Can Students Review Their Peers? Comparison of Peer and Instructor Reviews

Nea Pirttinen
University of Helsinki
Helsinki, Finland
nea.pirttinen@helsinki.fi

Juho Leinonen
University of Helsinki
Helsinki, Finland
juho.leinonen@helsinki.fi

ABSTRACT

Having students peer review each other's exercises is a common task in modern computing classrooms. In large classes, peer review might even partly replace traditional instructor-led review – and prior work has found some indications that the quality of peer reviews can be close to that of instructor reviews. In this work, we explore the difference between instructor and peer reviews of student-created programming exercises. One task in an introductory programming course was to have students design their own programming exercises – including an exercise description, model solution, and test cases – which were then reviewed by peers. After the course, we had two instructors review the same student-created exercises. We compare the scores given by the instructors and the students to analyze potential differences. Our results suggest that agreement between instructors and students as measured by inter-rater reliability is low, although differences between instructor and student review score distributions are not statistically significant. Additionally, instructors have more fluctuation in their reviews compared to students. Due to the rising popularity of peer reviews, more research is needed to examine to what extent they could complement traditional instructor-led review of exercises.

CCS CONCEPTS

• **Social and professional topics** → *Computing education*; • **Applied computing** → *Interactive learning environments*; • **Information systems** → *Crowdsourcing*.

KEYWORDS

peer review, inter-rater reliability, crowdsourcing, learnersourcing, contributing student pedagogy, crowdsourced programming assignments

ACM Reference Format:
Nea Pirttinen and Juho Leinonen. 2022. Can Students Review Their Peers? Comparison of Peer and Instructor Reviews. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol 1 (ITiCSE 2022), July 8–13, 2022, Dublin, Ireland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3502718.3524762

ITiCSE 2022, July 8–13, 2022, Dublin, Ireland

ACM ISBN 978-1-4503-9201-3/22/07…$15.00
https://doi.org/10.1145/3502718.3524762

1 INTRODUCTION

As enrollments to computing are rising, the number of students attending courses is also increasing. However, this has not always necessarily meant that those organizing the course get more resources for assessment. One way how instructors have approached this challenge is by introducing more peer review into their classes, where students review each other's answers.

The rising enrollments have called for large exercise pools, especially if the instructor wishes to vary which exercises students work on in different course iterations, for example, to avoid students plagiarizing answers from those in prior iterations. Crowdsourcing – sometimes called learnersourcing in the context of education [14] – is one potential way to create large exercise pools. In learnersourcing, students participate in the creation of course materials, for example, by creating exercises that can then be given to their peers (or used in future course iterations) as practice.

In this work, we study students' peer reviews of programming exercises created by other students. Specifically, the students are reviewing exercises designed and created by others, and not answers to instructor-created exercises. We compare students' peer reviews to reviews given by two instructors to analyze both where students and instructors agree, and where they disagree. If peer reviews given by students are found to be reliable enough and reasonable in quality, it would be possible to use a crowd of students to both create exercises and review them, which would leave more time for the instructor for e.g. supporting struggling students. Previous work has found that students can review exercises well [8], that novice programmers are able to give as good reviews as more experienced students to learnersourced programming exercises [22], that students tend to create exercises that cover a variety of course topics [2], and that students tend to follow instructions when creating learnersourced exercises [24]; although a previous comparison of peer and tutor feedback found that feedback given by tutors was more specific [7].

This article is organized as follows. In Section 2, we provide an overview of previous research on peer reviewing, especially in the context of computer science education and learnersourcing. In Section 3, we present our research methods and questions. Then, in Section 4, we go over the results and discuss them in Section 5. Lastly, we conclude the article in Section 6.

2 RELATED WORK

2.1 Peer Review

In the educational context, peer review is often used as a collaborative learning activity. By assessing their peers' work, students can not only hone their social and collaborative skills, but also gain new viewpoints and ideas into the course material through their peers'

answers [4]. They can also learn to identify mistakes in both their own and others' work, and learn to both give and receive feedback [28]. Peer review can also ease instructors' workload, as utilizing peer reviews can greatly increase the quantity of feedback students receive, especially during the course [7].

Multiple studies indicate that peer reviews can be accurate and provide valid feedback for the students. One aspect of peer review that is unique to computing education is the peer review of other students' code. In an extensive literature review of peer code review in higher education, Indriasari et al. [10] report both common benefits and difficulties, such as development of programming-related skills and low student engagement, respectively. Hamer et al. [7] note that while tutors were more specific in their feedback and wrote longer comments, differences between tutor and peer feedback was not significant in other valuable respects, such as giving advice. They also argue that the effect of peer reviewing does not depend on the feedback produced, but that the primary value is in the process of writing a review. In another study by Hamer et al. [8], it was noted that based on a lexical sophistication analysis, student feedback can be as good or even better than tutor feedback.

