An Integrated CDIO–Kolb Framework for Experiential Learning in Software Engineering

[1]Pudumalar S, [2]Parkavi R
[1,2]Department of Information Technology, Thiagarajar College of Engineering, Madurai
[1]spmit@tce.edu, [2]rpit@tce.edu

***Abstract*—Traditional lecture-based approaches in software engineering education often fall short in providing the experiential depth required to understand stakeholder needs, foster effective team collaboration, and develop essential documentation skills. This study introduces and implements the CKIL (CDIO–Kolb Integrated Learning) framework in a mini-project assignment designed for third-semester undergraduate software engineering students (n = 180). The CKIL framework integrates engineering lifecycle principles from the CDIO model with Kolb's experiential learning cycle, enabling students to engage in structured phases of problem identification, requirement elicitation, and system modelling through iterative reflection, active prototyping and testing. Team-based reflections, peer reviews, and exploratory use of digital tools—were embedded throughout the assignment. Implementation outcomes indicate enhanced student engagement, improved teamwork dynamics, and stronger attainment of course learning objectives. The study further explores the framework's influence on skill development, critical thinking, and the cultivation of lifelong learning competencies. findings underscore the value of blending traditional academic content with contemporary pedagogical practices, demonstrating that the CKIL framework offers a scalable and effective model for preparing engineering graduates to meet the complex demands of real-world software development environments.**

***Keywords*— CDIO, Kolb's Experiential Learning, Software Engineering Education, Active Learning strategies, CKIL Framework.**

***ICTIEE* Track*:* Innovative Pedagogies and Active Learning**

ICTIEE Sub-Track: Choose one from the list of sub-tracks mentioned in the Website
(Refer to the Paper Submission and Review Guidelines for more details.)

I. INTRODUCTION

In engineering instruction, specifically in fields like software engineering, the dominant pedagogical model is still primarily didactic—oriented toward lectures, slides, and static evaluations. Although this approach effectively conveys theoretical underpinnings, it tends to fail in developing the dynamic skills needed for actual software production, including collaboration, problem-solving, iterative thinking, and tool facility. Students often experience a gap between what they learn in the classroom and the dynamic, nimble, and tool-centric reality of industry practices. While curriculum structure and content coverage have improved, the absence of experiential depth gets in the way of learners to absorb and translate ideas throughout the software development lifecycle.

Modern efforts to close this gap involve flipped classrooms, project-based education, and gamified testing. These interventions tend to be discrete, though, without a unified pedagogical design that progressively scaffolds learning from grasping theoretical models to practicing design principles, working in teams, employing software tools, and reflecting on practice. Moreover, although active learning approaches are increasingly promoted, they are not necessarily systematically connected to frameworks of cognitive development or engineering project workflows, so resulting learning is periodic.

To meet these challenges, this research introduces a integrated pedagogical strategy: the CKIL (CDIO–Kolb Integrated Learning) framework. CKIL combines the ordered, lifecycle-foundation engineering pedagogy of CDIO (Conceive–Design–Implement–Operate) with Kolb's cycle of experiential learning (Concrete Experience, Reflective Observation, Abstract Conceptualization, and Active Experimentation). This combination creates both a project-based and cognitively founded learning pathway, allowing students to apply software engineering principles in an iterative, reflective, and real-world context.

This study aims to change an otherwise passive content delivery during a software engineering course by embedding the CKIL framework into its instructional design. Mini-projects inspired by real-world contexts engaged in by the students would require requirement analysis, modeling, design thinking, and collaboration—guided by iterative cycles of doing, reflecting, and improving. This approach not only closes the theory-practice gap but also gives learners lifelong learning skills, deeper conceptual understanding, and readiness for professional roles in software engineering.

The study assesses CKIL efficacy in an undergraduate software engineering course with 180 students across the principal software development stages. The results are measured using measures of student interaction, effectiveness of learning, skill acquisition, and outcome achievement—providing a scalable and flexible model for the engineering education of the 21st century.

