Building Bridges: Teaching Programming through Problem-Based Learning in Mechanical Engineering

Mrs.C.Sravanthi, Dr.PratibhaDharmavarapu, Mrs.A.Neeraja
Anurag University
sravanthimech@anurag.edu.in

***Abstract*—Teaching programming to mechanical engineering students, often seen as a computer science endeavor, requires a thoughtful approach to highlight its relevance in their field. The mechanical engineering teacher can bridge this gap by demonstrating how programming solves complex engineering problems. Using real-world examples such as simulating mechanical systems or automating data analysis, illustrates the practical benefits of programming skills. To engage students, the teacher implemented Problem-Based Learning (PBL) activities tailored to mechanical engineering contexts. Activities like "Trace it Out," where students identify code errors, and "Jumble the Code," which involves ordering or debugging code pieces, help students connect programming concepts to engineering projects. Solutions to real-world problems are presented as poster presentations, making learning interactive and hands-on. Community problems and solutions were documented and assessed using designed rubrics, reinforcing programming knowledge and developing problem-solving skills. Regular feedback and project-based assessments, rather than traditional exams, better measure students' understanding and ability to apply programming skills to engineering problems. This comprehensive approach—linking programming with mechanical engineering applications, mentoring, active learning, and continuous feedback—transforms students' perspectives, helping them realize the invaluable role of programming in their professional development as mechanical engineers. Providing access to online resources and encouraging participation in coding communities further enhances their learning experience.**

I. INTRODUCTION

In contemporary mechanical engineering education, the integration of programming skills is becoming increasingly essential. Mechanical engineering students often view programming as a domain confined to computer science, yet its applications in their field are vast and transformative. This paper explores a pedagogical approach that addresses this disconnect by embedding programming into the mechanical engineering curriculum through practical, real-world applications.

To achieve this, a mechanical engineering instructor utilized her domain expertise to design Problem-Based Learning (PBL) activities tailored specifically to the needs and interests of her students. These activities include coding exercises that solve engineering problems, interactive debugging tasks, and projects that require students to present solutions to real-time challenges. This hands-on approach not only makes learning more engaging but also underscores the practical benefits of programming in solving complex mechanical engineering problems.

The effectiveness of this method is further enhanced by regular feedback and project-based assessments, which focus on students' ability to apply programming skills in real-world contexts rather than merely testing theoretical knowledge. By fostering an environment of active learning and continuous improvement, this approach helps students develop critical problem-solving and analytical skills that are crucial for their future careers.

This study highlights how integrating programming with mechanical engineering education can transform students' perspectives, equipping them with the tools and confidence needed to excel in an increasingly digital and automated engineering landscape.

II. LITERATURE REVIEW

The integration of problem-based learning (PBL) and active learning strategies in engineering education has been widely researched and documented. This review synthesizes key findings from various studies, focusing on the application of these methods in enhancing programming skills among mechanical engineering students.

The studies of teaching programming to engineering students. Kannan et al. (2018) discusschallenges faced by faculty and students, emphasizing practical sessions and technology-enhanced learning. Lathigara et al. (2021) advocate for activity-based learning to enhance engagement and real-world problem-solving skills. Rajalingam et al. (2021) explore peer interaction in virtual classrooms, finding that peer mentoring and collaborative projects improve

engagement and outcomes. Thorat and Kshirsagar (2021) emphasize developing logic, problem-solving, and debugging skills through targeted strategies. Collectively, these studies underscore the need for innovative, interactive, and student-centered teaching approaches in programming education.

Savin-Baden (2003) and Woods (1994) provide comprehensive overviews of PBL, emphasizing its role in fostering critical thinking and problem-solving skills. PBL shifts the focus from traditional lecture-based instruction to student-centered learning, where students engage with real-world problems. This method has been shown to improve retention and application of knowledge, as students actively construct their understanding through problem-solving activities. The incorporation of PBL in engineering curricula encourages students to develop skills relevant to their professional careers, such as teamwork, communication, and self-directed learning.

Nikolic (2017) demonstrates that project-based learning (PBL) enhances student engagement and learning outcomes in engineering courses. By working on projects that simulate real-world scenarios, students can see the direct relevance of their studies to their future careers. This hands-on approach is particularly effective in mechanical engineering, where practical application of concepts is crucial.

