Integrating Professional Tools in Programming Education with MOOCs

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

***Abstract*—An increasing number of high school teachers use existing Massive Open Online Courses (MOOCs) concerning programming education. Most MOOCs focus on teaching the basics of a programming language and common concepts or patterns. MOOC platforms usually provide their own code execution environments and thus have full control over the features and appearance available to learners. However, only a subset of tools available to professional software engineers is used in introductory programming MOOCs. While the reduction of features is helpful to ease navigation for novices, we assume that learners benefit from more advanced features at a later stage in the learning process. To help students minimize bugs and conceptual mistakes, we intend to evaluate how pair programming could be enabled for remote peers in MOOCs with a synchronized editor and an additional communication channel. Further, we plan to use static program analysis to get more insights about the code written by learners and to provide early feedback about the coding style. One of our contributions will be to identify possibilities to integrate professional tools and methods in MOOCs supported by an evaluation from learners.**

Keywords—Programming, Computer Science Education, Pair Programming, Static Program Analysis, Online Learning, MOOC

I. Context

Modern computer science education in high schools and universities usually contain several lessons on programming education with practical exercises. Existing research, e.g., as done by Merchant et al. highlights the importance of individual experiences in learning [1]. Besides the individual engagement with the learning content, Blayney and Freeman point out that "a crucial part of any learning or assessment activity is the degree to which students receive timely and effective feedback" [2]. While high school teachers might give individual feedback to their students on demand, Staubitz et al. concluded that manual feedback and grading is impossible for Massive Open Online Courses (MOOCs) with huge enrollment numbers [3]. Therefore, in a MOOC, it is required to use automated feedback to support learners. While multiple-choice quizzes are an integral part of MOOCs and technically easy to verify, they do not cover the whole range of content learned in a programming course and are inadequate to practice coding [4]. As an alternative to automated feedback, peer reviews can be used. While learners perceive timely feedback for their own work as helpful, giving feedback for peers results in additional workload for them and thus cannot be used throughout the full course [5].

To offer specific support for learners, some online platforms dedicated to programming education include interactive workspaces. These allow students to write and execute their own source code without setting up a local development environment, such as *Codeboard*[1]. MOOC providers, such as *openHPI*[2], also incorporate their own code execution platform, e.g., *CodeOcean* [3]. The educational platforms usually offer only a subset of features commonly found in an integrated development environment (IDE) used by professional software engineers. Many high school teachers we interviewed prefer the increased simplicity of educational IDEs over feature-rich IDEs to introduce novices to programming, as our previous work suggests [4]. Further, teachers valued the automated feedback given to learners, which freed some of their time to support students struggling with the topic individually [4]. While automated feedback in programming MOOCs is often based on unit tests and thus provides an indicator whether the code written by students fulfills the given requirements, it does not provide insights about the complexity, the runtime or other software metrics used by professional software engineers.

Due to the limited features available in some educational IDEs, not all principals used by professionals can be taught: *CodeOcean*, for example, misses debug functionality to support learners in finding bugs in source code [4]. While a debugger is useful to squeeze bugs in a given code (i.e., after making a mistake), pair programming as part of extreme programming (XP) leads to less error-prone programs compared to individual work [6]. However, according to a preliminary survey we conducted, many high school teachers do not explicitly introduce pair programming to their students. Nevertheless, students are forced to work in pairs of two in most school lessons due to missing computers. For school homework and learners in MOOCs, pair programming is not available without further adaption due to the distance of learners.

II. Objective

This thesis aims to investigate how to enhance programming education with professional tools and methods used by software engineers, such as pair programming or static program analysis:

- RQ1. How might we use the concept of pair programming in an online environment with learners being remote?
- RQ2. Which adaption is required to integrate static program analysis in MOOCs and how might results be used?
- RQ3. How can MOOC participants apply their knowledge gained through pair programming and static program analysis in a larger software project?

III. Method

To address the research questions outlined above, we plan to develop a coherent concept in consultation with MOOC instructors and high school teachers. Furthermore, we aim to integrate the required features into the existing web-based code execution platform *CodeOcean* and evaluate these by offering a dedicated MOOC for the interested public. We intend to open the course for learners of all age groups starting with senior high school students (K10-K12) but not limited to those.

[1] https://codeboard.io

[2] https://open.hpi.de

For integrating pair programming in an online scenario, we aim to match learners anonymously with a peer (if desired) and provide a synchronized code editor enabling live collaboration. As support for learners new to pair programming, we plan to evaluate different forms of assistance in applying the method. The assistance might range from providing a synchronized code editor fully accessible by both users simultaneously to limiting the editor to one learner at a time. The latter might support novices in applying pair programming, where one person actively types code as the driver and the other, called the navigator, watches out for mistakes and acts as a brainstorming partner [7]. Additionally, the strict separation of both roles in conjunction with the anonymity of peers might help to reduce gender-based discrimination as otherwise common in Computer Science [8].

