Enhancing Software Engineering Education through Peer-to-Peer Project-Based Learning: An Empirical Study with Rust Programming

Minsuk Lee
Professor, School of Software, Kookmin University, Korea

Abstract

This empirical investigation delves into the integration of peer-to-peer approaches within project-based learning (PBL) methodologies for software engineering education. Specifically, it examines a case study of teaching the Rust programming language to 27 university students through a four-week intensive program. Students collaborated in small teams of three to four members, developing multi-user network games while utilizing GitHub for version control, Slack for team communication, and ChatGPT as a recommended learning assistant. The methodology employed mixed-methods data collection, including pre- and post-course surveys to assess knowledge progression, automated extraction of GitHub analytics and Slack metrics, and five-point scale peer evaluations of teammate contributions. Quantitative analysis examined response distributions, collaboration tool usage patterns, and correlations between engagement levels and learning outcomes. Utilizing established pedagogical theories and contemporary educational technology trends, this research demonstrates that peer-to-peer learning significantly enhances collaborative engagement (74% positive response), learning effectiveness (100% found AI-assisted peer learning effective), and project outcomes (72.8% of peer evaluations rated as positive or excellent). The research provides educators with evidence-based insights and practical guidelines to enhance the effectiveness of software engineering education through collaborative and experiential learning strategies, particularly in teaching emerging programming languages.

Keywords: software engineering education, peer-to-peer learning, project-based learning, collaborative learning, AI-assisted education

1. Introduction

Project-Based Learning (PBL) has emerged as a highly effective pedagogical approach for imparting practical skills and knowledge in Software engineering education. However, the traditional instructor-centric model of PBL often restricts opportunities for collaborative learning and real-world industry simulations. This research presents an empirical study investigating the integration of peer-to-peer learning approaches within project-based learning in the context of Software engineering education, specifically through a four-week intensive course teaching Rust programming language.

The software industry is increasingly demanding engineers who can rapidly acquire new technologies, collaborate effectively, and utilize modern tools, including AI assistants. Our study addresses this need by investigating how peer-to-peer learning, in conjunction with contemporary collaboration tools and AI assistance, can enhance traditional Problem-Based Learning (PBL) approaches. The overarching objective is to augment the depth and breadth of learning experiences while cultivating essential skills such as teamwork, critical thinking, and self-directed learning.

This investigation is guided by the following research questions:

RQ1: How can the integration of peer-to-peer learning enhance PBL environments in Software engineering education, particularly when learning new programming languages?

RQ2: What is the role of modern collaboration tools (GitHub, Slack) and AI assistants (ChatGPT) in supporting peer-to-peer learning within PBL?

RQ3: What are the quantifiable outcomes of peer-to-peer PBL in terms of student engagement, skill acquisition, and peer assessment?

2. Related Work

Recent systematic reviews and empirical studies have confirmed that collaborative and project-based methods positively impact programming skills, problem-solving, and teamwork among software engineering students [1][2][3]. Multidisciplinary group work and real-world projects facilitate the integration of classroom learning with industry practice [4][5][6].

Peer-to-peer learning, grounded in social constructivism and Vygotsky's zone of proximal development, emphasizes learning through rich interactions among peers. Effective peer learning interventions yield academic, social, and affective benefits across diverse contexts [7].

The integration of collaborative platforms, in conjunction with emerging artificial intelligence (AI) tools, substantially enhances the efficacy of project-based and peer-to-peer learning in programming education [6][7][8][9][10][11][12]. Empirical research substantiates that these tools augment collaboration, facilitate prompt technical feedback, and support self-directed skill acquisition.

Although both PBL and peer-to-peer learning have individually demonstrated their merits, empirical studies examining their synergy in teaching emerging programming languages remain limited. Furthermore, the role of AI assistants in P2P PBL contexts has not been thoroughly investigated. This paper bridges this gap by presenting quantitative evidence from a controlled educational experiment. We implemented a four-week intensive Rust programming course where 27 students with minimal prior knowledge collaborated in teams to develop text-based games. Our approach uniquely combined mandatory use of modern collaboration tools (GitHub and Slack) with encouraged adoption of AI assistants (ChatGPT) to create a technology-enhanced peer learning environment. We collected comprehensive data including platform activity metrics, self-reported learning outcomes, and detailed peer evaluations to assess the effectiveness of this integrated approach. Through this empirical study, we provide evidence-based insights into how P2P learning, when augmented with contemporary tools and AI assistance, can accelerate the acquisition of complex programming skills.