To guide the analysis, the following research questions were framed:

RQ1: How does the integration of the CKIL (CDIO–Kolb Integrated Learning) framework impact student engagement, collaboration, and conceptual understanding in undergraduate software engineering education?

RQ2: To what extent does the CKIL framework support the development of applied software engineering skills—such

Pudumalar S
Assistant Professor
Department of Information Technology, Thiagarajar College of Engineering
Madurai.spmit@tce.edu

as requirement analysis, modeling, and design thinking—compared to traditional lecture-based instruction?

II. Literature Review

Traditional lecture-based instruction in software engineering has long been critiqued for failing to provide the experiential depth required to develop essential engineering competencies. Several studies underscore that while lectures are effective for conveying theoretical knowledge, they are insufficient for cultivating collaborative problem-solving, iterative thinking, and stakeholder communication skills—capabilities essential for real-world software development (Bonetti et al.2025 and Zowghi 2011).

In their systematic mapping study, Bonetti et al. (2025) found that lecture-based models often lead to surface-level understanding, with limited opportunities for students to apply knowledge contextually. Ebadi et al. (2020) further highlight that requirement engineering (RE) education often lacks real-world simulation, resulting in a gap between classroom learning and industry expectations.

Responding to the limitations of lecture-centric pedagogy, many researchers have explored active and experiential learning models that foster deeper engagement, HOTS, and real-world skill acquisition among students. Konak et al. (2014) applied "Kolb's experiential learning cycle"—comprising Concrete Experience, Reflective Observation, Abstract Conceptualization and Active Experimentation—in the context of virtual computer laboratories. Their findings demonstrated that embedding learning activities within this structured cycle led to notable improvements in both learner performance and metacognitive awareness. Students not only performed better on assessments but also reflected more meaningfully on the learning process, allowing for iterative refinement of their skills.

Building on this, Devi and Thendral (2025) implemented Kolb's model in a theory-intensive engineering course, revealing that even in non-practical, concept-heavy subjects, experiential learning could significantly improve conceptual clarity and long-term retention. This reinforces the idea that experiential strategies are not confined to lab-based settings but can be applied across a spectrum of content types to stimulate meaningful learning.

The support for experiential learning's transformative potential is offered by the WIETE Transformative Learning Team (2023) and Albertini et al. (2024), who emphasize the synergistic effect of "reflection and iteration" in engineering education. These studies highlight that when students are given opportunities to engage in cyclic learning—where they actively participate, reflect, reframe understanding, and reapply knowledge—their critical thinking, problem-solving abilities, and intrinsic motivation improve. Importantly, students begin to assume greater ownership of their learning journey, becoming self-directed learners equipped for continuous growth.

Silva et al. (2021) introduced a four-stage project-based learning (PBL) approach tailored to software engineering. This model, which closely mirrors Kolb's learning cycle, structures student learning around real-world problem-solving, with an emphasis on stakeholder communication, team collaboration, and leadership development. Their results showed not only improved technical skills but also enhanced soft skills—an area often overlooked in traditional curricula. The iterative stages allowed students to apply theoretical ideas in real situations, promoting contextual learning and a deeper understanding of the software development process.

The CDIO framework, introduced by Crawley et al. (2014), presents a lifecycle-based model that mirrors engineering practice—moving through Conceive, Design, Implement, and Operate stages. It has been widely adopted to support project-based curricula and integrated assessment structures. Its relevance in software engineering education lies in providing students with a structured, authentic context to apply their knowledge.

Garcia et al. (2023) emphasize that collaborative learning models are most effective when grounded in structured pedagogical approaches like CDIO. Their mapping study identified gaps in standalone active learning techniques, advocating for models that provide scaffolding across multiple stages of a project.

Studies have also recognised the value of peer learning and stakeholder interaction in requirement engineering and design thinking. Connor et al. (2014) showed that peer-based review in RE projects helped students better internalise domain knowledge and client expectations. The use of pair learning and peer assessment in requirement elicitation (Jatit Engagement Team, 2025) revealed higher levels of participation and accuracy in requirement documentation.

Ahmed et al. (2025) explored integrating AI tools to simulate stakeholder interviews for elicitation practice, highlighting that technology-supported, realistic interaction significantly strengthens learning in the early stages of SDLC.

