Evaluating the Effects of Automated Test Results, Model Solutions, and Checklists on Peer Code Review: A Quantitative Study

Sven Strickroth
sven.strickroth@ifi.lmu.de
LMU Munich
Munich, Germany

Abstract

Peer code review is a valuable pedagogical approach that allows students to receive timely, personalized, and comprehensive feedback while fostering constructive collaboration and reducing educators' workload. In programming education, however, students often struggle to assess the correctness of peer solutions and provide very short feedback. To address these challenges, some studies provide support to students; however, systematic comparisons of support types remain scarce. This paper addresses this gap by comparing the effects of four types of support—no support, automated test results, model solutions, and checklists—on students' peer reviews and perceptions in a single introductory programming course. Our quantitative analysis of 622 peer reviews on review time, length, and correctness reveals that students are better at assessing actual correct submissions as correct than incorrect ones as incorrect, with test results and checklists slightly increasing their performance in identifying incorrect submissions. Overall, there seems to be no significant differences in correctness assessment performance and feedback length across the four types of support. Students are very confident in their assessments. They spent the least time on reviews when no support was provided, suggesting a cognitive disengagement rather than efficiency. While many students prefer model solutions due to reduced perceived uncertainty, this preference is not reflected in their self-reported confidence during reviews. Implications of different types of support are discussed.

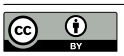
CCS Concepts

• **Social and professional topics** → **Computing education**; • **Applied computing** → *Collaborative learning*.

Keywords

peer review, code review, peer teaching, feedback, programming education

ACM Reference Format:
Sven Strickroth. 2026. Evaluating the Effects of Automated Test Results, Model Solutions, and Checklists on Peer Code Review: A Quantitative Study. In *Proceedings of the 2nd ACM Virtual Global Computing Education Conference V.1 (SIGCSE Virtual 2026), November 12–15, 2026, Virtual Event, USA*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3795867.3830991



SIGCSE Virtual 2026, Virtual Event, USA

ACM ISBN 979-8-4007-2506-7/2026/11
https://doi.org/10.1145/3795867.3830991

1 Introduction

Peer review is an established method in higher education in which learners of equal status provide feedback to each other [34]. It is a pedagogical method to initiate collaboration among students, exposing them to diverse solution approaches, and helping them learn to critically evaluate possible solutions and to provide feedback. Furthermore, peer review can be used to scale feedback processes in mass education [10, 24, 29] by enabling personalized and timely feedback that extends beyond the automated test results provided by traditional e-assessment systems, which typically focus on syntactic and functional correctness and neglect other quality dimensions [16, 18, 31]. Additionally, peer review plays a key role in professional software development [7], and the ability to critically assess code is likely to become even more important in a future where generative AI (GenAI) is routinely used to generate and refactor code. Therefore, learning to systematically assess whether (generated) code is correct, to judge its quality (e.g., whether it is robust or secure), and to provide constructive feedback are very important.

Despite its potential, peer review poses notable challenges: Prior work and practical experience indicate that students often struggle to identify relevant issues in peer solutions and to formulate specific, actionable feedback. As a result, reviews may remain very short, superficial, overly positive, partially incorrect, or focus on trivial aspects, thereby limiting the learning benefits for both reviewers and reviewees [10, 12]. To address these challenges, instructors may provide support during the peer review process, such as rubrics or checklists [12] that make evaluation criteria explicit and structure the review. Other forms of support, such as model solutions or test results, may also appear intuitively beneficial; however, their effects on students' reviews have not yet been systematically compared.

In this paper, we present a quantitative study of peer review in an introductory programming course, investigating how different types of support influence students' reviewing behavior and their perception of the peer review process. We systematically varied whether reviewers received automated test results, a model solution, a checklist, or no support, and we analyzed students' assessment, review texts, and subjective opinions about the process. This paper answers the following research questions: (**RQ1**) What effects do automated test results, model solutions, and checklists have on students' (syntactic/functional) assessment accuracy and their self-reported confidence? (**RQ2**) What effects do automated test results, model solutions, and checklists have on the review time and the length of the feedback? (**RQ3**) How do students perceive peer review and the different types of support?

By addressing these questions, we provide empirical evidence to inform the design of effective peer review activities in programming education and to support students in developing code review skills.

2 Related Work

Peer review has proven to be a highly effective pedagogical approach, enabling students to receive timely and comprehensive feedback while fostering constructive collaboration among students [5, 9, 12, 15, 33]. It has been linked to deeper and active learning [15, 21, 34]. Notably, giving feedback appears to lead to stronger learning gains than simply receiving it; particularly when learners provide detailed, explanatory, and constructive comments [37].