Active learning strategies, such as those reviewed by Prince (2004) and Hake (1998), have been shown to significantly improve student performance in STEM fields. These strategies involve interactive activities that require students to actively engage with the material, rather than passively receiving information. For example, "Trace it Out" and "Jumble the Code" activities help students understand programming concepts by debugging and organizing code, fostering deeper comprehension and problem-solving skills.

Lister et al. (2009) and Xu and Rajlich (2004) provide evidence that code tracing and debugging exercises are effective in developing programming skills. These activities help students understand the flow of a program and identify logical errors, which are crucial skills for any programmer. Lahtinen et al. (2005) and Simon and Lister (2011) further explore the difficulties faced by novice programmers and suggest that structured, scripted instruction can alleviate some of these challenges.

The use of visualizations in programming education, as discussed by Ben-Ari et al. (2003), aids in the comprehension of abstract concepts. Tools like Jeliot 3 allow students to visualize the execution of their code, making it easier to understand complex programming constructs. These tools can be particularly beneficial for mechanical engineering students, who may not have a strong background in computer science.

Prince and Felder (2006) compare various inductive teaching methods, including inquiry-based learning and problem-based learning, finding that these approaches are generally more effective than traditional teaching methods. Bonwell and Eison (1991) also advocate for active learning, suggesting that it creates a more engaging and effective learning environment. Their findings are supported by Freeman et al. (2014), who show that active learning strategies significantly increase student performance in science, engineering, and mathematics.

Blumenfeld et al. (1991) emphasize the importance of motivation in project-based learning. They argue that sustaining student interest through relevant and challenging projects is key to successful learning outcomes. Margaryan and Littlejohn (2008) discuss the use of digital technologies and online resources to support learning, highlighting the potential of these tools to enhance student engagement and provide additional learning opportunities.

III. PROBLEM DESCRIPTION

Mechanical engineering students often perceive programming as a domain exclusive to computer science, leading to a disconnect between their engineering education and the potential benefits of programming skills. This perception limits their ability to utilize programming in solving complex engineering problems. The challenge lies in demonstrating the relevance and practical applications of programming within the mechanical engineering field, engaging students in a way that underscores its necessity and benefits. Traditional teaching methods and assessments may not adequately measure students' understanding and application of programming skills in real-world engineering contexts. Therefore, there is a need for a pedagogical approach that effectively integrates programming into the mechanical engineering curriculum, making learning interactive and hands-on, and fostering continuous feedback and self-directed learning.

III. IMPROVEMENT IN C LANGUAGE TEACHING

Problem Based Learning

Problem-Based Learning (PBL) is a learner-centered educational method that focuses on real-world examples. Students engage with case studies to "find, analyze, and solve problems," guided by their teachers. During this process, students perform information searches and acquire essential knowledge necessary for resolving these issues. This approach has been shown to enhance students' understanding and retention of knowledge. In PBL, learners set their learning objectives based on the problems they encounter and employ self-learning to address and understand these challenges. This method helps students develop self-directed learning skills, fostering lifelong learning capabilities, as highlighted by Wen Xiangmin (2012).

IV. APPLICATION OF THE PBL METHOD IN C LANGUAGE TEACHING:

A. PBL Model Building

The "Programming for Problem Solving" course is structured around a Problem-Based Learning (PBL) model to

integrate programming into the mechanical engineering curriculum, fostering student engagement and practical understanding. Each group of students select or proposes a real-world or community problem, addressing it through a project presented via posters and detailed documentation. Regular feedback and assessments are provided using rubrics focusing on problem identification, solution effectiveness, code quality, and presentation skills. Students also have access to online resources and are encouraged to participate in coding communities. This approach ensures ultimately transforming students' perspectives on the role of programming in their professional development.

B.Teaching organization of PBL

The "Programming for Problem Solving" course curriculum comprises five distinct units. To enhance student engagement and learning, various activities were designed and implemented across these units. The activity, 'Trace it Out' was conducted in a classroom setting with 38 students, focusing on content from Units I and II. For Units III, IV, and V, the activities 'Jumble the Code and Code It' were employed.