Collectively, these studies provide strong evidence that experiential and project-based learning, when systematically implemented through structured models like Kolb's, significantly improves the quality and depth of student learning. By aligning Kolb's cognitive model with CDIO's engineering design structure, the CKIL framework offers a scalable solution to develop not only academic skills but also practical, team-based, and reflective abilities—bridging the ongoing theory–practice gap. They also indicate that such approaches are especially effective when combined with reflective and collaborative elements, providing the pedagogical foundation for frameworks like the CKIL model proposed in this study.

III. RESEARCH METHODOLOGY

The methodology aligns course design, Kolb's Experiential Learning Theory delivery, and assessment practices with two frameworks, suggesting that effective learning is a continuous, experience-driven process. Rooted in constructivism, it proposes that learners create knowledge by transforming their experiences through a four-stage cycle: Concrete Experience, Reflective Observation, Abstract Conceptualization, and Active Experimentation. When applied systematically, this

cycle promotes deep understanding, fosters metacognitive skills, and effectively connects theoretical concepts with practical application.

TABLE I
KOLB'S FOUR-STAGE CYCLE

STAGE	Description
Concrete Experience (CE)	Engaging in a specific, hands-on task or activity
Reflective Observation (RO)	Reviewing and reflecting on the experience
Abstract Conceptualization (AC)	Concluding and developing theories based on reflection
Active Experimentation (AE)	Applying new knowledge to future tasks and contexts

The CDIO (Conceive–Design–Implement–Operate) framework provides a structured, lifecycle-aligned approach to engineering education. It emphasizes:

- Conceive: Defining the problem, needs, and constraints.
- Design: Developing technical plans and models.
- Implement: Building, coding, testing, and integrating solutions.
- Operate: Deploying, maintaining, and reflecting on solutions in use.

CDIO promotes project-based, team-driven learning grounded in real-world engineering challenges. It aligns educational practice with industry expectations, preparing students to become innovative and socially responsible engineers. The application of CKIL in Course Planning, Content Delivery and Assessment.

A. Course Planning

1) Define learning outcomes using Bloom's Taxonomy, integrated with reflective components.
2) Structure the course plan around mini-projects aligned to SDLC phases.
3) Design assignments and lab activities to match CDIO stages and Kolb cycles.

B. Course Delivery

1) Concrete Experience: Use role plays, stakeholder interviews, and tool exploration activity
2) Reflective Observation: Encourage weekly team reflections, peer review, and group retrospectives.
3) Abstract Conceptualization: Guide students in theory development via concept mapping, classroom discussions, and guided questioning.
4) Active Experimentation: Engage students in sprint planning, design reviews, and test case generation.

C. Assessment Alignment

Assessment in a CKIL-based course is continuous, formative, and summative, aligning with each phase of Kolb's cycle and CDIO stage.

This alignment facilitates constructive alignment in course planning, ensuring that Intended Learning Outcomes (ILOs), Teaching-Learning Activities (TLAs), and Assessment Tasks (ATs) are coherently mapped and mutually reinforcing.

TABLE II
CKIL-BASED COURSE ASSESSMENT STRUCTURE

STAGE	Description
Concrete Experience (CE)	Lab work, experiments, field visits, demonstrations, simulations, case encounters
Reflective Observation (RO)	Reflective journals, lab reflections, observation reports, error analysis, self/peer review
Abstract Conceptualization (AC)	Written exams, derivations, conceptual questions, analytical problem-solving, case analysis
Active Experimentation (AE)	Projects, coding tasks, design/build tasks, simulation studies, prototype development

TABLE III
MAPPING CKIL FRAMEWORK TO SOFTWARE ENGINEERING COMPONENTS

CDIO Phase	Kolb Stage	Course Component
Conceive	Concrete Experience (CE)	Brainstorming project ideas, identifying problems, and stakeholder interviews
Conceive→ Design	Reflective Observation (RO)	Team discussions on feasibility, requirements gathering review
Design	Abstract Conceptualization (AC)	Creating SRS, user stories, modeling use cases and architecture
Implement → Operate	Active Experimentation (AE)	Coding, testing, peer reviews, client feedback integration

