Web-Based Collaborative Learning in CS1: A Study on Outcomes of Peer Code Review

Rolando Gonzalez and Andreas Biørn-Hansen

Department of Technology, Kristiania University College, Norway

Abstract

Based on a teacher-organized student-to-student code review session, we gathered both quantitative and qualitative data from 177 first-semester Information Technology undergraduate students to learn about their thoughts, experiences and outcomes from collaborative learning through an online tool in an introductory programming course. The students were given a programming exercise to solve using JavaScript in a Web-based IDE facilitating real time code-sharing for peer-evaluation of code based on five provided evaluation criteria: naming of artifacts in the code, formatting of code, use of data types, use of execution flow, and other comments. In the survey questionnaire, we employed a five-point Likert scale with an additional text field for qualitative feedback. For the qualitative free-text based answers, thematic coding was carried out to identify recurring themes and topics in the students' answers. Based on the students' feedback, our results indicate that the majority of the participants had positive experiences resulting in self-reported learning through collaborative work, peer-evaluation and problem solving.

1 Introduction

According to Ying and Boyer [20], there is a gap in research on tools that improve learning through collaboration for novice programmers in higher education. The tools that exist are difficult to use for students, since they are not created for beginners and the tools that are created for the sake of research are neither further developed nor used after the projects end. This happens despite the literature showing that collaboration has positive effects on learning.

Courses within the fields of science, technology, engineering, and mathematics (STEM) are widely considered notoriously difficult from a student's perspective. The number of students participating in introductory courses to programming, commonly referred to as Computer Science 1 (*CS1*) continue to grow [13]. A recent study on failure rates in CS1 courses, Bennedsen and Caspersen [3] found an average of 28% of students in such introductory courses failing [3]. Challenges reported by students were synthesized by Medeiros et al. [12] in their systematic literature review on teaching CS1. Motivation, engagement and problem solving were prominent factors contributing to the challenging nature of learning programming. Looking to previous research on teaching CS1, we find that making use of collaborative components such as pair programming and pair review improves code quality, retention rates and

This paper was presented at the NIK-2020 conference; see http://www.nik.no/.

confidence [20], motivation and soft skills [11]. Student motivation is important to attain a high retention rate. A study on the process of hiring and evaluating recently graduated computer scientists and software engineers emphasized the importance of soft skills, including communication and cooperation [10] – two core components of programming collaboration, which we were interested in researching further.

The study at hand is based on qualitative and quantitative data gathered from 177 first-semester undergraduate students in information technology during their CS1 course. The course is focused on providing the students with fundamental theoretical knowledge and hands-on experience with core components in programming, by use of JavaScript, such as data types, execution flow, loops, variables, syntax and structure.

Our aim is investigating the students' perceived learning outcome from real time collaborative learning using an online collaboration tool, while working with a programming task. The complexity of the task was not especially high and this was important, as the focus of the assignment was not solely on solving the programming problem, but also on reviewing and and discussing with others. The students were to use the Web-based collaborative programming tool Scrimba[1]. Scrimba allows for among other easy sharing of code, real time screen share, making a copy of a project to develop it and share the changes and comments. Based on this, we formed two research questions:

RQ$_1$: How do students perceive working in groups on a programming task through an online collaborative tool?

RQ$_2$: How does the collaborative effort affect the students' perceived learning?

To answer these questions, we carried out an analysis based on quantitative (Likert scale) and qualitative data with thematic coding. An important distinction between the study at hand and related work, is that we explore students' *perceived* learning (see RQ$_2$). We do not assess the impact of collaborative learning on the students' final grades in CS1, neither do we employ objective evaluation from the students' perspective. The objective of this study is to better understand the students' perception of collaborative components and how students perceive the impact on their learning in the context of problem solving in CS1.

In Section 2, we highlight seminal studies in collaborative learning and teaching introductory programming. Section 3 describes our research methods, the programming and peer evaluation assignment and instructions, and data analysis techniques. In Section 4, we present findings of the study, which are discussed and placed in context of previous research in Section 2. Finally Section 5 provides a conclusion and suggestions for future work.

