Journal of Engineering Education Transformations ,
Volume 39, No. 1 , July 2025 , ISSN 2349-2473, eISSN 2394-1707

10.16920/jeet/2025/v39i1/25127

Bridging the Academic-Industry Gap: A Role Play Peer Review Method for Teaching Software Engineering

Vijaylaxmi Bittal[1], Khalid Alfatmi[2]

[1, 2] Department of Computer Engineering, SVKM's Institute of Technology, Dhule, India
[1] vijaylaxmi.bittal@gmail.com
[2] alfatmi.khalid@gmail.com

Abstract- Traditional educational methods of teaching software engineering often stall to align with industry standards. Theoretical instruction, which emphasizes concepts and procedures, does not sufficient to prepare students for real-world applications. Consequently, students struggle to apply their knowledge in practical scenarios. This paper introduces the Role Play Peer Review Learning (RPL) approach to address this gap. In the RPL approach, students are assigned various roles within the software development cycle and engage in peer reviews. This method has been implemented in a classroom setting, and this paper presents the results of its application, demonstrating its effectiveness in bridging the gap between academic instruction and industry-level software engineering practices.

Keywords- Software Engineering, Role Play, Peer Review, Teaching & Learning.

Vijaylaxmi Bittal
Department of Computer Engineering,
SVKM's Institute of Technology, Dhule, India
vijaylaxmi.bittal@gmail.com

1. Introduction

In the rapidly evolving field of software engineering, the alignment between academic instruction and industry requirements is of most importance. However, traditional educational methods often fall short in preparing students to meet the practical demands of the industry. These traditional approaches typically emphasize theoretical instruction, focusing on concepts and procedures rather than hands-on, practical skills. As a result, students frequently encounter difficulties when attempting to apply their theoretical knowledge in real-world scenarios, leading to a significant gap between academic preparation and professional readiness.

The software engineering industry demands a blend of technical proficiency, practical experience, and the ability to work collaboratively within a team. Yet, academic programs have traditionally placed a greater emphasis on individual performance and theoretical understanding. This disengagement can leave graduates underprepared for the collaborative and dynamic nature of professional software development environments. Recognizing this critical gap, academic institutions are exploring innovative pedagogical approaches to enhance the relevance and applicability of their curricula.

One such innovative approach is the Role Play

Peer Review Learning (RPL) method, which this paper introduces and evaluates. The RPL approach is designed to simulate real-time software development processes within a classroom setting, thereby providing students with practical experience that mirrors industry practices. In this approach, students are assigned various roles that exist within the software development life cycle, such as developers, testers, project managers, and clients. By engaging in these roles, students gain a comprehensive understanding of the software development process from multiple perspectives.

Central to the RPL approach is the concept of peer review. Peer reviews are a standard practice in the software industry, serving as a important mechanism for ensuring code quality and facilitating knowledge sharing among team members. By incorporating peer reviews into the RPL method, students are not only required to produce work that meets professional standards but also to critically evaluate the work of their peers. This dual focus on creation and evaluation helps to deepen students' understanding of software engineering principles and practices.

The implementation of the RPL approach in a classroom setting has shown promising results. Students participating in this method have demonstrated improved ability to apply theoretical knowledge in practical contexts. They have also developed essential skills such as teamwork, communication, and critical thinking, which are crucial for success in the software engineering industry. The RPL method fosters an active learning environment where students can learn from their mistakes and successes in a collaborative setting, closely mimicking the real-world professional environment.

This paper presents the findings from the application of the RPL approach, providing empirical evidence of its effectiveness in bridging the gap between academic instruction and industry-level software engineering practices. Through a detailed analysis of student performance and feedback, the paper illustrates how the RPL method enhances the learning experience and prepares students more effectively for their future careers.

Hence, to engage students in learning Software Engineering principles, we have shown a small approach RPL which creates interest towards learning software engineering course. This method comprises assigning the distinct roles of Software Development Life Cycle (SDLC) to the student teams and conducting peer review to cross check.

This paper is structured as follows: Sect. 2 Related Work (Background) Sect. 3 Methodology, Sect. 4 Results and Discussion Sect. 5 Conclusion.

2. Related Work

Time and again there are various methods of teaching the SE course. The following methods are discussed with respect to the different challenges whether it be competencies, teaching methodology or online platform.

