Phased Classroom Instruction: A Case Study on Teaching Programming Languages

Sebastian Mader and François Bry
Institute of Informatics, Ludwig Maximilian University of Munich, Germany

Keywords: Technology-enhanced Learning, Active Learning, Flipped Classroom, Peer Review.

Abstract: This article describes a novel educational format called "phased classroom instruction" and its enabling technology specially tuned to the effective learning of formal languages in tertiary STEM education. Like flipped classroom, phased classroom instruction aims at promoting active learning. In contrast to flipped classroom, phased classroom instruction scales to large classes thanks to its associated enabling technology. The article reports on a real-life evaluation of the proposed format and of its enabling technology pointing to their effectiveness.

1 INTRODUCTION

Programming languages and all formal languages (like the language of chemical equations) used in STEM education can only be learned by sufficiently applying them. Language application, however, mostly falls short in tertiary STEM education which heavily relies on lectures (Stains et al., 2018) the large audiences of which make an active participation of students hardly possible. Even though lectures are commonly decried, they persist in tertiary STEM education for pragmatical reasons: Lectures scale and with steadily increasing numbers of students they often remain the last resort. As a consequence, "learning by applying" is delegated outside lectures leaving the students mostly alone with problems and misconceptions.

The necessity of "learning by applying" and more generally of "active learning" has often been stressed, not only for learning formal languages. According to Bonwell and Eison, in order to be actively involved in their learning, students need to "engage in such higher order thinking tasks as analysis, synthesis, and evaluation" (Bonwell and Eison, 1991, p. iii). This broad definition covers nearly every learning scenario that goes beyond lecturing, and is restricted by Prince to "activities that are introduced in the classroom" (Prince, 2004, p. 233). The teaching format "flipped classroom" has been introduced so as to promote active learning by flipping the activities traditionally performed during and outside classes: Students begin with learning on their own, or "self-learning", what is followed by classroom sessions of active learning (Bishop et al., 2013).

Flipped classrooms face two difficulties: First, the creation of learning material suited to self-learning is significantly more time-consuming than creating learning material for presence lectures (McLaughlin et al., 2014; Stelzer et al., 2010). Self-learning material must be more bulletproof and extensive than lecture material because with self-learning, no lecturer can provide complementary explanations. Second, flipped classrooms do not scale to several tens or even hundreds of students. The role of lecturers in flipped classrooms is the role of a "guide on the side" (King, 1993) which requires of lecturers to be aware of every students' progress and difficulties – a manageable task with twenty to thirty students that becomes more and more impossible as the number of students increases.

Building upon the experience gathered with flipped classrooms and exploiting possibilities offered by technology, a novel teaching format, "phased classroom instruction", has been designed. Phased classroom instruction fosters active learning like flipped classrooms but is, in contrast to flipped classrooms, deployable in large classes thanks to its technological support. With phased classroom instruction, a classroom session consists of mini-lectures lasting typically 15 to 25 minutes, followed by longer phases devoted to application of the knowledge conveyed in the mini-lectures in practical exercises.

Phased classroom instruction's practical exercises are worked on in a browser-based environment which

Mader, S. and Bry, F.
Phased Classroom Instruction: A Case Study on Teaching Programming Languages.
DOI: 10.5220/0007655702410251
In *Proceedings of the 11th International Conference on Computer Supported Education (CSEDU 2019)*, pages 241-251
ISBN: 978-989-758-367-4

is advantageous for lecturers and students alike. As of lecturers, the browser-based environment makes all currently worked on submissions available to the lecturers and makes it possible to get assistance from software for spotting students in need of help. As of students, the browser-based environment provides editors customized to different types of exercises, such as code editors with compiling functionality, editors for logical proofs, detecting erroneous expressions or even allowing only well-formed expressions. Such editors are able to provide immediate feedback instead – or in addition to – the lecturer, allowing short feedback loops even in large classes. Furthermore, the online availability of the students' submissions makes further work with the submissions easily possible, such as the lecturer referring to some of them in following mini-lectures or students reviewing other students' submissions.

Phased classroom instruction is similar to Frederick's proposal to alternate "mini-lectures and discussions" (Frederick, 1986, p. 47). Both formats differ inasmuch as Frederick's format does not include written students' submissions to exercises but instead discussions on the mini-lectures' content and as with Frederick's format active phases generally last no more than 10 to 15 minutes. Furthermore, Frederick's proposal is to be deployed in a non-digital classroom and does not discuss enabling technology.

The contributions of this article are threefold: First, a novel course format lessening the workload of teachers, second, the conception of a novel technology enabling the proposed instruction format in large classes, and third, a report on a real-life evaluation of the format and its enabling technology in a university computer science course pointing to their effectiveness.

This article is structured as follows: Section 1 is this introduction. Section 2 is devoted to related work. Section 3 introduces the course format, its enabling technology, and the implementation in a software development practical. Section 4 reports on an evaluation of course format and enabling technology. Section 5 summarizes the article and gives perspectives for future work.

2 RELATED WORK

The survey of flipped classroom research by Bishop et al. (Bishop et al., 2013) distinguishes two kinds of teaching activities: Activities taking place in the classroom and activities taking place outside of the classroom. For their survey, to qualify as flipped classroom, classroom activities have to consist of "interactive group learning activities" (Bishop et al., 2013, p. 4), while outside classroom activities have to consist of "direct computer-based individual instruction" (Bishop et al., 2013, p. 4). Their definition excludes formats that do not use videos for outside classroom activities as well as formats that include traditional lectures among classroom activities. Thus, according to their definition, the format proposed in this article does not qualify as flipped classroom even though it incorporates components of that format.