3. Study Design And Methodology

This study was conducted within a university-level software engineering course specializing in Rust programming. Students collaborated in small teams of three or four over a four-week period, utilizing GitHub and Slack for version control and communication. ChatGPT was recommended as a learning assistant. Data sources encompassed GitHub and Slack analytics, surveys, and peer assessments.

The course was structured as a four-week intensive program employing a hybrid pedagogical model that integrated multiple learning approaches. The program commenced with an initial knowledge assessment, wherein students self-reported their familiarity with Rust programming language, thereby establishing a baseline for measuring learning progress. The core of the course focused on peer-to-peer project work, where students formed small teams to collaboratively develop multi-user network games. This hands-on experience allowed students to apply Rust concepts in a practical and engaging context. To ensure systematic collaboration and create observable learning artifacts, the course mandated the use of GitHub for version control and code sharing among teams. Slack served as the primary communication platform for both synchronous and asynchronous discussions. Students were explicitly encouraged to utilize ChatGPT as a learning assistant, with instructors providing guidance on effective prompting strategies and appropriate use cases for AI support. However, unlike recent "vibe-coding" initiatives where entire projects could be generated with AI, students were not permitted to generate complete projects using AI. Instead, all AI-generated code snippets were required to be carefully reviewed by the students themselves to ensure comprehension and accuracy. The course concluded with a comprehensive peer assessment process, wherein students evaluated their teammates' contributions, collaboration quality, and overall impact on the project's success.

Our data collection strategy employed multiple complementary methods to capture both quantitative metrics and qualitative insights regarding the peer-to-peer learning process. We extracted detailed GitHub analytics, including commit frequencies, lines of code contributed, pull request activities, and code review participation, to comprehend individual contribution patterns and collaborative coding practices. Slack analytics provided insights into communication dynamics through message frequency analysis, channel participation rates, help-seeking behaviors, and the temporal distribution of team interactions throughout the project period. We administered comprehensive surveys at two critical points: pre-course surveys established baseline knowledge levels using binary assessments (know/don't know Rust), while post-course surveys measured self-reported capability improvements, perceived effectiveness of peer learning, utility of ChatGPT for skill acquisition, satisfaction with collaboration tools, and detailed time investment patterns, including meeting frequencies. The peer evaluation component asked students to assess each teammate using a five-point scale ranging from exceptional contribution to severe collaboration issues, providing direct insights into team dynamics and individual accountability within the peer-to-peer framework.

Quantitative analysis was conducted on survey responses and activity metrics. Response distributions were calculated, patterns in collaboration tool usage were identified, and correlations between engagement levels and learning outcomes were analyzed. Peer evaluation data was categorized to comprehend team dynamics and collaboration quality.

4. Results

Participants demonstrated substantial skill improvements in Rust programming, evidenced by both self-reported confidence and peer evaluations. A significant majority of students reported positive engagement with GitHub, Slack, and AI support, aligning with findings in recent studies [10][11]. Peer-to-peer learning, augmented by AI, enhanced project outcomes and collaborative dynamics, consistent with contemporary research on collaborative and AI-assisted learning in software engineering education.

4.1 Learning Outcomes

The initial knowledge assessment revealed that only one student among the 27 participants reported any prior experience with Rust programming, establishing an almost uniform starting point that provides an ideal condition for examining peer learning effectiveness without the confounding factor of varied expertise levels. After four weeks of intensive peer-to-peer project-based learning, the transformation in student capabilities was remarkable: 51.8% of students reported confident capability to use Rust independently, while 48.1% acknowledged moderate capability with some remaining challenges, and notably, not a single student reported inability to use the language. This complete elimination of the “cannot use” category represents a significant achievement in pedagogical terms, demonstrating that peer-to-peer learning can effectively support all students in achieving at least functional programming capability even in a language as complex as Rust, which is known for its steep learning curve due to unique concepts like ownership and borrowing. The near-even split between confident and moderate capability suggests that peer learning creates a supportive environment where students progress at different rates while still achieving meaningful competency, validating the inclusive nature of peer-to-peer approaches in accommodating diverse learning speeds.

