Students' Perceptions of Features and Qualities in Peer Code Assessment Tools

Somayeh Bayat Esfandani[1][0000−0002−3558−2202]
and Trond Aalberg[1][0000−1111−2222−3333]

Norwegian University of Science and Technology
Department of Computing Education
somayeh.b.esfandani@ntnu.no
trond.aalberg@ntnu.no

Abstract. Peer code assessment (PCA) empowers computer science students by enhancing their learning and equipping them with practical skills for professional work. However, instructors often face a scarcity of well-designed tools customized for this learning activity. The question addressed in this paper is how a peer code assessment tool should be designed to support students' programming learning experience. A case study was conducted in the context of a code reviewing task, employing interviews and observation as data generation methods. Research aimed to identify features that enhance student learning and uncover essential qualities of peer code assessment tools from students' perspectives. Informants considered inline comments, general comments, rubrics, threads, and code editor functionalities essential. They also highlighted the importance of utilizing a customized, user-friendly tool with a step-by-step process. The Self-Regulated Learning conceptual theory has been used as a theoretical lens and we have identified qualities and features that will improve the learning experience by increasing students' motivation and enabling them to follow their learning strategies. The findings can be used as design principles for developing peer code assessment tools.

Keywords: Peer Code Assessment , Code Review · Self-Regulated Learning · Peer Assessment.

1 Introduction

To enter the competitive software engineering market, computer science students need not only to master programming but also need to develop professional skills such as understanding peers' codes, inspecting bugs, evaluating the quality of a code, and giving and receiving feedback [1, 2]. Peer assessment [3] can assist students in developing these skills and getting prepared for the competitive software engineering market. It can be defined as "an arrangement in which individuals consider the amount, level, value, worth, quality, or success of the products or outcomes of learning of peers of similar status" [4]. Peer code assessment (PCA) can be seen as a sub-discipline of peer assessment where the artefact to assess

and comment on is programming code. PCA enables students to identify code defects, share ideas, explore alternative solutions, develop giving and receiving feedback, and learn from helpful feedback. Furthermore, it improves the efficiency of the assessment process and reduces instructors' workload [5].

Digital tools can facilitate the PCA practice, enhance learning, and allow for adapting this practice in large-scale classes [6–8]. However, the design of PCA tools requires sufficient knowledge about design requirements to ensure a straightforward learning experience [9, 10]. Existing tools that can be used for PCA include commercial learning management tools (LMS), test based assessment tools (TBA), generic peer assessment tools and industrial code review tools. However, LMS systems are not designed for code review, TBA tools do not support peer assessment, peer assessment tools do not support code level comments, and industrial code review tools are impractical for educational environments due to a lack of essential features such as grading, rubrics, etc. [10, 11] More specific purpose educational PCA tools or prototypes mentioned in prior studies are unfortunately limited [12], no longer available, such as PeerSpace, EduPCR, Eliph, or not maintained for years, such as MyPeerReview, CaptainTeach, and OSBLE. Therefore, there are limited tools available for educational PCA use [13]. Moreover, there are currently insufficient knowledge of how the new generation of educational tools that supports peer assessment, fulfill users' needs [14, 15]. Consequently, a significant knowledge gap exists regarding relevant information for selecting or designing appropriate educational PCA tools.

This study aims to provide insights into essential functional requirements, referred to as *features*, and non-functional requirements, referred to as *system qualities* [16] of PCA tools, from students' perspectives. So, the overarching question is, how to design PCA tools to support students' programming learning experience? This question is further divided into two sub-questions:

- RQ1: What features support students' learning while giving and receiving feedback?
- RQ2: What qualities do students consider essential in a peer code assessment tool?

An exploratory case study was utilized as a research strategy to answer these research questions. Semi-structured interviews and observations were used as data-gathering methods. An open coding technique was used for the transcriptions, and findings were analysed using the self-regulated learning (self-regulated learning) theory. Results offer fundamental insights into essential functionalities and qualities of a PCA tool from students' perspectives. The study findings can aid in establishing design principles or guide the design process of PCA tools tailored to the educational environment.

