CHALMERS
UNIVERSITY OF TECHNOLOGY

Building Collaborative Learning: Exploring Social Annotation in Introductory Programming

Downloaded from: https://research.chalmers.se, 2025-07-03 05:50 UTC

Citation for the original published paper (version of record):
Gomes, F., Dobslaw, F. (2024). Building Collaborative Learning: Exploring Social Annotation in Introductory Programming. Proceedings - International Conference on Software Engineering: 12-21. http://dx.doi.org/10.1145/3639474.3640063

N.B. When citing this work, cite the original published paper.

© 2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, or reuse of any copyrighted component of this work in other works.

(article starts on next page)

Building Collaborative Learning: Exploring Social Annotation in Introductory Programming

Francisco Gomes de Oliveira Neto
Chalmers and the University of Gothenburg
Dept. of Computer Science and Engineering
Gothenburg, Sweden
francisco.gomes@cse.gu.se

Felix Dobslaw
Mid Sweden University
Dept. of Quality Mngmt, Communication and Inf. Systems
Östersund, Sweden
felix.dobslaw@miun.se

ABSTRACT

The increasing demand for software engineering education presents learning challenges in courses due to the diverse range of topics that require practical applications, such as programming or software design, all of which are supported by group work and interaction. Social Annotation (SA) is an approach to teaching that can enhance collaborative learning among students. In SA, both students and teachers utilize platforms like Feedback Fruits, Perusall, and Diigo to collaboratively annotate and discuss course materials. This approach encourages students to share their thoughts and answers with their peers, fostering a more interactive learning environment. We share our experience of implementing social annotation via Perusall as a preparatory tool for lectures in an introductory programming course aimed at undergraduate students in Software Engineering. We report the impact of Perusall on the examination results of 112 students. Our results show that 81% of students engaged in meaningful social annotation successfully passed the course. Notably, the proportion of students passing the exam tends to rise as they complete more Perusall assignments. In contrast, only 56% of students who did not participate in Perusall discussions managed to pass the exam. We did not enforce mandatory Perusall participation in the course. Yet, the feedback from our course evaluation questionnaire reveals that most students ranked Perusall among their favorite components of the course and that their interest in the subject has increased.

CCS CONCEPTS

• **Applied computing** → **Collaborative learning**; **Computer-assisted instruction**; **Interactive learning environments**; • **Social and professional topics** → **Computing education**.

KEYWORDS

Social Annotation, Educational Technology, Computing Education

ACM Reference Format:
Francisco Gomes de Oliveira Neto and Felix Dobslaw. 2024. Building Collaborative Learning: Exploring Social Annotation in Introductory Programming. In *46th International Conference on Software Engineering: : Software Engineering Education and Training (ICSE-SEET '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3639474.3640063

This work licensed under Creative Commons Attribution International 4.0 License.

ICSE-SEET '24, April 14–20, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0498-7/24/04.
https://doi.org/10.1145/3639474.3640063

1 INTRODUCTION

Programming is one of the first topics taught in many engineering disciplines. Large classes of students typically have their first contact with programming when teachers explain basic programming constructs as statements and are shown examples of output produced by executing code [21, 29]. Most of that knowledge is not introduced to students before higher-level education. Moreover, most students are unfamiliar with the technological content knowledge associated with teaching and learning programming (e.g., development environments, installation of compilers or interpreters). Exercising those skills already in the first class can easily overwhelm students, particularly those without any prior knowledge of programming [29]. Besides the content itself, students must learn how to explain their algorithms to each other, i.e., explain to peers the steps that they followed to solve a specific problem [25, 28]. This is particularly challenging when students need to collaborate towards a solution in, e.g., a project course.

Teaching approaches focused on flipping the classroom or performing active learning have shown effective results in improving the exam results of students [3, 5]. Particularly, preparing for lectures by reading material or watching videos has been one of the main tools used to allow teachers and students to focus their time together on solving problems and discussing different solutions to the problems. However, those studies did not investigate the collaborative dimension of students working and learning together.

Social Annotation (SA) is a pedagogical approach that fosters collaborative learning among students, enabling them to jointly engage with course materials, discuss concepts, solve problems, and compare their annotations with peers [1, 18]. Recent research highlights the effectiveness of collaborative learning in computer science and programming education, with students showing increased engagement and improved learning outcomes [12, 24]. SA has garnered overwhelmingly positive student responses, underscoring its potential for enhancing the educational experience [4].

Social Annotation requires an online platform facilitating communication and knowledge sharing among students as they interact with course resources, such as textbooks, exercises, and video lectures. Feedback Fruits, Diigo, and Perusall are a few examples of such tools. For instance, in Perusall [1], instructors share course materials, allowing students to asynchronously and collaboratively generate annotations by highlighting specific sections within the

[1] https://www.perusall.com/

material — whether those annotations target timestamps in videos, sentences or paragraphs in text, or sections of a web page. These annotations serve as a medium for students to write their understanding, identify challenging concepts, and seek clarification through questions or comments.

This interactive annotation process encourages students to articulate their comprehension of concepts and pinpoint difficulties. These annotations trigger discussions among students, as other students provide their own explanations, hence fostering collaborative learning dynamics within the environment. Teachers can use such discussions to discover why certain ideas remain unclear to students and cover them in discussions with the entire class.

Our goal is to investigate the impact of social annotations in Perusall when teaching programming to first-year students in the Software Engineering and Management bachelor program at the University of Gothenburg (Sweden). Particularly, we aim to verify whether students sharing their understanding of the course material with their peers affects their performance in the course exam. We compare the results of 112 students who used Perusall to prepare for each lecture by completing reading assignments. Particularly, our report targets the following research questions:

> **RQ1: Do students of a programming course engage in non-compulsory social annotation activities?**
> *Yes.* Most students (on average 78% of 112 students) engaged in social annotations throughout all 18 course lectures. However, roughly 20% of those students created meaningful comments that showed their understanding of the topic.
>
> **RQ2: Does social annotation engagement have an impact on the students' grades and passing rates?**
> *Yes.* Students who engaged in social annotation by creating more meaningful comments had, proportionally, better grades and passing rates than those who were less engaged in social annotations.

Our results reveal that students who create meaningful comments in Perusall and engage in discussion with their peers have better grades in the course. Failure rates decrease for those students who use Perusall more in the course. Moreover, the course feedback questionnaire reveals a positive response from students using Perusall in the course, which aligns with existing findings in literature [12, 23]. On the other hand, the course feedback indicates that many of the students were not motivated to engage in Perusall discussions.