Indriasari et al. [12] compiled benefits, barriers, and challenges of peer code review in a systematic literature review: One the one hand, students frequently report being motivated to participate, citing opportunities to exchange ideas, explore alternative solutions, develop constructive critique skills, learn from feedback, and engage in collaborative learning [4, 22, 23, 27]. There are reports that students can detect and correct syntax errors in peer review and that it can further enhance code quality and stimulate discussion on higher-level design and implementation decisions [11, 14]. On the other hand, an often reported issue is low student motivation and engagement [10, 12, 36] Given feedback is often identified as quite short (median 14 words in a Haskell course [10], median 13 words in a large introductory Java course [27], or 25 tokens on average [8]). Also, the assessment quality is often reported to be quite low, such as about 22 % of reviews being partly incorrect [10] or having an accuracy of 64 % [27]. There are reports of improvements in students' review and programming skills over time [3, 11, 25, 27], but at the same time of no improvements in review skills (e.g., [35]). Besides low motivation, contributing factors include students' perceived or actual lack of understanding of the material (e.g., [28, 36]), a tendency to rush through reviews at the last minute (e.g., [25]), or too little time is allocated for peer review (e.g., [26]).

To support learners working on programming assignments, scaffolding such as example solutions [20] or automated test results (e.g., [6, 30]) has proven effective. For peer reviews, educators often provide students with checklists or rubrics [11, 12, 17, 24] to guide students and reduce cognitive load [32]. From the literature, it is unclear whether test results or model solutions were also provided for peer code review. Song et al. [24] found that longer descriptions in rubrics led to decreased grading accuracy. Koitz-Hristov [17] compared different types (individual, team, and pair reviews) of peer code review in a first-year algorithms course to determine students' perceptions in terms of their understanding of data structures, their programming skills, and their overall learning experience. She underlines the importance of checklists and mock reviews [17]. Besides these studies, the author is not aware of any empirical study comparing different types of support. Indriasari et al. [12] found that most studies are solely based on students' self-reports and were conducted in courses with fewer than 200 students. They conclude that more research is needed to assess the correctness of peer reviews [12]. These research gaps are addressed in this paper.

3 Context

The study was conducted in the first-semester introductory programming course at LMU Munich, Germany, in winter term 2024/2025. The course, taught in Java, covered imperative and object-oriented programming for computer science majors (minors could also take the course). About 650 students were enrolled.

The course included 14 voluntary weekly homework assignments (except during the Christmas break). The assignments were administered using the GATE system [30]. Each exercise sheet consisted of four to five tasks. Submitted solutions were not individually corrected manually. Instead, weekly exercise sessions led by student teaching assistants were held, during which solution approaches were discussed, an additional voluntary peer review was conducted, and GATE provided automated feedback based on syntactic analysis and black-box unit tests for assignments that were not peer reviewed. To avoid overloading students, peer review was conducted for only one assignment per exercise sheet. Peer review was also managed using the GATE system. Each submission from participating students was assigned two peer reviews, which had to be completed within one week (extended to three weeks during the Christmas holidays). To prevent passive or read-only participation, a strict exclusion policy was enforced: students who failed to submit a solution or to provide peer reviews twice were permanently excluded from the peer review process, except for justified cases (e.g., illness; as done in [27]).

The peer review form required students to assess their peers' submissions on syntactic correctness, overall correctness/completeness, readability (coding style), and understandability using a four-point Likert scale. In addition, students were asked to indicate their confidence in their assessment using the same scale and to provide written feedback in a text field, which could not be left empty. The files subject to peer review could be downloaded by reviewers for local testing. After submitting their reviews, students could discuss the feedback with both reviewers in a forum-like environment. To prepare students for the peer code review, a worked example was discussed during the lecture. No additional training was provided, and tutors were not actively involved in the peer review process.

Of the 343 students who submitted at least one peer review, 306 (90 %) voluntarily consented to their reviews being used for research. Based on local regulations no IRB review was necessary.

4 Methodology

The course comprised 14 assignment sheets (i.e., 14 peer review rounds); however, only 10 involved actual programming. Of these, three tasks included substantial UML modeling and design freedom and therefore could not be evaluated automatically (i.e., syntactic assessment applied to both UML and Java code). For the final two tasks, design freedom similarly limited automated evaluation to syntactic correctness only. Furthermore, the peer review task on exercise sheet six had to be excluded from the analysis because the corresponding unit test was found to be incorrect in some cases. Consequently, six tasks were included in the analysis (cf. Table 1).

To compare the effects of different support types, students were randomly assigned to one of the four support types in each peer review round: *no support*, *automated test results*, *model solution*, or *checklist*. To ensure a sufficient number of participants per type, higher priority was given to *checklist*s and *model solution*s, as these require higher-order cognitive skills, and difficulties with *no support* are already known. Hence, the students were not always evenly distributed. The first code review included only two conditions, namely (syntactic and functional) *test results* and *model solution*. Task 6 was designed to include syntactic and functional *test results*

as well as the other three support types. Tasks 8 and 11 included three conditions (*no support*, *checklist*, and *model solution*). Tasks 12 and 13 included all four support types, with *test results* limited to syntactic feedback only. Due to a low number of participants, task 14 included only two conditions (*no support* and *model solution*). Table 1 summarizes the tasks and the number of students assigned to each support type. The automated test results were presented exactly as students were accustomed to in GATE: Result of syntax (in case of failure: compiler error output) and/or test cases covering common and edge cases, contrasting expected and actual results (the same tests were used to judge final correctness in this study). The model solutions consisted of uncommented code. The checklists comprised generic and task-specific items, such as for Task 12: "The implementation is syntactically correct.", "The code is well formatted and consistent.", "The inheritance hierarchy of the 'IllegalISBNException' classes is correct.", and "The ISBN validity checks are correct and cover all cases." No score was calculated.