2 Background

2.1 Peer Code Assessment (PCA) Tools

Peer code review is a widely adopted best practice for ensuring code quality in the software industry. It also helps novice developers align with the software

quality standards of their community [6, 17]. Both general and inline comment features are common in industrial code review tools [18]. Inline comments allow reviewers to highlight specific issues with specific line(s) of code, while general comments address broader concerns about the overall code [19]. Familiarizing students with code review practices during their education benefits both students and the industry by preparing them for real-world development processes [20]. To effectively integrate peer code review into educational settings, assessment tools should incorporate commonly used features such as inline and general comments, enabling students to experience the same code review process as the industry. In an educational setting, it is also relevant to include features commonly found in more general peer assessment tools. The use of predefined rubrics can guide the students in constructing relevant feedback. Solutions for reacting upon feedback they receive by flagging or discussion threads, can contribute to trust and further learning.

To integrate peer code review in the educational settings, various digital tools with different aims and features have been developed and explored. For instance, PeerSpace fostered peer networks by providing chat, discussion, and competition features [12]. EduPCR, which was designed based on the incentive strategy model, offered shareable comments and iterative reviews [21], MyPeerReview was developed to offer a usable and learnable tool for supporting multiple program files [13], CaptainTech facilitated assessment of multiple deliverables at various stages of in-progress assignments [22], Eliph aimed to enhance problem-solving by offering code history visualization [10]. OSBLE was designed to support online and face-to-face code reviews [23]. The reviewed studies didn't consider students' perceptions in the design process of the tools, which is crucial for designing a useful tool [24]. However, a study by Alkhalifa and Devlin [25] took into account the students' perceptions of designing their prototype, but they solely focused on rubrics. Similarly, reviewed studies concentrated on specific aspects of PCA, providing features to demonstrate the impact of those particular factors. Not to mention that these tools are either unavailable or not maintained for use.

2.2 Self-Regulated Learning (SRL)

self-regulated learning is defined as students' capability to actively participate in their learning process metacognitively, motivationally, and behaviourally. It encompasses three phases: forethought, performance, and self-reflection. During the forethought phase, learners set goals and devise plans to attain them. Motivation plays a critical role here; without it, success in this phase is greatly hindered. In the performance phase, students follow their task strategies to accomplish the learning task. In the self-reflection phase, learners assess their performance, often by comparing themselves to peers or receiving feedback, and adapt new learnings by reacting to them [26]. self-regulated learning does not occur automatically, but it may be improved in an "enabling environment" that provides support and feedback from teachers and peers, appropriate digital tools, and the chance to practice it [27]. Peer assessment enhance self-regulated learning, allowing students to actively participate in their learning while practicing self-evaluation,

reaction, and feedback exchange [28]. Therefore, self-regulated learning is used as a theoretical lens in this study.

3 Research Method

3.1 Research Strategy

An exploratory case study research strategy has been adopted in this research, as it is recommended for contemporary events investigations when related behaviors are not manipulable. Experiments were not chosen as they manipulated associated behaviors and required a data-driven hypothesis, which wasn't available due to inadequate literature. Furthermore, we aim to gain in-depth insights, which is impossible by the survey method [29, 30].

3.2 Case Description

In the study, we used the CodeGrade (https://www.codegrade.com) tool as case, a commercial test based assessment tool that also includes support for peer feedback [14]. CodeGrade provided key features for the study, including inline comments, rubrics, and more. Additionally, it offered a free trial, making it accessible for our research. Screenshots of the tool was used to create a high-fidelity prototype to simplify observation sessions. This decision was made to prevent prolonged observation sessions due to tool settings. The prototype, built in Figma, connected all relevant pages of the scenario tasks, including assignment criteria, uploading assignments, giving feedback, and receiving feedback, ensuring it closely replicated the real tool. To mitigate unfamiliarity, we ensured informants were exposed to the tool before the observation sessions.

