# Video Conferencing as a Peephole to MOOC Participants

## Understanding Struggling Students and Uncovering Content Defects

***Abstract*— Distance education gained considerable attention with the rise of Massive Open Online Courses (MOOCs). Given the significant role collaboration plays in practical computer science education on campus, it becomes evident that nowadays online course platforms mostly lack the necessary collaborative capabilities. We present a solution to support collaborative programming through video conferencing for practical exercises employed in MOOC contexts. Two user surveys show that although users value the possibilities, privacy concerns remain. We therefore propose to additionally use video conferencing technology to face another challenge: MOOCs usually are conceptualized and produced to a large extent before the actual course runtime. Reaction on current events within the course is possible but requires insights on students' problems. Course conductors can use the tutoring mode in our WebIDE to understand struggling students and potentially uncover topics that lack additional background material or need additional training exercises.**

***Keywords—MOOCs; video conferencing; tutoring; programming assignments; content adaption***

## I. INTRODUCTION

Running a MOOC is usually an intensive and busy time for the involved, often rather small, teaching team. Supervising and nurturing discussions, fixing glitches in the course material and keeping things running is enough to fill each workday during the runtime of a course. Particularly courses with experimental or interactive parts require additional efforts to fix and enhance the tooling used. Therefore, the majority of content is produced before the course runtime. During the course runtime, course conductors mostly moderate the forums, record additional "office hours" videos and supervise the helpdesk to interact with the students. The comparison of campus centered teaching activities and distance education shows several differences.

While the core principles of teaching remain the same, the surrounding conditions in a MOOC are different. Pea describes that collaborative efforts and the sharing of different perspectives are required to acquire knowledge [1]. This has been missing until the recent trend to integrate collaborative concepts into MOOCs [2, 3, 4]. Group-based experiences are supposed to improve "satisfaction, persistence and intellectual and social development" [5]. They are therefore relevant not only to on-campus courses, but to students taking part in online courses as well. Chen et al. measured in 2008 that, compared to on-campus students, remote participants taking the classes online were at least as engaged when it came to asking questions or contributing to the class discussion. Yet they were significantly less involved in working with other students to prepare class assignments or projects [5]. Since then, multiple approaches to improve collaboration among MOOC students, such as openHPI's Collab Spaces [6] or Stanford's Talkabout [4], have emerged.

Even though this apprenticeship approach is beneficial to the students' learning outcome, it is not feasible to mentor and support the thousands of students that participate in online courses individually [7, 8, 9]. Although support forums exist, those are often impersonal. Moreover, asking a question interrupts the students' workflow – one has to leave the editor to post on the forum and then check regularly to see if there is an answer. Still, writing code collaboratively has been promoted in the form of pair programming in the last years to help programmers share learnings and improve their code's quality [10, 21].

We propose *CodePilot*, a prototypical video conferencing solution integrated into our web-based execution environment *CodeOcean*[1], to support remote tutoring sessions. Leveraging students' knowledge by enabling them to mentor their peers and by encouraging them to share their recorded discussions also fosters the scalability of MOOCs.

On the basis of *CodePilot*, we want to address the following research questions:

[1] available at: https://github.com/openHPI/codeocean

 DOI: 10.1109/TALE.2017.8252312

1. Can we transfer collaboration concepts from on-campus education to MOOCs, and especially programming MOOCs?
2. What issues hinder remote collaboration and how can they be circumvented?
3. Are participants willing to share their content and thus actively create further course content?

## II. Related Work

The field of remote tutoring in MOOCs touches different research areas, which we will examine with regards to their related work.

### A. *Media Richness and Media Synchronicity Theory*

To evaluate the effectiveness of different approaches with regards to improving dissemination of knowledge, media richness and its advancement, media synchronicity theory offer approaches [11]. Media covering more senses and offering a stronger immersion generally support easier content adoption [12]. We argue that a stronger personal commitment and increased interactivity, as induced by one-to-one interaction with another human, further increases learning rates notably.