4.2 Peer Learning Effectiveness

The student perception data on peer learning effectiveness reveals overwhelmingly positive outcomes that have significant implications for software engineering pedagogy. Notably, 74% of students rated peer learning as effective or very effective, while 26% found it moderately effective. Remarkably, no student deemed it ineffective. This unanimous recognition of at least moderate value suggests that peer-to-peer approaches address fundamental learning needs that traditional instructor-centered methods may overlook. Specifically, they provide students with the opportunity to explain language concepts and knowledge in their own words and learn from peers who recently overcame similar conceptual challenges. The high effectiveness ratings validate our first research question by demonstrating that peer-to-peer approaches not only enhance PBL environments theoretically but also produce measurable satisfaction and perceived value among learners. The absence of any “ineffective” ratings is particularly noteworthy given the challenging nature of Rust and the compressed four-week timeframe. This suggests that peer support serves as a crucial scaffold when learning complex technical concepts under time pressure.

4.3 AI-Assisted Peer Learning

Perhaps the most impressive finding of our study was the universal endorsement of ChatGPT as a learning tool. Notably, 100% of students reported its effectiveness in learning Rust, a finding rarely observed in educational technology research. Students reported diverse use cases, including syntax clarification, debugging assistance, concept explanation, and code optimization suggestions. This effectively treated ChatGPT as an always-available peer tutor that complemented human peer interactions. This finding addresses our second research question by demonstrating that AI tools do not merely support but fundamentally amplify peer-to-peer learning effectiveness. This creates a hybrid learning ecosystem where human peers provide emotional support, motivation, and collaborative problem-solving, while AI assists with immediate technical queries and conceptual clarifications. The complete consensus on AI effectiveness suggests a generational shift in learning preferences, where students naturally integrate AI assistance into their learning workflows. Educational frameworks that fail to acknowledge this reality risk becoming disconnected from actual student practices.

4.4 Collaboration Tool Effectiveness

The assessment of technology-mediated collaboration revealed that 88.8% of students found GitHub and Slack effective for project coordination. Students also voluntarily adopted additional tools, including video conferencing platforms, Google Docs/Notion for documentation, and Discord for informal discussions. This high satisfaction rate demonstrates that modern software development platforms naturally align with peer-to-peer learning needs by providing persistent, searchable, and transparent collaboration spaces that extend learning beyond scheduled class hours. The variety of supplementary tools adopted by students indicates that peer learners actively seek and create multiple communication channels to support different aspects of collaboration—from formal code reviews in GitHub to casual problem-solving discussions in Discord. The effectiveness of these platforms in supporting peer learning validates that the infrastructure of modern software development inherently facilitates educational collaboration, suggesting that teaching students to use these professional tools simultaneously prepares them for industry practices while enhancing their immediate learning experience.

4.5 Time Investment and Engagement

Analysis of student time investment patterns provided valuable insights into the commitment levels necessary for successful peer-to-peer PBL. Teams met between 2 and 12 times over the four-week period, with a median of 6 meetings. Weekly time investment ranged from 4 to 30 hours, including class time, with a median of 9 hours. The correlation between higher time investment and more positive peer evaluations suggests that peer-to-peer learning naturally establishes accountability mechanisms where visible commitment directly impacts team dynamics and peer perceptions. The wide range in both meeting frequency and time investment indicates that peer-to-peer PBL accommodates diverse team working styles—some teams preferring frequent short check-ins while others opted for fewer but longer collaborative sessions. This flexibility in engagement patterns, while still achieving positive learning outcomes across the spectrum, demonstrates that peer-to-peer approaches can adapt to diverse student schedules and preferences, a crucial consideration for modern educational environments where students often balance multiple commitments.