This paper is structured as follows. Section 2 presents related work regarding teaching programming, social annotation, and previous studies with Perusall. Section 3 provides context to our investigation by sharing course details, such as student population and course structure.[2] We detail the data collection and analysis in Section 4. We present results and findings from our research questions in Section 5, followed by a discussion involving student feedback, lessons learned and the limitations of our report (Section 6). Lastly, we conclude and outline future work in Section 7.

[2]Some details were omitted due to the double-blind review process. Some of the course artefacts (feedback form, example of the exam, course material can be shared after the reviewing process.)

2 RELATED WORK

Learning programming goes beyond the skill of writing code according to a syntax (theory), it also requires students to trace the execution by predicting outputs and state changes [16, 21], as well as explaining the code to other programmers [28]. Those skills are connected but distinct from one another, such that instruction models aim to refine them. For instance, the Theory of Instruction model proposed by Xie et al. [29] highlights four programming skills based on reading and writing code using knowledge at a machine level (semantics) or at task level (templates). Those four skills are progressively obtained by the student by reading semantics, writing semantics, reading templates, and writing templates.

Our goal is not to propose or evaluate such models, rather we investigate the prospects of social annotations that can later be used in combination with such models to bring forward the aspect of metacognitive thinking. In Perusall, students are encouraged to share their cognitive processes of understanding when posing questions or providing answers within shared course materials [20], and prompting students to engage in metacognitive thinking can enhance their abilities in reading and writing code [14, 15].

Perusall, or social annotation, has not been widely investigated in the context of teaching and learning programming yet. Meyer and Müller found significant challenges in implementing Perusall in a Data Structures and Algorithms course [19]. The primary difficulty was in maintaining student motivation to annotate materials and participate in online discussions. Similarly, we observed that many of our students failed to produce annotations that effectively demonstrated their understanding of the subject matter. These experiences underscore the necessity of promoting a shift towards a culture of continuous learning

Other subject areas have shown promising results in using SA. In a comparative study, Suhre et al. identified a positive correlation between active participation and examination results with Perusall in eight different courses in social sciences [27]. They further found that engagement can be fostered through the tool if certain criteria are respected including stimulating texts and assignment formulations, appropriate group sizes, the a-priori providing of good annotation examples, as well as timely feedback from instructors. Those findings align with other studies focused on social annotation with other platforms, where researchers see improvements in learning engagement [8], attention [7], peer communication, and sense of community [10].

The papers above provide insights into diverse methods for evaluating student performance and learning. In this experience report, we establish correlations between Perusall activity and students' exam performance to discern distinctions among groups of students actively engaged in social annotation. Although exam scores offer a limited perspective on student performance [6], they have been employed in prior research that investigates Perusall and social annotation, enabling the exploration of student engagement and learning [7, 13, 20]. This approach allows us to draw parallels with our findings. In future studies, we intend to explore additional dimensions of student performance, including their sense of belonging and learning progression throughout the course.

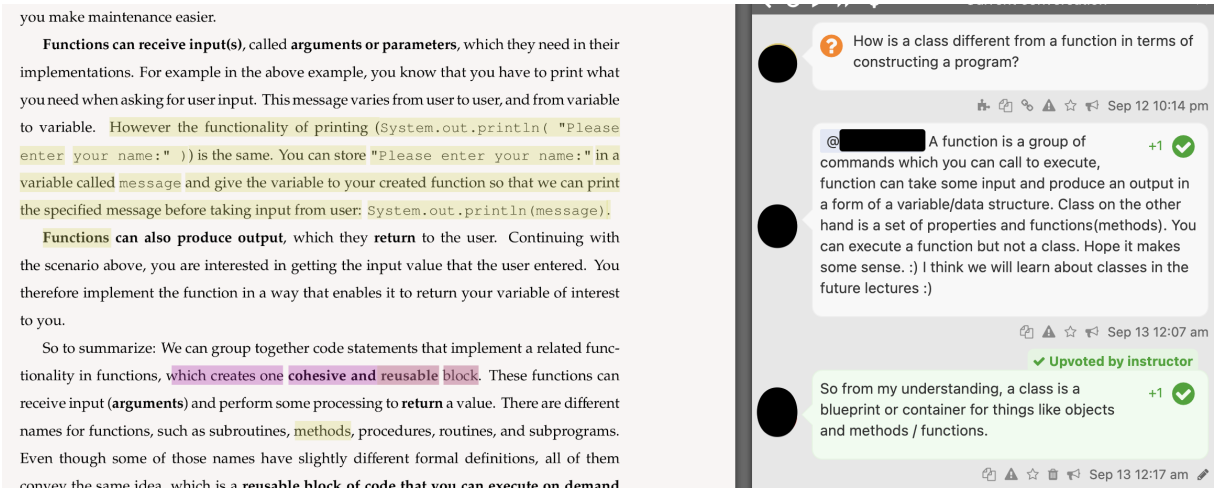

Figure 1: Example of student interaction in Perusall. Three students comment on the highlighted (pink) annotation in the course material about the lecture on Functions in Java. Students help each other understand the difference between reusable code for simple tasks (functions) and structural abstractions (classes).

3 CASE COURSE: CONTEXT AND SCOPE

We investigate the impact of Perusall in a course taught in the first study period of an international bachelor program in Software Engineering and Management at the University of Gothenburg (Sweden). The course is on Object-oriented Programming (OOP) and covers the following learning outcomes: (i) basics in procedural programming (e.g., printing, conditionals, loops, arrays and functions), and (ii) core concepts of OOP (e.g., classes, objects, encapsulation, polymorphism). The programming language taught in the course is Java.

Course structure: The course instance took place during 10 weeks in 2022 and had 143 registered students. Students were expected to dedicate 20 hours per week to the course, which would include time in lectures, laboratory sessions (focused on practical exercises), and self-studies at home. Students were offered three 2-hour lecture sessions, and three 2-hour lab sessions a week — all of which *non*-compulsory. For course completion, the students must submit: (i) three programming assignments done in groups of up to three; and (ii) a final individual written hall exam where students score between 0–100 points. The course was taught on campus by one course responsible, and eleven teaching assistants.

Student background: No entry requirements in programming or computer science applied, as this was the students' first programming course in the Program. Nonetheless, students may or may not have had previous programming knowledge (e.g., during their high-school education), leading to a heterogeneous sample of student backgrounds. Students took one other course in parallel in discrete mathematics with the same expected workload.