At the outset of the course, the classroom was divided into groups, each consisting of 5 to 6 students. The instructor provided a list of real-world and community problems, allowing each group to either select a problem from the list or propose their own. Each group undertook a real-world problem as a project, which they subsequently presented through poster presentations and detailed documentation. The activities were assessed using specifically designed rubrics to evaluate each team's performance. This structured approach ensured that students applied programming concepts to practical engineering problems, fostering an interactive and hands-on learning environment.

TABLE I
ORGANIZATION OF PBL PROCESS

Curriculum	Learning Outcomes	Assessment Methods	Other Assessment Method
UNIT-I	Discuss algorithms and flowcharts for various real-world applications (L1)	Trace it out	Project on real world problems showcasing the identified problem and solution by poster presentation and then documentation
UNIT-II	Express the usage of various operators useful in Program development(L2)	Trace it out	
UNIT-III	Design programs involving decision and iterative structures(L3)	Jumble the code and code it	
UNIT-IV	Apply the concepts of code reusability with the help of Functions(L4)	Jumble the code and code it	
UNIT-V	Analyse the concepts of Arrays and Strings to resolve real world problems(L4)	Jumble the code and code it	

V. TEACHING ASSESSMENT OF PBL

Assessing Project-Based Learning (PBL) involves evaluating both the process and outcomes of student projects using comprehensive criteria. Teachers assess how effectively students collaborate within teams, solve problems critically, and conduct research relevant to their projects. Evaluation extends to the quality of the final product or solution developed, assessing the clarity and persuasiveness of presentations, and reviewing the thoroughness of project documentation. Individual contributions are also considered, with attention given to how well students fulfill their roles and reflect on personal growth. Clear rubrics with descriptors ranging from "excellent" to "need improvement" are invaluable for ensuring consistency and fairness in assessment, guiding students in understanding expectations and areas for improvement.

TABLE II
RUBRICS FOR ASSESSMENT METHOD –PROJECT BASED

Criteria	Weightage	Excellent (10M)	Very Good (8M)	Good (6M)	Need Improvement (4M)	Poor (2M)
Design	(10M)	The algorithm and flow chart are very well designed and presented clearly, which meets all the needs of the project.	The algorithm and flow chart are very well designed but some needs of the project are not met	The algorithm and flow chart are designed but needs of the project are not met.	The algorithm and flow chart does not convey the needs of the project	Partially completed either algorithm or flowchart
Implementation	(10M)	Code is functional and refined and ready for future enhancements. Code is executed by passing all test cases	Code is functional, Code is executed but not passing one of the test cases	Code is partially functional not passing any of the test cases.	Code does not execute because of errors.	Code is not implemented yet.
Presentation and Documentation	(10M)	Explanation of the project by the student is very clear and understandable, completed the presentation and report submission in specified time. Every aspect of the project is clearly documented.	Explanation of the project by the student is clear and understandable. Project is documented and submitted in Stipulated time. Every aspect of the project is clearly documented.	Explanation of the project by the student is clear and understandable. Some of the aspects of the projects are not clearly documented.	Explanation of the project by the student is not clear and not documented properly	Student was not able to explain the project .

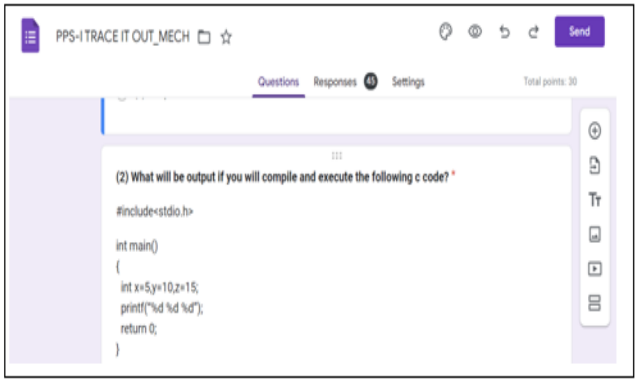
TABLE III
RUBRICS FOR ASSESSMENT METHOD –DESIGNING OF POSTER