2 Related Work

Collaborative programming is seen as an effective learning approach for novice programmers to improve their computational thinking skills. By receiving and providing peer teaching, alternative ways of solving programming solutions can be achieved. It coincides with constructivist theories where one sees the learner as creating their knowledge through actively participating in social interaction

[1]Scrimba website: `https://scrimba.com/`

with peers [19]. Wing [18] states that *"Computational thinking is reformulating a seemingly difficult problem into one we know how to solve"*. Wing further explains how Computational thinking is about different activities on different levels of abstraction ranging from finding errors to using heuristics to find best ways to solve an issue. The student must work on their ability to think in multiple levels of abstraction while at the same time being imaginative. There are different manners of making the students work together, ranging from group discussions to pair programming. Pair programming is seen as an effective manner of learning versus working alone, resulting in higher quality of code, more confidence, performing better on exams and in addition letting the student enjoy the process more [5] [20]. However, even though the literature claims collaborative learning is beneficial, students are often forced to learning programming individually, according to Shadiev et al. [15]. The students seldom interact with peers and teachers and this may lead to students learning the subject in a wrong manner in addition to not developing certain thinking skills which are beneficial in programming.

Based on a systematic literature review of studies on approaches to teaching introductory programming, Vihavainen et al. [17] highlight cooperative learning as the most beneficial method for improving student pass grades. Also Beck and Chizhik [2] and Ying and Boyer [20] suggest numerous benefits of cooperative learning, including student retention and student achievements. However, according to Luxton-Reilly et al. [11] and their comprehensive survey on introductory programming, there is a lack of studies evaluating the effect of cooperative and collaborative learning. While we do not assess the impact of collaborative learning on the participants' grades, we do focus on two other aspects discussed by Luxton-Reilly et al. [11]: collaboration and communication. These aspects were also discussed by Urness [16] in an experiment on peer assessment in randomly assigned groups, stating that the most common complaint among their students was lack of response from their peers. Such behaviour may be expected in light of Shadiev et al. [15]'s findings that students often lack interaction with peers and teachers. Ying and Boyer [20] also report on a lack of research on the use of tools and how the collaborative process works.

Although previous research has highlighted numerous benefits of cooperative learning and variations of peer programming and peer assessment in CS1, also criticism of the teaching methods is to be found in the body of knowledge. An experiment on peer assessment in a Web development course by Aalberg and Lorås [1] found that the students do not necessarily trust the comments provided by their peers, and that the quality of the comments provided varied significantly. For our current study, these findings are particularly interesting as the participating students provided comments and feedback to their peers in an iterative manner, thus were able to see if their peers included or modified their code based on the provided comments. Another challenge related to pair programming is highlighted in a synthesis of surveyed literature provided by Salleh et al. [14]. The authors state that the actual pairing of peers is considered challenging, and that grouping based on skill level and personality type may generate beneficial results for the students.

Our motivation for conducting this research is to uncover perceived benefits and drawbacks of peer assessment from the perspective of students enrolled in our CS1 course. The importance of developing skills in communication and cooperation from an early stage in the computer science education is at the core of the project, in the

context of doing so through online collaborative tools.

3 Research Method

Based on related work in cooperative learning, peer reviewing and literature on constructivist activities in CS1, we developed a programming assignment along with a set of assertions for the participating students to answer. In this section, we elaborate on the assignment, questionnaire and analysis.

Pedagogical Philosophy

Constructivism is the underlying pedagogical philosophy under the assumptions we make when organizing the activities which we research in this paper. In constructivism, learning is something that happens as an active process in the student, and in collaboration with others. In our research we gave a task to students which required for them to first apply what they already have learnt, then expand their knowledge as the task was novel, and in dialogue with peers discuss and reflect on different aspects of the code. The goal of the social interactions is cognitive development and deep learning [6]. The assignment given to the students opened up for them helping each other, discussing and sharing ideas both orally and written forcing them to put into words the knowledge they had and obtained.

Assignment on Programming and Peer Assessment

The context we explore is how students in their first year of a Bachelor in Information Technology experience collaborating in groups on a programming task through an online collaboration tool called Scrimba[2], which opens up for easily sharing and collaborating on code. Our research is exploratory in the sense that the tool has opened for cases that we have not previously assessed and evaluated in teaching situation. However, we build upon previous experience [8] [9]. We assigned the students of an introductory programming course (CS1) an obligatory task which they were to solve through means of collaborative learning in groups of 2 or 3 peers. They worked on the same task, then gave each other feedback on each other's codes and went back to coding and so back and forth. The task was given to the students after 8 of 12 lectures, after having been introduced to most of the fundamental concepts in programming. They were given 1 week to solve the assignment and each student was to deliver their own version of the solution. The activity which the students were given thus opened up for the students to program on their own, present and explain their code, receive and give feedback, discuss and continue coding based on feedback. Taking into account the experiences provided by Urness [16] that students in their experiment would complain about peers not answering emails, we urged our CS1 students to start solving the assignment during a 2-hour lab exercise immediately following lectures when most students were already at campus. However, it was not given that all students managed to fulfill the requirements of the assignment and coding exercise within the 2 hours, and thus had to continue their work beyond the lab exercise. This process is illustrated and further detailed in Figure 1.