### B. *Cooperative and Collaborative Learning*

We distinguish between cooperative and collaborative learning. According to Panitz, cooperative learning is aimed towards reaching a pre-defined goal, while collaborative learning is more of a "personal philosophy, not just a classroom technique" [13]. We stick to this definition and will refer to goal-oriented teamwork as cooperation, while collaboration refers to open discussions and general advancements. In order for teamwork to be effective, team members have to share a common goal, which might be set externally (e.g. a task to be solved) or motivated internally (create a certain program). If such a goal is missing, teamwork will end too early or might not start at all. The groups that form can be classified into two kinds, informal cooperative learning groups, that are short lived and exist to discuss a recent problem or solve a present task, and base groups, that are more formal and exist for a longer period, mostly solving a larger exercise or attending the whole course together [14, 15].

When conducting remote cooperation, additional steps have to be taken to prevent the failure of the intended teamwork due to additional organizational (scheduling) or technical (missing hardware, software or accounts) issues.

### C. *(Remote) Pair Programming*

Our focus on programming MOOCs causes additional requirements to the assistive tooling supplied for collaboration. Jointly developing programs, i.e. writing and discussing code, requires a shared view on the present state of the source code. This can either be achieved by screen sharing, or by editor synchronization. While screen sharing, for example via Screenhero[2], yields some benefits as transmission of all visible elements including the mouse pointer, screen sharing in general also wastes additional bandwidth and might come with delays or otherwise poor user experience due to blurry pictures preventing small fonts to be read.

In his research challenges for global software development, Herbsleb underlines the importance of exploiting project memory and knowledge of team members [16]. This can be transferred to the e-learning context as well – here, the project memory consists of course- or assignment-related knowledge of all students. While pair programming can help to share knowledge from one team member to another [17] and while there are plenty of tools that enhance access to project memory [16, 18], xMOOCs usually only feature forums to allow students to share their knowledge. In addition to the benefits of knowledge sharing, McDowell et al. found that dropout rates of students using pair programming were reduced [19].

### D. *Other Courses, Platforms*

There have been some former attempts to utilize video conferencing in xMOOCs, e.g. Collab Spaces in openHPI or Talkabout in Coursera [6, 4]. While these attempts highlight the importance of collaboration and the positive effects of connecting students, they currently have a number of drawbacks based on their proprietary foundation, relying on Google Hangouts and thus requiring an additional account and preventing a deeper integration into the course platform to enhance grouping quality or better feedback. This also prevents recordability and might lead to legal issues, as data sovereignty is also not given and content is shared with third parties.

[2] available at: https://screenhero.com/

openHPI's collab spaces focus on creating purposeful groups. Three different kinds of groups are usually distinguishable: study groups, that progress through the course together and are connected by external factors (language, location, age, employer), topic focused groups that evolve around a certain (often times specific or especially demanding) topic, and teams. Teams are formed when a certain task has to be solved in cooperation.

Joseph and McKinsey surveyed the adoption of remote pair programming in 2013, and came to the conclusion that organization is one of the major issues [20, 21]. Also, they encountered problems with participants just wanting a "free ride", joining remote programming sessions with the primary goal to copy solutions for the exercises.

For production usage, a distinguished tool, such as Talkabout, to form groups and plan shared time slots is therefore recommended [4]. In their experiments with Talkabout, Kulkarni et al. found that students collaborating in diverse discussions were significantly more likely to also answer quizzes and score higher on exams [4]. They also underline that a pedagogical concept that accompanies the group discussions is important: Depending on the course type and the expected learning results, certain strategies can, for example, increase sharing of self-references or encourage students to re-evaluate their own opinion. Omitting explanations of all agenda items encourages students to ask questions about them. When testing a very strict and rigid agenda, Kulkarni et al. found that students mentioned that the discussions were less motivating and that they were less inclined to meet the same group again. While Coursera, the platform that the respective courses were conducted on, uses an open-source, web based development environment for some of their programming classes, there is no deeper integration of this into Talkabout. Talkabout also did not try this concept in a programming related class.