To answer the first two research questions, the submitted peer reviews and associated metadata were analyzed quantitatively: The actual syntactic correctness of the submissions was determined using the OpenJDK 17 compiler. This result was then compared with the students' assessments. Functional correctness was evaluated using unit tests derived from the task specification; note that syntactic errors also cause unit tests to fail. To measure the needed time for peer review, a Unix timestamp was included in the HTML form and subtracted from the current time when processing the HTTP POST request. The number of words was determined by splitting the review on whitespace ("\s+") and counting the resulting tokens.

Only non-parametric significance tests are used, as they do not require e.g. normally distributed data (cf. [1, 13]). Note that these tests are typically stricter than their parametric counterparts [13].

To answer the third research question, we conducted a voluntary online survey at the end of the term asking students about the lecture, the GATE system, and peer review in particular. All survey items and responses were translated from German for this paper.

5 Results

Overall, 3,108 peer review assignments were issued, of which 2,691 were fulfilled by 306 distinct students. For the analyzed tasks, there were 719 peer review assignments, of which 622 were submitted by 168 distinct students (cf. Table 1). Participation decreased noticeably over the course of the semester, as reflected in declining numbers of submissions, completed peer reviews, and participating students.

The reviewed submissions are mostly syntactically correct (91 %). Syntactic correctness ranges from 87 to 100 % across tasks and remained relatively stable throughout the semester. In contrast, functional/complete correctness ranges from 17 % (task 11: collections and object-oriented modeling) to 51 % according to the tests (43 % overall). Notable is the low frequency in task 11, which appears to be primarily due to incorrectly sorted collections (in reverse order) and missing methods. Overall, files were downloaded 43 times.

5.1 Students' Assessment Performance

Students assessed submissions for syntactic and overall correctness/completeness using a four-point Likert scale (1 = lowest, 4 = best). For comparison with the actual test results, these ratings were mapped to a binary scale to construct contingency tables. A rating of 4 defined syntactic correctness as lower values indicate (minor) issues. Alike, for overall correctness/completeness, both syntactic and overall correctness/completeness had to be rated as 4 (cf. [27]).

Table 2 summarizes students' performance in assessing the syntactic correctness. Overall accuracy (ACC) is .91. Students were more accurate in identifying syntactically correct submissions (true positive rate TPR = .95) than syntactically incorrect ones (true negative rate TNR = .56). All classifications are statistically significant ($p \leq .001$); however, the *model solution* condition yielded only a medium effect size compared to large otherwise [19].

Across support types, accuracy ranges from .88 (model solution) to .95 (test results). The true positive rate (TPR) remained stable at about .95 across all types. The true negative rate (TNR) varies substantially, ranging from .33 for *model solutions* to .80 for *test results*, and from .65 to .68 for *no support* and *checklists*. No significant differences in assessment performance were found across the four support types ($\chi^2(3, N = 622) = 4.694$, $p = .191$), nor between the *no support* condition and all other support conditions combined (Fisher's exact test, $\chi^2(1, N = 622) = .732$, $p = .429$).

For the four tasks that are automatically testable for functional correctness, students' overall correctness classification performance is reported in Table 3. Overall, accuracy is .6, with a significantly higher TPR (.92) than TNR (.35). Among the support types, the *tests results* achieved the highest accuracy (.7) and TPR (.98), whereas the highest TNR (.45) can be observed for the *checklists* type – the other support types range between .3 and .38. For each support type, the classification performance is statistically significant ($p \leq .001$). Effect sizes are large for *test results*, moderate for *no support* and *checklists*, and small for *model solutions* [19]. No statistically significant differences in classification performance across support types can be detected (Fisher's exact test, $\chi^2(3, N = 566) = 4.563$, $p = .206$). However, an exploratory analysis revealed that students who downloaded files (n = 35) assessed overall correctness significantly more accurately (ACC = .83, TPR = 1, TNR = .71) than those who did not (ACC = .59, TPR = .92, TNR = .32; $\chi^2(1, N = 566) = 8.076$, $p = .004$, $V = .12$) with a small effect. Also, restricting the *test results* condition to task 3, where students had access to unit tests, there still is no significant difference (ACC = .71, TPR = .98, TNR = .4).

Finally, there are no notable differences when correctness is already assumed with ratings ≥ 3.

5.2 Students' Confidence in their Assessment

The majority of students rated their confidence in their evaluation with the highest possible score (median m = 4, arithmetic mean $\bar{x} = 3.67$). Throughout the semester (cf. Table 1), students' mean confidence rating ranges from 3.6 to 3.72, while the median consistently remained 4, indicating no observable trend.

Similarly, across the four support types (cf. Table 4), the median confidence is constantly 4, with mean values ranging from 3.59 to 3.72. A Kruskal-Wallis test indicates that there is no difference across the four conditions ($\chi^2(3, N = 662) = 2.026$, $p = .560$).