The field of Software Engineering (SE) education has long been grappling with the question of how to effectively teach complex, context-sensitive competencies that mirror those required in professional environments. Klopp et al. (2020) expressed the fundamental question of whether e-learning can indeed teach the essential generic competences necessary for effective team- and project-based work in SE. They told these competencies as including communicative and professional cooperation skills, the ability to structure one's work, personal competencies, understanding complex processes, and applying knowledge to new situations.

The need for improved teaching strategies to address these competencies is evident when examining the challenges highlighted by previous studies. For instance, “Work in Progress: Teaching-Obstacles in Higher Software Engineering Education” (2017) argued that SE, with its inherently complex content, should be taught using problem-based learning (PBL). This method provides students with the opportunity to engage with conceptual concepts through real-world applications, making the content more comprehensible. However, PBL is not always feasible for every topic or teaching mode, and educators must continue to explore new strategies to overcome the persistent challenges in SE instruction. To aid this, a teaching obstacle map was developed to offer a systematic overview of the issues faced in SE education.

Challenges are particularly pronounced at the undergraduate level, where students difficult to understand and visualize complex SE concepts and theories, such as software system modeling

(Malaysia. Kementerian Pengajian Tinggi et al., n.d.). To address these difficulties, the incorporation of cooperative learning methods (CLM) has been suggested as a means to better help students analyze and visualize organizational problems, system analysis, and system modeling. The pilot project approach, specifically within requirement engineering courses, has demonstrated promising results by integrating CLM strategies.

The structural evolution of SE programs is another critical aspect to consider. Chatley (2019) discussed how their program adapted teaching methods to align with the tools and techniques used in the real world by professional software developers. They demonstrated the effectiveness of aligning teaching with lean software delivery principles, creating tighter feedback loops in traditional lecture-based courses and using project-based courses where students applied agile methods to build software in teams over several months.

The impact of project-based learning (PBL) has been further validated by studies such as those by Gransbury et al. (2023), who found that traditional methods of teaching SE often result in low student engagement. By shifting to a PBL approach, where teams were formed based on specific project needs, student engagement improved significantly. Similarly, Thelma Colanzi et al. (2023) observed that using PBL in their SE course motivated students and encouraged a deeper interest in learning SE. These findings echo the positive feedback from Alva et al. (2018), who highlighted the benefits of Problem-Based Learning and Peer-Assisted Learning in SE courses, showing that students responded well to the more interactive and collaborative learning experiences.

Abirami A.M et al. (2021) suggested Active Learning Strategies and Blended Learning Approach for Teaching Undergraduate Software Engineering Course. Involves experimental study of application of the various active learning techniques such as discussion forums, tech talks followed by quiz for such topics. Impact of the application of the improved content delivery plan on the course outcomes attainment by the learners has also been observed and presented.

Moreover, Yang et al. (2022) emphasized the importance of bridging the gap between academia and industry by applying industrial software engineering practices in university courses. This preliminary approach included a competency model specifically designed for implementation in Chinese universities, aiming to bring educational practices more in line with the expectations of the software industry. Chia et al. (2022) also contributed to this discussion by surveying students' sentiments towards their computing science syllabi, identifying gaps between industry needs and academic training, particularly in developing professional skills.

Moharir et al,Awati et al and Dol et al presented case studies based on peer review and role play. Also shown effectiveness of these in teaching and learning process.

Given the insights from these studies, it is clear that while significant strides have been made to improve SE education through various teaching strategies like PBL, CLM, and Problem-Based Learning, there is still room for refinement and integration. This gap between educational practice and industry demands has led us to propose an integrated approach called RPL (Reflective Practice Learning). This approach aims to bridge the divide between academia and the industry by embedding practices that are more representative of real-world software engineering. By promoting reflective learning and integrating it with project-based methodologies and cooperative techniques, RPL seeks to offer a more holistic approach that prepares students to meet industry challenges effectively.

Based on the related work and literature survey we came to know that still there are lot of opportunities to suggest an effective learning approach to learn Software engineering course. Hence, we suggested an integrated approach RPL which reduces the gap between industry and academia in terms of adopting software engineering practices.

3. Methodology

Our effective learning approach of Software Engineering through RPL proposed the following strategy, and it is shown in Figure 1.