Additionally, it became apparent that just putting people together does not lead to effective progress. Staubitz et al. concluded that an elaborated team composition increases learning success [13].

## III. CONCEPT

Our system should meet the following requirements:

1. Pair programming support: Lagless synchronization of the source code and program output is crucial to enable a natural development flow.
2. Apprentice becomes master: by encouraging students to help each other, we want to offer advanced students additional options to grow their knowledge and will thereby reduce the workload on the teaching team and tutors.
3. Reproducibility and rehearsability: the ability to re-watch tutoring videos and use them as additional content will foster the effectiveness of tutoring for wider audiences.
4. No additional plugins, no additional accounts: installing software or registering for 3rd party services will discourage usage and thus hinder adoption.
5. Pairing of Participants: Students asking for help and tutors should be automatically matched for the best potential outcome.

Pair programming allows us to put the driver (person writing code) to be set up in the so-called "zone of proximal development" [8]. In pair programming, the driver and the observer, the person tasked with guiding, reflecting and commenting on the code, usually switch roles after a certain amount of time. For our main use case tutoring, we will work with fixed roles: the tutor being the observer and the student being the driver. Having a tutor guiding and helping on problems, allows students to progress from tasks they can do alone to tasks they can solve with external help. There will remain tasks that are still too hard, but usually exercises that are demanding and require the participant to leave his comfort zone yield the best learning results [22, 23]. Also the eXtreme Apprenticeship model suggests using scaffolding and mentors to help students. In the context of children education, scaffolding has been described as "the way the adult guides the child's learning via focused questions and positive interactions" [24]. For this reason, we decided to limit the code synchronization to one site, forcing the tutor to explain all necessary source code changes instead of directly implementing them. Another benefit of this "guided programming", where the student is lead by the tutor but needs to solve all tasks himself, is that it effectively circumvents students falling into

pitfalls that would leave beginners stuck. Debugging sometimes feels cumbersome and demotivates students [10]. Experienced tutors can explain rather cryptic compiler errors and stack traces and help during debugging, allowing the student to focus to concentrate on their program design and algorithms. By limiting our approach to pairs of one participant and one tutor, we implicitly prevent the „free ride“ problem mentioned by McKinsey.

The second requirement, apprentices becoming masters, provides for the specific MOOC setting. While tutoring from teaching assistant to student works for university course scale, the proportions in MOOCs require a much larger group of tutors, preferably available over all time zones. Recruiting motivated and skilled students to take over the role of tutors is therefore necessary. In addition, students might feel more comfortable with receiving help from a fellow student, instead of the teaching staff [25]. Naumann et al. found that students are willing to contribute to these forums and often solve issues without the intervention of teaching staff, e.g. by sharing pre-existing knowledge or providing links to external information [26]. Also the findings of Staubitz et al. [27] support this conclusion. The fact that the ability to teach fellow students can be built up is further demonstrated by Coursera’s usage of *Community Teaching Assistants* – successful participants of former courses who volunteer to teach future classes [28]. In order to gain scalability, we propose to build up a potential pool of tutors through the tutoring itself. The main reason not to give everyone the option to answer open requests, is that a synchronous audio and video connection implies potential threats with regards to privacy. As the tool supplier, we therefore want to ensure control over at least one side of the tutoring sessions. However, given the workload on the teaching team, we also admit that additional persons are required, as well. Starting from the group of course conductors and platform owners, we aim to unlock the tutoring backend for participants that we had a positive session with and who seem to be qualified with regards to knowledge as well as attitude. We are further confident that tutors do not need to be on expert level content wise, since oftentimes it is already enough to just give a subtle hint, if necessary at all. Sometimes it’s already sufficient just to phrase the problem and to recapitulate the steps taken, to uncover a potential solution oneself.