2.2 Peer Review in Learnersourcing

Contributing Student Pedagogy (CSP) [6] encourages students to contribute into the learning of other students, and to value the contributions of others. A closely related idea, learnersourcing [14], is a form of crowdsourcing where the crowd consists of students, and the sourced artefacts are used somehow by peers either on the same or a future course iteration. In computer science education, learnersourcing appears in many forms, such as student-created multiple-choice questions [1, 13], programming exercises [3, 21], SQL exercises [17], and open-ended questions [18, 26].

Many learnersourcing systems use peer review at some point of the artefact creation or usage process to evaluate the validity and quality of the student-created artefacts. In CrowdSorcerer [21], which is a programming exercise learnersourcing tool, students peer review each other's programming exercises, and Pirttinen et al. have reported in a study that novice students can be as good reviewers as more experienced students [22]. Denny et al. inspected the coverage of course topics in a student-generated multiple-choice question repository collected with PeerWise [1], and reported that despite having the freedom to choose any topic on the course for their exercises, the students created a repository that covered all the major topics on the course.

The accuracy of peer reviews has also been studied extensively. Studies have reported both tendencies to overrate [20, 27] and underrate [8, 25] scoring in peer assessment. Regarding self-assessment, Stefani [27] found that high-achieving students tend to underestimate, and low-achieving students overestimate their performance.

The accuracy of peer assessment can be adjusted in a multitude of ways. Panadero and Alqassab [20] report the use of a rubric as a support tool for peer review. In their study, all students overrated their peers' performance, but those using a rubric gave more valid review scores. The number of peers can also affect peer review accuracy. Reily et al. [25] aggregated the final peer review scores, collected from multiple students, and reported accurate results when compared to tutor scoring.

3 METHODS

3.1 Learnersourcing Tool

In this study, we used CrowdSorcerer [21], a computing education tool similar to CodeWrite [3], for learnersourcing. In CrowdSorcerer, students can create programming exercises according to instructions given by the instructor of the course. The tool, embedded into online course materials, guides students through full programming exercise creation with exercise description, code template and model solution, and test cases. Model solution is the full, completed code. Code template contains the basic structure of the program, such as class and main method declarations, and possible example input lines that the person completing the programming exercise can see before finishing their answer. The created exercises are submitted to a test server that checks if the given program compiles and passes the student-created tests. Any possible error messages are relayed back to the student, or, if all tests pass, the student receives information that their exercise has been successfully finished. One particular feature of CrowdSorcerer is the focus on teaching testing [12, 23].

The tool also supports peer review features. In the peer review phase, students are given a created exercise in full, and a set of review statements, as well as an open feedback form. Students are not required to try and complete the programming exercise they are reviewing themselves, though the tool does allow downloading the exercise being reviewed as a ZIP file.

3.2 Context and Data

This study was conducted on an introductory Java programming course in the spring of 2019. The course consists of a total of 14 weeks, and teaches the typical introductory Java programming topics, such as variables, conditionals, loops, functions, objects, and object-oriented programming. The course uses an online textbook which contains integrated programming exercises and other practices, the tool described in Section 3.1 included. The programming exercises are generally small, so the students complete some tens of exercises each week, as opposed to completing fewer, larger projects. This iteration of the course was organised both as a MOOC, available for anyone interested in programming fully free of charge, and as a regular university course with weekly lectures and walk-in laboratories. All the participating students used the same materials.

The programming exercises were created on week 12 of the course, and peer reviewed on week 13. Students were asked to create a programming exercise according to the following instructions:

Create a programming exercise that can be used to practice hashmaps, for example, how to find information from a hashmap. The person completing the exercise should be required to program one or more class methods in their answer.

Write an exercise description, model solution and at least three tests. Your method will be placed in the class Submission, which means that the class methods will be in format Submission.method(). Mark the model solution lines that will be hidden from the programmer by using the checkboxes on the left.

When creating the exercise description, try to be as precise as possible. The programmer needs to know what the name of the method they are programming should be, what the method should return, and

what parameters are given to the method. In addition, you can give example code or example inputs that can be used to test the program.

In addition to these instructions, students were provided with an example of a method and accompanied unit tests that were done according to the instructions. Creating an exercise was not mandatory, but awarded students the same number of points as completing a typical programming exercise on the course.