3.3 Data Gathering

To ensure triangulation in data generation [29] , we opted for systematic observation of informants and semi-structured interviews. Systematic observations allowed us to observe their actions and challenges while using the tool. This approach provides a deeper understanding of participants' actions, insights, and perceptions. During the study sessions, the voices and screens of informants were recorded to make transcription and review easier. Since students are the primary users of PCA tools, their perceptions serve as an essential source of knowledge [24]. Therefore, four (female) computer science students were recruited for the study; they worked as research assistants in the Excited SFU. Table 1 shows the informants' information. They signed "informed consent" and had withdrawal rights. Informants' choices were influenced by their relevance to the research topic and the ease of recruitment.

	Year of Education	PCA experience	Programming skill	Confidence in Feedback Giving
I1	fifth	Yes	Very good	Very Good
I2	fifth	Yes (Forms)	Average	Good
I3	fifth	Yes (BlackBoard)	Lower than average	Good
I4	third	Yes (EduFlow)	Lower than average	Good

Table 1. Research informants' information

3.4 Data Analysis

Qualitative analysis was used in this study; interview and observation records were transcribed, and as the study was in the preliminary stage, the open coding method opted to identify key themes and categories [31]. Coding was facilitated using Microsoft Excel, where repetitive patterns and words associated with the features and qualities of the tool were identified and organized into codes, such as: "motivation", "inline comment", "general comment", "rubric", "strategy", "thread", "usability", and so forth.

4 Findings

The study findings are reported in four sections: mapping findings to self-regulated learning theory, feedback-giving features, feedback-receiving features and tool qualities.

4.1 Mapping Findings to self-regulated learning Theory

The relationship between study findings and self-regulated learning is depicted in Figure 1. Boxes A1 and A2 represent features that help students adhere to their task strategies associated with the performance phase of self-regulated learning. Box B1 illustrates the qualities of tools that enhance students' motivation linked to the forethought phase of self-regulated learning, while Box B2 demonstrates the ordering effect of grading on the self-reflection phase.

4.2 Feedback-giving Features

Informants' feedback-giving strategy starts by running the code and assessing output to identify errors; they expect the tool to provide a code editor feature for running code, highlighting syntax errors, and tracing code for computational and logical issues. They also perceived the auto-test feature as useful during observation sessions. *"I3: The tool can support me by being able to run the code, so I don't have to bring the code into another editor to run it."*

Participants mentioned that they may use Artificial Intelligence (AI) tools to find bugs or issues they didn't identify and ensure the correctness of the feedback or rephrase feedback. However, some participants believe that using AI for writing the whole feedback contradicts the purpose of PCA. *"I3: I think that's*

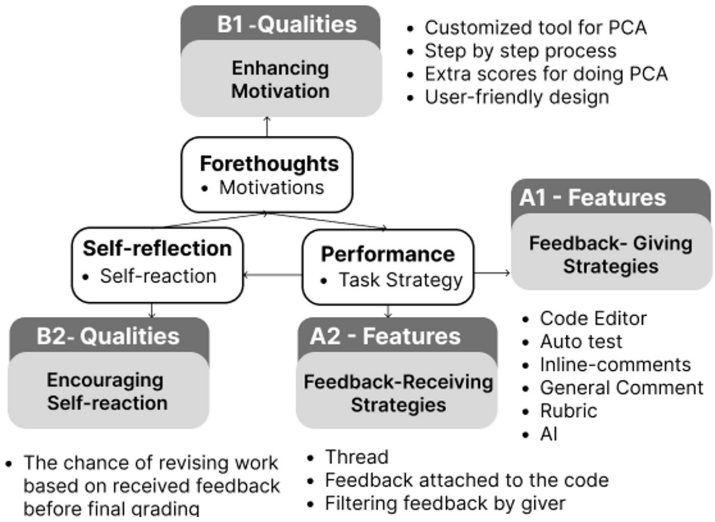

Fig. 1. Mapping the self-regulated learning theory phases to the study findings.

kind of contradicting the purpose of the peer feedback. If you wanted feedback from AI, you could do it yourself."