Flipped classrooms have already been deployed and evaluated in STEM education: Amresh et al. (Amresh et al., 2013) introduced a flipped classroom in an introductory computer science class of 39 students. Their evaluation shows that while the flipped classroom improved examination results, some students were overwhelmed and intimidated by the format. Gilboy et al. (Gilboy et al., 2015) applied a flipped classroom to two classes on nutrition: Outside of the classroom, students learned from mini-lectures and written material while all of the in-classroom time was devoted to active learning in form of a "jigsaw classroom" (see (Aronson, 2002)). In an evaluation, the majority of students preferred the classroom learning activities to a traditional lecture of similar duration.

While the aforementioned studies represent flipped classrooms adhering to Bishop et al.'s definition, other studies examined flipped classrooms in which some kind of lecture took place during the in-classroom activities: Stelzer et al. (Stelzer et al., 2010) introduced flipped classrooms in an introductory course in physics attended by 500 to 1,000 students. Classroom activities were conducted in groups of 24 students. At the beginning, the course format did not include any lectures among the classroom activities, but it was later adapted to contain a small lecture at the classroom sessions' beginning recapitulating the outside classroom activities. The authors observed a positive influence of the educational format on examination results. Furthermore, the course was perceived by the students as less difficult than the same course taught in a traditional manner. McLaughlin et al. (McLaughlin et al., 2014) used a flipped classroom approach to teach pharmaceutics to 162 students. Their approach included on-demand micro-lectures to "reinforce and, if needed, redirect students' learning" (McLaughlin et al., 2014, p. 3).

The "Taxonomy of Educational Goals" by Bloom (Bloom, 1956) defines a hierarchy of six educational goals aimed at comparing and classifying of educational formats and content: Knowledge, Comprehension, Application, Analysis, and Evaluation. Each of the aforementioned goals has all previously men-

tioned goals as precondition, the rationale being that nothing can be applied without first being known and comprehended. Every goal except Application is further subdivided into more specific goals. The taxonomy was revised by Krathwhol (Krathwohl, 2002): The goals are expressed as verbs and reordered resulting in the goals Remember, Understand, Apply, Analyze, Evaluate, and Create. In the revised taxonomy, Remember is not equal to Knowledge from the original taxonomy, in fact Knowledge is broken up into four dimensions: Factual Knowledge, Conceptual Knowledge, Procedural Knowledge, and Metacognitive Knowledge. These knowledge dimensions are considered orthogonal to the formerly mentioned goals, resulting in a two-dimensional taxonomy. This taxonomy makes possible to express distinct objectives such as "analyze factual knowledge" and "analyze conceptual knowledge".

Peer review supports students in attaining parts of the learning goal Analyze, as this goal concerns itself with "making judgements based on criteria and standards" (Krathwohl, 2002, p. 215). Peer review consists in students providing feedback on their peers' work. This feedback can either replace the lecturer's feedback or extend it. Peer review is reflexive in the sense of making reviewers reflect on their own work (Topping, 1998; Lundstrom and Baker, 2009; Williams, 1992) and has been shown to have a positive impact on reviewers' own writing (Lundstrom and Baker, 2009). In a study in tertiary STEM education, Heller and Bry (Heller and Bry, 2018) used peer review for providing feedback on coding assignments. Their study showed that in the majority of cases, the delivered peer review was correct and that the majority of the students found helpful to deliver peer reviews, but that they found only sometimes helpful the peer reviews they received on their own work.

To support students giving peer review, scoring rubrics can be provided to students (Cho et al., 2006). A rubric is "a scoring guide to evaluate the quality of students' constructed responses" (Popham, 1997, p. 72). Jonsson and Svingby (Jonsson and Svingby, 2007) conclude in their survey, that the use of scoring rubrics in peer review can further support students' learning.

According to Hattie and Timperley (Hattie and Timperley, 2007), feedback is "information provided by an agent (...) regarding aspects of one's performance and understanding" (Hattie and Timperley, 2007, p. 81) and effective feedback should answer three questions (Hattie and Timperley, 2007, p. 87):

- "Where Am I Going?" (Hattie and Timperley, 2007, p. 88): With *feed up,* a task can be contextualized showing learners to what end a certain concept should be learned.
- "How Am I Going?" (Hattie and Timperley, 2007, p. 89): The *feed back* dimension gives information about students' performance on a task and how their performance relates to some performance goal.
- "Where to Next?" (Hattie and Timperley, 2007, p. 90): With *feed forward,* learners' can be given an outlook where to next, e.g., by providing them with further sources of information on concepts not understood or on related concepts.

Feedback can be provided on four levels (Hattie and Timperley, 2007):

- Task-level feedback is given pertaining students' work on a task, e.g., the correctness of a mathematical computation.
- Process-level feedback aims to give information about the processes that are involved in achieving a goal, e.g., giving learners information about what rules to apply to simplify a mathematical equation.
- Self-regulation feedback aims to provide feedback about students' self-regulation skills and metacognitive knowledge, e.g., the skill to self-evaluate one's own work.
- Self-level feedback is unrelated to the task and only pertains the student, e.g., "You are very talented."

With the exception of self-level feedback, each of the aforementioned feedback levels is effective depending on the learner and the situation in which the feedback is given. (Hattie and Timperley, 2007) As of the timing of feedback, Hattie and Timperley conclude from surveying various studies that task-level feedback should be provided as soon as possible and process-level feedback should be delayed so as not to inhibit the construction of the learners' autonomy.