During the peer review, students were given ten review statements that they answered on a five-point Likert-like scale consisting of faces ranging from frowning to neutral to smiling. The review statements were as follows:

- The model solution corresponds to the exercise description
- The code is clean
- The model solution and the code template are separated correctly
- The exercise is creative
- The exercise is suitably difficult
- The exercise description corresponds to the instructions
- The exercise description is clear
- The test cases are reasonable
- The test coverage is on the expected level
- The test names are descriptive

Students were given three exercises for review, two of their peers' and their own. If a student did not create an exercise during the previous week, they were given an additional exercise created by a peer to review instead. As with the exercise creation process, the peer review was not mandatory, and contrary to creating an exercise, no points were awarded for reviewing. All the reviews were submitted separately, so the students could also choose to complete fewer than three reviews.

3.3 Research Questions and Approach

Our research questions are as follows:

RQ1. How many students participated in the creation and peer review of crowdsourced exercises?

RQ2. To what extent do instructors and students agree with each other in their reviews?

RQ3. What characteristics are there in exercises that are highly rated by...
- **RQ3.1** ...only students?
- **RQ3.2** ...only instructors?

For RQ1, we compare the number of students on the course to the number of students who completed the programming exercise creation task, as well as the number of students who reviewed exercises.

For RQ2, 50 student-created exercises were chosen at random for two instructors to review. These exercises were all finished, meaning that the exercise compiles and the given model solution passes the student-provided unit tests. The reviews were completed using the same review statements that the students used for peer review, presented in Section 3.2. All of the analysis for RQ2 and RQ3 focuses on these randomly selected 50 exercises.

Before reviewing, the instructors discussed some of the criteria that could affect the grading of the statements, but all the reviews were done independently. After reviewing the first 10 exercises, the instructors also had a discussion about some edge cases and other noteworthy observations. Neither of the instructors reviewing the exercises in this work were the course instructor on the course in which this study was conducted.

After the instructors had reviewed each of the 50 student-created exercises, we calculated the average of the scores they had given for individual review statements (see the end of Section 3.2) for each exercise, resulting in a single score per exercise for both instructors. For the students, we first calculated the average for the review statement individually for each student, and then an average of these averages per exercise. In the end, for each exercise, we had three scores: instructor #1 score, instructor #2 score, and peers' score.

In our case, the concrete aim in learnersourcing is to create a pool of exercises that are of good enough quality to include in future course iterations. Thus, for both the instructors and the students, we consider exercises that they would "include" with two thresholds: average score greater than 3.0 (>3) and average score greater than 4.0 (>4) on a scale of 1-5. The threshold of >3 was chosen because the instructors hypothesized that these exercises may be reasonably good, and could be included as small exercises in future course iterations with some modifications. Exercises rated higher than the second threshold of >4 are hypothesized to be excellent exercises that could likely be included in future course iterations with either no or very minor modifications.

To evaluate agreement between the instructors and students, we calculate inter-rater reliability for their "include" decisions for both thresholds separately using Krippendorff's alpha [15]. Krippendorff's alpha measures the extent of agreement between any number of reviewers, $1 \geq \alpha \geq -1$. We inspect the alpha agreement to the review score averages between the two instructors, and between students and instructors separately (students and instructor #1, students and instructor #2).

The results are evaluated using the guidelines by Krippendorff [16] where it is outlined that an $\alpha > 0.667$ can be used to draw tentative conclusions, while $\alpha > 0.800$ is required for stronger conclusions. We also applied the Mann–Whitney U test to examine whether differences between reviewers' score averages are statistically significant. We chose Mann-Whitney U as the data being compared is ordinal (can be ordered) but not interval (difference between values is not necessarily equal).

For RQ3, we inspect some of the student-created exercises for qualitative analysis, aiming to find some common features these exercises may have. Specifically, we examined exercises that only students or instructors would include with the thresholds explained above (>3 and >4 average review score). For both thresholds, we examine four categories: both (students and instructors) include, only instructors include, only students include, and neither include. For the instructors, we considered an exercise to be included by the instructors only if both instructors included it based on the above criteria.