Next to sorting out potential legal issues and carefully selecting potential participants that act as additional teaching assistant, the prototype has to offer an environment in which students feel confident in asking as well as answering questions.

Good reproducibility and rehearsability allows also students whose skill level is above the average and simply do not encounter further issues and students that lack the technical requirements or the extroversion to ask questions publicly, thus admitting perceived deficits to strangers. Synchronizing a recording of the video conference with the editor content of the student allows other students to track applied code changes and the discussion that led to these changes. Also adding a further channel to convey information improves the media richness.

Apart from the primary use case within programming assignments, the software should also offer general availability of video conferencing to be used within arbitrary group tasks.

## IV. IMPLEMENTATION

The prototype of our proposed tutoring solution was implemented as a Ruby on Rails application that integrates into our code execution platform *CodeOcean* via an iFrame. For the actual video conferencing part, we rely on the open source project *Jitsi Meet*. With respect to the workflow, our prototype has to support the actions shown in Figure 1.

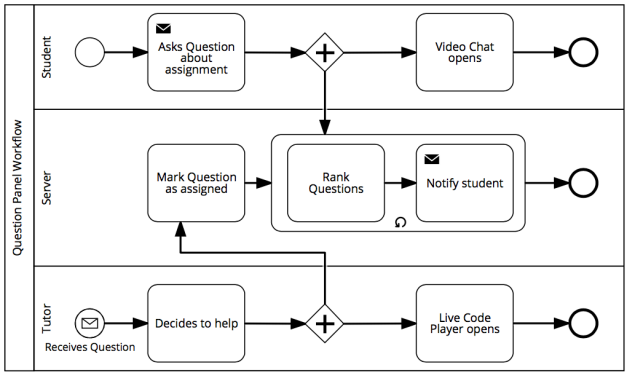


Fig. 1. Main Workflow for Tutoring Sessions

In order to realize the flows, we modeled the required data as shown in Figure 2. The core part is the question, which has attached participations of users (in our case: two users). Technically, to also support other use cases like public demonstration

sessions or group discussions, there could also be more participations. Whenever the student starts a program run, we save the execution result to allow for playback later on. All changes to the source code are saved as deltas, to support a seamless playback and synchronization with the tutor. If the session ends, a recording entry is stored and all participants are surveyed for their opinions (helpfulness of the tutor, could the problem be solved, allowance to use the recordings, additional free text).

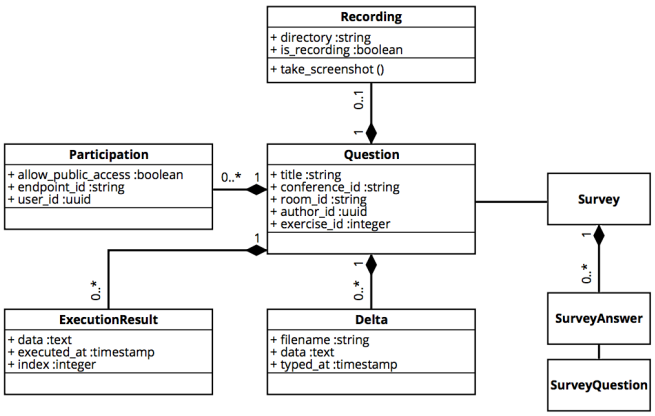


Fig. 2. Data Model Used for *CodePilot*

For participants, the user interface of our prototype presents itself as shown in Figure 3.