Scaffolding is defined by Wood et al. (Wood et al., 1976) as a "process that enables a child or novice to solve a problem (...) which would be beyond his unassisted efforts" (Wood et al., 1976, p. 90). One way to provide scaffolding is by feedback. Merriënboer et al. (Van Merriënboer et al., 2003) further specify scaffolding as a combination of "performance support and fading" (Van Merriënboer et al., 2003, p. 5), the support first being provided to the students while they are working towards a goal and later being gradually withdrawn, or "faded". Scaffolding can be provided in person by instructors which generally requires them to work with a single student or a small group of students or provided by software what can

accommodate more students and be used in out-of-classroom learning scenarios. Automatic scaffolding is often provided in form of feedback and can be found in intelligent tutoring systems and adaptive learning environments.

The "Test My Code" environment by Vihavainen et al. (Vihavainen et al., 2013) provides feedback in form of both automated tests and incremental exercises, i.e., exercises partitioned into guidance-giving subtasks. Pedro et al. (Sao Pedro et al., 2014) developed an automatic scaffolding system which supports students in developing data collection skills by means of simulations of scientific experiments. Their environment detects off-track students and provides them with feedback to help them get back on track. In their study, they showed that the automatic scaffolding provided by their system helped students to develop data collection skills.

Automatic scaffolding is explored for non-STEM subjects as well: He et al. (He et al., 2009) propose a system for automatic assessment of text summaries produced by students which provides feedback in the form of key points missing in students' summaries. Yang (Yang, 2015) provides computer-generated feedback about a concept map created by students about a text's content. Yang observed that the feedback provided by their system had a positive impact on the students' reading comprehension as well as on their summary-writing skills.

Didactic reduction is very similar to scaffolding and fading. Didactic reduction is a term coined by Grüner (Grüner, 1967) and refers to breaking down a concept into its most basic parts ("scaffolding") while still retaining its functionality. Later on, the more advanced parts can be put back step by step ("fading") until the concept is available in its full complexity. In tertiary STEM education, new concepts are often embedded into other concepts making it hard for students to grasp the actual concept to learn. With didactic reduction, other concepts can be omitted at first, and later on be reintroduced step by step.

The article Heller et al. (Heller et al., 2018) describes how various course formats can be realized by Backstage.[1] Phased classroom instruction is briefly mentioned, however, without referring to its evaluation and implementation first reported about in this article.

[1] https://backstage2.pms.ifi.lmu.de:8080

3 PHASED CLASSROOM INSTRUCTION AND ITS ENABLING TECHNOLOGY

The following section introduces the course format, its technological support, and the actual implementation in the accompanying lecture of a software development practical.

3.1 Format

A session in the format consists of one or more blocks each of them consisting of three phases:

1. a lecture (subsequently *mini-lecture*) of about 15 to 25 minutes introducing new concepts to the students
2. an extensive practical exercise where students work in groups of two to four students putting the newly acquired concepts to use. During this phase, the lecturer stands ready to provide struggling teams with support
3. a peer review where each team is assigned another team's submission for review

Mini-lectures minimize the amount of passive listening and therefore counteract the problem of students' attention dropping during lectures after about 25 to 30 minutes (Stuart and Rutherford, 1978). To restore students' attention, Young et al. suggest that "short breaks or novel activities may temporarily restore attention to normal levels" (Young et al., 2009, p. 52), which leads to the next part of the format, the exercise for students to work on in groups. Depending on the subject taught, the exercise can take different forms: From a larger coding exercise, a mathematical proof, to the creation of a larger body of text about some topic. Working on exercises in groups leverages benefits of collaborative learning, such as an improvement in academic achievement (Prince, 2004).

The combination of lecture, exercise, and peer review brings the majority of Bloom's taxonomy (in the revised form) into the classroom: For "Knowledge", the mini-lectures provide both the factual and conceptual knowledge required for the exercise. If exercises are formulated in a scaffolded way, they can provide procedural knowledge: Breaking a bigger exercise into smaller subtasks shows students the problems the bigger problem is composed of and by that one possible approach to solve the problem. Peer review teaches students the ability to evaluate and correct work of others, and therefore supports students' meta-cognitive knowledge through its reflective nature.

The dimensions “Remember” and “Understand” are covered by the mini-lectures, “Apply” and (parts of) “Analyze” by exercises, and (parts of) “Analyze” and “Evaluate” through peer review. Therefore, the format should cover six of the seven steps of Bloom’s taxonomy. The last of step, “Create”, would arguably require more extensive problems and a longer duration, which can hardly be implemented in a lecture.

3.2 Enabling Technology

The enabling technology has to provide lecturers with a group organization component and with means of keeping an overview of the groups’ work even in larger classes. The technology should make it easy for lecturers to identify groups in need of support. To this goal, all students’ submissions should be be available to the software. This is achieved by providing students with web-based editors integrated into the software to work out their submissions.

The enabling technology is available in the learning and teaching platform Backstage which orchestrates the process starting from assigning the exercises to the teams, providing them with an appropriate editor for creating their submission, and finally randomly assigning each team another team’s submission for peer review. The number of participants is not limited by the technology, as Backstage was built to support a few hundred students, but by the format itself: It is likely that after reaching a certain number of students, collaboration will be heavily impaired due to sheer number of students being present in the classroom. With increasing number of students, the number of teams that require personal support through the lecturer is bound to increase as well, which at some point becomes too much for a single lecturer to handle.