Informants characterized inline comments as specific, detailed comments highlighting minor bugs and errors. They perceived inline comments as more understandable and clearer, making it easier for them to solve the problems. However, they acknowledge minor learning from inline comments. *"I1: Inline comments are more like specific things on the actual code line, which you can learn from, but I feel you don't learn the big concepts from inline comments".*

Regarding general comments, they perceived them as holistic comments on the entire assignment, highlighting the generic issues in their code, such as readability, performance, and runtime. Furthermore, they perceived the learning impact of these comments as substantial. *"I2: I think that it would be more valuable for my knowledge if I received some good comments or feedback on the rubrics or general comments to use further, not just on this project".* On the other hand, rubrics are perceived as specific, formal, and easy-to-understand overall comments on students' assignments. Informants viewed it as categorized feedback on different topics, simplifying providing feedback. *"I4: A rubric is categorized according to different topics in the assignments. So, I think it's easy to understand".*

4.3 Feedback Receiving Features

Informants preferred to get the feedback attached to their code. This approach allowed them to see the code and feedback together, reducing needed clicks and facilitating a more straightforward review.

Informants value the thread feature, which allows them to respond to comments, ask questions, express disagreement, or say thank you. *"I3: I may reply to the feedback to thank you or ask what you meant about this. So, I don't feel like I'm maybe writing to a wall, and you understand that you don't get any response back".*

4.4 Tool Qualities

Informants found that giving feedback becomes more straightforward when the tool breaks down the process into smaller steps. *"I2: The tool makes the process easier because it kind of breaks down the review process, which is nice to have in smaller steps".* They find customized tools for PCA more beneficial, feeling that instructors considered their needs, resulting in higher motivation. *"I1: It feels more useful because it's not just something the teacher tells us to do; I think it's because they've set up this tool to actually work with the assignment. I think this tool will help me, and I have been motivated to do it".*

Informants believe that peer code assessment as a mandatory task without any extra scoring incentive will not be taken seriously by students and they will not put sufficient effort into it, leading to low-quality feedback with minimal learning impact. *"I3: People don't really take it too seriously because They get it approved as long as they do it".* They also believe that receiving feedback after getting an assignment score discouraged them from modifying their code based on the feedback. *"I1: I guess my submission is already submitted, so. I would revise it if this was like a draft".*

Informants believe that utilizing a user-friendly and accessible tool that eliminates downloading and opening different files will enhance their motivation for performing PCA tasks. During observation sessions, they expressed dissatisfaction with usability problems and emphasized the importance of a self-explanatory design for smoother task execution. *"I3: if the system is kind of easy to use, then it's like easier to give feedback as well, and if it's like a complicated system, it's probably like not really motivating to use it".*

5 Discussion

This study aimed to provide insights into the design or selection of a PCA tool to support the programming learning experience; self-regulated learning theory was chosen as a relevant framework for explaining the learning impact of the findings. To achieve this objective, investigating essential functional features was necessary (RQ1). As students devise and follow strategies for performing learning tasks during the performance phase of self-regulated learning , providing features that align with their strategy can support them through the learning process [26] . Therefore, we examined students' feedback-giving and feedback-receiving strategies. They considered features such as code editor, auto test, inline comments, general comments, and rubric essential for following their feedback-giving strategies, and thread and attachment of feedback to the code

as crucial for feedback-receiving tasks (Fig. 1). The significant learning impact of rubrics is aligned with the findings of Alkhalifa et al.'s study [25]. Their study outlined the primary stages of peer assessment as assignment submission, feedback-giving, and feedback-receiving phases. However, it didn't explore the students' detailed strategies and processes for performing PCA tasks. Furthermore, their study focused on the rubric as a commenting feature and didn't examine the other relevant features.