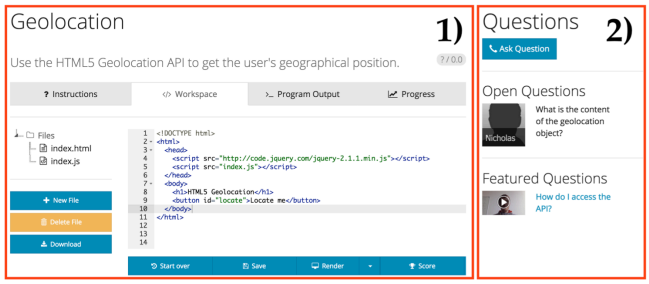


Fig. 3. User Interface for the Student

On the left side (1), users see the usual controls of our execution platform. The only difference is that changes made in the editor during an active conference session are recorded and transferred to the tutor via a synchronous webRTC connection. We currently restrict the synchronization direction to be unidirectional towards the tutor. In order to potentially enable full bidirectional synchronization for full pair programming, additional methods like operational transform (OT) are required [29]. On the right side (2), the main compartment of our prototype is embedded. We show open questions and existing recordings here. If the student clicks "Ask question", an input box to phrase the question opens up. Afterwards, the student is forwarded to an empty meeting room that opens up in the iFrame (2). The coding environment (1) stays as it is; the progress on the exercise is not interrupted in any way. If an existing recording is chosen, which is available under "Featured Questions" the current progress is persisted and the recording is played back in fullscreen, to make room for the recorded code to be presented. With regards to usability, we kept the controls and new elements as minimal as possible and got only positive feedback so far.

## V. Evaluation

We conducted several qualitative interviews as well as 3 quantitative surveys with a varying number of participants and questions. The first small study was conducted in a controlled environment and had eleven participants, students and young professionals, use the tool for different tasks. We asked them for their perception which factors would be most important for an optimal tutoring. The overall result can be seen in Figure 4. As a basis, so of highest importance, the participant - tutor pairs have to share a common language and participants wish to have the tutor to have mastered the skillset to be learned. On top of that, students then prefer tutors that they had contact with, for example in person, in prior sessions or in forum discussions. Ranked of least importance was the user role, meaning the position and occupation within a course, so being a student, an entitled teaching assistant or part of the core group actually conducting the course. When presenting potential questions for tutors, these considerations should be represented in the pairing and ranking algorithms.

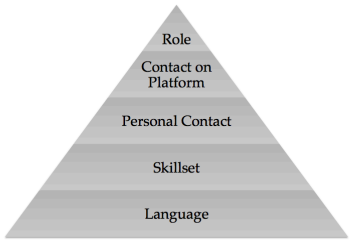


Fig. 4. Factors for Optimal Tutoring

Over all conducted sessions, the tutor was always the dominant speaker, meaning he had more or louder parts during the conversations. This met our expectations, since we expected that the tutors

explain the comparably shorter questions of the student in comparably greater detail. For longer sessions, we expect the ratio to shift towards the student, as potentially in longer sessions the tutor acts as a facilitator, guiding the student to work his way to the partial solutions himself, instead of answering a specific question and therefore solving the problem rather quickly. In our sessions, however, all questions were solved in less than eight minutes, which means we got no data concerning that, yet.

When asking video conferencing participants under which circumstances they are willing to share their content on the platform, the results were hesitant but overall promising. Only one participant stated that he would never share the content. The others were ranging from reluctant (majority) to proactive (minority). Further inquiring, what stopped them from sharing, they answered that they were not sure whether the content was good enough, whether the questions were interesting and whether the results were helpful enough to be regarded as teaching material. All participants agreed that being individually contacted by the teaching team and asked for permission would further convince them to share their content. External reassurance by the teaching team would therefore dispel their doubts about usefulness and quality. Wishes for the ability to re-watch the recording before granting the ok for publishing, and in best case to have basic cutting functionality to trim the videos to core parts and remove unwanted parts were voiced. Alternatively, also the much simpler function to simply remove the video stream was requested. As implementing video editing capabilities into the web application was outside our focus, we gathered further feedback on the idea to remove the video stream. In the end, we discarded that option because several participants mentioned that the video was a vital part for them and created a sense of trust in the discussants.