Backstage provides students with editors that support them in solving specific types of exercises, such as a coding editor for programming tasks, a logic editor for creating logic proofs, and an editor for induction proofs. Each of the editors creates automated feedback and in this way provides scaffolding to students. The logic editor, e.g., disallows incorrect steps in a proof and the code editor automatically runs tests on the students’ submission. Didactic reduction is implemented by focussing on the important concept to be exercised: The code editor removes compiler and file management from the equation and the proof editors focusses solely on which rules to apply, not on the structure of the resulting proof. Groups choose by themselves how to collaborate: Work together on a single computer or split the task into subtasks and work on those subtasks in smaller groups. When working on two or more computers, Backstage supports students in quickly exchanging their worked on artifacts and creating their final group submission.

The feedback and the scaffolding provided by those editors can potentially replace lecturer interventions for advanced groups, which in turn, frees up the lecturer’s time, allowing them to support groups that need their help.

During the exercise phase, Backstage provides lecturers with an overview of the currently worked-on submissions, allowing them to stay on top of a larger number of submissions and identify struggling groups much faster as opposed to the alternative of walking around glancing at screens and worksheets. In a large classroom with extensive exercises, a more reduced view on the classroom is required: Aggregated measures, such as the percentage of passing tests, can be shown to the lecturer. After the exercise has been finished, the online availability of all submissions allows teachers to adapt their teaching to an exercise’s result: Weaknesses and misconceptions as well as positives can immediately be discussed by the means of students’ submissions.

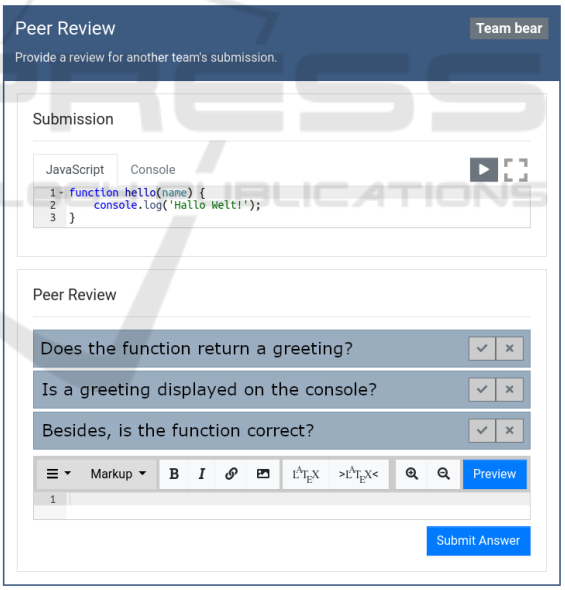

Figure 1: Interface in which a peer review is given: Another group’s submission displayed in the same editor the submission was worked on (top), the checklist of conditions a correct submission has to fulfill (middle), and a text area for further comments (bottom). The checklist items have been translated from German.

For peer review, Backstage provides the reviewing team with the submission to review in a “try-out” environment (if applicable), and with a checklist of conditions that a correct submission has to fulfill, such as “Has each case of a proof by induction an induction

hypothesis?" For each checklist item, the reviewing team has to decide if the solution fulfills the condition. Those checklist items serve two purposes: First, they act as a simplified scoring rubric which guides students in giving feedback. Second, they allow to generate correctness feedback for exercises for which correctness cannot be automatically determined: If each of the checklist items is marked as fulfilled, the submission is seen as correct. An example for the view in which the peer review is conducted can be seen in Figure 1: In the top, another teams solution can be seen, the aforementioned checklist of conditions can be seen in the middle, and in the bottom there is a text area where further comments can be provided by the reviewing team.

3.3 Implementation

A total of five sessions using the format described above were developed for the accompanying lecture of a software development practical. In this software development practical, students are tasked to develop a game in groups of four using the programming language JavaScript. Each session was built for a 135 minute lecture with a 15 minute break halfway through. The lecture material was created from scratch. The creation of eleven mini-lectures and eleven exercises amounted to around 60 hours. A first evaluation took part in the winter term 2018/19. For the initial iteration of the practical, only sixteen students were admitted in order to initially test the mini-lectures and exercises before using the course format and material for a greater number of students. Students worked in groups of four.

To teach students the techniques required for the development of their own game, mini-lectures were structured around the game "Snake".[2] In each mini-lecture, one technical aspect of game development was explained using Snake and implemented in the subsequent exercise in each group's personal version of Snake. Groups always worked on the same code, so that at the end each group had implemented their own version of Snake from scratch.

The exercises were worked on in a specialized JavaScript editor which can be seen in Figure 2. It enabled students to write and run code directly from their web browser. The utilized version of the JavaScript editor did not support unit tests, i.e., the correctness of a submission was determined solely by peer review or the students themselves. Indeed, for the exercises at hand, groups could easily check for correctness themselves as the game can either be played correctly or not.

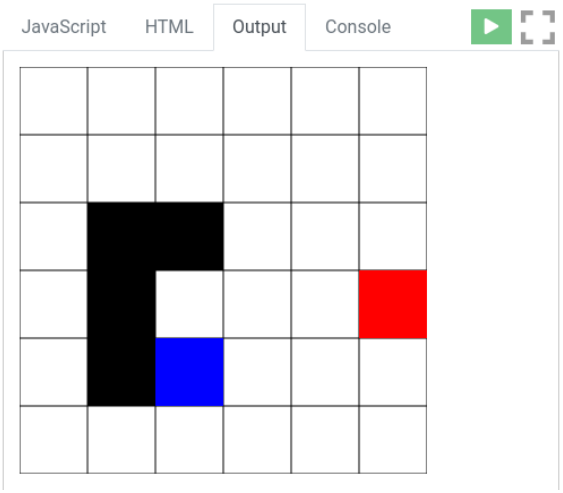

Figure 2: The JavaScript editor used by the teams for creating their submissions showing the output of a group's submission. The tabs at the top allow to switch between the editor's functionalities. In the image, *output* is selected, which shows the result of the program's execution: The black rectangle is the snake's body, the blue rectangle its head, and the red rectangle the apple the snake has to eat.