Perusall and social annotation: For each lecture, students were instructed to prepare by reading the material in the Perusall platform in the form of *reading assignments*. Students complete these reading assignments by creating meaningful annotations, reading the material until the end, and engaging with the material (e.g., scrolling, highlighting text, etc.). The reading assignments are the main component of Perusall that fosters collaborative interaction between students before the lecture. Therefore, the completion of reading assignments will be our main metric to measure the social annotation element in the course.

To motivate students to complete reading assignments, the course instructor offered bonus points to students. A maximum of eight bonus points towards the exam were given to students creating *meaningful annotations* or engaging in discussion in the Perusall material for the eighteen lectures.[3] For each lecture, a 0.5 bonus point could be obtained. Thus, the maximum amount of points was achieved completing any 16 of the 18 Perusall tasks.

We used Perusall's definition of meaningful annotations and shared examples with the students at course start.[4] In short, meaningful annotations are comments or questions that showcase the student's comprehension of the concepts discussed in the material. We exemplify a meaningful student exchange in Perusall through Figure 1 where three students discuss what a Function is. All three students received the bonus for that lecture as they made multiple similar comments throughout that lecture's material.

To cope with the large number of students, we used Perusall's algorithm that automatically assess the quality of annotations of students based on the content quality, the number of students' replies, length of text, among other features extracted from the annotation [11]. The course instructor chose the holistic scoring strategy defined in Perusall which considers the annotations created, and whether the student has read the entire material before the lecture.[5] The course instructor also determined that students needed to create at least two meaningful annotations to complete

[3]Students were also informed that bonus points could not be used to cross the passing threshold of 50 points, i.e., a student could not pass the course through bonus points.
[4]https://support.perusall.com/hc/en-us/articles/360034824694-How-is-annotation-quality-defined-in-Perusall-
[5]https://www.perusall.com/hubfs/downloads/scoring-details.pdf

the reading assignment to foster discussion threads and communication between groups of students. The practical exercises and student-teacher interactions happened mainly during lectures. In case a student disagrees with Perusall's automatic score, the course instructor can revise and override Perusall's decision. Fine-tuning Perusall's accuracy is beyond the scope of our study, therefore, we acknowledge and discuss some limitations associated with its automated grading system in our threats to validity.

Lecture format: All lectures were hosted on campus. An average of 80 students showed up to class (55% attendance rate). Each two-hour lecture was divided into two parts with a 10-min break in between. Part one was a Mentimeter[6] session with multiple-choice questions regarding the Perusall material. We chose Mentimeter for its simplicity and to allow for anonymity. Students were told that the quiz participation had no influence on the grade and was optional. For each question, the answer statistics were presented live to the students, and the teacher initiated a discussion about the student's reasoning, particularly when the answers were not converging to the correct option. The second part of the lecture focused on applying the lecture topics with the help of one or two coding exercises solved together with the class.

Written hall exam: Students did a four-hour written exam with various questions focusing on tracing code, writing small classes or functions, and explaining the application and trade-offs of topics covered in the course. Students received between 0–100 points based on the quality of their answers. We applied the four-level grading scale below. The exams were anonymised by the examination office and graded by the course responsible.

- **Fail (U):** Assigned to students that scored less than 50 points in the exam.
- **Pass (3):** Given to students that scored between 50 and 69 points.
- **Pass with merit (4):** Given to students that scored between 70 and 84 points.
- **Pass with distinction (5):** This is the highest grade in the scale and is given to students that received points greater than or equal to 85.

Course evaluation: At the course's outset, five students volunteered to become student representatives who are the contact point for all students when the student collective wants to offer feedback regarding the teaching and learning throughout the course. Nonetheless, all students have direct channels to communicate with the course instructor. In the last week of lectures, *all* students receive a questionnaire following the SEEQ feedback template [17]. The questionnaire is closed before the written exam to reduce the risk of bias introduced by the examination experience. The course responsible and student representatives meet on two occasions: the first time halfway into the course to reflect on the course status and the teaching methods for possible intervention; the second meeting was a retrospective with the presence of the program manager and study administrators where the SEEQ questionnaire results were discussed. The information collected from those instruments helped the instructor to understand: (i) some of the main obstacles in the usage of Perusall, (ii) the frequency and level of satisfaction from students engaged in peer instruction, (iii) students' reactions to the teaching methods, and (iv) the self-reported impact on their learning.

[6]https://www.mentimeter.com/

Prior Knowledge in Programming: One of the main challenges in teaching first-year programming courses is the variance among students regarding their prior knowledge of programming. On one hand, having prior knowledge can lower students' motivation to engage in discussion about topics they are already familiar with. On the other hand, those students can also share their experiences with novice students to help them learn. Typically, programming is not taught in primary or secondary school, even though reality might change, given the benefits of introducing students earlier to programming [26, 29]. The instructor estimated the prior knowledge of students by sending them an anonymous questionnaire with various programming-related questions before the first lecture (each question had an option "*I do not know/I cannot answer yet*". From the sample of 115 respondents, 34% of students could not answer what a String is, and 67% of students did not know what an if-statement is. Both topics are basic programming constructs taught in the first week of the course. Therefore, we argue that prior programming knowledge is not a prevalent factor influencing our analysis in this instance of the course.

4 RESEARCH METHODOLOGY

To prevent an unfair teaching environment and the risk of favoring or disadvantaging a particular group of students, we chose *not* to conduct a controlled experiment. Instead, we gave students the choice by making social annotation an optional part of the course. Therefore, to answer our research questions, we used the individual results of the Perusall reading assignments together with the student's exam results. The feedback from the course evaluation questionnaire is used to discuss the qualitative aspects of the student's feedback about using social annotation. We refer to Perusall activity as the *outcome* of the reading assignments made by each student. Each lecture had a corresponding reading assignment. There were three outcomes for each reading assignment:

- **Skipped:** The student did not create a single annotation for that reading assignment, or they did not even read the material in Perusall.
- **Incomplete:** The student created at least one annotation in the material, but the content was not assessed as meaningful by Perusall's algorithm, i.e., the annotation did not convey the student's understanding of the subject covered.
- **Completed:** The student made at least two comments on annotations that were classified as meaningful according to Perusall's algorithms. These comments can be questions they asked, answers provided to other students or comments in discussion threads.

We compared those different types of activities in relation to the students' exam results (both points and grade). We analysed the exam results *without* adding the bonus points from Perusall since this would otherwise introduce a bias towards passing students. We analysed the results of 112 students considering the intersection between those registering for Perusall during the course instance

Table 1: List of topics covered in the course.