4.6 Peer Assessment Results

The peer evaluation data, comprising 114 individual assessments, provides valuable insights into team dynamics and collaboration quality within peer-to-peer project-based learning (PBL) environments. The distribution revealed that 43.9% of evaluations rated teammates as exceptional, 28.9% as good collaborators, 18.4% as acceptable, 6.1% as problematic, and only 2.6% (representing just 3 evaluations) as severely problematic. The cumulative 72.8% positive rating (exceptional plus good) demonstrates that peer-to-peer PBL predominantly fosters successful collaborative relationships, creating environments where students not only acquire technical skills but also develop professional collaboration capabilities. The low percentage of severely negative evaluations (2.6%) suggests that the structured nature of peer-to-peer PBL, combined with transparent contribution tracking through GitHub and regular communication via Slack, minimizes free-riding and encourages meaningful participation. The emotional intensity of the evaluation language used by students—ranging from “gift from heaven” to “would take leave”—indicates that peer collaboration in PBL contexts creates strong interpersonal dynamics that likely enhance both motivation and accountability, factors crucial for successful software engineering practice.

5. Discussion

Findings underscore the synergistic relationship between Project-Based Learning (PBL), peer learning, and technology-enhanced environments in software engineering education. Professional development tools such as GitHub and Slack facilitate visible and accountable collaboration. The integration of artificial intelligence (AI), for instance, ChatGPT, not only provides technical guidance but also amplifies peer support structures, expanding Vygotsky’s Zone of Proximal Development (ZPD) concept to encompass intelligent digital scaffolding [7][8][10].

Our findings extend social constructivist theory by demonstrating that AI agents can participate in the zone of proximal development, traditionally occupied solely by human peers and instructors. The combination of human peers and AI assistance creates a more enriching learning environment than either alone could provide.

The peer evaluation results indicate that peer-to-peer PBL naturally establishes accountability mechanisms. The predominance of positive evaluations (72.8%) suggests that peer pressure and mutual dependency foster engagement and contribution.

Based on our empirical findings, we propose the following evidence-based guidelines:

- Structured Onboarding: Initiate with explicit expectations regarding peer collaboration and tool utilization.

- AI Integration: Explicitly encourage and train students in utilizing AI assistants as learning partners.
- Visible Collaboration: Mandate the use of platforms that facilitate transparent contributions (e.g., GitHub).
- Regular Peer Assessment: Implement periodic peer evaluations to uphold accountability.
- Flexible Meeting Structures: Permit teams to self-organize meeting frequency based on their requirements.
- Time Expectation Management: Clearly communicate the anticipated time investment (9 hours per week in our study).

Despite overall success, several challenges emerged:

- Uneven Participation: The 8.7% of negative peer evaluations suggests that some students struggled with collaboration. Mitigation: Implement early intervention strategies based on GitHub/Slack activity metrics.
- Time Management: The wide range of time investment (3-15 hours) indicates varying commitment levels. Mitigation: Establish minimum participation thresholds.
- Over-reliance on AI: Some students may use ChatGPT as a crutch rather than a learning tool. Mitigation: Design assessments that test understanding, not just code production.

Although promising, the results are limited by sample size and cultural context. Future research should examine the generalizability of the approach to other programming languages, the long-term retention of knowledge, and a comparative analysis with instructor-led models. The evolving impact of AI on collaborative learning necessitates continued, methodologically rigorous exploration [10][13].

6. Conclusion

This study investigated the integration of peer-to-peer learning within project-based learning (PBL) for software engineering education using Rust programming. Guided by three research questions, the findings elucidate both pedagogical and technological implications.

For RQ1 (Peer-to-peer learning in PBL environments), the results demonstrate that peer learning transforms instructor-dependent models into collaborative ecosystems where all students attain functional programming proficiency. Despite minimal prior experience, every participant achieved at least moderate proficiency in Rust, confirming that peer collaboration fosters inclusive and accelerated learning progression.

For RQ2 (Modern collaboration tools and AI in supporting peer learning), the findings reveal that such tools serve as structural and cognitive scaffolds. GitHub and Slack ensured transparency and asynchronous coordination, while ChatGPT acted as an always-available peer tutor, amplifying conceptual understanding and reducing hesitation in seeking assistance.

For RQ3 (Measurable outcomes in engagement, skill acquisition, and peer assessment), quantitative evidence demonstrates strong engagement, substantial competency gains, and positive collaboration quality.

These results indicate that peer-to-peer PBL nurtures accountability, cooperation, and sustained motivation.