4 RESULTS

4.1 Student Participation

In total, 9707 students were enrolled on the course. Out of these, 1358 were active on week 12 (the week of exercise creation prompt), and 1299 on week 13 (the respective peer reviews). A student is

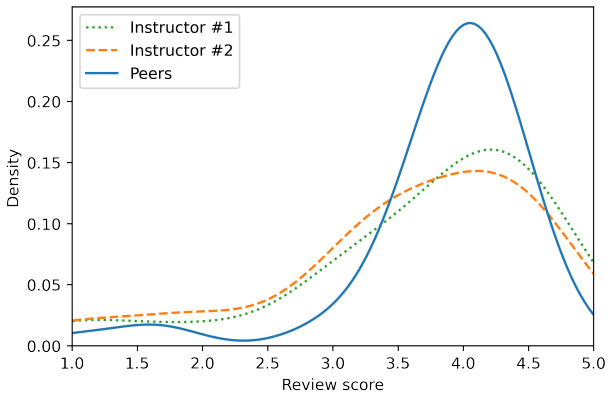

Figure 1: Distribution of review scores for peers (students) and the two instructors.

Table 1: The number of exercises that both students and instructors, only instructors, only students, or neither would include with an inclusion threshold of peer review average greater than 3.0.

Both include: 39	Only instructors include: 0
Only students include: 7	Neither include: 4

Table 2: The number of exercises that both students and instructors, only instructors, only students, or neither would include with an inclusion threshold of peer review average greater than 4.0.

Both include: 12	Only instructors include: 7
Only students include: 11	Neither include: 20

considered active during a week if they collected at least one point from the week's exercises.

Out of the active students on week 12, 405 students (29.8%) tried creating an exercise, meaning that they submitted it to the test server for evaluation. Out of these, 333 students (82.2%) finished the exercise, meaning that they eventually submitted an exercise that compiled and passed student-created unit tests. Those who did not had some errors in their submission, such as non-compiling code, a model solution that did not pass the tests, or a timeout occurred due to an infinite loop.

Out of the active students on week 13, 807 students (62.1%) reviewed at least one exercise. In total, the exercises from week 12 received 2290 reviews, meaning that on average each student-created exercise received approximately seven peer reviews.

4.2 Student and Instructor Agreement

We report the agreement using two review score average thresholds: exercises that received an average greater than 3.0 (>3), and exercises that received an average greater than 4.0 (>4). How many exercises are included in the set using this criteria is summarized in Tables 1 and 2, and inspected in more detail in Section 4.3. The distribution of the review score averages of both peer and instructor reviews is summarized in Figure 1. The Krippendorff's alphas between the instructors and the peer reviewers are outlined in Table 3.

Table 3: The inter-rater reliability between instructors and peers with review score average thresholds of greater than 3.0 and greater than 4.0.

Inclusion threshold	Instructor #1 – instructor #2	Instructor #1 – peers	Instructor #2 – peers
>3	$\alpha = 0.88$	$\alpha = 0.51$	$\alpha = 0.51$
>4	$\alpha = 0.76$	$\alpha = 0.12$	$\alpha = 0.36$

When measured with Krippendorff's alpha, the inter-rater reliability between the two instructors with average review score >3 was $\alpha = 0.88$, and with >4, $\alpha = 0.76$, where the both are considerably over the threshold of $\alpha > 0.667$ outlined by Krippendorff for making tentative conclusions about content [16]. When comparing the average review scores, the Mann-Whitney test results are $U = 1153.5$ and $p = 0.25$. The average difference between the average scores for exercises between the instructors was 0.28 (on a 1-5 scale).

The inter-rater reliability between the students' peer reviews and instructor #1 when average review score >3 was $\alpha = 0.51$, and with >4, $\alpha = 0.12$. Both α values are noticeably lower than the $\alpha > 0.667$ threshold suggested by Krippendorff [16]. For the Mann-Whitney test, $U = 1246.0$ and $p = 0.49$. The average difference between the average scores between students' peer reviews and the first instructor's reviews was 0.51.

The inter-rater reliability between students and instructor #2 with average review score >3 was $\alpha = 0.51$, and with >4, $\alpha = 0.36$. Both α values are noticeably lower than the $\alpha > 0.667$ threshold suggested by Krippendorff [16]. For the Mann-Whitney test, $U = 1113.5$ and $p = 0.17$. The average difference between students' peer review scores and the second instructor's reviews was 0.54.

4.3 Exercise Characteristics

In this section, we take a look into the exercises in Tables 1 and 2 that only students or instructors included with either greater than 3.0 or greater than 4.0 threshold. The purpose was to examine whether these exercises have any noticeable similarities in, for example, their contents or structure.