[2]Scrimba Website: `https://scrimba.com`

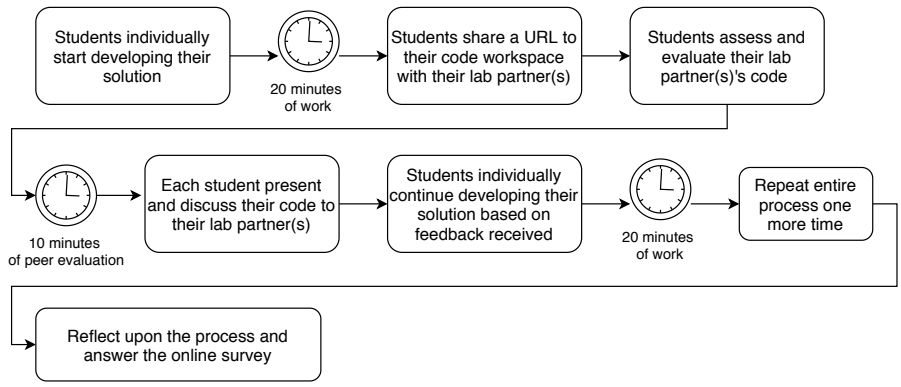

Figure 1: Instructions on executing the task.

Data Collection and Analysis

A folder containing program source code along with the feedback and comments provided to their peers was expected in the students' hand-in. Subsequent to the hand-in, the students were to answer an online survey questionnaire regarding their experiences and perceived learning outcomes. Our study is concerned with the results from the survey questionnaire rather than the aforementioned program source code and comments between student pairs. For this questionnaire, we developed five assertions ($A_{1...5}$) for the students to rank on a five-point Likert scale. Thematically, the assertions cover the use of the online collaborative tool Scrimba (RQ_1), the providing and receiving of peer evaluation (RQ_1 and RQ_2), and the overall perceived learning outcome (RQ_2).

- A_1: Scrimba made it easy for us to exchange ideas on how my code should be changed.
- A_2: A result of the evaluation phase was that I learned more about JavaScript.
- A_3: I made use of the feedback I received from my peers in the two evaluation phases.
- A_4: Scrimba helped me in providing guidance to the other group members.
- A_5: The use of Scrimba in this group project supported my learning of JavaScript.

Along with the assertions, we included a text field where the students were to *"Reflect upon the collaborative process in Scrimba for exchange of ideas and improvement of program code"*. These answers were translated from Norwegian to English with explicit focus on retaining sentiment and perceptions. The combination of quantitative Likert-point based data and free form text answers reflecting on the process gave us a considerable dataset for analysis. We take a descriptive approach to the quantitative data, analysing the students' assertions to find patterns and provide a holistic picture of perceived learning outcome. As for the qualitative data, the process leaned on thematic analysis using Bryman's Four stages, referred to among other by Carpendale et al. [4], along with Gibbs [7]' material on qualitative analysis. Our process started reading through the free form text responses, noting ideas, and rereading several times before creating and refining codes. Afterwards, we began identifying major themes and dimensions. The final stage was relating our findings to existing theories and understandings, in addition to adding one's own interpretation, which we present and discuss in Section 4.

4 Findings and Discussion

Throughout this section we provide and discuss our findings based on quantitative and qualitative data. The quantitative findings are based on the five point Likert scale questions from the questionnaire showing the percent-wise distribution of answers, while the qualitative findings arise from thematic coding based on the open answer reflection question. An aggregated view of the quantitative questionnaire data is displayed in Figure 2. We refer to this figure throughout the section to follow.