1. Formation of team and assigning roles
2. Selecting Problem Statement
3. Set the sprint cycle

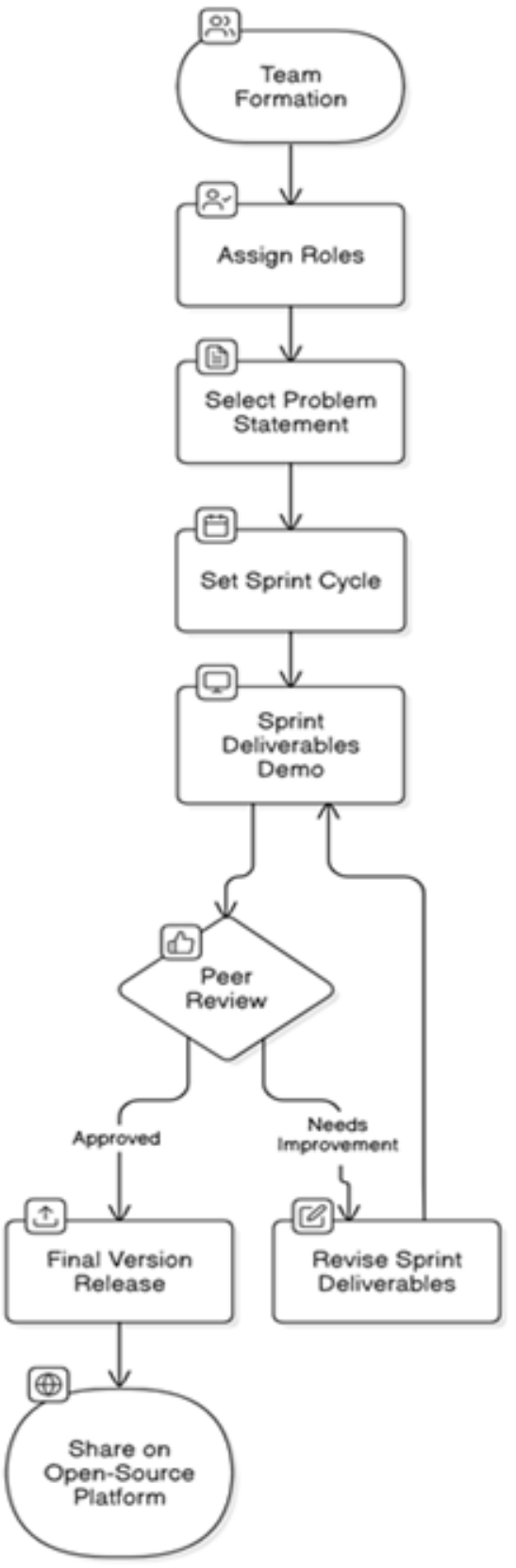

Fig. 1 : Process Diagram

4. Demonstration of sprint deliverables and taking the review from peer team

5. Releasing final version of software product and sharing in open-source platform

1. Formation of team and assigning roles

Team is formed with distinguished roles of SDLC (Software Development Life Cycle) as Team Leader, System Designer, Software Developer and Tester. Each role is assigned with predefined task as shown in Table I.

Table I :
Role And Task Details

Role Name	Task
Team leader	Oversees the entire software product, ensuring it stays on track in terms of timeline, budget, and scope. Coordinate software development activities and communicate with stakeholders.
System Designer	Designs the complete structure of the software system, ensuring that all components work together well. Creates a blueprint that guides developers.
Software Developer	Writes and maintains the code based on the design specifications and requirements. Collaborates with team members to implement features and fix bugs.
Tester	Develops test cases and performs testing (manual or automated) to ensure the software is free of defects. Validates that the product meets the defined requirements and quality standards.

2. Selecting Problem Statement

Suggested that all teams choose development of complex systems such as LinkedIn, Instagram, Spotify, and e-commerce platforms as their problem statements. This will allow them to fully apply software engineering principles at an industry-standard level.

3. Set the sprint cycle

The Agile Software Development method is an iterative and incremental approach to software development that emphasizes flexibility, collaboration, and customer satisfaction. Agile methods prioritize working software over comprehensive documentation, adapt to changing requirements, and foster close collaboration among

cross-functional teams. This approach is highly valued for its responsiveness to change and focus on delivering functional software in short cycles, known as sprints or iterations. Informed all teams to use

Agile Softtware Devlopment method for their development process.

Setting up the optimal length for each sprint, commonly 1-4 weeks. A 1-week sprint is decided for all teams to balance feedback frequency and development time. Details shown in Table II.

