Therefore, the goal of our evaluation was to get early feedback from typical lessons using our prototype.

To minimize the side effects of using a web-based programming environment and online videos, we conducted our evaluation with a class that was learning Java with a free MOOC on openHPI. In that course, videos are followed by multiple choice questions and practical programming exercises in *CodeOcean*. For a first test run, we prepared two worksheets on the Java topics “methods” and “inheritance” in tight consultation with the teacher and inspired by the openHPI MOOC. Both worksheets were used in the same 90-minute lesson, starting with the worksheet on “methods”. From teachers and previous iterations of our online courses, we know that the “inheritance” topic is one of the more difficult concepts in programming education. By starting with an easier repetition on “methods”, we were able to reduce possible distraction introduced through the digital worksheets for the second, more complex subject. Each worksheet

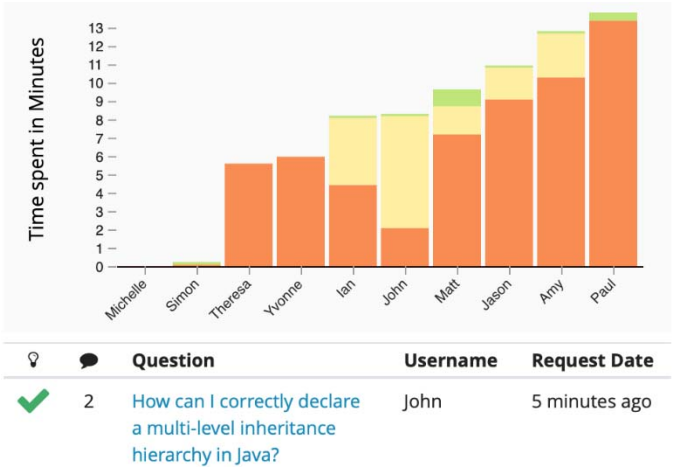

Fig. 2. The live dashboard available to teachers during a lesson in *CodeOcean*. The graph represents the working time with different colors indicating the current progress of students: orange (< 50%), yellow (50%-89%), green (⩾ 90%). In addition, recent questions are shown.

consisted of a mix of texts, images, videos, multiple-choice quizzes and several practical programming exercises in-lined.

A. Methodology

The designed worksheets were used during a regular lesson by a combined class of 21 students in their last two years of high-school. We decided not to join the class in person to prevent any disturbance due to our attendance. Therefore, it was the teacher’s responsibility to introduce the topic as he normally does and to guide the students through the material. After the lesson, the students were asked to provide anonymous feedback through our survey. Furthermore, we consulted the teacher directly to learn more about the in-class situation and his impressions. In the survey given to students, we asked what they liked or disliked about the concept in general. Additionally, we asked which advantages or disadvantages the different mediums, i.e., the schoolbook, traditional worksheets, static PDF worksheets, and our approach offer. Moreover, we assessed the Net Promoter Score (how likely it is that students would recommend the digital worksheet to friends) as described by Reichheld [25]. We further embedded modules from a questionnaire called “modular evaluation of key Components of User Experience” (meCUE) [26].

Eight weeks after the first test with two worksheets, we conducted a second evaluation with a third worksheet on the Java topic “abstract classes”. This time, we also included a reference to the schoolbook as students had to work on their own with the teacher being absent. The experiment was complemented by a survey with three questions comparing the book and the embedded video with regard to the comprehensibility, the perceived enjoyment and the preferred source for the repetition of content.

Furthermore, we introduced our prototype with sample documents and learning analytics to teachers on didacta, one of the largest European education trade fairs. During this event, we gathered different ideas from teachers and asked for feedback using the standardized User Experience Questionnaire (UEQ) [27]. It includes a benchmark based on 401 studies and about 18.000 participants to compare results from own studies.