TABLE IV
KOLB CYCLE STAGE ALIGNMENT FOR COURSE ACTIVITIES

TOPIC - ACTIVITY	Kolb Cycle Stage
Assignment 1 – Requirement Engineering & Problem Definition (Stakeholder identification, requirement gathering, feasibility analysis, user stories, use-case diagram)	CE, RO, AC, AE
Assignment 2 – System Design & Tool Exploration (DFD, UML, DevOps mind map, test case writing, peer assessment, SRS compilation)	CE, RO, AC, AE
Joint Teaching -Testing and Automated testing tool usage Software Quality Models – SEI-CMM	AC/AE
Final Review, Evaluation & Peer Feedback	RO, AE

IV. IMPLEMENTATION

The course was designed as a three-month program comprising 36 instructional hours, divided into six modules and aligned with six course outcomes. Each module was designed to progressively develop students' understanding of core software engineering concepts through experiential learning activities and iterative feedback. For Requirement engineering module,students conduct user interviews (CE),

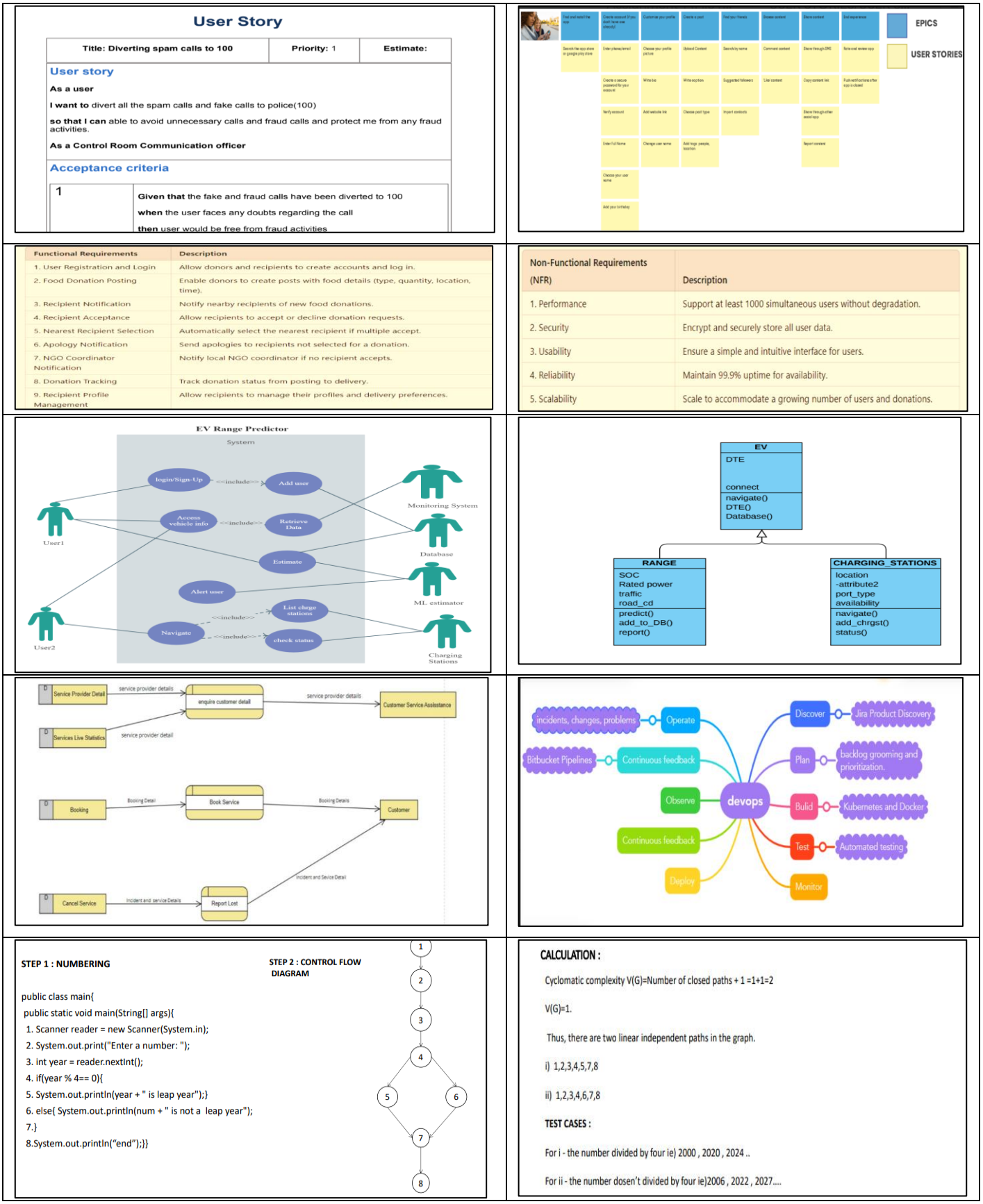

Fig. 1. Sample of assignment 1 and assignment 2 submission showing requirement gathering, generating function and non-functional requirements, user stories, use case diagram, class diagram,data flow diagram, test case generation using various tools.

reflect on feedback (RO), document user stories and SRS (AC), and prototype wireframes or UML diagrams (AE).In the

design module, students begin by analyzing project goals and exploring system requirements to conceptualize the architecture (Concrete Experience). They then reflect on different design approaches and challenges faced during peer discussions and critiques (Reflective Observation). Based on this reflection, they develop design artifacts such as data flow diagrams, UML