Table II :
Sprint Details

Sl No	Sprint	Deliverables
1	Week1	SRS (Software Requirement Specification)
2	Week2	System Design (UML diagrams)
3	Week3	Initial Version of system (Codebase)
4	Week4	Design of Testcases (Test Report)

4. Demonstration of sprint deliverables and taking the review from peer team

Demonstration of sprint deliverables as per sprints decided is shown in Figure 2

Ensure that all deliverables are complete and ready to be demonstrated. This includes code, features, documentation, and any other outputs. Outline the key points to showcase, including the main functionalities and any new features. Create a summary of the peer team's feedback and share it with the development team. If there are any changes or improvements suggested, integrate them into the next sprint planning or backlog refinement.

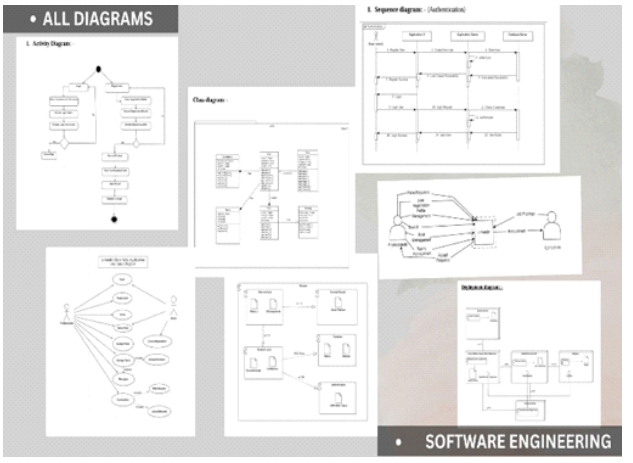

Fig. 2 : Demonstration of Sprint Deliverables and Taking The Review From Peer Team.

5. Releasing final version of software product and sharing in open-source platform

Conduct a final code review to ensure that code quality meets your standards and is ready for public release. Decide on which open-source platform to share your software, such as GitHub, GitLab, or

Table III :
System Repositories and Post Details

System Name	Repository Link	Linked in Post Link
Social Connect	https://lnkd.in/g95VdyHv	https://www.linkedin.com/posts/chaitanya-dusane119381259_softwareengineering-teamworksocialconnectactivity7268175274893025280 6vjV?utm_source=share&utm_medium=member_desktop
Spotify Clone	https://lnkd.in/gvstCwPu	https://www.linkedin.com/posts/vaibhav-patil1b4871257_softwareengineering-teamwork-studentproject-activity7270465954801692673 2nxP?utm_source=share&utm_medium=member_desktop
Eraser.io	https://lnkd.in/d5jAb5UB	https://www.linkedin.com/posts/mohammad-anasinam_softwareengineering-sdlc-teamwork-activity-7271152945675583488nLbf?utm_source=share&utm_medium=member_desktop

Bitbucket. Set up a new repository to include the final version of your software. Repositories and post details shown in Table III.

4. Results And Discussion

Once all theory essentials are delivered about SDLC, these below mentioned questions were asked before and after the conduction of RPL in classroom. This entire process took 4 weeks to complete, and it found all teams were active in learning their functionality as well peer reviewing other teams.

1) Do you know the importance of SDLC Process?

2) Do you know how SRS is prepared?

3) Are you able to prepare software design for a given problem statement?

4) Are you able to decide technological stack as per requirements?

5) Are you able to identify testing strategies?

6) Are you able to document each step of SDLC?

Table IV :
Before 'RPL'

Q No	No of students	Yes	Somewhat	No
1	69	10	02	57
2	69	0	12	57
3	69	0	8	61
4	69	0	3	66
5	69	0	3	66
6	69	0	2	67

Table V :
After 'RPL'

Q No	No of students	Yes	Somewhat	No
1	69	69	0	0
2	69	67	2	0
3	69	68	1	0
4	69	66	3	0
5	69	66	3	0
6	69	67	2	0

Learning outcomes extracted from learning feedback received from each team participants

1) Discovered how various teams function within the software development cycle, how they respond to client requests and how they work together to produce a product that meets customer expectations.

2) Visualize on accurate environment of software development and boost practical knowledge of understanding SDLC.

3) Learning was engaging and unique for the team as well as other team members.

The overall performed activity was delightful

Conclusion

The software engineering course is a fundamental course of the computer science branch. Software engineers will apply programming knowledge and engineering principles to deliver customer required software product. Engaging students in the class for learning software engineering course in an effective way is a challenging task for many course instructors and academicians. Hence, we have given RPL which exhibits engaging students in learning SDLC in a joyful manner and displayed many meaningful learning outcomes which are aligned with Software Engineering principles. This small effort helps the academicians in delivering Software Engineering course in an intended manner.