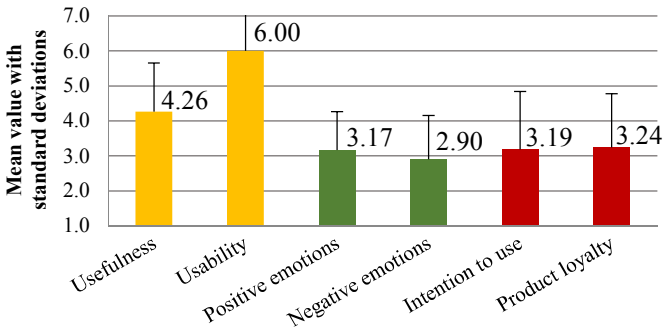

Fig. 4. Results of meCUE modules I (yellow), III (green) and IV (red). (N=7)

B. Results

After the first lesson including our worksheets, the teacher reported that his students got along well. He valued the deep integration and material mix we provided and repeatedly expressed the intention to create equivalent worksheets himself. From his observation, students revisited previous parts of the worksheets more often while working on the programming exercises than they did in the setting with the MOOC.

From the 21 students who accessed the first worksheet, nine students participated in our voluntary survey. Eight students valued the general concept and the overall design of our interactive worksheets. The students further appreciated the provided overview and structure on the content of a lesson. They also valued the general availability of online content, as this allowed them to access content independent of their current location, e.g., while commuting in a bus on their way to school. We received a Net Promoter Score (NPS) of -38 (ranging from -100 to +100), based on eight answers in our survey. In contrast, answers to the meCUE show an overall score of 2.4 on average (ranging from -5 to +5; N=7). Other scores in the meCUE (Fig. 3; ranging from "strongly agree" = 7 to "strongly disagree" = 1, with their respective average score based on seven replies) were usability (6.0), usefulness (4.3), positive emotions (3.2), negative emotions (2.9), intention to use (3.2) and product loyalty (3.2).

The second survey comparing the schoolbook with videos from a MOOC only got three answers: One student preferred the video for learning and attributed it was more enjoyable than the schoolbook, while the other liked both the same. The two students agreed to use both resources for repeating the content. A third student preferred the schoolbook in all categories and expressed a general rejection of learning with digital resources.

Out of 17 teachers who tried our prototype (student and teacher role), twelve filled the UEQ. The scales tested turned out to be mostly rated good or excellent (cf. Fig. 4). The stimulation (mean 1.7) provided through the use of interactive worksheets was rated the best, together with attractiveness (1.7), dependability (1.5), efficiency (1.5) and novelty (1.1) rated good. The perspicuity (1.4) was seen above average with the highest improvement potential, compared to the UEQ benchmark.

C. Discussion and interpretation of the results

The results gathered with the questionnaires are a first evaluation and motivation to further research the impact of interactive worksheets. The limited responses from students and teachers allow cautious conclusions to be drawn. The responses of

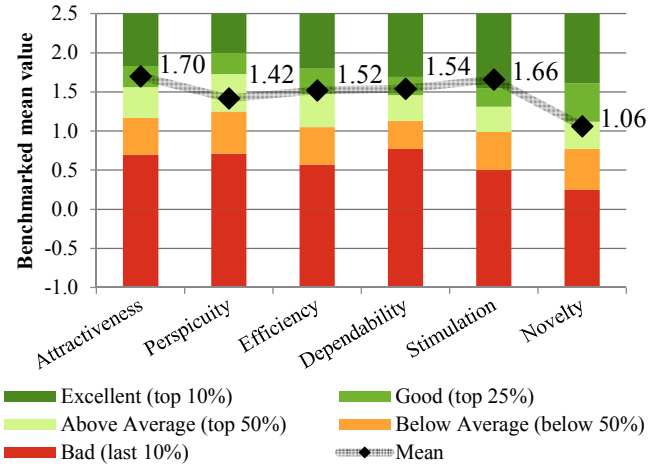

Fig. 3. Measured scales of the UEQ showing mean values in comparison with the method benchmark. (N=12)

students show contradictions such as the difference in the NPS (-38) and the meCUE overall experience (2.4). Free text answers also showed minor contradictions, hinting that students might have misinterpreted parts of our survey. The negative NPS might be explained by technical issues caused by connection problems from within the school network as the overall experience was valued, which was also confirmed by the teacher.