ID	Topic of the lecture:
L01	Variables, Types and Expressions
L02	User Input
L03	Conditionals
L04	Loops
L05	Arrays
L06	Basics in OOP
L07	Reference variables
L08	Encapsulation and immutable objects
L09	Collections - Lists, Sets and Maps
L11	Composition and Aggregation
L12	Inheritance
L13	Polymorphism
L14	Abstract Classes
L15	Exceptions and Error Handling
L16	Interfaces in Java
L17	OOP Design Principles: SOLID
L18	Files

and taking the written exam.[7] We anonymised the data set by removing all identity information from the records used throughout our analysis.

For simplicity, the plots discussed in our results include IDs for each lecture. Table 1 maps each lecture ID to the corresponding subject covered by the reading material. For the remainder of the paper, we use the term assignments to refer to the *reading assignments* in Perusall. Throughout our discussions, we consider that a student who completed a reading assignment conveyed their understanding of the topic to other students, which is one of the main goals of social annotation.

4.1 Scientific Ethics and Data Availability

The University of Gothenburg is a State University under Sweden's principle of publicity (in Swedish, offentlighetsprincipen) which ensures transparency with the population.[8] Therefore, the public can request all exams (digital or printed) via a transparency office at the University. Students are made aware of such principles when admitted to the University. Nonetheless, we anonymise the data shared in this paper. To comply with scientific ethical guidelines, we also asked students and teachers for consent to use their course data (e.g., annotations, Perusall login data, exam results) during the first week of the course. We clarified that students could opt out of the study at any moment.

We share the files and scripts relevant to this experience report in our analysis package in Zenodo [2].[9] The CSV files include the exam points, grades and the classification of Perusall's annotation per student and lecture. We also share the report of the course evaluation.

[7] Students who did not complete previous course instances also take the exam. Similarly, a subset of students participated in the course but decided to skip the written exam.
[8] https://medarbetarportalen.gu.se/service-support/for-arbetsgivare/8.universitetet-ar-en-statlig-myndighet/, available in Swedish.
[9] https://doi.org/10.5281/zenodo.10483184

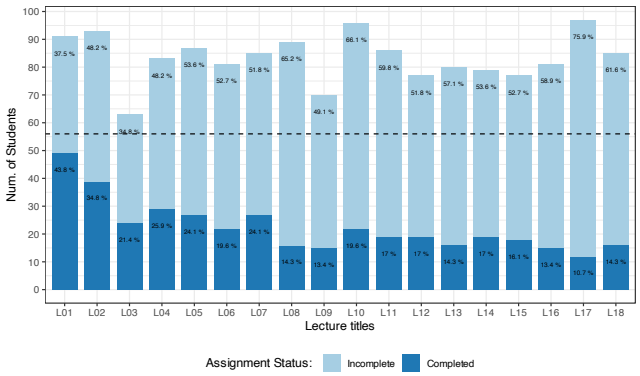

Figure 2: The overall distribution of social annotations with non-compulsory peer instruction. The dashed line intercepts the y-axis at half of the number of students (n = 56 students).

5 RESULTS

Table 2 presents the results of the exam without any association to the Perusall activity. This allowed us to understand how the class performed. Based on the exam grade distribution, 45% of the students failed the written exam. The one course parallel to this one showed a smaller yet similar failing rate (roughly 30%). Below, we analyse our research questions by relating those percentages to the Perusall activity of students.

Table 2: Distribution of exam grades. Students who failed the exam received a grade of U. Passing students received grades 3, 4 or 5 (highest grade).

Grades:	U	3	4	5
Number of students:	51	37	17	7
Percentage of students:	45.5%	33.0%	15.2%	6.2%

5.1 RQ1: Level of Engagement in Social Annotation

Figure 2 contains an overview of students' annotation patterns per reading assignment. More than 50% of students participated in social annotation (Incomplete + Completed) with variations depending on the topic. An average of 78% attempted or completed the interactions with the material. The lecture on loops (L03) had the least social annotation (44% of the students skipped it), whereas L17 (OOP Design) had the most engagement (only 13% skipped it). While the majority of the engaged students did not complete the reading assignments, students used the material throughout the course with no consistent signs of decline.

On the other hand, there was a decline in the proportion of students completing the reading assignments, hence indicating that fewer were making meaningful annotations/comments in Perusall. The drop was higher after L03 (loops) from 44% to 21% which is then sustained in different topics. The completion rate stayed consistently below 20% after L08 (Encapsulation) and the other core

OOP concepts. Note that the drop in completion rate did not affect the drop in reading assignments engagement as the proportion of incomplete assignments varies roughly between 40–60% throughout all lectures. Below, we illustrate two contrasting annotations from different students about overriding methods in Lecture 12 (Inheritance).

> "not very clear what does it mean functionality?". (Classified as Incomplete.)

> "Why is [overriding] risky? I get that it could become a problem if a subclass needs to override a method, but doesn't. Is there any risk if everything works even if the subclass does not override the method and the superclass method is executed instead?". (Classified as Completed.)

The first comment was classified as incomplete because the student is not clear whether they mean the functionality of a shown piece of code or *how* Inheritance and overridden methods work. In contrast, the student that completed the assignment conveys their current understanding of overriding risks (*"I get that it could..."*) and their struggle to realise different risky scenarios (*"... even if the subclass does not override..."*).

The lower percentage of completed assignments can be attributed to a variety of reasons. For instance, the learning curve to create meaningful annotations, lack of time to dedicate to social annotations, or a lack of accuracy in Perusall's automated algorithm to detect meaningful comments related to programming. Determining an accurate decline in the quality of annotations requires a more extensive and manual qualitative analysis of the text written by students which is outside the scope of our experience report. Therefore, we summarise our RQ1 findings below.

> **RQ1:** More than 50% of students took part in the non-compulsory social annotation activities in Perusall. However, the percentage of students making meaningful comments or annotations decreased over time.

5.2 RQ2: Impact of Social Annotations on Grades and Passing Rates

We measure social annotation based on the number of completed reading assignments. The reason behind our choice is that Perusall's algorithm mainly grades students based on the quality of their annotations which is one of the affecting factors in social annotation [7, 13]. Therefore, we assumed that students completing assignments have provided more insights to help their peers learn about the subject. Figure 3 shows the correlation between the points obtained in the exam and the number of completed reading assignments. For students with none/little annotations, i.e., few completed assignments, no grade impact could be observed. As we increase the number of completed assignments, note that the number of students who failed started to decrease. Nonetheless, there were still many students who passed the exam without completing many reading assignments.