Regarding PCA tools' qualities responding to RQ2, findings are mostly related to improving students' motivations, which can enhance learning by facilitating the forethought phase of self-regulated learning. Informants believe that performing PCA tasks with a customized, user-friendly PCA tool with a step-by-step process is motivating [9]. Furthermore, they think that considering additional grades for PCA tasks improves their motivation; this finding is aligned with the Zimmerman and Alkhalifa studies [25, 32]. Additionally, considering grading as a final step of the PCA process will encourage self-reaction in the self-reflation phase of self-regulated learning by motivating students to revise their work based on received feedback. This finding is also aligned with Alkhalifas' study [25]. The current study has contributed to detailed student-driven information about the essential features and qualities needed in PCA tools.

6 Conclusion

A case study was conducted to respond to research questions regarding the essential features and qualities of PCA tools; the findings offered crucial features that support feedback-giving and receiving tasks such as code editor, thread, inline commenting, general and rubric comments, etc. Findings also suggested some qualities that increase students' motivation, such as the effect of the order of steps, scoring, usability, etc. Regarding the study's validity, we acknowledge limitations related to the number, diversity, and experience of informants. Additionally, we recognize that the CodeGrade features may have influenced the strategies devised by the informants. Furthermore, we admit that conducting a case study on a tool actively used in a specific course would yield more realistic results. Due to the unavailability of such a case, we just opted for the available and relevant CodeGrade tool. We also didn't delve into all non-functional requirements, which can be considered in future studies. Also, recruiting more students with varying experience levels is suggested. Additionally, informants could encounter a broader range of features, or the new generation of AI-based features could be explored. In the study, the main emphasis has been on features related to giving feedback. In future work we also plan to study challenges related to receiving feedback from multiple peers such as how to present multiple and possible contradicting inline comments anchored to the same code fragments.

References

1. Guoping Rong, Jingyi Li, Mingjuan Xie, and Tao Zheng. The effect of checklist in code review for inexperienced students: An empirical study. In *2012 IEEE*

25th Conference on Software Engineering Education and Training, pages 120–124. IEEE, 2012.

2. Pavlína Wurzel Gonçalves, Gül Calikli, Alexander Serebrenik, and Alberto Bacchelli. Competencies for code review. *Proceedings of the ACM on Human-Computer Interaction*, 7(CSCW1):1–33, 2023.
3. Filip JRC Dochy and Liz McDowell. Introduction: Assessment as a tool for learning. *Studies in educational evaluation*, 23(4):279–98, 1997.
4. Keith Topping. Peer assessment between students in colleges and universities. *Review of educational Research*, 68(3):249–276, 1998.
5. Theresia Devi Indriasari, Andrew Luxton-Reilly, and Paul Denny. A review of peer code review in higher education. *ACM Transactions on Computing Education (TOCE)*, 20(3):1–25, 2020.
6. Andrew Luxton-Reilly. A systematic review of tools that support peer assessment. *Computer Science Education*, 19(4):209–232, 2009.
7. Y Benjelloun Touimi, N Faddouli, S Bennani, and M Khalidi Idrissi. Peer assessment in the context of project-based learning online. *World Academy of Science, Engineering and Technology International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, 7(1), 2013.
8. Hongli Li, Yao Xiong, Charles Vincent Hunter, Xiuyan Guo, and Rurik Tywoniw. Does peer assessment promote student learning? a meta-analysis. *Assessment & Evaluation in Higher Education*, 45(2):193–211, 2020.
9. Harri Hämäläinen, Ville Hyyrynen, Jouni Ikonen, and Jari Porras. Applying peer-review for programming assignments. *Int. J. Inf. Technol. Secur*, 1:3–17, 2011.
10. Jungkook Park, Yeong Hoon Park, Suin Kim, and Alice Oh. Eliph: Effective visualization of code history for peer assessment in programming education. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 458–467, 2017.
11. Torgeir Sandnes Laurvik. Design process behind an educational review system for student submissions-applying knowledge from professional code reviews to an educational assessment setting. Master's thesis, NTNU, 2021.
12. Cen Li, Zhijiang Dong, Roland Untch, Michael Chasteen, and Nathan Reale. Peerspace-an online collaborative learning environment for computer science students. In *2011 IEEE 11th International Conference on Advanced Learning Technologies*, pages 409–411. IEEE, 2011.
13. Ville Hyyrynen, Harri Hämäläinen, Jouni Ikonen, and Jari Porras. Mypeerreview: an online peer-reviewing system for programming courses. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, pages 94–99, 2010.
14. Kamila Misiejuk, Barbara Wasson, and Kjetil Egelandsdal. Using learning analytics to understand student perceptions of peer feedback. *Computers in human behavior*, 117:106658, 2021.
15. H Løje, N Qvistgaard, and PE Jensen. Implementing peer feedback in practice–a case study. In *49th SEFI Conference: Blended Learning in Engineering Education: challenging, enlightening–and lasting?*, pages 1444–1448. European Society for Engineering Education (SEFI), 2021.
16. Zahra Shakeri Hossein Abad, Oliver Karras, Parisa Ghazi, Martin Glinz, Guenther Ruhe, and Kurt Schneider. What works better? a study of classifying requirements. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 496–501. IEEE, 2017.