Each mini-lecture began with a "why?" contextualizing its content, providing the students with a learning goal and why this learning goal should be attained, acting as general feed forward. Each mini-lecture closed with a review of the exercise that followed, where the exercise and the steps it consists of were reviewed. The exercise description shown to students was structured in a similar way: The main exercise and steps on the way providing a form of instructional scaffolding. Everything was displayed at once, i.e., the visibility of a step was not dependent on completing the preceding steps.

The exercises were worked on in groups of four students sitting next to each other. Students could either collaborate on a single computer or work on two or more computers and bring their results together afterwards.

4 EVALUATION

Phased classroom instruction has been evaluated as course format for a JavaScript software development practical during the winter term 2018/19 (see Section 3.3).

The sessions took place during the first five weeks of the term. During the following weeks, the students implemented a larger software project independently

[2] https://en.wikipedia.org/wiki/Snake_(video_game_genre)

putting the concepts learned during the sessions to use. During this phase, each team was supported by weekly meetings with a teaching assistant.

4.1 Method

Data for the evaluation was collected using a survey and taken directly from the Backstage system as well. The survey was conducted during the final session of the practical and consisted of six parts:

1. Four questions referring to the students' course of study, current semester, gender, and team they were in.
2. Six questions measuring the students' attitude towards the course format and its elements.
3. Six questions measuring the students' attitude towards the content and structure of mini-lectures and exercises.
4. Six questions measuring the students' attitude towards the enabling technology.
5. Five questions measuring the students' programming proficiency using an adapted version of the survey by Feigenspan et al. (Feigenspan et al., 2012).
6. Three questions in form of free-text questions, asking about what they liked most, what could be done better and for further comments.

For parts (2), (3), and (4), a six-point Likert scale from strongly agree to strongly disagree with no neutral choice was utilized. All submissions and peer reviews were retrieved directly from the Backstage system. A single lecturer determined for each team and exercise the point in time – if at all – in which the exercise was solved correctly and whether the peer review delivered was correct.

The correctness of an exercise was determined in a strict way: A submission was seen as correct, if and only if the whole task was solved correctly. For peer review, the lecturer determined which of the checklist items (see Section 3.2) the submission properly fulfilled and compared their judgement to the group's review. A review was seen as correct if, and only if, lecturer and group judgement were identical. The written comments were not taken into account for determining a peer review's correctness.

Data from the first lecture had to be excluded from the evaluation, because internet connection broke down during the lecture, making it nearly impossible for students to turn in their submissions and provide peer review.

4.2 Results

All of the sixteen students completed the questionnaire, six of whom were female, ten male. In each session, at least three members of each team were present and the majority of the time all of the participants were present.

Figure 3 shows the students' attitude towards the course format: The vast majority of students strongly agreed that the immediate exercise and the discussion with their team mates helped their understanding of the topic and none of the students would have preferred to have a traditional lecture. Regarding peer review, students found their own reviewing of the submissions of other teams more helpful than the reviews they received for their own submission.

Figure 4 shows the students' attitude towards the content and structure of mini-lectures and exercises: Most students agreed that the mini-lectures were sufficient to solve the exercises and preferred exercises building upon each other to independent exercises. The majority of the students' did not find the exercises too difficult or too big.

The results for the students' attitude towards the enabling technology can be seen in Figure 5: While the majority of the students found that the JavaScript editor helped them getting started with JavaScript, a number of students thought the opposite. Furthermore, around a third of the students would have preferred to solve the exercises in a real development environment as opposed to the JavaScript editor on Backstage. Students thought that the views for submitting a solution and giving peer review were clearly designed and the course format was supported well by Backstage.

The overall correctness for each team's submission for every exercise can be seen in Figure 6. If a team failed to finish a task during the lecture, they were tasked to complete it as a team outside class. A correct task, finished outside class, is indicated by a yellow square. Generally, the correctness of the submissions decreases rapidly after the first two lectures and somewhat recovers for the last lecture.

The correctness of the peer reviews can be seen in Figure 7: In the majority of cases, the peer review was correct and there are no big differences in the number of correctly delivered peer reviews between the groups.

Figure 8 shows how long each group took to turn in a correct (or, otherwise final) submission determined by the lecturer as described in Section 4.1. Submissions that were submitted after 100 minutes are omitted so that only submissions that were done during the session are represented. Generally, dura-