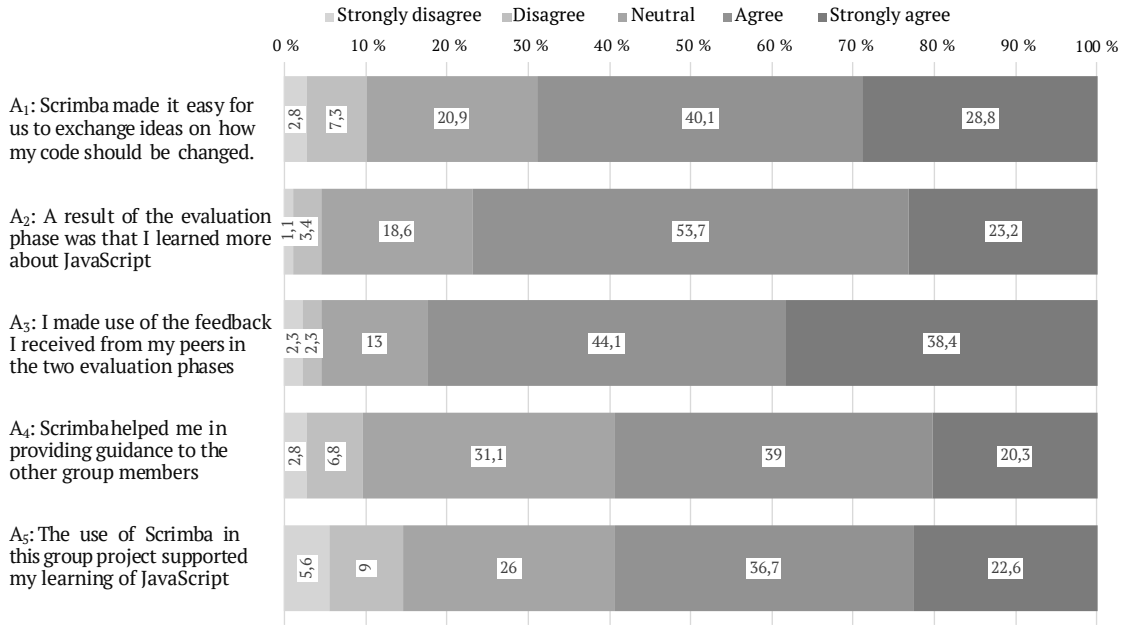

Figure 2: Percentage-wise results from the questionnaire assertions.

The process of thematic analysis resulted in the set of codes and associated overarching themes, the result of which is found in Table 1. These themes form the structure of the current section, where we investigate respectively the process of "obtaining knowledge", the "collaborative process", and the "working process".

Obtaining knowledge

The students reported on how the collaborative task let them gain insight into how their peers solved their own assignment, and how different their solutions could be. A student reported that *"The members of the group had all approached the task differently, and all the code was working"*. The expression of having different ideas in solving a case coincides with the thematic analysis done by Celepkolu and Boyer. Celepkolu and Boyer's findings also show that collaborative programming results in students observing other's solutions, comparing ideas and getting new ideas in the process.

As for the comparison of the coded solutions, and the resulting changes, our Likert scale statement on whether the 'tool helped students to exchange ideas on how the code should be changed' received 40.1% on agree and 28.8 on strongly agree. Celepkolu and Boyer reports on how code quality improves in situations of collaborative coding. Students in our study reported that the collaborative process led to a better result as it opened for various solutions to the same problem: *"We*

Meaning	Codes	Themes
Experiencing getting insight into alternative ideas and ways of solving the assignment.	Alternatives	Obtaining knowledge
Increased learning and understanding through reflection. Got to test different ideas.	Learning	
The student experienced their code improved after receiving feedback.	Improvement	
The student focused on receiving feedback.	Received feedback	Collaborative process
The student focused on giving feedback.	Gave feedback	
The group had positive discussions on the assignment.	Discussion	
The collaboration worked well.	Collaboration	
Focus on coding in itself. Being able to code efficiently.	Coding	Working process
Focusing on being able to find errors and lacks in code in the process.	Finding errors and lacks	
Easy to share and show code to the others through the tool.	Sharing of code	
Experienced technical issues, or lacks in the tool, for some making the tool hard to work in.	Technical issues	

Table 1: Themes and codes alongside their meaning, as derived from the qualitative free form questionnaire.

had different solutions and ideas to present, which lead to a better result. I learnt much!". We can expect the process to have been that as they see each other's codes and compare, they will undergo a process of selection, where the student uses their peers' codes as inspiration for improvement. It helps them get closer to do judgements on what is better or worse code.