When offered to either directly ask a question or first watch a potentially matching, recorded tutoring sessions, 9 out of 11 participants watched a potentially suited video before or instead of asking for help. Asking for help requires participant to overcome an individual hurdle, as they feel they are eventually disturbing another person and they have to show their face together with their lack of specific knowledge, which is something especially participants in higher career positions sometimes have grown unfamiliar with and thus can feel intimidated [30]. Allowing to copy the code of recorded sessions however re-introduces the "free ride" problem, potentially causing participants to cheat without learning anything. Asking our surveyed participants for the reason they are copying the final source code, they stated that they understood the concept, but had however mostly problems with the proper syntax.

Our evaluation further showed that if a question has already been answered, students are more likely to watch the recorded session than to ask the question again.

Based on our initial findings, we activated the tool in the aftermath of our Introduction to Java[3] course and also in an Introduction to Python[4] course. Students were not able to communicate with other students, just the teaching team had access to the tutor backend. While the survey responds were positive, only a fraction of the course audience actually tried to start a conference. Technical problems prohibited about half of the sessions, because students could not get a proper connection, or immediately left the session (possibly not patient enough to wait for the tutor to connect, or scared by the fact that their webcam was activated after they allowed the access). The sessions that did take place were mostly conducted without a webcam on the participant side, sometimes even without a microphone. The tutors streamed a live picture and answered by voice, however the students mostly lacked the technical capabilities or were too deeply concerned about their privacy [31]. In our recent Java course in 2017, we did not offer tutoring due to a lack of potential time, but we used the opportunity to gather some additional insights about participants' views on tutoring. 1836 participants answered the survey, each question could be answered individually. 38% of the students voiced that they would not use that opportunity because they had doubts about their privacy, and 19% of all students lacked the technical capabilities to take part in a tutoring session. When being asked what they would

[3] German course "Java für Einsteiger" conducted in 2015, available at: https://open.hpi.de/courses/javaeinstieg2015/

[4] German course "Spielend Programmieren lernen 2015!", conducted in 2015, available at: https://open.hpi.de/courses/pythonjunior2015/

use the tool for, 21% answered they would like to get feedback from the teaching team, 12% would like to get feedback from their fellow students. Another interesting finding is that 10% would be willing to contribute for feedback from a tutor. We purposely did not state whether the contribution should be monetary or in terms of another service to the benefit of the platform. From our point of view, such a service could also be the allowance to use the recorded session as additional teaching material.

Based on these findings, we currently use the tool to get a glimpse into the students work processes and compensate what is done in on-campus settings by just watching over the shoulder: uncovering oftentimes trivial problems, that simply were not on teachers minds during the conception of the didactical concept.

## VI. Focus shift and Current Usage

Starting from our concept to establish a scalable pool of voluntary tutors with the course conductors as the seed and foundation in order to improve programming courses with individual help on shortcomings, we shifted our focus with regards to the current usage of *CodePilot*. Missing technical equipment and privacy concerns on student side lowered the effective demand for tutoring. While we might be able to compensate that with a different approach with regards on how we integrate it into the course, we then suspect other issues to arise instead. By globally announcing public office hours, we will most likely be overwhelmed with participants who will then be disappointed and demotivated if they do not make it into a session and waited in vain. Offering the tutoring rather silently without additional voluntary tutors and inviting users just by announcing it in the forums, lead to only few participants, as only a fraction of participants read the forums and even fewer then decided to start working on the exercises, encountered problems and on top of that had the courage to ask for help. As of now, we are nonetheless sticking to the silent tutoring approach, but see the main benefit in another factor besides the knowledge gain of the participant. For us as course conductors, an important benefit is the insights that we gain about the individual problems of the respective participant. The direct dialog with students uncovered several shortcomings that were perceived as minor by the individual participant and were therefore not posted in the forum, but affected several students and hindered their progress. Such a shortcoming was for example the missing knowledge on how to enter curly brackets via the German keyboard layout on Windows and MacOS. The opportunity of virtually watching over the shoulder of a participant also enables teachers to detect limiting factors or misunderstandings that the students do not notice themselves, as for example superfluous variables, which were used in instructional videos and that had been perceived as necessary for a certain setup by the student, however had no usage in the actual exercise. The virtual glimpse into actual real-time learning processes of some students allows making didactical improvements, uncovering likely misunderstandings of de-facto correct but ambiguous explanations and filling previously unnoticed shortcomings of the supplied material.