Criteria	Weightage	Excellent (15M)	Very Good (12M)	Good (9M)	Need Improvement (6M)	Poor (3M)
Presentation	(15M)	Explanation of the topic is clear. Student responded to all the questions with Accurate answers.	Explanation of the topic is clear. Student was able to answer most of the questions.	Explanation of the topic by the student is clear. Student could answer some of the questions.	Explanation of the topic by the student is not clear. Student was not sure about the answers given to questions.	Explanation of the topic by the student is not clear. Student was not able to answer the questions.
Poster	(15M)	Poster is well designed using graphics. labels are in readable format.	Poster is well designed. Labels are in readable format but the look of the poster needs some improvement.	Poster is well designed. Labels are not clear.	Poster designed is not up to the mark. Labels are not clear.	Poster is designed partially.

TABLE IV
RUBRICS FOR ASSESSMENT METHOD -JUMBLE CODE AND CODE IT

Criteria	Weightage	Excellent (10M)	Very Good (8M)	Good (6M)	Need Improvement (4M)	Poor (2M)
Code arrangement	(10M)	Program is well sequenced and properly separates code in and out of the draw loop .	Program separates code in and out of the draw loop. May contain some incorrectly sequenced code.	Program is executed through the draw loop but some code is improperly placed in and out of the draw loop.	Draw loop is not used for execution of program.	Unable to arrange the code.
Time taken for code Arrangement	(10M)	Code arrangement is done in with in 15 Min.	Code arrangement is done in 15 to 20 Min.	Code Arrangement is done with in 30 min.	Code Arrangement is done in 45 min.	Unable to arrange the code.
Code Implementation	(10M)	Code is functional and refined and Code is executed by passing all test cases	Code is functional, Code is executed but not passing one of the test cases	Code is partially functional not passing any of the test cases.	Code does not execute because of errors.	Code is not implemented yet.

TABLE V
SAMPLE QUESTION FOR TRACE IT OUT ACTIVITY



VI. PBL REFLECTIONS

A. Outcomes

- The ability to write algorithms and create flowcharts for real-world scenarios has improved, enabling students to apply learned concepts to problem-solving.
- Skills in tracing outputs have advanced, proving useful during placements, and students' analytical skills have been enhanced through the "trace it out" method.
- Problem-solving skills have developed, allowing students to connect concepts to the real world through the "code it out" assessment method.
- The ability to organize code within a specified time has increased, and early understanding of logic and syntax has improved through the "jumble code" method.
- Communication, creativity, and leadership qualities have been enhanced during poster and project presentations.

B. Challenges

- Make students take the ownership of their learning.
- Continuous motivation of passive students.
- Student resistance in initial days as they compare with the non PBL courses.
- Resolving conflicts in the group.
- Time constraints.

C. Student Reflections

- We liked the project assessment because we learnt more by this method.
- We liked the jumble code because arranging the code in limited time is fun learning.
- The experience of teamwork was interesting.
- Poster presentation improved our creative skills.
- It is more knowledgeable and gave practical experience.
- I recommend this PBL implementation for other subjects like Physics and Engineering Mechanics.

D. Teacher Reflections

- Faculty autonomy as well as student autonomy.
- Emerging challenges and equity.
- Valuable insights from conflict resolution.
- Continuous internal evaluation.
- Transparency in the evaluation strategy