On the Likert scale 39% agreed, while 20% strongly agreed on the collaboration tool enabling them providing guidance to others. An interesting finding is how some students reported being able to use the collaborative tool as a way to test different ideas while working on which feedback to give. This can be interpreted as that students see something in their peer's codes that may not be optimal and then try to rewrite it in different ways, before giving feedback orally. This is interesting in two manners: 1) The student providing feedback employs their knowledge and put it in context of another student's code, and 2) the student receiving feedback will receive a modified version of their code. I.e., it is guidance that is given as an expansion of the student's abilities, manifested as code, at that moment. 82.5% reported on the Likert scale having used feedback they received from peers to improve their solution.

[5] reports in their thematic coding that collaborative coding makes coding more enjoyable and fun, and Celepkolu and Boyer [5] and Ying and Boyer [20] both reported how collaborative coding improved confidence in a number of students. Few of our students reported directly that it was fun: *"It was fun to see how*

other solved it.", however several expressed how the collaboration worked well and were content with the learning process. Neither did we find any direct expressions of improved confidence. We believe our questions may not have opened up for expressing emotions experienced in the collaborative process. However, our Likert scale show how 76.9% either agree or strongly agree on having learned more about coding and one may see a link between the feeling of having learned and confidence in one's coding skills.

Some students reported that they finished the task very quickly: *"...The idea is good and one gets many different feedback[s], but since the assignment wasn't complex enough it led to having a good suggestion already after the first round [of feedback]"*. It can be worth mentioning that for learning to take place one must make tasks that are adaptive to level of competence so that students can obtain knowledge according to their level. The majority of students reported of learning, however, the students discussing the low level of difficulty may be above-average students.

Regarding levels of competency affecting student collaboration, Salleh et al. [14] report how students working in groups where their peers are on the same level of competency may learn better. In addition, the level of engagement may affect the process. Celepkolu and Boyer [5] mentions how some students came in group with people they saw as freeriders including people who were not necessarily so motivated to learn. One of our students reports that *"...not all are in groups with engaged student that one can discuss and exchange knowledge with. In my case I mostly helped the others solve the task, that resulted in me not getting so much feedback on my own code"*, showing a possible case of asymmetry in both competency and engagement.

Collaborative process

The overall results indicate that the majority of students had good experiences collaborating through the Web-based IDE, thus improving the processes of sharing, showing and discussing code (68,9% [strongly] agree). They were able to both give and receive feedback, although we see from the comments that a number of the students report mainly either giving or receiving feedback. The feedback was about strengths and weaknesses in the code in addition to variations on how to code the different parts of the solution. The feedback was often given both textually, as comments in the code, and orally afterwards in the demonstration and discussion part. Some students reported it made it easier to note comments first and then express them orally, *"We chose to send the code to each other for evaluation and feedback. Scrimba worked fine. The feedback was mostly spoken verbally, as we were sitting in groups."*

For students solving the assessment remotely, i.e., not while sitting in close proximity to each other, Scrimba aided also in this scenario, *"Scrimba worked fine even when we sat at home in the evening. As long as the net is good, Scrimba works well."* Some students also commented on the need for an external voice chat for efficient communication, and would have liked to see this integrated in Scrimba.

Some students also reported that Scrimba did not necessarily aid in the learning of programming, but rather may have helped in terms of collaborative efforts, *"[...] I don't think Scrimba alone supported my JavaScript learning, but it made the collaboration a little more effective."* This is to a certain degree reflected in the

questionnaire results (A_5), showing that 26% were neutral on the assertion that Scrimba supported the learning of JavaScript, while 59,3% (strongly) agreed.

It is interesting to note how the students differ in reporting the collaboration process, focusing on either giving or receiving feedback. One reason for this may be the level of competence the student has in programming and hence, their ability to help others. This may indicate that some may have mostly helped others and other mostly received help. This for sure would vary according to the group compositions, as others reported learning through both giving and receiving feedback,

> *"[...] liked to get someone else's views on my code. The fact that I had to explain what I did here and there made me better understand my code setup. Also reading another's code was good, [...] giving feedback was also educational. I thought the use of scrimba worked well, [...]"*

These findings are noteworthy when seen in context of previous research. Our results indicate that 83% of the participants made use of the feedback from fellow students, while Aalberg and Lorås [1]'s study indicate trust issues between pairs in pair programming, respectively regarding the feedback provided by their peers. We did not find indications of mistrust between peers or the feedback provided in our experiment, instead both those who predominantly either providing or received feedback found the assessment helpful, and the great majority of students making use of feedback received.