## VII. Future Work and Conclusion

The future work presented here mostly suits the initial focus of connecting students, as well as the shifted focus on uncovering didactical problems. First, we think that an automatic detection of the topics covered within a tutoring session will be a worthwhile effort. While complete and coherent speech to text transcription is hard to realize, we believe that detecting specific keywords is possible. They will allow for automatic clustering of recordings and improved searchability of content. Second, analyzing the discussion dynamics yields potential. With regards to the social course status (student tutor, teaching assistant, course conductor) and age, there might be patterns that occur in tutoring sessions. Using the active speaker detection, we can analyze the speech ratio and determine whether we rather have on open discussion or a more counseling based session. Determining factors to improve student - tutor matching is mainly helpful for the initial approach. Participants having posted in the same forum threads is most likely not the best metric to determine suited partners. Probably, also similar scores, a similar progress in the course and previous contact via asynchronous commenting in the programming environment might be additional factors. As those factors have not proven efficient yet, we currently simply put the request in queues and assign the next free tutor to the next student.

The technical requirements for our prototype have been met. The evaluation of our prototype implementation shows that students value the benefits of our solution. When receiving advice, participants favor language and professional skills over status and previous contact. Concerning the research questions, we currently cannot transfer the on-campus collaboration concepts to MOOCs because many participants lack the technical capabilities to actively participate in video conferences. When encouraged by the teaching team, most participants were willing to share their produced content. Although we shifted from our initial goal on intent due to the current status quo of our audience, we are confident that we created an auxiliary tool, elevating our efforts to better understand our participants and opening up further room for research. The feedback of the tutored participants was positive without exception, and the masses that could not be reached directly benefitted transitively through better material, mostly without even noticing where our elaborated guesses which parts to improve, originated from.

## REFERENCES


[1] R. D. Pea, Practices of distributed intelligence and designs for education. *Distributed cognitions Psychological and educational considerations*, pages 47–87, 1993.

[2] T. Staubitz, J. Renz, C. Willems, and C. Meinel, Supporting social interaction and collaboration on an xMOOC platform, In *Proc. 6th International Conference on Education and New Learning Technologies (EDULEARN14)*, Barcelona, Spain, 2014.

[3] T. Pfeiffer, Collaborative learning in a MOOC environment, Master's thesis, Hasso Plattner Institut, Potsdam, Germany, 2014.

[4] C. Kulkarni, J. Cambre, Y. Kotturi, M. S. Bernstein, and S.R. Klemmer, Talkabout: Making distance matter with small groups in massive classes. *Proceedings of CSCW 2015: ACM Conference on Computer Supported Collaborative Work*, 2015.

[5] P. Chen, R. Gonyea, and G. Kuh. Learning at a distance: Engaged or not? *Innovate*, 4(3), April 2008.

[6] T. Staubitz, T. Pfeiffer, J. Renz, C. Willems and C. Meinel, Collaborative Learning in a MOOC Environment, *8th International Conference of Education, Research and Innovation (*ICERI*) Seville, Spain, 2015.*

[7] A. Vihavainen, M. Paksula, and M. Luukkainen, Extreme apprenticeship method in teaching programming for beginners, in *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 93– 98. ACM, 2011.

[8] L. S. Vygotsky, *Mind in society: The development of higher psychological processes*, Harvard university press, 1980.

[9] C. Meinel, C. Willems, J. Renz, and T. Staubitz. Reflections on enrollment numbers and success rates at the openHPI MOOC platform, In *Proceedings of the European MOOC Stakeholder Summit (eMOOCs),* Lausanne, Switzerland, 2014.

[10] L.Williams, R. Kessler, W. Cunningham, and R. Jeffries, Strengthening the case for pair programming. *IEEE software*, 17(4), pages 19–25, 2000.