We further draw three main conclusions from the text comments of the students received in the surveys: (1) When comparing interactive worksheets with MOOCs, students highlight the improved structure and the enhanced overview offered through the worksheets. (2) They value the mix of different content types on a single web page and see no additional advantage in accessing the MOOC platform instead. (3) Traditional worksheets are seen as static and inflexible; however, students are sometimes allowed to use traditional worksheets for their reference in a written exam. As digital devices are usually not allowed, we derived the need to work on a print mode in the future. Overall, students value the integration of online learning content and the introduction of hybrid classes. In their opinion, MOOCs offer state-of-the-art knowledge while schoolbooks can be outdated.

The biggest advantages of interactive worksheets are voiced by teachers. They favor the efficiency offered by *edtr.io* to create individual worksheets with customized programming exercises. Our qualitative survey shows that our concept meets the requirements computer science teachers have and the results of the UEQ support the proposed design. Due to the insights provided by learning analytics, teachers feel better prepared when planning upcoming lessons and gain a better understanding of the reasons why their students struggle with given exercises. During lessons, teachers observed that students learned individually at their own pace and rearranged the topic order to their needs. Therefore, teachers reported that the worksheets helped to change their own role from an instructor of the class to an individual tutor. The automated feedback freed up some of their time, which they were now able to spend on helping individual learners. The positive feedback of teachers using our initial prototype of digital worksheets in their lessons, combined with the repeatedly voiced demand to create own worksheets, underlines the relevance and practical applicability of our concept.

VII. Outlook and Future Work

Identified shortcomings such as the missing usability of digital worksheets during exams should be solved with a feature to print out individual filled worksheets or the material of a complete lecture series. While a simple print-out of an interactive worksheet provides students with a copy of their progress, many of the advantages of digital worksheets cannot be preserved. Traditionally, the exam is also paper-based, resulting in the absence of features students used during learning. Future work should, therefore, concentrate on the special requirements during exams and introduce an "exam-mode" for interactive worksheets. In this scenario, teachers need to get increased control over resources accessed by students and the features enabled in the programming exercises, such as the set of limitations available in *edtr.io* and *CodeOcean*, e.g., by introducing a time limit and an explicit submission handling.

A cockpit view for teachers, showing the students' progress and allowing presentation and discussion of individual solutions is valuable in lessons and exam situations. The presentation view enabling teachers to do an in-class comparison of different solutions should be evaluated against currently used approaches.

For exam usage of our tool, we plan to further improve the technical stability. Despite already being stable in MOOC usage, we consider shifting from our current custom container pooling to Docker Swarm or Kubernetes to reduce maintenance effort and minimize error scenarios. Future work will also concentrate on ways how to provide debug functionality for novices.

As the adoption of interactive worksheets highly depends on the ease of use for teachers to create own worksheets and the variety of already existing documents, we will further investigate options to share worksheets and embed other types of pre-existing Open Education Resources (OER). Lastly, we will rerun our experiments with larger study groups to achieve reliable and potentially generalizable outcomes.

VIII. Conclusion

Computer science education in high-schools includes practical programming exercises which impose technical requirements on school computers. Therefore, teachers like online programming environments that can be used without any local setup. Those environments are available as stand-alone web applications or integrated into MOOCs including learning content and beginner-friendly programming exercises. Computer science teachers value the online content and partially reference it in traditional worksheets, which only offer a cumbersome and static user experience. To enable teachers to adapt learning content to the respective needs of their class, we developed the concept of interactive worksheets with embedded programming exercises. On the basis of our experiments, we are able to answer our research questions:

RQ1. How can we enable teachers to reuse and adapt exercises (e.g., from MOOCs) and create their own interactive worksheets?

Teachers not only need access to the same authoring tools available to MOOC instructors but require further technical help in creating equivalent exercises. In addition, teachers should be enabled to access and share exercises with colleagues. In our concept, all exercises can be embedded in an interactive worksheet with a customizable integration providing teachers with fine-granular control on the features available to students.

RQ2. Which tooling support do teachers need to help students struggling with given programming exercises?