Working process

A quantity of students reported how the tool made it easy to share the code and also show the code to their group underway in the process. With the tool they no longer had to zip folders and send it via some software, but rather just share a link. Also the real time aspect of the tool, where they could see how their peers would comment and write code in real time, was an important part in the working process as the ideas could be created and conveyed there and then.

> *"The 'Fork' tool made it incredibly easy to exchange each other's code so we could compare and try to extend each other's code."*

Some students found the tool in itself to be a bit hard to work in and that it lacked some functionality that was in other tools they had previously used. In addition some of the students experienced some technical problems due to the Wi-Fi connection, in some cases not being able to share their links and in the worst cases losing work they had done. This is as reported in the research of Ying [20] where a common issue with digital collaboration tools is that they require a stable connection. If there is not a stable connection, issues such as code loss and errors in code merging may happen in many tools.

We also noticed how a number of students in their open answers first and foremost expressed how the work had been about finding error and lacks in their own and others' codes. They expressed they would get help to find errors in their own code, and where more code-centric in this manner, as opposed to other students focused more on the process of collaboration and for example sharing ideas and improvements. A student reported that *"I noticed how having many eyes on my code helped me find solutions faster, and the process of finding mistakes went easier."*

A question arising for the course lecturer is what kind of focus she wants for the student to have to learn well.

Regarding technical difficulties we saw that some of the respondents referred to others tools being better. The students had used another tool for programming, Brackets, for seven lectures before they used Scrimba. It may be the case that these students either found the transition from one tool to another challenging, however, we believe some of the students making comments on the tool itself also may have had experience from before beginning their Bachelor and gotten experiences and preferences regarding tools. Some students explicitly mentioned having programming experience beyond the current CS1 course and reported that Scrimba did not necessarily facilitate learning any better than alternative non-collaborative tools:

> *"I have programmed earlier in other languages and do not think Scrimba helps anything on my learning of Javascript, I prefer Visual Studio Code when I code otherwise. Probably works fine for beginners, and went well for us, but we didn't learn any more."*

Revisiting the research questions

This section revisits our two research questions. The implications are of interest for teachers that would want to make use of online tools for collaborative programming.

RQ$_1$: How do students perceive working in groups on a programming task through an online collaborative tool?

Having a tool that lets you easily exchange code and expand on it in real time opens for collaborative processes not else possible. Scrimbas functionality made possible dynamic, interactive and effective collaborative learning process, where the students in the groups both could give and receive feedback. The entire group could follow along on their screens in real time while their peers showed, discussed and changed code on the fly. The result was an effective exchange of ideas and ultimately the students perceived they learned and that their solutions became better. Some of the students suggested an integration of voice chat into the functionality when they work apart from each other which we find a good idea since the collaboration requires discussion.

RQ$_2$: How does the collaborative effort affect the students' perceived learning?

Both our qualitative and quantitative findings show that the majority of students perceived the collaborative efforts as educative and helpful through giving and receiving feedback, sharing ideas, discussing and improving their codes. 83% made use of the feedback they were given to improve their solutions. Some students focused on error finding, which makes us ask ourselves what level of learning they obtained. Learning lies in both giving and receiving feedback in a process where the student codes, reflects on their own and their peers code, and finally articulates their thoughts textually and orally. As students inherit different levels of competency, adaptive assignments could be beneficial to interest also more advanced students.

5 Conclusion

We surveyed 177 first-semester IT undergraduate students on their perceptions and thoughts on cooperative learning through peer code review using an online collaboration tool. Our findings indicate that the vast majority of the students first of all found the online tool helping the collaboration process. In addition they found the collaboration itself educational, both in terms of providing and receiving feedback on their code assignment resulting in sharing of ideas and learning with and from each other.

As for limitations, we did not record any personal information about the participants, for instance age, gender, perceived skill level and past experience with programming. Neither did we investigate potential impact of peer assessment on the participants' final grade in the CS1 course. Another possible limitation is how a number of students experienced issues related to Internet connectivity.

Suggestions for Future Research

We suggest exploring possibilities and shortcomings in already existing online collaborative programming tools. A comprehensive survey and evaluation of such tools in educational setting could be of great importance for educators and students going forward. Secondly, explore what new functionality tailored to collaborative situations in education could be added. This also implies that there may be a need for development projects where both researchers and developers are involved.