VII. RESULT ANALYSIS

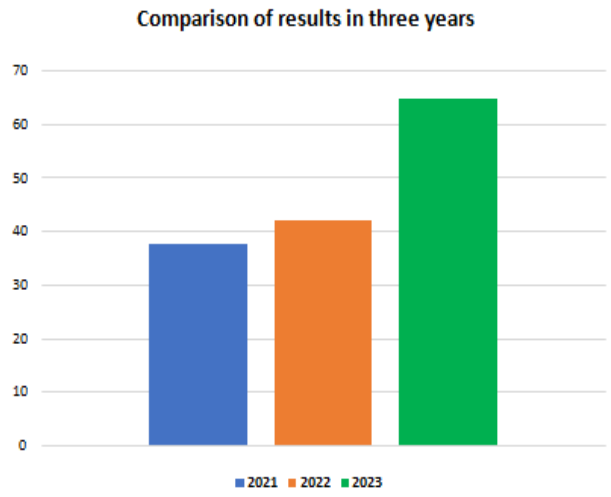


Fig 1. Comparison of results in three years

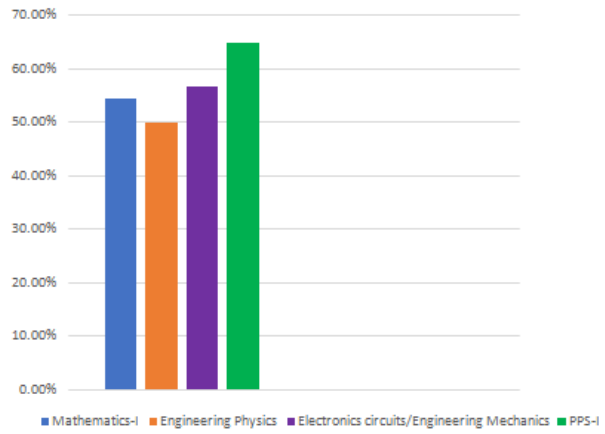
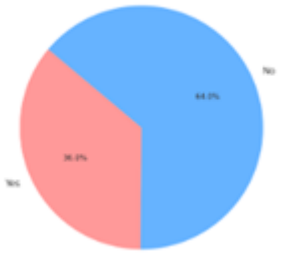
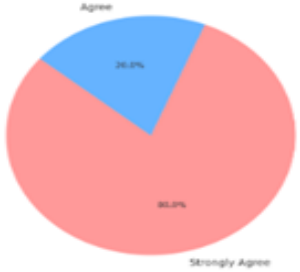
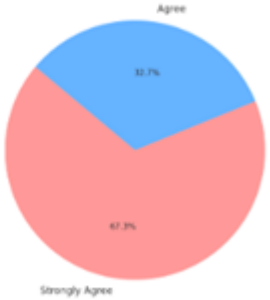
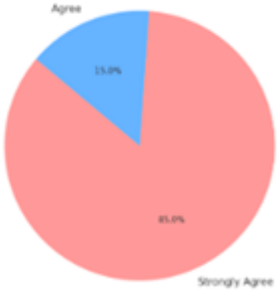


Fig 2. Comparison of results between PBL and Non PBL Courses.

The comparison over the past three years indicates a noticeable improvement in student performance, particularly after the introduction of Project-Based Learning (PBL) in the curriculum. Students in PBL courses showed improved coding skills, problem-solving abilities, and practical application of programming concepts. Through PBL, students were able to work on real-world coding projects, which deepened their understanding of topics such as data structures, algorithms, and debugging techniques. These results indicate that PBL allows students to engage with programming in a more interactive and applied manner, thereby enhancing their confidence and proficiency in coding.

VIII. FEEDBACK ANALYSIS

Questionnaires and responses given by students	**Analytical Research**
1. Do you know C language before this course? 	36% students learnt C language before
2. Do you feel that the activity trace it out helps in recalling the concepts learnt in programming language 	80% strongly agree 20% agree
3. Do you feel that the activity jumble code and code it improves problem-solving ability. 	74% strongly agree and 36% agree
4. Did you feel comfortable with your team members in poster making and presentation 	85% strongly agree 15% agree
5. Did you satisfy with the project of problem identification and finding solution 	88% strongly agree and 12% agree

	92% strongly agree and 8% agree

IX. CONCLUSIONS

Based on the evidence presented, it can be concluded that integrating programming into the mechanical engineering curriculum through context-specific, problem-based learning (PBL) significantly enhances students' engagement and comprehension of programming's relevance to their field. The use of real-world engineering problems and active learning strategies, such as "Code It" and "Trace it Out," fosters critical thinking and problem-solving skills, crucial for professional success in mechanical engineering. Moreover, the shift towards project-based assessments over traditional exams better measures students' ability to apply programming in practical scenarios. The final outcomes of this educational intervention are likely that such methods will lead to a deeper integration of programming skills into mechanical engineering practice, potentially serving as a model for similar interdisciplinary teaching approaches.