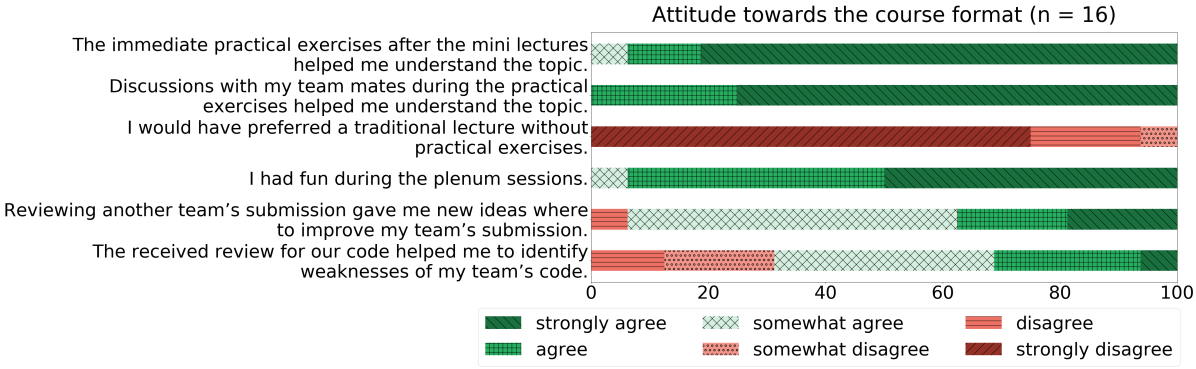

Figure 3: Students' responses to the section referring to their attitude towards the course format.

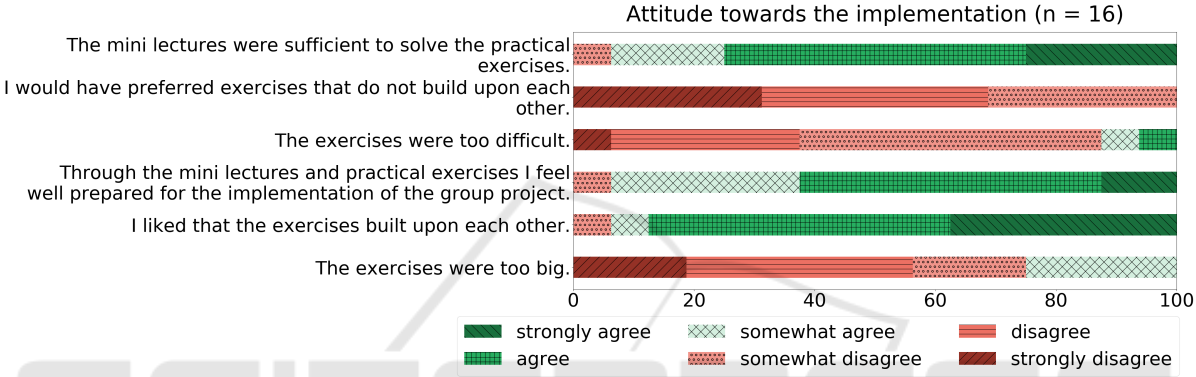

Figure 4: Students' responses to the section referring to their attitude towards the implementation of the course format.

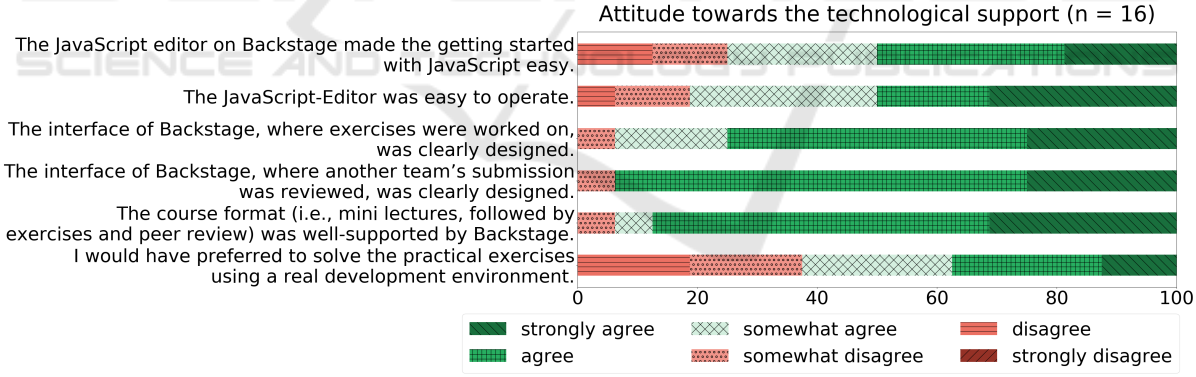

Figure 5: Students' responses to the section referring to their attitude towards the technological support.

tions for the first two sessions fit the duration proposed by the course format. The exercises in the fourth session were not solved by a single team in time, with the times normalizing again for the final session.

On a scale from 0 to 9, the average coding proficiency was rated with 3.69 (Min: 1, Max: 9, Var: 7.30), participants reported a coding experience of in average 3.47 years (min: 0, Max: 10, Var: 9.16). Eight of the participants reported that they code in their free time besides the assignments for the courses they are attending.

4.3 Discussion

Generally, the course format was well-liked by the students, with students preferring the mini-lectures combined with practical exercises over the traditional lecture and reporting better understanding through the immediate practice and discussion with their team mates. The results for peer review are as expected from other studies on peer review: Students seem to

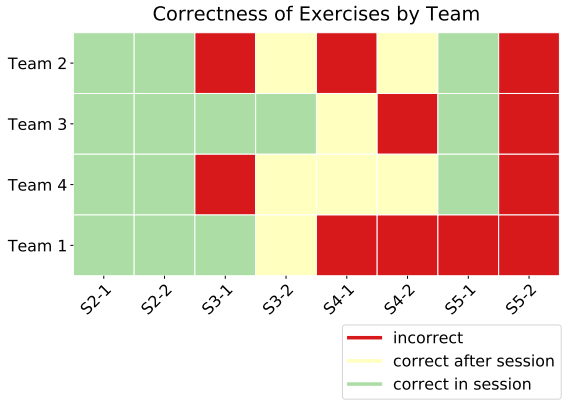

Figure 6: Correctness of each team's submission for each exercise.