17. Rakesh Shukla, Ashish Sureka, Rushikesh Joshi, and Rajib Mall. A report on software engineering education workshop. *ACM SIGSOFT Software Engineering Notes*, 37(3):26–31, 2012.
18. Toshiki Hirao, Raula Gaikovina Kula, Akinori Ihara, and Kenichi Matsumoto. Understanding developer commenting in code reviews. *IEICE TRANSACTIONS on Information and Systems*, 102(12):2423–2432, 2019.
19. Felipe Ebert, Fernando Castor, Nicole Novielli, and Alexander Serebrenik. Confusion detection in code reviews. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 549–553. IEEE, 2017.
20. Saikrishna Sripada, Y Raghu Reddy, and Ashish Sureka. 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*, pages 3–6. IEEE, 2015.
21. Yanqing Wang, Yu Jiang, Min Chen, and Xin Hao. E-learningoriented incentive strategy: Taking edupcr system as an example. *World Trans. Eng. Technol. Educ*, 11(3):174–179, 2013.
22. Joe Gibbs Politz, Shriram Krishnamurthi, and Kathi Fisler. Captainteach: a platform for in-flow peer review of programming assignments. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 332–332, 2014.
23. Christopher D Hundhausen, Anukrati Agrawal, and Pawan Agarwal. Talking about code: Integrating pedagogical code reviews into early computing courses. *ACM Transactions on Computing Education (TOCE)*, 13(3):1–28, 2013.
24. Sari Kujala, Marjo Kauppinen, Laura Lehtola, and Tero Kojo. The role of user involvement in requirements quality and project success. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 75–84. IEEE, 2005.
25. Amal Alkhalifa and Marie Devlin. Student perspectives of peer assessment in programming courses. In *Proceedings of the 2021 Conference on United Kingdom & Ireland Computing Education Research*, pages 1–7, 2021.
26. BJ Zimmerman. Attaining self-regulation: A social cognitive perspective. *Self-regulation: Theory, research, and applications/Academic*, 2000.
27. Simon Cassidy. Self-regulated learning in higher education: Identifying key component processes. *Studies in Higher Education*, 36(8):989–1000, 2011.
28. Pei-Lin Hsu and Kuei-Hsiang Huang. Evaluating online peer assessment as an educational tool for promoting self-regulated learning. In *Multidisciplinary Social Networks Research: Second International Conference, MISNC 2015, Matsuyama, Japan, September 1-3, 2015. Proceedings 2*, pages 161–173. Springer, 2015.
29. Briony J Oates, Marie Griffiths, and Rachel McLean. *Researching information systems and computing*. Sage, 2022.
30. Robert K Yin. *Case study research: Design and methods*, volume 5. sage, 2009.
31. Shahedul Huq Khandkar. Open coding. *University of Calgary*, 23(2009):2009, 2009.
32. Barry J Zimmerman. Self-regulating academic learning and achievement: The emergence of a social cognitive perspective. *Educational psychology review*, 2:173–201, 1990.