To verify whether the number of completed assignments affected the exam results, we divided our sample into two groups and

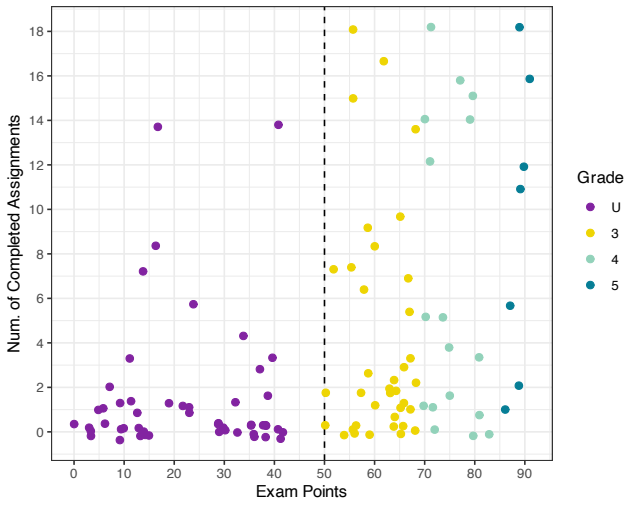

Figure 3: Exam points in correlation to the number of completed Perusall assignments. The dashed vertical line denotes the passing threshold (50 points).

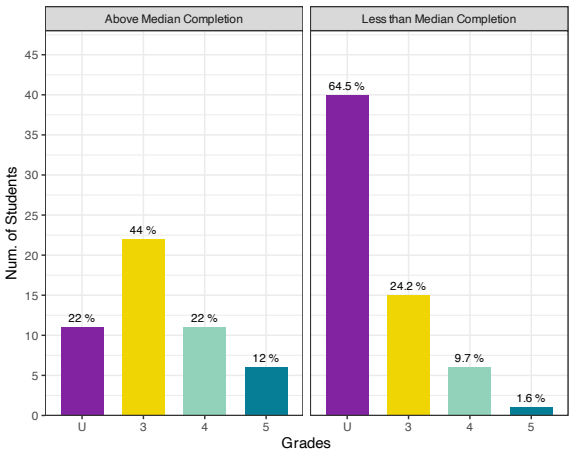

Figure 4: Percentage of students per each grade based on the expected number of completed assignments from passing students (more than 2 assignments). Students who completed above the median have better grades compared to those who do not.

compared the proportion of students for each grade. We chose the median number of completed assignments for all passing students to divide the groups. Our reasons were two-fold: (i) the median conveys the expected number of completed assignments to pass the exam; (ii) the median divides the sample into two student groups of roughly equal size such that a similar proportion of students in all grades indicates that students would pass/fail the exam independently of their Perusall interactions. The median number of reading assignments completed for the passing students was 2, resulting in the two student cohorts in Figure 4.

Of the 62 students who completed less than two reading assignments, 64.5% (40) failed the exam. This proportion was almost three times higher than the proportion of failing students who completed at least two assignments (22%). Moreover, the proportion of students in all passing grades is significantly higher in the group of students that completed the expected number of assignments to pass, particularly for the better grades 4 (6 vs. 11) and 5 (1 vs. 6).

> **RQ2.1:** We observed a distinct grade distribution among students who actively participated in social annotation by completing the required number of assignments for passing. Notably, we identified a positive correlation between engagement and the distribution of exam scores and final grades. As students increased their involvement in social annotations within Perusall, we noted a decrease in the number of exam failures. Furthermore, a significant trend emerged, revealing that the majority of students who completed fewer than two assignments ended up failing the exam, while those who completed at least two assignments exhibited proportionally better performance in their exam grades.

Figure 5 contrasts the proportion of passing and failing students based on their *corresponding* number of completed assignments. We see that the largest proportion of failing students are those who did not complete a single assignment (x = 0), which aligns with our observations above. Moreover, when completing more than 4 assignments the cumulative number of students passing the exam (25) is much higher than the number of students failing (5). Focusing on the middle range of completed assignments (4–9), few students fail and many more pass in that range. For the students highly engaged in social annotation (above 9 lectures) the correlation is even more apparent as only 2 (out of 19) students failed the exam.

> **RQ2.2:** The majority of students who failed did not complete any reading assignments. After completing more than 4 assignments, the proportion of students passing the exam (22.5%) is much higher than those that failed (4.5%).

6 DISCUSSIONS AND LESSONS LEARNED

Here, we complement our quantitative analysis above with a qualitative analysis of the course feedback provided by students. The course evaluation questionnaire reveals some qualitative aspects of the usage and the response of students to using Perusall as a tool. We also cover connections between Perusall and the student's learning and summarise our findings. We end the section with a summary of lessons learned and the limitations of our observations.

Course evaluations at our University are anonymous and use the Student Evaluation of Educational Quality (SEEQ) template [17]. We extended the questionnaire with a few questions focusing on the social annotation aspect of the course. A subset of questions relevant to our discussion and their corresponding answers is presented in Table 3. The response rate was 17% (25 out of 142 students), which is low. One of the reasons for the low response rate is that the questionnaire was only available for students in the last 2-weeks of the course. Moreover, students reported that they received few reminders to complete the course evaluation.

Most students (64%) would often or always read the material available in Perusall, but only three students stated that they create annotations in Perusall at the same frequency. Below, we share the statement from a student reporting that the need to annotate the material added a distraction to their studies, despite seeing their benefits when reading discussions from other students. Perusall has options to hide comments and annotations, but students did not receive a walk-through or demonstration of Perusall's features in course start.

> "Personally, I enjoyed being able to ask questions directly in Perusall and receiving answers. However, my personal learning style implies highlighting key concepts I find important, and sometimes in Perusall there were full paragraphs highlighted with a question, and it distracted me from the material". (Student)

After L12 (Inheritance), student representatives in the course asked the teacher to create anonymous annotations, which is an option for Perusall. The anonymity allows teachers to see the identity of students creating or replying to comments, but students do not see each other. We see a slight increase in activity from students after that, but this is still lower than some assignments before the anonymity was enabled.