X. REFERENCES

Kannan, S., Sumathi, D., & Prabakaran, T. (2018). A study on challenges and opportunities in teaching programming subject to first-year computer science and engineering students: In the perspective of faculty and student. *Journal of Engineering Education Transformations, 31*(3). ISSN 2349-2473, eISSN 2394-170.

Lathigara, A., Tanna, P., & Bhatt, N. (2021). Activity based programming learning. *Journal of Engineering Education Transformations, 34*, Special issue.

Rajalingam, S., Kanagamalliga, S., Karuppiah, N., &Puoza, J. C. (2021). Peer interaction teaching-learning approaches for effective engagement of students in virtual classroom. *Journal of Engineering Education Transformations, 34*, Special issue.

Thorat, S. A., & Kshirsagar, D. P. (2021). Developing logic building, problem-solving, and debugging programming skills among students. *Journal of Engineering Education Transformations, 34*, Special issue.

Savin-Baden, M. (2003). Facilitating Problem-Based Learning: Illuminating Perspectives. Open University Press.

Woods, D. R. (1994). Problem-Based Learning: How to Gain the Most from PBL. McMaster University.

Chiou, R., & Hsu, C. (2013). Incorporating Computational Tools in Mechanical Engineering Curricula: Integrating MATLAB and SolidWorks. *ASEE Annual Conference & Exposition*.

Anderson, J., & Finelli, C. J. (2014). A Systematic Review of the Literature on Teaching Engineering Design through Project-Oriented Capstone Courses. *Journal of Engineering Education*, 103(4), 472-496.

Felder, R. M., & Silverman, L. K. (1988). Learning and Teaching Styles in Engineering Education. *Engineering Education*, 78(7), 674-681.

Nikolic, S. (2017). Enhancing Student Learning through Project-Based Learning in a First-Year Engineering Course. *IEEE Transactions on Education*, 60(3), 197-203.

Prince, M. (2004). Does Active Learning Work? A Review of the Research. *Journal of Engineering Education*, 93(3), 223-231.

Hake, R. R. (1998). Interactive-Engagement Versus Traditional Methods: A Six-Thousand-Student Survey of Mechanics Test Data for Introductory Physics Courses. *American Journal of Physics*, 66(1), 64-74.

Wiggins, G. P., & McTighe, J. (2005). Understanding by Design. Association for Supervision and Curriculum Development (ASCD).

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active Learning Increases Student Performance in Science, Engineering, and Mathematics. *Proceedings of the National Academy of Sciences*, 111(23), 8410-8415.

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., &Palincsar, A. (1991). Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning. *Educational Psychologist*, 26(3-4), 369-398.

Margaryan, A., & Littlejohn, A. (2008). Are Digital Natives a Myth or Reality? University Students' Use of

Digital Technologies. *Computers & Education*, 56(2), 429-440.

Lister, R., Fidge, C., & Teague, D. (2009). Further Evidence of a Relationship between Explaining, Tracing and Writing Skills in Introductory Programming. *ACM SIGCSE Bulletin*, 41(3), 161-165.

Xu, L., &Rajlich, V. T. (2004). Empirical Validation of Test-Driven Pair Programming in an Introductory Computer Science Course. *IEEE Transactions on Education*, 47(2), 197-204.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A Study of the Difficulties of Novice Programmers. *ACM SIGCSE Bulletin*, 37(3), 14-18.

Simon, B., & Lister, R. (2011). Computing Education: The Case for Scripted Instruction. *ACM Transactions on Computing Education (TOCE)*, 11(4), 24.

Ben-Ari, M., Levy, R., &Uronen, P. A. (2003). A Suite of Visualizations for Object-Oriented Program Analysis and Design. *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, 133-137.

Bonwell, C. C., & Eison, J. A. (1991). Active Learning: Creating Excitement in the Classroom. *ASHE-ERIC Higher Education Report No. 1*. Washington, DC: George Washington University.

Prince, M. J., & Felder, R. M. (2006). Inductive Teaching and Learning Methods: Definitions, Comparisons, and Research Bases. *Journal of Engineering Education*, 95(2), 123-138.