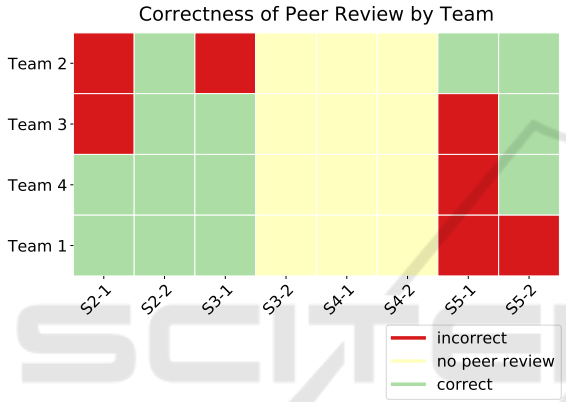

Figure 7: Correctness of each team's peer review for each exercise.

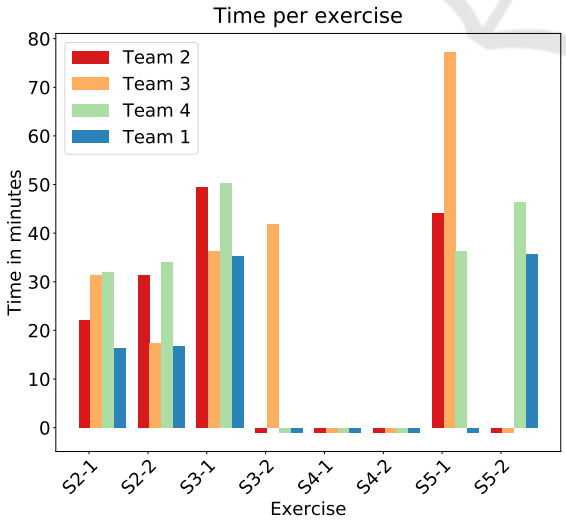

Figure 8: Time to submission for which correctness was determined for each team.

profit more from looking at other teams' submissions and providing review as from the review they receive from other groups. While the provided peer reviews seem to be mostly correct, a wrong review led to a team's submission being shown as incorrect, which could have led to this attitude. To solve this, the quality of the peer review has to be improved, so that false negatives are the exception. One way to increase the quality of peer reviews and therefore the value for reviewees is proposed by Heller and Bry: Assigning submissions of struggling students (here: groups) to well-performing students (Heller and Bry, 2018). In the present scenario, an imaginable assignment strategy would be to assign submissions of teams who failed the previous exercise to teams who successfully solved said exercise.

The majority of students somewhat disagreed that the exercises were too difficult. This, in combination with the fact that the exercises for the third session were not solved by a single team during the designated time, could be an indicator that some of the exercises should be made easier for the next iteration of the practical. During the third lecture, object-oriented programming with JavaScript was introduced, which is different to that of the Java programming language taught earlier in the curriculum which the students are accustomed to. This could possibly explain why the students were struggling in this session. Subsequently, the next exercise could not be solved as well, because it was dependent on a working submission for the preceding task.

The results shown in Figure 5 show that Backstage supported the course format well and that the views for submission and peer review were clearly designed. While most of the students thought that the JavaScript editor made their start with JavaScript easier, there are a few students who disagreed with this statement and an even larger number of students who would prefer to use a real development environment for the exercises. The authors' hypothesized that this may be caused by previous experience and being accustomed to the feature set offered by full-fledged development environments. A correlation analysis between all values for coding proficiency and those two questions did not reveal any correlation, which suggests that the reason for this attitude most likely lies somewhere else.

The previous experience of the participants varied greatly, which is reflected in the completion times for exercises as well: Groups' completion times are often differing in the amount of 10 to 15 minutes, which opens up the possibility of further collaboration in the class room: peer teaching. Groups who already completed their submission can support other groups in solving the exercise.

5 CONCLUSION AND PERSPECTIVES

This article introduced a course format and its enabling technology which intends to bring active learning to large class tertiary STEM education. The format combines mini-lectures with extensive exercises worked on in groups. The enabling technology supports the lecturer in staying on top of larger classes by providing them the currently worked on submissions and aggregated values about the groups' submissions for an easy overview. Students are supported by editors specialized to certain types of exercises which can provide scaffolding in form of automated feedback.

An evaluation was carried out with a small number of participants to first collect feedback about format and course material, improving both, before running another iteration with a larger number of students in the summer term 2019. The evaluation has shown that students heavily favor the active approach with none of them preferring a traditional lecture over the new course format. A problem with the exercise design was identified through the evaluation: For one session, all teams failed to complete the exercises, most likely due to a new, unclear subject matter and too extensive exercises. The students found the enabling technology to support the format well and its views clearly designed.

To address the issue of teams not finishing exercises, two approaches are imaginable: Exercises should be scaffolded in a way that shows teams the steps to a complete solution, with the current step in the best case always being an attainable step. In the evaluated format, all steps were visible at once, which most likely led groups to focus on more than one step at once which could be solved by showing students only one step at a time. Another approach is to provide struggling groups (e.g., groups not solving the previous exercise correctly) a more detailed template or a smaller task to work on, so that at the end of each session each group has achieved something which provides a sense of achievement.

The format seems to have potential, but for the format to work for large class teaching, the enabling technology has to be able to reliably calculate aggregated measures that allow lecturers to identify struggling groups or general problems, because with increasing exercise complexity and classroom size even the most experienced lecturer can hardly have an overview – and an understanding – of all submissions, even if they are presented to them in an easy way. How such an aggregated measure can be calculated has to be examined in further studies.