class diagrams, and interface layouts (Abstract Conceptualization). Finally, they apply this understanding by building detailed design documentation and refining their models through iterative feedback and tool-based experimentation (Active Experimentation). These assessments were not just tools for engagement and immediate feedback but were also integrated into the formal evaluation process. The assignment plan for the course and its mapping with Kolb Cycle stage is given the table IV.

V. RESULTS AND DISCUSSION

The assessment outcomes from both Assignment 1 and Assignment 2 provide valuable insights into student engagement, understanding, and skill acquisition throughout the course. The highest score in Assignment 1 is 38 and lowest score is 8,similarly for assignment2 39 and 24. In Assignment 1: Most students scored between 60–80, with a few outliers above 90. In Assignment 2: Scores shifted slightly higher, with a peak around 70–85. The course outcomes of the course are

CO1 Compare traditional and agile software process models
CO2 Identify user stories, Story map, functional and non-functional requirements for any given problem
CO3 Prepare design documents with standards for the given requirements
CO4 Develop test cases using appropriate testing techniques for an application
CO5 Explain the scope of the software maintenance problem and demonstrate the use of version controlling and tracking mechanisms.
CO6 Demonstrate DEVOPS life cycle processes and introduce state of art tools used in large scale software systems.

The score of the students for Assignment 1 and Assignment 2

TABLE V
ASSIGNMENT 1 AND ASSIGNMENT 2 MARKS

Course Outcome	Average Score (A1)	Average Score (A2)
CO1	72	76
CO2	68	74
CO3	71	77
CO4	69	73
CO5	70	80
CO6	75	82