5.3 Feedback Length

The median length of all reviews is 20 words ($\bar{x} = 33$), corresponding to 128 characters ($\bar{x} = 238$). Eleven reviews consist of exactly

Table 1: Summary of tasks and peer review metrics (A: peer review assignments, R: completed reviews, U: unique students delivering peer reviews, S: unique submissions, TR: reviews supported by test results, MS: reviews supported by model solutions, CL: reviews supported by checklists, SC: syntactic correct submissions in %, FC: fully correct submissions in %, C: average self-reported confidence, W: average/median number of words, T: review time in seconds, DL: downloaded files)

No. Exercise Sheet & Assignment	#A	#R	#U	#S	#TR	#MS	#CL	SC	FC	C$\bar{x}$	W$\bar{x}$	Wm	T$\bar{x}$	Tm	#DL
4 Count chars 't', 'e', 'l' in array	330	289	159	176	75	140	–	.90	.49	3.67	33	20	605	240	18
8 Guess number (binary search)	154	138	79	85	–	32	49	.91	.51	3.72	40	24	488	264.5	7
11 Java Collections & OO modeling	80	74	43	46	–	19	32	.87	.17	3.61	41	18	553	274	4
12 Custom Exceptions (ISBN10/13 checks)	76	65	38	41	7	25	24	.90	.34	3.60	31	19	378	181	6
13 Multi threaded counter	55	45	26	30	5	17	20	.97	–	3.64	40	18	523	221	4
14 Tic-Tac-Toe (incl. Swing GUI)	24	11	6	8	–	8	–	1.00	–	3.64	28	9	334	135	4
Overall	719	622	168	386	87	241	125	.91	.43	3.67	36	20	539	243	43

Table 2: Students' syntactic correctness assessment performance (TP: true positives, FP: false pos., FN: false negatives, TN: true negatives, TPR: true positive rate, TNR: true negative rate, ACC: accuracy, V: Cramér's V, *: p = .001, **: p < .001)

Support condition	TP	FP	FN	TN	ACC	TPR	TNR	V
No support**	146	5	7	11	.93	.95	.69	.609
Tests result**	79	1	3	4	.95	.96	.80	.653
Model solution*	206	14	14	7	.88	.94	.33	.270
Checklist**	104	6	4	11	.92	.96	.65	.643
Overall**	535	26	28	33	.91	.95	.56	.502

Table 3: Students' overall correctness assessment performance, only tasks 4–12 (Legend cf. Table 2)

Support condition	TP	FP	FN	TN	ACC	TPR	TNR	V
No support**	69	60	5	29	.60	.93	.33	.316
Tests result**	42	24	1	15	.70	.98	.38	.455
Model solution*	85	85	10	36	.56	.89	.30	.233
Checklist**	34	37	4	30	.61	.89	.45	.352
Overall**	230	206	20	110	.60	.92	.35	.317

Table 4: Self-reported confidences in the assessments, feedback text length, and time needed to provide feedback (#: number of peer reviews, $\bar{x}$: arithmetic mean, m: median, C: confidence, W: number of words, T: review time in s)

Support condition	#	C$\bar{x}$	Cm	W$\bar{x}$	Wm	T$\bar{x}$	Tm
No support	169	3.70	4	33	18	320	169
Test results	87	3.62	4	30	20	397	231
Model solution	241	3.70	4	39	21	768	269
Checklist	125	3.59	4	37	22	490	286
Overall	622	3.67	4	36	20	539	243

one word, 42 reviews contain more than 100 words (6.7 %), and 11 contain more than 200 words (1.8 %). The longest feedback text comprises 448 words and includes a complete model solution with additional comments. The median number of words seems to decrease over the course of the semester (cf. Table 1, Spearman's $\rho(4) = -.841$, p = .0036), whereas no such trend is observable for the mean number of words ($\rho(4) = -.486$, p = .329).

Across the four support types, median feedback lengths range from 18 to 22 ($\bar{x}$ between 30 and 39, cf. Table 4). A Kruskal-Wallis test indicates no difference in the feedback text length across the four support types ($\chi^2(3, N = 622) = 4.399$, p = .221).

5.4 Time Needed to Provide Feedback

On average, students needed about 9 minutes ($\bar{x}$ = 539 s, m = 243 s ≈ 4 min.) with a minimum of 10 s and a maximum of 59,765 s (one outlier). Review time seems to vary slightly across the tasks (cf. Table 1), but this variation is not statistically significant ($\rho(4) = -.714$, p = .111). There is a positive correlation between the review time and number of words in the feedback ($\rho(620) = .560$, p < .001) with a large effect size. Additionally, there is a statistically significant difference between the review time depending on whether students correctly assessed overall correctness (m = 272 s) or not (m = 194 s; UTest: Z = 56121, p < .001) with a small effect size (r = .16, [19]), but not for syntactic correctness assessment (m = 243 s vs. m = 219 s).