The course evaluation questionnaire also includes two questions about the different teaching practices used in the course: (Q9) *"What are the three things you liked the **least** about the course?"*, (Q10) *"What are the three things you liked the **most** about the course?"*. Only one student listed Perusall and social annotation as one of the things they liked the least in the course. Particularly, the student was unsatisfied with the amount of time spent during the lecture quizzes (which typically cover the content from Perusall). Also, this student is more interested in a more traditional format of lectures where explanations are delivered predominantly by the lecturer rather than by their colleagues. Related work also reports that students struggle to adopt social annotations and move towards continuous learning [19].

> "The discussions about the questions on Perusall. I think they were unnecessarily long and took away precious time from the lectures. I personally was interested in listening to the explanations from my lecturer, not from my peer that may be as lost as me.". (Student)

In contrast, a dominating number of students responded positively to the usage of social annotation and practical exercises during the lectures. From 18 text responses: 8 students (40%) explicitly mentioned Perusall as one of the things they liked the most in the course, and 11 students (61%) mentioned that the lecture quizzes and discussions helped them understand programming better. Particularly, students emphasised the scope, size and quality of the reading material created by the instructor for this course.

> "The provided material on Perusall was on a nice level, and it was never unclear what to read before each lecture or where we were in the course. And lastly, the quizzes were a good

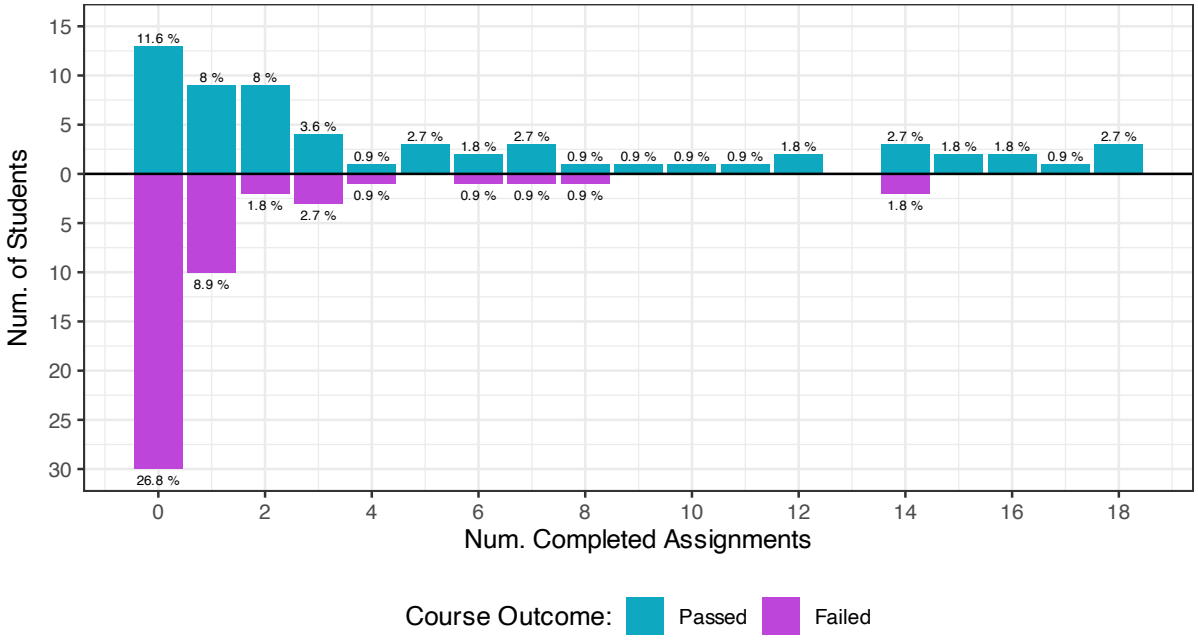

Figure 5: Course passing statistics per lecture. Completing many of the assignments (x-axis) has a large impact on passing the course, while little activity results in a high risk of failing. The "negative" y-axis is used simply to emphasise the difference between students who passed and those who failed.

Table 3: The responses for a subset of questions from the course evaluation questionnaire. 25 students answered the questionnaire using a Likert scale with 5 levels detailed below. For each question, the median answer is highlighted in bold and blue.

ID	Question Description	1	2	3	4	5
	1:Never. 2:Rarely. 3:Sometimes. 4:Often. 5.Always					
Q1	How often did you read the material before the lecture (Perusall or offline)?	1	2	6	**6**	10
Q2	How often did you create or respond to annotations in Perusall?	8	**9**	5	1	2
Q3	How often did you attend the lectures?	1	0	0	2	**22**
	1:Strongly disagree. 2:Disagree. 3:Neutral. 4:Agree. 5:Strongly Agree					
Q4	Reading the material before the lectures helped me better understand the lessons	1	2	6	**7**	9
Q5	The lecture quizzes made me better understand the concepts being taught.	2	2	4	**10**	6
Q6	Students are encouraged to ask questions and are given meaningful answers.	0	0	1	9	**15**
Q7	I have learned something that I consider valuable.	0	1	1	8	**15**
Q8	My interest in the subject has increased as a consequence of this course.	1	1	2	**9**	12

> measure of what had been understood or needed more practice, and also led to some nice discussions.". (Student)

Most of our analysis focuses on the correlation between Perusall annotations and exam points, such that we cannot use those measures to confidently infer causation between social annotation and students' learning. When analyzing the correlation between exam scores and students' learning, it is important to consider potential confounding factors. For instance, students who engage in social annotation might already be highly motivated and diligent, inherently contributing to their higher exam scores. Additionally, students vary in study habits or access to additional educational resources outside the course, which might influence exam scores independently of the course's teaching methods. Such an analysis would require other instruments, such as a thematic analysis of students' annotations throughout the course, as well as exercises or assessments that can show more of a progression throughout the course. We aim to perform those analyses in future work.

For the scope of this paper, we evaluate the learning based on the self-reported satisfaction from the course evaluation. The results suggest that social annotation correlates with the student's learning satisfaction. Note that 23 students (92%) agree or strongly agree

that they have learned something valuable in the course (Q7) and that, similarly, their interest (84%) in programming increased as a consequence of the course (Q8). Therefore, we summarise our findings and lessons learned in the points below:

- Most students who engaged in social annotation passed the exam and, proportionally, showed higher grades. This is also reported in other areas such as physics [20] or multimedia applications [7].
- Completing more reading assignments in Perusall is correlated with higher passing rates.
- Most students listed that Perusall, the lecture quizzes and class discussions among the three things they liked the most in the course.
- More than 90% of the course evaluation respondents agree that they learned something that they consider valuable, and their interest in programming has increased after the course.
- Social annotation can leverage flipped classroom approaches. Many students read the material before the lecture and were motivated to engage in active learning during classes (e.g., Mentimeter quizzes and discussions).