ACKNOWLEDGEMENTS

The authors are thankful to Maximilian Meyer for the implementation of the JavaScript editor as part of his master's thesis (Meyer, 2019).

REFERENCES

Amresh, A., Carberry, A. R., and Femiani, J. (2013). Evaluating the effectiveness of flipped classrooms for teaching CS1. In *Frontiers in Education Conference, 2013 IEEE*, pages 733–735. IEEE.

Aronson, E. (2002). Building empathy, compassion, and achievement in the jigsaw classroom. *Improving academic achievement: Impact of psychological factors on education*, pages 209–225.

Bishop, J. L., Verleger, M. A., et al. (2013). The flipped classroom: A survey of the research. In *ASEE national conference proceedings, Atlanta, GA*, volume 30, pages 1–18.

Bloom, B. S. (1956). *Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain*. Longmans.

Bonwell, C. C. and Eison, J. A. (1991). *Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports.* ERIC.

Cho, K., Schunn, C. D., and Wilson, R. W. (2006). Validity and reliability of scaffolded peer assessment of writing from instructor and student perspectives. *Journal of Educational Psychology*, 98(4):891.

Feigenspan, J., Kästner, C., Liebig, J., Apel, S., and Hanenberg, S. (2012). Measuring programming experience. In *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on*, pages 73–82. IEEE.

Frederick, P. J. (1986). The lively lecture – 8 variations. *College teaching*, 34(2):43–50.

Gilboy, M. B., Heinerichs, S., and Pazzaglia, G. (2015). Enhancing student engagement using the flipped classroom. *Journal of nutrition education and behavior*, 47(1):109–114.

Grüner, G. (1967). Die didaktische Reduktion als Kernstück der Didaktik. *Die Deutsche Schule*, 59(7/8):414–430.

Hattie, J. and Timperley, H. (2007). The power of feedback. *Review of educational research*, 77(1):81–112.

He, Y., Hui, S. C., and Quan, T. T. (2009). Automatic summary assessment for intelligent tutoring systems. *Computers & Education*, 53(3):890–899.

Heller, N. and Bry, F. (25-28 September 2018). Peer teaching in tertiary STEM education: A case study. In *The Challenges of the Digital Transformation in Education - Proceedings of the 21st International Conference on Interactive Collaborative Learning (ICL2018)*, volume 2, page to appear. Springer.

Heller, N., Mader, S., and Bry, F. (2018). Backstage: A versatile platform supporting learning and teaching format composition. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*, page 27. ACM.

Jonsson, A. and Svingby, G. (2007). The use of scoring rubrics: Reliability, validity and educational consequences. *Educational research review*, 2(2):130–144.

King, A. (1993). From sage on the stage to guide on the side. *College teaching*, 41(1):30–35.

Krathwohl, D. R. (2002). A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218.

Lundstrom, K. and Baker, W. (2009). To give is better than to receive: The benefits of peer review to the reviewer's own writing. *Journal of second language writing*, 18(1):30–43.

McLaughlin, J. E., Roth, M. T., Glatt, D. M., Gharkholonarehe, N., Davidson, C. A., Griffin, L. M., Esserman, D. A., and Mumper, R. J. (2014). The flipped classroom: a course redesign to foster learning and engagement in a health professions school. *Academic Medicine*, 89(2):236–243.

Meyer, M. (2019). A browser-based development environment for javascript learning and teaching. Master thesis, Institute of Computer Science, LMU, Munich.

Popham, W. J. (1997). What's wrong-and what's right-with rubrics. *Educational leadership*, 55:72–75.

Prince, M. (2004). Does active learning work? a review of the research. *Journal of engineering education*, 93(3):223–231.

Sao Pedro, M. A., Gobert, J. D., and Baker, R. S. (2014). The impacts of automatic scaffolding on students' acquisition of data collection inquiry skills. *Roundtable presentation at American Educational Research Association*.

Stains, M., Harshman, J., Barker, M., Chasteen, S., Cole, R., DeChenne-Peters, S., Eagan, M., Esson, J., Knight, J., Laski, F., et al. (2018). Anatomy of STEM teaching in north american universities. *Science*, 359(6383):1468–1470.

Stelzer, T., Brookes, D. T., Gladding, G., and Mestre, J. P. (2010). Impact of multimedia learning modules on an introductory course on electricity and magnetism. *American Journal of Physics*, 78(7):755–759.

Stuart, J. and Rutherford, R. (1978). Medical student concentration during lectures. *The lancet*, 312(8088):514–516.

Topping, K. (1998). Peer assessment between students in colleges and universities. *Review of educational Research*, 68(3):249–276.

Van Merriënboer, J. J., Kirschner, P. A., and Kester, L. (2003). Taking the load off a learner's mind: Instructional design for complex learning. *Educational psychologist*, 38(1):5–13.

Vihavainen, A., Vikberg, T., Luukkainen, M., and Pärtel, M. (2013). Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 117–122. ACM.

Williams, E. (1992). Student attitudes towards approaches to learning and assessment. *Assessment and evaluation in higher education*, 17(1):45–58.

Wood, D., Bruner, J. S., and Ross, G. (1976). The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100.

Yang, Y.-F. (2015). Automatic scaffolding and measurement of concept mapping for efl students to write summaries. *Journal of Educational Technology & Society*, 18(4).

Young, M. S., Robinson, S., and Alberts, P. (2009). Students pay attention! combating the vigilance decrement to improve learning during lectures. *Active Learning in Higher Education*, 10(1):41–55.