Table 4 reports median and average review durations for each support type. The *model solution* condition has the largest mean due to the mentioned outlier ($\bar{x}$ = 768 s, m = 269 s). The *checklist* condition has the largest median ($\bar{x}$ = 490 s, m = 286 s). The shortest durations are observed for the *no support* condition ($\bar{x}$ = 320 s, m = 169 s). The *test result* condition lies in the middle ($\bar{x}$ = 490 s, m = 231 s). A Kruskal-Wallis test indicates a statistically significant difference in review time across the four support types ($\chi^2(3, N = 622) = 16.594$, p = .001). Post-hoc comparisons using Dunn's method with a Bonferroni correction indicate that the median review times are significantly higher when *checklists* (p = .016) or *model solutions* (p = .001) were provided compared to *no support*. However, there are no significant differences between the other support types (p ≥ .371).

5.5 Survey

A total of 40 students responded to at least one item in the post-semester evaluation survey. In the following, only students who took part in peer review at least once are considered (n = 32).

The majority of students (82 %) were 18–21 years old. Half of the students (50 %) identified as male, 38 % as female, and two skipped that question. Almost all students were computer science majors

Table 5: Overview of survey results (ratings from 1 to 5 best)

Item	n	$\bar{x}$	m
The feedback from my fellow students is helpful to me.	24	3.33	4
Feedback helps me discover other ways of solving problems.	23	3.96	4
Peer feedback has motivated me to work on the exercises regularly.	23	3.00	3
I feel that I have helped other students.	24	3.04	3
Overall, I found peer feedback valuable.	23	3.61	4
The model solutions were particularly helpful for giving peer feedback.	22	3.91	4
The checklists were particularly helpful for giving peer feedback.	21	3.86	4
The test results were particularly helpful for giving peer feedback.	20	3.75	4

(91 %), one student reported to be a CS minor, and two students did not answer this question. The majority of respondents (56 %, n = 18) participated in at least 5 peer review rounds (m = 5, $\bar{x}$ = 6.1).

The overall questionnaire consisted of five parts, whereas only the peer review and general parts are analyzed and summarized here. Table 5 shows the results of the closed questions that had to be answered on a Likert-scale from 1 to 5 (1: "do not agree at all" to 5: "fully agree") with an extra option for "cannot judge/do not want to answer." The majority of students found peer feedback quite valuable (m = 4) and also helpful in discovering alternative problem-solving approaches (m = 4). However, many students were quite unsure whether they had helped others (m = 3) and whether peer review had motivated them to work on exercises regularly (m = 3). When the given answers are compared across the two groups who took part in at least five peer review rounds and those who did not, there are no statistically significant differences.

The majority of students considered having a sample solution (m = 4), checklist (m = 4), and test results (m = 4) all useful in guiding their feedback. To differentiate between the three support types, students were also asked which support type for peer review they would prefer if only one were available and to argue why. This item was answered by 23 students: Six students could not decide (26 %), three students preferred the checklists (13 %), four students preferred the test results (17 %), and ten students preferred the model solutions (43 %). When restricted to students who took part in at least five peer review rounds, n = 17 students answered this item: Three students could not decide (18 %), three students preferred the checklists (18 %), two students preferred the test results (12 %), and the majority of nine students preferred the model solutions (53 %).

Fifteen students provided a reason for their preference (evaluated using thematic analysis to find common reasons per preference, cf. [2]): The three students who preferred the checklist, all said that it provides a quick overview of the most relevant aspects of the task – one student added that it is superior to the model solution, because s/he "noticed errors more frequently that would only have been apparent with a sample solution if I had looked very closely". Three students preferred the tests. One student argued that s/he does not need the checklists as they are pretty self-explanatory and that s/he does not "need a model solution, as it can narrow your perspective, but the test result, i.e., automated, is helpful for validating your own thought processes (do I get the same result, is there a syntax error somewhere, etc.)". The remaining two students contended that test results offer the most effective means for evaluating code functionality. Seven of the ten students who preferred the model solution reported higher certainty (about whether their own solution or feedback is correct). Notable additions are "it provides a full understanding even if I could not solve the task", "if you are unsure about your own solution, you then have to wait until you receive feedback yourself before you can give feedback.", peer review without a model solution would be "impossible", because otherwise they has no basis for it, and "You can see everything from that [the model solution]. The test results are helpful, but not as much as the model solution itself." Finally, one student noted that his/her "own programming skills are limited" and s/he "often had difficulty providing appropriate feedback" that "goes beyond the obvious." Finally, one student who could not state a clear preference argued "The checklist was helpful for checking whether you had considered everything, but many people didn't know what to write about it. Model solutions weren't very common, so I can't choose them as the most important aid. [...] if I had to choose, I would rather choose one than have none at all."

6 Discussion

To address RQ1, students' performance in assessing syntactic and functional correctness, as well as their self-reported confidence, were analyzed across four support conditions. Overall, students' accuracy was about one-third higher for assessing syntactic correctness than for overall correctness. This difference is unsurprising, as syntactic correctness is a prerequisite for functional correctness, and functional errors may additionally arise from unconsidered edge cases. Across both syntactic and functional correctness, students were more accurate at identifying correct submissions than at identifying incorrect ones. This finding is consistent with prior work in Java-based settings [27], but contrasts with results reported for Haskell [10]. Given that the majority of submissions were syntactically correct, high true positive rates and overall accuracy could be achieved even by consistently rating submissions as correct. However, only 43 % of submissions were fully correct, while the true positive rate for functional correctness remained close to 90 %, indicating that students still frequently overestimated correctness.