There were seven exercises that only students rated greater than 3.0 on average, and eleven exercises that only students rated greater than 4.0. Two of these were edge cases where one of the instructors scored the exercise exactly at 3.0, which excluded the exercise from the "both include" set, and one additional exercise was an edge case in the >4 threshold. Since all these exercises share the same issues, these features are reported as one, regardless of the threshold. Generally, it seemed that if the exercise description and model solution are clear and seem functional at a glance, students tend to grade test-related statements well, regardless of the actual quality of the test cases.

In cases where there were clear shortcomings in the exercise, e.g., the model solution and code template were exactly the same,

or the exercise description was noticeably too short and vague, the instructors were more strict in their scoring. While students tended to score the statements "The model solution and the code template are separated correctly" and "The exercise description is clear" in these types of exercises as 3 (mid-point of the scale), both of the instructors gave the score of 1 or 2, depending on the case.

In addition, there were seven exercises that only the instructors included in the >4 set. Most of these exercises were on the more difficult side of the student-created exercises, at least seemingly. These exercises had longer, more detailed exercise descriptions, and in some cases, the code templates had several lines of example code for testing purposes. Even if the method to be programmed was not complicated or too difficult for students at this point of the course, many peer reviewers rated these exercises not suitably difficult, specifying in the open feedback that the exercise seemed too hard. Most of these exercises also received lower scores from the students for the statement "The exercise description is clear", even though the instructors did not consider the descriptions to contain irrelevant information.

5 DISCUSSION

One concern in prior work has been whether students will participate in peer review [11], and as such, e.g. gamification has been proposed as a potential way of increasing peer review participation [9, 11]. However, in our case, more students participated in the peer review process than creating a programming exercise (62.1% and 29.8%, respectively), even though peer reviewing did not award any course points, unlike creating an exercise. Students might be curious of their peers' exercises, even if they did not use the tool themselves for the creation process. It is also possible that peer reviewing is seen as a less laborious task, and worth the effort, even if it does not give any points towards the final grade.

Considering prior work that has examined the use of CrowdSorcerer [12], slightly fewer students created exercises in this iteration of the course (29.8% in this study versus 38% in [12]). We have two possible hypotheses for this. First, the prior study focused on a course that was organized locally, whereas our study focuses on a course offered simultaneously as a MOOC and as a local university course. It is possible that MOOC participants are less likely to participate in learnersourcing compared to local university students. Second, in this study, we examined data collected later in the course compared to the prior study (weeks 12 and 13 in this study, weeks 2 to 7 in the earlier study). It is also possible that students are more likely to participate in learnersourcing if the activity is situated earlier in the course.

Comparing the distributions of review scores for students and instructors (Figure 1), one noticeable difference between the students and the instructors is that the student review score distribution seems to be slightly bimodal with peaks near 1.5/5.0 and 4.0/5.0 while the distributions for both instructors are unimodal with peaks near 4.1/5.0. This suggests that students might deal more in absolutes compared to the instructors. One possible explanation is that students can accurately identify exercises that are either very good or very poor, but have a harder time reviewing mediocre exercises. This result potentially explains the findings of prior work that has found that students tend to both under [8, 25] and overrate [20, 27] their peers in peer review.

Looking at Figure 1, students seem to have a tendency to rate the exercises created by their peers as "pretty good" (4/5), but give fewer very good (5/5) or very bad (1/5) scores compared to the instructors. These results differ from prior work that found that peers' gave harsher ratings compared to tutors or instructors [8]. Overall, we found that – considering suggested thresholds for "good agreement" as measured by Krippendorff's alpha [16] – students and instructors had low agreement with each other. On the other hand, there were no statistically significant differences in review score as measured by a Mann-Whitney U test: this also raises the question about what methodologies would be the most appropriate for evaluating how similarly instructors and students review exercises.

Possible ways of increasing agreement between students and instructors would be to have a better rubric outlining *how* an exercise should be reviewed [20]. In our case, students (and instructors) relied on a set of review statements without explicit guidelines for how to score these individual statements. Another possible improvement suggested by prior work is to aggregate peer reviews [25] – however, we found poor agreement between instructor reviews and aggregated student reviews.

Student reviews seemed to give good scores across all the review statements if the exercise description and model solution were clear and concise, even if the given test cases were irrelevant for the model solution or, in some cases, did not test anything. In these cases, the instructors gave the test-related review statements lower scores (1-2), while students still gave these statements fairly good scores (3-4). While testing has been introduced to the students since week 3 in small steps, students might not see the relevancy of testing, or be hesitant to review test cases, thinking that they do not have the skill set to do so.