References

[1] Trond Aalberg and Madeleine Lorås. Active learning and student peer assessment in a web development course. In *Proc. 2018 Norwegian Conference for Education and Didactics in IT subjects (UDIT)*. BIBSYS, August 2018.

[2] Leland Beck and Alexander Chizhik. Cooperative learning instructional methods for cs1: Design, implementation, and evaluation. *Trans. Comput. Educ.*, 13(3), August 2013. URL `https://doi.org/10.1145/2492686`.

[3] Jens Bennedsen and Michael E Caspersen. Failure rates in introductory programming: 12 years later. *ACM Inroads*, 10(2):30–36, April 2019.

[4] Sheelagh Carpendale, Søren Knudsen, Alice Thudt, and Uta Hinrichs. Analyzing qualitative data. In *Proc. 2017 ACM ISS*, ISS 17, pages 477–481, New York, NY, USA, 2017. ACM. URL `https://doi.org/10.1145/3132272.3135087`.

[5] Mehmet Celepkolu and Kristy Elizabeth Boyer. Thematic analysis of students' reflections on pair programming in cs1. pages 771–776, 2018.

[6] Catherine Fosnot. *CONSTRUCTIVISM: THEORY, PERSPECTIVES, AND PRACTICE*. 2005.

[7] Graham Gibbs. *Analyzing Qualitative Data*. 2012.

[8] Rolando Gonzalez and Per Lauvaas. An experience report using scrimba: An interactive and cooperative web development tool in a blended learning setting. In *Proc. 2017 Norwegian Informatics Conference*. BIBSYS, 2017.

[9] Per Lauvaas and Rolando Gonzalez. Teaching introductory web development using scrimba: An interactive and cooperative development tool. In *ECEL 2018*. ACI Academic Conferences International, 2018.

[10] Per Lauvaas and Kjetil Raaen. Passion, cooperation and JavaScript: This is what the industry is looking for in a recently graduated computer programmer. In *Proc. 2017 Norwegian Informatics Conference*. BIBSYS, 2017.

[11] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and et al. Introductory programming: A systematic literature review. In *Proc. Comp. 23rd Annual ACM Conf. ITiCSE*, ITiCSE 2018 Companion, pages 55–106, New York, NY, USA, 2018. Association for Computing Machinery. URL `https://doi.org/10.1145/3293881.3295779`.

[12] Rodrigo Pessoa Medeiros, Geber Lisboa Ramalho, and Taciana Pontual Falcão. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Trans. Education*, pages 1–14, 2019.

[13] National Academies of Sciences, Engineering, and Medicine. Assessing and responding to the growth of computer science undergraduate enrollments. , washington, DC. Technical report, The National Academies Press, 2018.

[14] Norsaremah Salleh, Emilia Mendes, and John Grundy. Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *IEEE Trans. Software Eng.*, 37(4):509–525, July 2011.

[15] Rustam Shadiev, Wu-Yuin Hwang, Shih-Ching Yeh, Stephen J. H. Yang, Jing-Liang Wang, Lin Han, and Guo-Liang Hsu. Effects of unidirectional vs. reciprocal teaching strategies on web-based computer programming learning. *J. Educational Computing Research, Vol. 50(1)*, pages 67–95, 2014.

[16] Timothy Urness. Assessment using peer evaluations, random pair assignment, and collaborative programming in CS1. *J. Computing Sciences in Colleges*, 25 (1):87–93, 2009.

[17] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. A systematic review of approaches for teaching introductory programming and their influence on success. In *Proc. Tenth Annual Conf. ICER*, ICER 14, pages 19–26, New York, NY, USA, 2014. ACM. URL `https://doi.org/10.1145/2632320.2632349`.

[18] Jeannette M Wing. Computational thinking. *COMMUNICATIONS OF THE ACM*, pages 33–35, 2006.

[19] Bian Wu, Yiling Hu, A.R. Ruis, and Minhong Wang. Analysing computational thinking in collaborative programming: A quantitative ethnography approach. *Wiley J. of Assisted Learning*, pages 421–434, 2019.

[20] Kimberly Michelle Ying and Kristy Elizabeth Boyer. Understanding students needs for better collaborative coding tools. april 2020. URL `https://dl.acm.org/doi/pdf/10.1145/3334480.3383068`.