Support types affected error detection differently. For syntactic correctness, the highest true negative rate (.8) was observed when *test results* were provided, whereas the lowest (.33) occurred when *model solutions* were available. A similar pattern emerged for overall correctness, where *checklists* yielded the highest true negative rate (.45) and *model solutions* the lowest (.3). This suggests that *checklists* better support the identification of potential issues, even when items are formulated at a higher level of abstraction. In contrast, *model solutions* may cognitively overload students and/or shift their focus toward overall structure or outcomes, potentially reducing attention to specific errors or edge cases. This interpretation is consistent with the absence of performance advantages for *model solutions*, despite students' subjective preference for them.

Unexpectedly, false positives and false negatives occurred even when *test results* were provided. This indicates difficulties in interpreting test output, limited trust in automated feedback, or a

tendency to judge submissions as "good enough" despite failing strict automated tests. This warrants further investigation.

It was hypothesized that students' self-reported confidence would be higher when specific support was provided. Contrary to this expectation, mean confidence was highest (3.7) in both the *no support* and *model solution* conditions and lowest for *checklists* (3.59), although these differences are not statistically significant. Survey responses suggest that *model solutions* reduce perceived uncertainty, yet this subjective impression is not reflected in assessment performance or confidence data, indicating a potential mismatch between perceived and actual effectiveness.

Overall, the lack of statistically significant performance differences across support conditions was unexpected. Given the relatively small sample sizes per condition, particularly for *test result* support, future studies with larger cohorts and more review instances are necessary to draw more robust conclusions.

Regarding RQ2, students spent the most time on reviews when *checklists* were provided, quite closely followed by *model solutions*. This is understandable, given the effort required to read the checklist or study the given solution. Students without any support spent the least time on reviews (with overall review durations comparable to prior work [27]), yet this remains counterintuitive. It potentially indicates cognitive disengagement rather than efficiency. Students without support might have been cognitively overloaded or unsure how to evaluate correctness and, therefore, engaged less deeply with the submissions. While longer review times do not necessarily imply higher quality, they may indicate deeper cognitive processing.

The median length of feedback texts (20 words) is comparable to values reported in related work (m = 14 words [10], m = 13 words [27], $\bar{x}$ = 25.25 tokens [8]). Feedback tended to be longest when *checklists* were provided, though this difference is not statistically significant. Future research should qualitatively analyze the content to examine how checklist items are reflected in students' comments.

Regarding RQ3, most students who answered perceived all support types as helpful, with a preference for *model solutions*. Students frequently cited reduced uncertainty as a reason for this preference, although this is not supported by performance or confidence data (only a tendency is visible). This suggests that other factors, such as knowing "the correct" solution or perceived usefulness, which do not manifest in accuracy, play a more important role for students.

Despite accuracy benefits, *test results* likely limit opportunities for critical engagement, which are most important for learning in peer review (cf. [37]). Moreover, tests are costly to design, difficult to make comprehensive, and unsuitable for tasks with design freedom. While students seem to prefer *model solutions*, overreliance on them may encourage treating a single solution as authoritative, thereby limiting creativity and exploration of alternative approaches.

Overall, educators should use *checklists* (also recommended in [17] for preparing students for peer review). Checklists can guide attention without prescribing solutions, though the optimal level of detail remains an open question. Furthermore, a potential middle ground between students' wishes and effectiveness may be the use of *model solutions* only in specific cases when both the submissions to review and the student's submission are incorrect. Future work should explore adaptive or fading support mechanisms and different interface designs (e.g., integrated code execution for manual testing) to better balance guidance and critical engagement.

7 Limitations

The study was conducted in an authentic setting at a single German university in a single course using the programming language Java with a strict exclusion policy to prevent read-only participation and an overall declining participation throughout the semester. Furthermore, participation in peer review was voluntary. Hence, results may be biased by self-selection. Survey participation was very low (n = 32, 19 %). All reviews were analyzed as being independent.

The required time is not a perfect measure as it may overestimate the actual time needed (e.g., if students open the form, collect a coffee, and submit later). Furthermore, this measure does not account for students who open the review form, do some inspection, leave it, and reopen it later to submit their review. Yet, it provides insight, especially into whether students rush through the reviews. The number of words may also overestimate the actual length of the feedback text, especially when dealing with source code. Yet it provides insight into the typical length, as students often write very short reviews containing only a few words. Also, it says nothing about the quality. As students were not supervised while writing peer feedback, it is unknown whether generative AI was used for programming or authoring the feedback, or whether others helped, but the results are comparable to related work. A bug in the unit test resulted in only one assignment having unit test results, potentially limiting generalization. Not all students may have experienced all conditions, because these were assigned randomly in each round.