[11] A.R. Dennis and J.S. Valacich, Rethinking media richness: towards a theory of media synchronicity, In *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences*, pages 10 pp., 1999.

[12] E. Dale, The cone of experience. In Audio-visual methods in teaching. (pp. 37-51), New York: Dryden Press, 1946.

[13] T. Panitz, Collaborative versus cooperative learning: A comparison of the two concepts which will help us understand the underlying nature of interactive learning, 1999.

[14] K. Smith. Cooperative learning: Making "group-work" work. *New directions for teaching and learning*, (67), pages 71–82, 1996.

[15] T. Staubitz and C. Meinel, Collaboration and Teamwork on a MOOC Platform: A Toolset. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale* (L@S '17). ACM, New York, USA, pages 165-168. DOI: https://doi.org/10.1145/3051457.3053975

[16] J.D. Herbsleb, Global software engineering: The future of socio-technical coordination, In *Future of Software Engineering*, FOSE '07, pages 188–198, Washington, DC, USA, 2007. IEEE Computer Society.

[17] H. Holz and F. Maurer. Knowledge management support for distributed agile software processes, In *Advances in Learning Software Organizations*, pages 60– 80, Springer, 2003.

[18] D. Cubranic, G.C. Murphy, J. Singer, and K.S. Booth, Hipikat: A project memory for software development, In *IEEE Transactions on Software Engineering*, 31(6), pages 446–465, 2005

[19] C. McDowell, L. Werner, H. Bullock, and J. Fernald, The effects of pair-programming on performance in an introductory programming course. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '02, pages 38–42, New York, USA, 2002. ACM.

[20] Agile Ventures, Remote Pair Programming Analysis, Retrieved July 22, 2017, from https://www.agileventures.org/remote-pair-programming/analysis

[21] J. McKinsey, Remote Pair Programming in a Visual Programming Language, Technical Report No. UCB/EECS-2015-139, University of California at Berkeley, 2015

[22] S.I. Leberman and A.J. Martin. Does pushing comfort zones produce peak learning experiences? *Australian Journal of Outdoor Education*, 7(1):10, 2002.

[23] M. Brown, Comfort zone: Model or metaphor. *Australian Journal of Outdoor Education*, 12(1), pages 3–12, 2008

[24] N. Balaban, Seeing the child, knowing the person. *To become a teacher*, pages 49–57, 1995.

[25] B. Goldschmid and M.L. Goldschmid, Peer teaching in higher education: a review, *Higher Education*, 5(1): pages 9–33, 1976.

[26] F. Naumann, M. Jenders, and T. Papenbrock. Ein Datenbankkurs mit 6000 Teilnehmern. *Informatik-Spektrum*, 37(4): pages 333–340, 2014.

[27] T. Staubitz, J. Renz, C. Willems, C. Meinel: Supporting Social Interaction and Collaboration on an xMOOC Platform, In *Proceedings of 6th Annual International Conference on Education and New Learning Technologies* (EDULEARN2014), Barcelona, 2014.

[28] K. Papadopoulos, L. Sritanyaratana, and S.R. Klemmer, Community TAs scale high-touch learning, provide student-staff brokering, and build esprit de corps. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 163–164. ACM, 2014

[29] C.A. Ellis and S.J. Gibbs. Concurrency control in groupware systems. In *ACM SIGMOD Record*, volume 18, pages 399–40 , 1989

[30] M. Lowis, T. Staubitz, R. Teusner, J. Renz, C. Meinel and S. Tannert, Scaling Youth Development Training in IT Using an xMOOC Platform, In *Proceedings of 45th Annual Frontiers in Education Conference* (FiE2015), El Paso, TX, USA, 2015

[31] T. Staubitz, R. Teusner, J. Renz and C. Meinel, An Experiment in Automated Proctoring, In *Proceedings of the European MOOC Stakeholder Summit (eMOOCs)*, Graz, Austria, 2016.