In summary, the convergence of human collaboration and AI-assisted peer learning represents a pedagogical shift in software engineering education. By orchestrating peer interaction, digital collaboration platforms, and intelligent assistance, educators can design scalable, student-centered learning environments that emulate professional software development and prepare learners for the collaborative, AI-enriched workplaces of the future.

REFERENCES

1. J. Zhang, L. Zhang, G. Zhang and Y. Yu, “From Theory to Practice,” Frontiers in Computing and Intelligent Systems, Vol. 8. No. 1, pp. 130–133, 2024. DOI: https://doi.org/10.54097/jv09dw34

2. R. Garcia, C. Treude and A. Valentine, “Application of collaborative learning paradigms within software engineering education: A systematic mapping study,” in Proceedings of the 55th ACM Technical Symposium on Computer Science Education, Vol 1, pp. 366–372, 2024. DOI: https://doi.org/10.1145/3626252.3630780

3. P. J. Roig, S. Alcaraz, K. Gilly, C. Bernad and C. Juiz, “Project-based learning for assessing a course on computer programming,” International Journal for Cross-Disciplinary Subjects in Education, Vol. 15, No. 2, pp. 4957-4964, 2024. DOI:https://DOI.org/10.20533/ijcdse.2042.6364.2024.0611

4. H. Alsmadi, G. Kandasamy, A. A. Kafri and K. F. Zahirah, “Empowering computing students through multidisciplinary project based learning (PBL): Creating meaningful differences in the real world,” Social Sciences & Humanities Open, Vol. 10, 2024. DOI: https://doi.org/10.1016/j.ssaho.2024.101180

5. A. J. Vicente, T. A. Tan and A. R. Yu, “Collaborative approach in software engineering education: An interdisciplinary case,” Journal of

Information Technology Education: Innovations in Practice, Vol. 17, pp. 127-152, 2018. DOI: https://doi.org/10.28945/4062

6. C. Hundhausen, P. Conrad, O. Adesope and, A. Tariq, “Combining GitHub, chat, and peer evaluation data to assess Individual Contributions to Team Software Development Projects,” ACM Transactions on Computing Education, Vol. 23, No. 3, pp. 1-32, 2023. DOI: https://doi.org/10.1145/3593592

7. L. S. Vigotsky, M. Cole, V. Jolm-Steiner, S. Scribner and E. Souberman, Mind in society: development of higher processes, Harvard University Press, pp. 141-152, 1978. DOI: https://doi.org/10.2307/j.ctvjf9vz4

8. M. Khan, M. A. Akbar and J. Kasurinen, “Integrating LLMs in Software Engineering Education: Motivators, Demotivators, and a Roadmap Towards a Framework for Finnish Higher Education Institutes,” arXiv preprint arXiv:2503.22238, 2024. DOI: https://doi.org/10.48550/arXiv.2503.22238

9. C. Sengul, R. Neykova and G. Destefanis, “Software engineering education in the era of conversational AI: Current trends and future directions,” Frontiers in Artificial Intelligence, Vol. 7, 2024. DOI: https://doi.org/10.3389/frai.2024.1436350

10. Y.-M. Yan, C.-Q. Chen, Y.-B. Hu and X.-D. Ye, “LLM-based collaborative programming: Impact on students’ cognitive load and computational thinking and self-efficacy,” Humanities and Social Sciences Communications, Vol. 12, 2025. DOI: https://doi.org/10.1057/s41599-025-04471-1

11. P. Patani, S. Tiwari and S. S. Rathore, “The impact of GitHub on students' learning and engagement in software engineering course,” Computer Applications in Engineering Education, Vol. 32, No. 5, 2024. DOI: https://doi.org/10.1002/cae.22775

12. K. J. Topping, “Trends in peer learning,” Educational Psychology, Vol. 25, No. 6, pp. 631-645, 2005. DOI: https://doi.org/10.1080/01443410500345172

13. K. Ko, “Effectiveness of ChatGPT in programming education for non- Computer Science majors,” Journal of Digital Contents Society, Vol. 25, No. 9, pp. 2517-2524, 2024. DOI: https://doi.org/10.9728/dcs.2024.25.9.2517