8 Conclusions and Future Work

This paper quantitatively compared four support types for peer code review—*no support*, *automated test results*, *model solutions*, and *checklists*—in an introductory programming course. The results indicate that students were generally more accurate at correctly identifying correct submissions than incorrect ones, with an overall assessment accuracy of 91 % for syntactic and 60 % for overall correctness. However, there seem to be no statistically significant differences in students' overall performance in assessing the syntactic and functional correctness of submissions or review length ($\bar{x}$ = 20 words) across the four support types. Yet, *test results* and *checklists* seem to slightly increase students' performance in identifying incorrect submissions, whereas *model solutions* seem to have a slight negative effect on identifying errors in submissions. Although students reported in a survey that they prefer model solutions because they reduce uncertainty, this preference could not be confirmed by self-reported confidence during review (no significant difference). The review duration was shortest when *no support* was provided (likely indicating cognitive disengagement instead of efficiency), and longest when a *checklist* or a *model solution* was provided. A statistically significant difference in review duration across support types was found, particularly when *checklists* or *model solutions* are provided, the review duration is longer than in the *no support* condition. Overall, students appreciate any type of support, and *checklists* seem to be the support type to recommend.

Further research is necessary on peer code review, such as using a larger pool of assignments (especially with unit tests), a qualitative analysis of feedback texts, fading support for peer reviews, or a think-aloud study on how students use different types of support.

I thank all students who participated and also shared their data!

References

[1] Ralf Bender, Stefan Lange, and Andreas Ziegler. 2007. Wichtige Signifikanztests. *Deutsche Medizinische Wochenschrift* 132 (2007), e24–e25. doi:10.1055/s-2007-959034

[2] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. doi:10.1191/1478088706qp063oa

[3] Tamaike Brown, Mourya Reddy Narasareddygari, Maninder Singh, and Gursimran Walia. 2019. Using Peer Code Review to Support Pedagogy in an Introductory Computer Programming Course. In *2019 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–7. doi:10.1109/FIE43999.2019.9028509

[4] Michael de Raadt, David Lai, and Richard Watson. 2007. An Evaluation of Electronic Individual Peer Assessment in an Introductory Programming Course. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research* (Koli National Park, Finland) *(Koli Calling '07, Vol. 88)*. Australian Computer Society, Inc., AUS, 53–64.

[5] Dominik Dolezal, Alexandra Posekany, Christoph Roschger, Gottfried Koppensteiner, Renate Motschnig, and Robert Pucher. 2018. Person-Centered Learning using Peer Review Method – An Evaluation and a Concept for Student-Centered Classrooms. *International Journal of Engineering Pedagogy (iJEP)* 8, 1 (feb 2018), 127–147. doi:10.3991/ijep.v8i1.8099

[6] Eduard Frankford, Tobias Antensteiner, Michael Vierhauser, Clemens Sauerwein, Vivien Wallner, Iris Groher, Reinhold Plösch, and Ruth Breu. 2025. A Survey on Feedback Types in Automated Programming Assessment Systems. *ACM Transactions on Computing Education* 26, 1 (Dec. 2025), 1–35. doi:10.1145/3773911

[7] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. 2003. *Fundmentals of Software Engineering*. Pearson Higher Edication.

[8] 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. doi:10.1080/02602938.2014.893418

[9] Stephanie J. Hanrahan and Geoff Isaacs. 2001. Assessing Self- and Peer-assessment: The students views. *Higher Education Research & Development* 20, 1 (may 2001), 53–70. doi:10.1080/07294360123776

[10] Niels Heller and Francois Bry. 2019. Organizing Peer Correction in Tertiary STEM Education: An Approach and its Evaluation. *International Journal of Engineering Pedagogy (iJEP)* 9, 4 (2019), 16–32. doi:10.3991/ijep.v9i4.10201

[11] Christopher Hundhausen, Anukrati Agrawal, Dana Fairbrother, and Michael Trevisan. 2009. Integrating Pedagogical Code Reviews into a CS 1 Course: An Empirical Study. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (Chattanooga, TN, USA) *(SIGCSE '09)*. ACM, 291–295. doi:10.1145/1508865.1508972

[12] Theresia Devi Indriasari, Andrew Luxton-Reilly, and Paul Denny. 2020. A Review of Peer Code Review in Higher Education. *ACM Transactions on Computing Education* 20, 3 (sep 2020), 1–25. doi:10.1145/3403935

[13] Amandeep Kaur and Robin Kumar. 2015. Comparative analysis of parametric and non-parametric tests. *Journal of computer and mathematical sciences* 6, 6 (2015), 336–342.

[14] C.F. Kemerer and M.C. Paulk. 2009. The Impact of Design and Code Reviews on Software Quality: An Empirical Study Based on PSP Data. *IEEE Transactions on Software Engineering* 35, 4 (jul 2009), 534–550. doi:10.1109/tse.2009.27

[15] Nafiseh Taghizadeh Kerman, Seyyed Kazem Banihashem, Mortaza Karami, Erkan Er, Stan van Ginkel, and Omid Noroozi. 2023. Online peer feedback in higher education: A synthesis of the literature. *Education and Information Technologies* 29, 1 (Nov. 2023), 763–813. doi:10.1007/s10639-023-12273-8

[16] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2018. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM Transactions on Computing Education* 19, 1, Article 3 (Sept. 2018), 43 pages. doi:10.1145/3231711