References

Alva, H. H., Uma, B., Shruthi, D. V., & Saroja, C. (2018). Enhancing learning outcomes in software engineering course through problem based learning and peer assisted learning. Journal of Engineering Education Transformations, 2018(Special Issue). https://doi.org/10.16920/jeet/2018/v0i0/120948

Chatley, R. (2019). Applying Lean Learning to Software Engineering Education. Agile and Lean Concepts for Teaching and Learning. https://doi.org/10.1007/978-981-13-2751-3_14

Chia, Y. X., Loh, K. H., Ong, Z. Y. B., Lim, J. X., Ooi, J. K., Cao, Q., Yau, P. C., & Lim, L. H. I. (2022).

Sentiments Analysis and Feedback among Three Cohorts in Learning Software Engineering Modules. Proceedings - 2022 IEEE International Conference on Teaching, Assessment and Learning for Engineering, TALE 2022. https://doi.org/10.1109/TALE54877.2022.00025

Gransbury, I., Brock, J., Root, E., Catete, V., Barnes, T., Grover, S., & Ledeczi, A. (2023). Project-Based Software Engineering Curriculum for Secondary Students. ACM International Conference Proceeding Series. https://doi.org/10.1145/3605468.3605501

Klopp, M., Gold-Veerkamp, C., Abke, J., & Borgeest, K. (2020). Teaching generic competences in software engineering via E-Learning - An evaluation. Proceedings of 2020 IEEE International Conference on Teaching, Assessment, and Learning for Engineering, TALE 2020. https://doi.org/10.1109/TALE48869.2020.9368446

Malaysia. Kementerian Pengajian Tinggi, Institute of Electrical and Electronics Engineers, Regional Conference on Engineering Education & Research in Higher Education (7th : 2017 : Kuala Lumpur, M., International STEA Education Conference (1st : 2017 : Kuala Lumpur, M., & Innovative Practices in Higher Education Expo (4th : 2017 : Kuala Lumpur, M. (n.d.). 2017 7th World Engineering Education Forum : WEEF 2017 : proceedings : 13-16 November 2017, Berjaya Times Square Hotel, Kuala Lumpur, Malaysia.

Work in progress: Teaching-obstacles in higher software engineering education. (2017). IEEE Global Engineering Education Conference, EDUCON. https://doi.org/10.1109/EDUCON.2017.7943067

Yang, X., Zhang, J., Cai, W., Ran, C., & Huang, H. (2022). Preliminary Exploration of Incorporating Industrial Software Curriculum into University Software Engineering Education. Proceedings - 2022 IEEE International Conference on Teaching, Assessment and Learning for Engineering, TALE 2022. https://doi.org/10.1109/TALE54877.2022.00117

Leo Natan Paschoal ,João Pedro S. M. Ruiz,Simone R. S. Souza" An Open Educational Resource Supporting Mutation Testing Teaching". SBQS '23: Proceedings of the XXII Brazilian Symposium on Software QualityNovember 2023Pages 291–30 ACM

Thelma Colanzi, Leandro Silva ,Andressa Medeiros, Paulo Gonçalves, Eniuce Souza, Douglas Farias,

Greicy Amaral "Practicing the Extension in Software Engineering Education: an Experience Report" SBES '23: Proceedings of the XXXVII Brazilian Symposium on Software Engineering September 2023 Pages 514–523 ACM

Abirami A.M, Pudu Malar S ThiruchadaiPandeeswari "Active Learning Strategies and Blended Learning Approach for Teaching Undergraduate Software Engineering Course" Journal of Engineering Education Transformations, Volume 35 , No. 1 , July 2021 , ISSN 2349-2473, eISSN 2394-1707

Moharir, M., Agavekar, R., Bhore, P., Kadam, H., & Bewoor, A. (2022). Effective Implementation of Peer Review as an Active Learning Technique to Attain Course Outcome: A Case Study. Journal of Engineering Education Transformations, 36(Special Issue 1), 63– 72. Scopus.

Awati, J. S., Kulkarni, S. S., & Patil, S. K. (2019). Energetic teaching activity role play and round quiz: A case study. Journal of Engineering Education Transformations, 33(Special Issue 1), 84–90. Scopus. https://doi.org/10.16920/jeet/2019/v33i1/149028

Dol, S. M., & Halkude, S. A. (2018). Improving critical thinking skill of students using aRPiGDs: An effective and alternative method to role play. Journal of Engineering Education Transformations, 2018(Special Issue). Scopus. https://doi.org/10.16920/jeet/2018/v0i0/120909