Based on the experience reported in this paper, we make the following **recommendations** to instructors interested in introducing social annotation and Perusall to their programming courses:

- **Consider material length and scope:** Given that students in the analyzed course had approximately 24 hours between lectures to prepare and annotate the associated reading materials, it is crucial for these materials to be both concise and succinct. In this particular course, the average content for each lecture encompassed approximately 10 pages, comprising text and Java code examples
- **Investigate incentives for social annotation:** More than half of the students consistently engaged with Perusall throughout the course. However, the percentage of completed reading assignments dropped over time. Increasing the number of bonus points, or making social annotation compulsory can encourage engagement.
- **Demonstrate the social annotation platform early:** Students are not familiar with social annotation platforms in education, which can create initial barriers to engagement. Additionally, they may not be used to articulate their cognitive processes while writing their notes. To mitigate this, demonstrating Perusall, along with illustrative examples of both effective and ineffective annotations, can diminish the learning curve associated with social annotation.
- **Explain the social annotation platform in the first weeks:** Students are not familiar with social annotation platforms, which can increase the friction of creating annotations in the first weeks of the course. Moreover, they are not necessarily critical about conveying their cognitive processes while studying. Showing how a tool like Perusall works, as well as some examples of meaningful (and not so meaningful) annotations can reduce the learning curve to social annotation as a practice.
- **Enable anonymous annotations:** Initially, students may hesitate to openly express their uncertainties or questions to their peers. We observed a small increase in the number of annotations following the introduction of anonymous annotation, albeit introduced later in the course. It is worth considering that implementing anonymous annotations from the outset could have fostered greater engagement early on.

6.1 Limitations

There are some limitations in our analysis. One of the main construct validity threats is focusing on one instrument to indicate learning performance in the course. Written exams have limitations in conveying the learning of students due to various factors such as anxiety due to time constraints or cultural biases [6, 22]. On the other hand, exam results or grades provide a consistent way to compare trends across many instances of courses and have been used to evaluate teaching in software engineering education in the literature [5]. Using exam points also allowed us to compare our findings with other results from literature [7, 13, 20]. We mitigate the limitation in using exam scores by focusing our conclusions on the correlations between points, grades and Perusall activity without inferring a direct causation to learning.

Even though the instructor can override Perusall's automated grading, there are also risks with students receiving bonus points without making substantial comments (false positives). We did not make adjustments for those cases to avoid reducing the grade of the student. In our findings, most students did not earn bonus points as their comments were deemed insubstantial by Perusall. Less than five students requested a score review, and of those, just two had their scores modified. Nonetheless, we plan to explore Perusall's accuracy in future research further.

Moreover, course evaluation feedback is also subject to various factors, such as the student population (e.g., student bias), the impact of exam results, and the phrasing of the questions. We mitigate these factors by: (i) using the standardised SEEQ template [17] and collecting course feedback before the written exam is performed. Moreover, course evaluations can be a useful tool to gather information about the student's experiences and can offer insights on how to improve course quality [9].

In our analysis, the heterogeneity of students' background knowledge in programming is a key internal validity threat. To assess this, we conducted an entry questionnaire at the course's outset, revealing that 67% of students were unfamiliar with basic concepts like "conditionals." This suggests a relatively uniform knowledge level among participants. However, our results are not broadly generalizable due to the specific context of our study. Despite this, our large student sample and the range of topics align with those in many university programming courses. Future iterations of this course will incorporate student feedback to refine these teaching activities.

7 CONCLUSIONS AND FUTURE WORK

We report on our experience in introducing optional social annotation in a programming course via reading assignments using Perusall. We analyse whether social annotation has an impact on the student's grades and satisfaction. Our findings suggest that many (in our case, the majority of) students actively engage with the optional material and that a significant correlation to passing grades can be observed. However, only a subset of the annotations

done by students are classified as meaningful. We also observe that many students interact with the material but leave assignments incomplete. Therefore, teachers aiming to use social annotation should account for the additional study time and effort required by students between lectures. Moreover, students might require guidance on how to use social annotation platforms and examples of how to create meaningful annotations.

Moreover, students were positive about the use of Perusall and, particularly, its combination with the quizzes and practical sessions during lectures. Students report that they felt that they learned something valuable in the course and that their interest in programming has increased. We argue that this latter aspect is particularly important, as an increased interest in programming can help students keep their motivation and engagement as they progress through different courses in the program. Although not the main focus of this study, the significant reduction in grading demands through the use of an AI grading tool indicates potential for scalability. With 140 students enrolled in the course and using Perusall, manually assessing each student's contributions would be impractical, even with numerous teaching assistants. We believe that teachers' time is better spent engaging in Perusall discussions, offering comments, and using these interactions to inform the preparation of exercises and practical sessions.

Future studies aim to compare our results to previous and future instances of the courses. Another important question is the relationship between social annotation and the other practical components of the course, such as the programming assignments done in groups or the lab sessions where students interact with teaching assistants. Lastly, we aim to compare our results to other topics in software engineering that require critical thinking and assessment in other complex constructs, such as software architectural design patterns, test specifications, or planning software development sessions. Most of these activities are typically carried out in teams and require consensus among participants, hence there might be an inherent component of collaborative learning in those activities that can be enhanced with tools such as Perusall.

REFERENCES

[1] Catherine H Crouch and Eric Mazur. 2001. Peer instruction: Ten years of experience and results. *American journal of physics* 69, 9 (2001), 970–977.

[2] Francisco Gomes de Oliveira Neto and Felix Dobslaw. 2024. *Analysis Package: Building Collaborative Learning: Exploring Social Annotation in Introductory Programming.* https://doi.org/10.5281/zenodo.10483184

[3] Scott Freeman, Sarah L. Eddy, Miles McDonough, Michelle K. Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. 2014. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences* 111, 23 (2014), 8410–8415. https://doi.org/10.1073/pnas.1319030111 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.1319030111

[4] Fei Gao. 2013. A case study of using a social annotation tool to support collaboratively learning. *The Internet and Higher Education* 17 (2013), 76–83.

[5] Lucas Gren. 2020. A Flipped Classroom Approach to Teaching Empirical Software Engineering. *IEEE Transactions on Education* 63, 3 (2020), 155–163. https://doi.org/10.1109/TE.2019.2960264

[6] Jason A Grissom, Demetra Kalogrides, and Susanna Loeb. 2015. Using student test scores to measure principal performance. *Educational evaluation and policy analysis* 37, 1 (2015), 3–28.