To provide students with opportunities to improve their coding style, we aim to provide learners with other possible solutions after submitting their code for final grading. We expect to raise awareness of the readability and maintainability of source code by asking students to compare two solutions which passed the same unit tests. Throughout our study, we want to compare the solution preferred by learners and the static program analysis. By using anonymized code of other learners, we suppose to also provide a benefit for the two creators of the involved source code examples by processing the gained feedback. The additional feedback could also be used in conjunction with the *Request for Comment* feature available in *CodeOcean* to get further help from peers [9].

During the course runtime, we intend to split the participants into groups and offer them different levels of tooling support in an A/B test. In that scenario, the control group will learn with the unaltered experience while the other(s) will be used to increase the learning process. To prevent negative impact on participating high school students within one class and to minimize the mutual influence, we aim to group students of the same class together. By comparing the groups of learners (e.g., concerning their time invested, the points achieved and results of the static program analysis), we expect to identify whether our proposed changes improve the learning outcome gains. In addition, we will kindly ask all learners to participate in a survey to get direct user feedback.

IV. Results

Based on a preliminary survey with high school teachers and students as well as our previous work, we are aware of further requirements currently not fulfilled to enhance teaching effectiveness in programming education. These requirements have been used to formulate the research questions listed above in Section II.

As I have just started to work in my first year as a Ph.D. candidate, I aim to continue exploring the research area first and further clarify the design of the experiments described above. I will then work on *RQ1* and focus on pair programming incorporating the insights Teusner et al. previously described regarding video-conferencing as a form of direct communication within a MOOC [10].

Continuing with *RQ2*, I want to focus on tooling support to help learners improve their style of code before providing students with an opportunity to apply their knowledge in a software project larger than the usual exercises used in MOOCs so far (*RQ3*). Depending on the results gathered so far, the software project will incorporate the methods described above (such as pair programming and static program analysis) as well as peer feedback.

V. Conclusion

Our previous work focused on satisfying the basic needs high school teachers have when introducing blended learning with MOOCs to their classes. While teachers aim to use programming environments designed for beginners, they also profit from further customizations to control the features available to students. However, existing web-based programming environments used in MOOCs, such as *CodeOcean*, lack support for extending programming education with professional tools and methods, such as pair programming or static program analysis. As a result, interested MOOC participants including high school students are unable to experience these techniques themselves in the familiar programming environment. We expect that programming learners familiar with basic concepts would profit from an early introduction to the tools and methods used in the software industry. Nevertheless, the designated usage in MOOCs requires special adaption of these techniques.

Therefore, this thesis aims to provide further insights into the usage scenarios, the benefits and the limitations of these techniques in MOOCs. Our vision is to incorporate teaching about coding styles as a matter of course and offer students the possibility to practice them while learning more profound programming concepts.

Acknowledgment

I would like to thank my advisor Ralf Teusner for his excellent support and the time he invested while working on his own thesis. Many thanks also to my supervisor Prof. Meinel and the members of the *openHPI* and *HPI Schul-Cloud* teams.

References

[1] 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.

[2] 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.

[3] 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.

[4] S. Serth, R. Teusner, J. Renz, and M. Uflacker, "Evaluating Digital Worksheets with Interactive Programming Exercises for K-12 Education," in *2019 IEEE Frontiers in Education Conference (FIE)*, Cincinnati, OH, USA, 2019, pp. 1–9.

[5] C. E. Kulkarni, M. S. Bernstein, and S. R. Klemmer, "PeerStudio: Rapid Peer Feedback Emphasizes Revision and Improves Performance," in *Proceedings of the Second (2015) ACM Conference on Learning @ Scale - L@S '15*, Vancouver, BC, Canada, 2015, pp. 75–84.

[6] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "The Effects of Pair-Programming on Performance in an Introductory Programming Course," p. 5.

[7] N. Nagappan *et al.*, "Improving the CS1 experience with pair programming," *ACM SIGCSE Bull.*, vol. 35, no. 1, p. 359, Jan. 2003.

[8] M. Marklund and S. Gustavsson, "'Why Am I Even Doing This?': The Experiences of Female Students in CS from an Insider Perspective," in *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, Auckland, New Zealand, 2018, pp. 77–81.

[9] R. Teusner, T. Hille, and T. Staubitz, "Effects of automated interventions in programming assignments: evidence from a field experiment," in *Proceedings of the Fifth Annual ACM Conference on Learning at Scale - L@S '18*, London, United Kingdom, 2018, pp. 1–10.

[10] R. Teusner, N. Wittstruck, and T. Staubitz, "Video conferencing as a peephole to MOOC participants: Understanding struggling students and uncovering content defects," in *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, Hong Kong, 2017, pp. 100–107.