[17] Roxane Koitz-Hristov. 2025. Peer Code Review Methods: An Experience Report from a Data Structures and Algorithms Course. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025)*. ACM, 610–616. doi:10.1145/3641554.3701799

[18] Xiao Liu and Gyun Woo. 2020. Applying Code Quality Detection in Online Programming Judge. In *Proceedings of the 2020 5th International Conference on Intelligent Information Technology (ICIIT 2020)*. ACM, 56–60. doi:10.1145/3385209.3385226

[19] Andrey Lovakov and Elena R. Agadullina. 2021. Empirically derived guidelines for effect size interpretation in social psychology. *European Journal of Social Psychology* 51, 3 (April 2021), 485–504. doi:10.1002/ejsp.2752

[20] Kasia Muldner, Jay Jennings, and Veronica Chiarelli. 2022. A Review of Worked Examples in Programming Activities. *ACM Transactions on Computing Education* 23, 1 (Dec. 2022), 1–35. doi:10.1145/3560266

[21] David Nicol, Avril Thomson, and Caroline Breslin. 2013. Rethinking feedback practices in higher education: a peer review perspective. *Assessment & Evaluation in Higher Education* 39, 1 (may 2013), 102–122. doi:10.1080/02602938.2013.795518

[22] Ryan Rybarczyk and Lingma Acheson. 2019. Interactive Peer-Led Code Reviews In CS2 Curricula. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 659–665. doi:10.1145/3287324.3287442

[23] Joanna Smith, Joe Tessler, Elliot Kramer, and Calvin Lin. 2012. Using peer review to teach software testing. In *Proceedings of the ninth annual international conference on International computing education research*. ACM, 93–98. doi:10.1145/2361276.2361295

[24] Xiangyu Song, Seth Copen Goldstein, and Majd Sakr. 2020. Using Peer Code Review as an Educational Tool. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20)*. ACM, 173–179. doi:10.1145/3341525.3387370

[25] Saikrishna Sripada, Y Raghu Reddy, and Ashish Sureka. 2015. In support of peer code review and inspection in an undergraduate software engineering course. In *2015 IEEE 28th conference on software engineering education and training*. IEEE, 3–6. doi:10.1109/CSEET.2015.8

[26] T. Stalhane, C. Kutay, H. Al-Kilidar, and R. Jeffery. 2004. Teaching the process of code review. In *2004 Australian Software Engineering Conference. Proceedings.*, Vol. 5. IEEE, 271–278. doi:10.1109/aswec.2004.1290480

[27] Sven Strickroth. 2023. Does Peer Code Review Change My Mind on My Submission?. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2023)*. Association for Computing Machinery, 498–504. doi:10.1145/3587102.3588802

[28] Sven Strickroth and Imen Azaiz. 2025. Qualitative Analysis of Peer Reviews of a Large Introductory Programming Course. *Computer Science Education* 36, 2 (2025), 354–374. doi:10.1080/08993408.2025.2450587

[29] Sven Strickroth and François Bry. 2022. The Future of Higher Education is Social and Personalized! Experience Report and Perspectives. In *Proceedings of the 14th International Conference on Computer Supported Education - Volume 1: CSEDU*, Vol. 1. INSTICC, SciTePress, 389–396. doi:10.5220/0011087700003182

[30] Sven Strickroth, Hannes Olivier, and Niels Pinkwart. 2011. Das GATE-System: Qualitätssteigerung durch Selbsttests für Studenten bei der Onlineabgabe von Übungsaufgaben?. In *Tagungsband der 9. e-Learning Fachtagung Informatik (DeLFI)*. Gesellschaft für Informatik e.V., Bonn, Germany, 115–126. https://dl.gi.de/handle/20.500.12116/4740

[31] Sven Strickroth and Michael Striewe. 2022. Building a Corpus of Task-based Grading and Feedback Systems for Learning and Teaching Programming. *International Journal of Engineering Pedagogy (iJEP)* 12, 5 (Nov. 2022), 26–41. doi:10.3991/ijep.v12i5.31283

[32] John Sweller. 1994. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction* 4, 4 (1994), 295–312. doi:10.1016/0959-4752(94)90003-5

[33] Harald Søndergaard and Raoul A. Mulder. 2012. Collaborative learning through formative peer review: pedagogy, programs and potential. *Computer Science Education* 22, 4 (2012), 343–367. doi:10.1080/08993408.2012.728041

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

[35] Scott Alexander Turner, Manuel A. Pérez-Quiñones, and Stephen H. Edwards. 2018. Peer Review in CS2. *ACM Transactions on Computing Education* 18, 3 (sep 2018), 1–37. doi:10.1145/3152715

[36] Yanqing Wang, LI Yijun, Michael Collins, and Peijie Liu. 2008. Process improvement of peer code review and behavior analysis of its participants. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. ACM, 107–111. doi:10.1145/1352135.1352171

[37] Yong Wu and Christian D. Schunn. 2023. Passive, active, and constructive engagement with peer feedback: A revised model of learning from peer feedback. *Contemporary Educational Psychology* 73 (April 2023), 102160. doi:10.1016/j.cedpsych.2023.102160