Many of the exercises in the "only instructors include" category were reviewed as too difficult by the students. On closer inspection, these exercises were either very suitable as exercises for this point of the course, or they were actually very simple, but with more background information or example input lines than contained in a typical student-created exercise. It is possible that if students review the exercises in a hurry, they do not pay close attention to what the exercise actually asks to implement, but get "frightened" by a longer description and decide that the exercise is too complicated.

Multiple studies have reported that students review exercises similarly to tutors or instructors [7, 8, 17], and that novices review similarly to more experienced students [22]. The results of this study contradict with these previous findings. It may be that our review statements affect the results, as they were not intended as a rigorous review rubric, but to provide some general overview into what to consider when reviewing the exercises' content.

5.1 Limitations

As a threat to external validity, we acknowledge that our results do not necessarily generalize to other contexts. It is possible that simply the context of learnersourcing affects reviews in a way that is not applicable for reviewing traditional exercises. Additionally, as students know that they are reviewing programming exercises created by other students, they might review differently as opposed

to reviewing programming exercises that they would know to have been created by a teacher.

Regarding internal validity, we acknowledge that we have no ground truth about the quality of the exercises. However, in this work, we are only investigating whether students and instructors agree in their reviews, i.e. perceive the exercises similarly, and do not address which group is better at recognizing a "good" exercise. It can be argued that instructors should recognize suitable exercises more easily, as they likely have more experience and thus should be well aware of what constitutes a good exercise. However, prior work has found the possible existence of an "expert blind spot" [5, 19], meaning that experts can be blind to things that are more apparent to novices. Similarly, it could easily be argued that students are better evaluators of their own subjective experiences, e.g. related to the difficulty of a program – which also was one of the aspects of the created exercises in which students and instructors had most differences in their reviews.

Peer reviews are subjective, and the students did not have a rubric or rigorous guidance to the review process. The instructors, on the other hand, did discuss the review statements beforehand, meaning that the reviewing process is not fully comparable. Additionally, some of the review statements, such as "The exercise is creative" are likely to be scored very differently by each individual. While the instructors mostly analyzed whether the exercise was a direct copy of a programming exercise in the course material as a measurement of creativity, students could have rated based on much more subjective factors, such as their sense of humor.

Lastly, we acknowledge that the reviews are Likert-like data, which is ordinal, but not interval data. When interpreting the results, it should be acknowledged that we have used averages as a part of the reporting.

6 CONCLUSION

In this work, we compared the peer reviews of student-created programming exercises to instructors' reviews in order to analyze potential differences in reviewers' perceptions, and to see where instructors and students agree and disagree in their reviews. Summarized, our research questions and their answers are as follows:

RQ1. *How many students participated in the creation and peer review of crowdsourced exercises?* **Answer:** Over 60% of the active participants of the course participated in the peer review, which was considerably more compared to the number of students – about 30% – who participated in the learnersourcing of programming exercises. Notably, students were given course points to participate in the exercise creation, but not the peer review.

RQ2. *To what extent do instructors and students agree with each other in their reviews?* **Answer:** We found that the agreement between the instructors was reasonably high. Comparing student and instructor reviews, however, we found conflicting results. On one hand, agreement measured by inter-rater reliability was low; but on the other hand, the differences in review scores between the instructors and students were not statistically significant.

RQ3. *What characteristics are there in exercises that are highly rated by...*

- **RQ3.1** *...only students?* **Answer:** The exercises are generally good, but, for example, have some crucial information missing in the problem description or poor unit tests.
- **RQ3.2** *...only instructors?* **Answer:** The exercises are at least seemingly more complicated than the average learnersourced exercise, and thus rated as too difficult by the students, but not by the instructors.

Our results are slightly contradictory to some prior work [8, 17] that has found students' reviews to be close to instructor/tutor reviews. From the instructors' perspective, this means that as the agreement between instructors and students reviews was low in our case, it should be considered carefully whether peer reviews are valid for all contexts they are currently used in. However, this low agreement might, at least partially, be explained by the bimodal distribution of students' reviews. Considering the bimodal distribution, peer reviews could still be a valid review method for a coarse binary evaluation (e.g. "exercise is adequate / exercise needs work"), even in our context.

Future work should more closely examine potential factors, such as what is being reviewed and how the review process is conducted, that could illustrate when and how peer review works best. In addition, in our future work, we are interested in examining whether there is a difference in how different student demographics, such as novices or more experienced students, use learnersourcing tools for both exercise creation and when reviewing their peers' creations. This would also include a closer inspection of student subpopulations: are there, for example, groups that agree with the instructors significantly more or less? Similarly, we are interested in whether there are differences between those who attempt the exercises they review exercises and those who review the exercises without attempting them – for example, are reviews by those who did not attempt the exercise more inexact?