Tooling support enhances the student and teacher interaction concerning the following three aspects:

(1) High-level overview as well as detailed insights into students' progress: Besides access to student submissions on demand, teachers benefit from the developed dashboard featuring a limited set of learning analytics of their students. Given a visualization of the students' score together with programming errors they made, teachers gain a better understanding of potential problems. Independent of single worksheets, existing Learning Record Stores (LRS) in schools offer customizable views for evaluating long-term progress. Hence, they can support struggling students in a targeted and faster way as well as better prepare upcoming lessons.

(2) Just-in-time feedback for interventions: Live processing of learning data, as featured by our tool and supported by the evaluation, is helpful for teachers to answer upcoming questions and to detect potential misconceptions.

(3) Content-driven inquiries: During a lesson, teachers and students do not wish any technical support with regards to their communication, as they value the direct, oral communication for emerging problems. However, students benefit from tools to help each other or contact the teacher in case of questions while working on their homework. *CodeOcean* supports commenting on lines of source code to provide teachers and students with the full context of a question.

RQ3. Which of the results can be transferred to the general MOOC context?

The improved structure of the worksheets was appreciated by students and our evaluation suggests that it supports learners in revisiting learning material while solving programming exercises. MOOCs will also benefit from grouping learning material of the same topic to ease navigating in the content.

Furthermore, our research suggests that the concept of study groups should be applied to MOOCs, enabling a tutor to guide learners through the course. In this scenario, tutors could be given control over the content visibility or deadlines and might also get access to learning analytics of their study group to enable informed interventions.

In this paper, we introduced and evaluated interactive worksheets with embedded programming exercises. Our prototype of a worksheet editor features an extensible plugin architecture for various learning tools. By combining text, videos and interactive content including practical programming exercises and multiple-choice quizzes, we improve the status quo for students as well as teachers: Students learn individually at their own pace. Teachers switch their role to tutors and individually support their students based on well-informed analytical insights. Feedback is positive and shows the vast potential of introducing digital content in the classroom.

References

[1] C. Meinel, J. Renz, M. Luderich, V. Malyska, K. Kaiser, and A. Oberländer, *Die HPI Schul-Cloud: Roll-Out einer Cloud-Architektur für Schulen in Deutschland*. Potsdam: Universitätsverlag Potsdam, 2019.

[2] T. Staubitz, R. Teusner, and C. Meinel, "MOOCs in Secondary Education - Experiments and Observations from German Classrooms," in *2019 IEEE Global Engineering Education Conference (EDUCON)*, Dubai, UAE, 2019, p. 10.

[3] A. Othman, A. Impes, and C. Pislaru, "Online Interactive Module for Teaching a Computer Programming Course," in *Proceedings of the 12th European Conference on e-Learning ECEL 2013*, 2013.

[4] Z. Merchant, E. T. Goetz, L. Cifuentes, W. Keeney-Kennicutt, and T. J. Davis, "Effectiveness of virtual reality-based instruction on students' learning outcomes in K-12 and higher education: A meta-analysis," *Comput. Educ.*, vol. 70, pp. 29–40, Jan. 2014.

[5] C. Hagedorn and C. Meinel, "Exploring the Potential of Game-Based Learning in Massive Open Online Courses," in *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, Timisoara, Romania, 2017, pp. 542–544.

[6] G. Beauchamp and S. Kennewell, "Interactivity in the classroom and its impact on learning," *Comput. Educ.*, vol. 54, no. 3, pp. 759–766, Apr. 2010.

[7] D. L. Leslie-Pelecky, "Interactive worksheets in large introductory physics courses," *Phys. Teach.*, vol. 38, no. 3, pp. 165–167, Mar. 2000.

[8] P. Blayney and M. Freeman, "Individualised interactive formative assessments to promote independent learning," *J. Account. Educ.*, vol. 26, no. 3, pp. 155–165, Sep. 2008.

[9] P. Blayney and M. Freeman, "Automated formative feedback and summative assessment using individualised spreadsheet assignments," *Australas. J. Educ. Technol.*, vol. 20, no. 2, pp. 209–231, Aug. 2004.

[10] V. Isomöttönen, A.-J. Lakanen, and V. Lappalainen, "K-12 game programming course concept using textual programming," in *Proceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE '11*, Dallas, TX, USA, 2011, p. 459.