[7] Yueh-Min Huang, Tien-Chi Huang, and Meng-Yeh Hsieh. 2008. Using annotation services in a ubiquitous Jigsaw cooperative learning environment. *Journal of Educational Technology & Society* 11, 2 (2008), 3–15. http://www.jstor.org/stable/jeductechsoci.11.2.3

[8] Wu-Yuin Hwang, Chin-Yu Wang, and Mike Sharples. 2007. A study of multimedia annotation of Web-based materials. *Computers & Education* 48, 4 (2007), 680–699. https://doi.org/10.1016/j.compedu.2005.04.020

[9] David W Johnson, Roger T Johnson, and Karl Smith. 2007. The state of cooperative learning in postsecondary and professional settings. *Educational Psychology Review* 19 (2007), 15–29.

[10] Jeremiah H. Kalir. 2020. Social annotation enabling collaboration for open learning. *Distance Education* 41, 2 (2020), 245–260. https://doi.org/10.1080/01587919.2020.1757413

[11] Gary King, Eric Mazur, Kelly Miller, and Brian Lukoff. 2019. Instructional support platform for interactive learning environments. US Patent 10,438,498.

[12] Cynthia Bailey Lee, Saturnino Garcia, and Leo Porter. 2013. Can peer instruction be effective in upper-division computer science courses? *ACM Transactions on Computing Education (TOCE)* 13, 3 (2013), 1–22.

[13] Jian-Wei Lin and Yuan-Cheng Lai. 2013. Harnessing Collaborative Annotations on Online Formative Assessments. *Journal of Educational Technology & Society* 16, 1 (2013), 263–274. http://www.jstor.org/stable/jeductechsoci.16.1.263

[14] Dastyni Loksa and Amy J. Ko. 2016. The Role of Self-Regulation in Programming Problem Solving Process and Success. In *Proceedings of the 2016 ACM Conference on International Computing Education Research* (Melbourne, VIC, Australia) *(ICER '16)*. Association for Computing Machinery, New York, NY, USA, 83–91. https://doi.org/10.1145/2960310.2960334

[15] Dastyni Loksa, Lauren Margulieux, Brett A. Becker, Michelle Craig, Paul Denny, Raymond Pettit, and James Prather. 2022. Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use. *ACM Trans. Comput. Educ.* 22, 4, Article 39 (sep 2022), 31 pages. https://doi.org/10.1145/3487050

[16] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. 2008. Relationships between Reading, Tracing and Writing Skills in Introductory Programming. In *Proceedings of the Fourth International Workshop on Computing Education Research* (Sydney, Australia) *(ICER '08)*. Association for Computing Machinery, New York, NY, USA, 101–112. https://doi.org/10.1145/1404520.1404531

[17] Herbert W Marsh. 1982. SEEQ: A reliable, valid and useful instrument for collecting students' evaluation of University teaching. *British journal of educational psychology* 52, 1 (1982), 77–95.

[18] E. Mazur. 1997. *Peer Instruction: Getting Students to Think in Class.* American Institute of Physics, 981–988. https://doi.org/10.1016/0378-4371(79)90165-1

[19] M. Meyer and T. Müller. 2019. If it were that easy: First experiences of introducing a social learning platform in an undergraduate CS course. In *ICERI2019 Proceedings* (Seville, Spain) *(12th annual International Conference of Education, Research and Innovation)*. IATED, 10688–10697. https://doi.org/10.21125/iceri.2019.2624

[20] Kelly Miller, Brian Lukoff, Gary King, and Eric Mazur. 2018. Use of a Social Annotation Platform for Pre-Class Reading Assignments in a Flipped Introductory Physics Class. *Frontiers in Education* 3 (2018), 1–11. https://doi.org/10.3389/feduc.2018.00008

[21] Greg L. Nelson, Benjamin Xie, and Amy J. Ko. 2017. Comprehension First: Evaluating a Novel Pedagogy and Tutoring System for Program Tracing in CS1. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (Tacoma, Washington, USA) *(ICER '17)*. Association for Computing Machinery, New York, NY, USA, 2–11. https://doi.org/10.1145/3105726.3106178

[22] Anthony J. Nitko. 1996. *Educational assessment of students.* ERIC, Prentice-Hall Order Processing Center, P.O. Box 11071, Des Moines, IA 50336-1071.

[23] Elena Novak, Rim Razzouk, and Tristan E Johnson. 2012. The educational use of social annotation tools in higher education: A literature review. *The Internet and Higher Education* 15, 1 (2012), 39–49.

[24] Leo Porter, Cynthia Bailey Lee, Beth Simon, and Daniel Zingaro. 2011. Peer Instruction: Do Students Really Learn from Peer Discussion in Computing?. In *Proceedings of the Seventh International Workshop on Computing Education Research* (Providence, Rhode Island, USA) *(ICER '11)*. Association for Computing Machinery, New York, NY, USA, 45–52. https://doi.org/10.1145/2016911.2016923

[25] Yizhou Qian and James Lehman. 2017. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 1–24.

[26] Mara Saeli, Jacob Perrenet, Wim MG Jochems, and Bert Zwaneveld. 2011. Teaching programming in Secondary school: A pedagogical content knowledge perspective. *Informatics in education* 10, 1 (2011), 73–88.

[27] Cor Suhre, Koos Winnips, Vincent Boer, Pablo Valdivia, and Hans Beldhuis. 2019. Students' experiences with the use of a social annotation tool to improve learning in flipped classrooms. In *Fifth International Conference on Higher Education Advances.* UPV Press, Valencia, Spain. https://doi.org/10.4995/HEAD19.2019.9131

[28] Jacqueline L. Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, P. K. Ajith Kumar, and Christine Prasad. 2006. An Australasian Study of Reading and Comprehension Skills in Novice Programmers, Using the Bloom and SOLO Taxonomies. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52* (Hobart, Australia) *(ACE '06)*. Australian Computer Society, Inc., AUS, 243–252.

[29] Benjamin Xie, Dastyni Loksa, Greg L Nelson, Matthew J Davidson, Dongsheng Dong, Harrison Kwik, Alex Hui Tan, Leanne Hwa, Min Li, and Amy J Ko. 2019. A theory of instruction for introductory programming skills. *Computer Science Education* 29, 2-3 (2019), 205–253.