Regarding the student-given reviews, we are interested in inspecting the written reviews more closely, for example, by comparing textual peer and instructor feedback similarly to a prior study by Hamer et al. [7]. Additionally, it would be valuable to study other ways of aggregating student reviews in addition to averaging. While prior work has found that aggregating student peer reviews seems accurate [25], it is possible, for example, that student reviews would correlate more with instructor reviews if the lowest and highest reviews given by students were removed before aggregation (similar to a trimmed mean).

Altogether, our results provide novel insights into the types of exercises that students and instructors might disagree on when reviewing, and some evidence on peer reviews not necessarily matching instructor reviews in some contexts.

ACKNOWLEDGMENTS

We are grateful for the doctoral research grant awarded by Jenny and Antti Wihuri Foundation to the first author.

REFERENCES

[1] Paul Denny, Andrew Luxton-Reilly, and John Hamer. 2008. The PeerWise System of Student Contributed Assessment Questions. In *Proceedings of the*

Tenth Conference on Australasian Computing Education - Volume 78 (Wollongong, NSW, Australia) *(ACE '08)*. Australian Computer Society, Inc., AUS, 69–74. https://dl.acm.org/doi/10.5555/1379249.1379255

[2] Paul Denny, Andrew Luxton-Reilly, John Hamer, and Helen Purchase. 2009. Coverage of course topics in a student generated MCQ repository. In *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*. 11–15. https://doi.org/10.1145/1562877.1562888

[3] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. 2011. CodeWrite: Supporting Student-Driven Practice of Java. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. Association for Computing Machinery, New York, NY, USA, 471–476. https://doi.org/10.1145/1953163.1953299

[4] Nancy Falchikov. 2007. *Rethinking Assessment in Higher Education*. Routledge. Chapter 11: The place of peers in learning and assessment.

[5] Philip J. Guo, Julia M. Markel, and Xiong Zhang. 2020. Learnersourcing at Scale to Overcome Expert Blind Spots for Introductory Programming: A Three-Year Deployment Study on the Python Tutor Website. In *Proceedings of the Seventh ACM Conference on Learning @ Scale* (Virtual Event, USA) *(L@S '20)*. Association for Computing Machinery, New York, NY, USA, 301–304. https://doi.org/10.1145/3386527.3406733

[6] John Hamer, Quintin Cutts, Jana Jackova, Andrew Luxton-Reilly, Robert McCartney, Helen Purchase, Charles Riedesel, Mara Saeli, Kate Sanders, and Judithe Sheard. 2008. Contributing Student Pedagogy. *SIGCSE Bull.* 40, 4 (nov 2008), 194–212. https://doi.org/10.1145/1473195.1473242

[7] John Hamer, Helen Purchase, Andrew Luxton-Reilly, and Paul Denny. 2015. A comparison of peer and tutor feedback. *Assessment & Evaluation in Higher Education* 40, 1 (2015), 151–164. https://doi.org/10.1080/02602938.2014.893418

[8] John Hamer, Helen C Purchase, Paul Denny, and Andrew Luxton-Reilly. 2009. Quality of peer assessment in CS1. In *Proceedings of the fifth international workshop on Computing education research workshop*. 27–36. https://doi.org/10.1145/1584322.1584327

[9] Theresia Devi Indriasari, Andrew Luxton-Reilly, and Paul Denny. 2020. Gamification of student peer review in education: A systematic literature review. *Education and Information Technologies* 25, 6 (2020), 5205–5234. https://doi.org/10.1007/s10639-020-10228-x

[10] Theresia Devi Indriasari, Andrew Luxton-Reilly, and Paul Denny. 2020. A Review of Peer Code Review in Higher Education. 20, 3, Article 22 (sep 2020), 25 pages. https://doi.org/10.1145/3403935

[11] Theresia Devi Indriasari, Andrew Luxton-Reilly, and Paul Denny. 2021. Improving Student Peer Code Review Using Gamification. In *Australasian Computing Education Conference*. 80–87. https://doi.org/10.1145/3441636.3442308

[12] Vilma Kangas, Nea Pirttinen, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2019. Does creating programming assignments with tests lead to improved performance in writing unit tests?. In *Proceedings of the ACM Conference on Global Computing Education*. 106–112. https://doi.org/10.1145/3300115.3309516