[11] P. Hubwieser, M. Armoni, and M. N. Giannakos, "How to Implement Rigorous Computer Science Education in K-12 Schools? Some Answers and Many Questions," *ACM Trans. Comput. Educ.*, vol. 15, no. 2, pp. 1–12, Apr. 2015.

[12] R. Griffiths, M. Chingos, C. Mulhern, and R. Spies, "Interactive Online Learning on Campus: Testing MOOCs and Other Platforms in Hybrid Formats in the University System of Maryland," Ithaka S+R, New York, May 2014.

[13] M. J. Israel, "Effectiveness of Integrating MOOCs in Traditional Classrooms for Undergraduate Students," *Int. Rev. Res. Open Distrib. Learn.*, vol. 16, no. 5, pp. 102–118, Sep. 2015.

[14] M. Caulfield, A. Collier, and S. Halawa, "Rethinking Online Community in MOOCs Used for Blended Learning," 06-Oct-2013. [Online]. Available: https://er.educause.edu/articles/2013/10/rethinking-online-community-in-moocs-used-for-blended-learning. [Accessed: 08-Feb-2019].

[15] C. Matthies, R. Teusner, and G. Hesse, "Beyond Surveys: Analyzing Software Development Artifacts to Assess Teaching Efforts," in *2018 IEEE Frontiers in Education Conference (FIE)*, San Jose, CA, USA, 2018, pp. 1–9.

[16] C. Monroy, V. Snodgrass Rangel, and R. Whitaker, "A Strategy for Incorporating Learning Analytics into the Design and Evaluation of a K-12 Science Curriculum," *J. Learn. Anal.*, vol. 1, no. 2, pp. 94–125, 2014.

[17] B. Williamson, "Digital education governance: data visualization, predictive analytics, and 'real-time' policy instruments," *J. Educ. Policy*, vol. 31, no. 2, pp. 123–141, Mar. 2016.

[18] P. Blikstein, "Using learning analytics to assess students' behavior in open-ended programming tasks," in *Proceedings of the 1st International Conference on Learning Analytics and Knowledge - LAK '11*, Banff, Alberta, Canada, 2011, p. 110.

[19] M. Berland, R. S. Baker, and P. Blikstein, "Educational Data Mining and Learning Analytics: Applications to Constructionist Research," *Technol. Knowl. Learn.*, vol. 19, no. 1–2, pp. 205–220, Jul. 2014.

[20] P. Long and G. Siemens, "Penetrating the Fog: Analytics in Learning and Education," *Educ. Rev.*, vol. 46, no. 5, pp. 30–40, 2011.

[21] T. Rohloff, D. Sauer, and C. Meinel, "On the Acceptance and Usefulness of Personalized Learning Objectives in MOOCs," in *Proceedings of the Sixth Annual ACM Conference on Learning at Scale - L@S '19*, Chicago, IL, USA, 2019, p. 10.

[22] C. Meinel and C. Willems, *openHPI: the MOOC offer at Hasso Plattner Institute*. Potsdam: Universitätsverlag Potsdam, 2013.

[23] T. Staubitz, H. Klement, R. Teusner, J. Renz, and C. Meinel, "CodeOcean - A versatile platform for practical programming excercises in online environments," in *2016 IEEE Global Engineering Education Conference (EDUCON)*, Abu Dhabi, 2016, pp. 314–323.

[24] T. Staubitz, R. Teusner, and C. Meinel, "Towards a repository for open auto-gradable programming exercises," in *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, Hong Kong, 2017, pp. 66–73.

[25] F. F. Reichheld, "The One Number You Need to Grow," *Harv. Bus. Rev.*, p. 12, 2003.

[26] M. Minge, M. Thüring, I. Wagner, and C. V. Kuhr, "The meCUE Questionnaire: A Modular Tool for Measuring User Experience," in *Advances in Ergonomics Modeling, Usability & Special Populations*, vol. 486, M. Soares, C. Falcão, and T. Z. Ahram, Eds. Cham: Springer International Publishing, 2017, pp. 115–128.

[27] B. Laugwitz, T. Held, and M. Schrepp, "Construction and Evaluation of a User Experience Questionnaire," in *HCI and Usability for Education and Work*, vol. 5298, A. Holzinger, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 63–76.