[13] Hassan Khosravi, Kirsty Kitto, and Joseph Jay Williams. 2019. RiPPLE: A Crowdsourced Adaptive Platform for Recommendation of Learning Activities. *Journal of Learning Analytics* 6, 3 (2019), 91–105. https://doi.org/10.48550/arXiv.1910.05522

[14] Juho Kim. 2015. *Learnersourcing: improving learning with collective learner activity*. Ph. D. Dissertation. Massachusetts Institute of Technology. https://dspace.mit.edu/handle/1721.1/101464

[15] Klaus Krippendorff. 1970. Estimating the Reliability, Systematic Error and Random Error of Interval Data. *Educational and Psychological Measurement* 30, 1 (1970), 61–70. https://doi.org/10.1177/001316447003000105

[16] Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, Inc.

[17] Juho Leinonen, Nea Pirttinen, and Arto Hellas. 2020. Crowdsourcing Content Creation for SQL Practice. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. Association for Computing Machinery, New York, NY, USA, 349–355. https://doi.org/10.1145/3341525.3387385

[18] Andrew Luxton-Reilly, Beryl Plimmer, and Robert Sheehan. 2010. StudySieve: A Tool That Supports Constructive Evaluation for Free-Response Questions. In *Proceedings of the 11th International Conference of the NZ Chapter of the ACM Special Interest Group on Human-Computer Interaction* (Auckland, New Zealand) *(CHINZ '10)*. Association for Computing Machinery, New York, NY, USA, 65–68. https://doi.org/10.1145/1832838.1832849

[19] Mitchell J Nathan, Kenneth R Koedinger, Martha W Alibali, et al. 2001. Expert blind spot: When content knowledge eclipses pedagogical content knowledge. In *Proceedings of the third international conference on cognitive science*. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.352.8217

[20] Ernesto Panadero, Margarida Romero, and Jan-Willem Strijbos. 2013. The impact of a rubric and friendship on peer assessment: Effects on construct validity, performance, and perceptions of fairness and comfort. *Studies in Educational Evaluation* 39, 4 (2013), 195–203. https://doi.org/10.1016/j.stueduc.2013.10.005

[21] Nea Pirttinen, Vilma Kangas, Irene Nikkarinen, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Crowdsourcing Programming Assignments with CrowdSorcerer. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (Larnaca, Cyprus) *(ITiCSE 2018)*. ACM, New York, NY, USA, 326–331. https://doi.org/10.1145/3197091.3197117

[22] Nea Pirttinen, Vilma Kangas, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Analysis of Students' Peer Reviews to Crowdsourced Programming Assignments. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling 2018)*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3279720.3279741

[23] Nea Pirttinen and Juho Leinonen. 2019. Integrating CrowdSorcerer: Lessons Learned. In *Proceedings of SPLICE 2019 workshop Computing Science Education Infrastructure From Tools to Data at 15th ACM International Computing Education Research Conference*. National Science Foundation (NSF). https://cssplice.github.io/ICER19/proc/SPLICE_2019_ICER_paper_9.pdf

[24] Nea Pirttinen and Juho Leinonen. 2021. Exploring the Complexity of Crowdsourced Programming Assignments. In *Seventh SPLICE Workshop at SIGCSE 2021 "CS Education Infrastructure for All III: From Ideas to Practice"*. https://cssplice.github.io/SIGCSE21/proc/SPLICE2021_SIGCSE_paper_1.pdf

[25] Ken Reily, Pam Ludford Finnerty, and Loren Terveen. 2009. Two Peers Are Better than One: Aggregating Peer Reviews for Computing Assignments is Surprisingly Accurate. In *Proceedings of the ACM 2009 International Conference on Supporting Group Work* (Sanibel Island, Florida, USA) *(GROUP '09)*. Association for Computing Machinery, New York, NY, USA, 115–124. https://doi.org/10.1145/1531674.1531692

[26] Sam Saarinen, Shriram Krishnamurthi, Kathi Fisler, and Preston Tunnell Wilson. 2019. Harnessing the Wisdom of the Classes: Classsourcing and Machine Learning for Assessment Instrument Generation. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 606–612. https://doi.org/10.1145/3287324.3287504

[27] Lorraine A.J. Stefani. 1994. Peer, self and tutor assessment: Relative reliabilities. *Studies in Higher Education* 19, 1 (1994), 69–75. https://doi.org/10.1080/03075079412331382153

[28] Keith Topping. 1998. Peer Assessment Between Students in Colleges and Universities. *Review of Educational Research* 68, 3 (1998), 249–276. https://doi.org/10.3102/00346543068003249