CO1 registered the highest average in both assignments, with 72% in Assignment 1 (A1) and a further improvement to 76% in Assignment 2(A2). This indicates that students have a firm grasp of the fundamental concepts and can apply them effectively across different problem contexts. The improvement between assignments suggests that students benefited from the iterative learning process, incorporating feedback from A1 to enhance their performance in A2. CO2 scores started at a slightly lower 68% in A1 but rose to 74% in A2, reflecting a substantial improvement. The initial lower performance could be attributed to the relatively higher cognitive demand of the CO2 aligned tasks in A1, possibly requiring deeper analytical reasoning. The marked improvement in A2 suggests that targeted interventions—such as additional problem-solving exercises, peer discussions, and instructor feedback—helped bridge the initial gap. CO3 demonstrated stable and strong attainment, improving from 71% in A1 to 77% in A2. The high scores across both assessments highlight students' ability to consistently apply learned concepts, especially in application- and synthesis-oriented tasks. This sustained high performance may be linked to the experiential learning components embedded in the course, enabling students to connect theory with practice effectively. CO4 attained 69% in A1 and 73% in A2, showing only modest improvement. While the upward trend indicates some learning progression, the scores remain below those of CO1 and CO3. This suggests that the skill areas associated with CO4likely involving higher-order problem solving, integration of multiple concepts, or creative design—require more reinforcement. CO5 jumped from 70% in A1 to 80% in A2, representing one of the largest gains among all COs. This suggests that the second assignment effectively reinforced skills in this outcome, possibly due to practical, hands-on components that improved student engagement and application accuracy. CO6 started strong at 75% in A1 and reached 82% in A2, the highest among all COs. This indicates exceptional mastery in the skills targeted by CO6, which may involve capstone-level tasks integrating knowledge from multiple course areas. It is observed that all COs showed positive progression from A1 to A2.The largest improvements were observed in CO5 (+10%) and CO6 (+7%).CO2 and CO4 show moderate attainment but present opportunities for targeted pedagogical interventions.

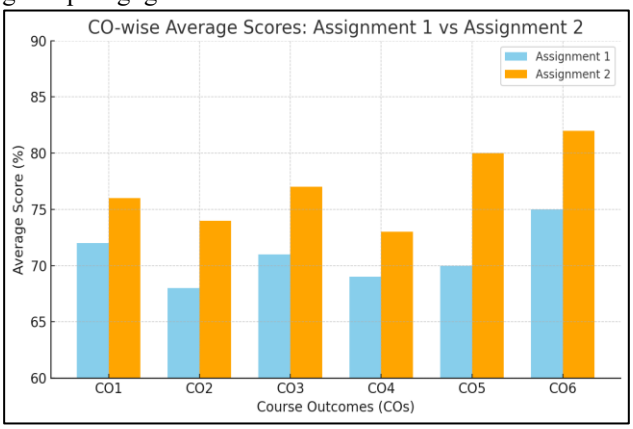

Fig. 2. Course outcome wise average Scores for Assignment 1 and Assignment 2

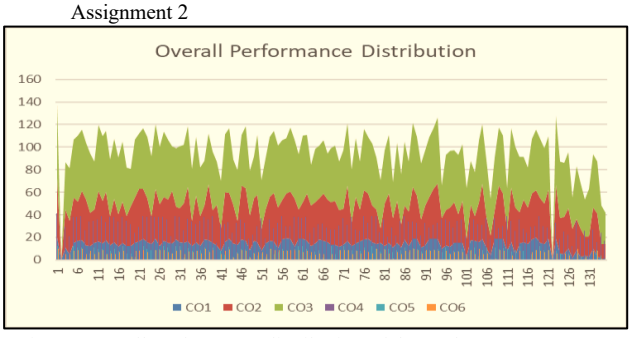

Fig. 3. Overall performance distribution of the students

VI. Conclusion

The analysis of CO-wise attainment across two sequential assignments demonstrates a positive trajectory in student performance for all six Course Outcomes. The data shows that CO1, CO3, CO5, and CO6 are strong, with consistently high scores and significant improvements, particularly in CO5 (+10%) and CO6 (+7%). This suggests that practical, hands-on components and iterative feedback mechanisms in the second assignment were highly effective in reinforcing learning. While CO2 and CO4 showed improvement, their attainment levels remained comparatively moderate, indicating the need for targeted interventions such as additional practice sessions, focused tutorials, and peer-assisted learning. The overall trend reflects successful engagement with the course content, effective application of the Kolb experiential learning cycle, and a high degree of adaptability among students in applying conceptual knowledge to practical tasks

REFERENCES

Ahmed, M., Patel, R., & Srinivasan, K. (2025). Enhancing requirement elicitation skills through AI-supported stakeholder simulations in software engineering education. *International Journal of Engineering Education*, 41(2), 245–260.

Albertini, S., Rossi, F., & Martelli, L. (2024). Reflection and iteration as drivers of transformative learning in engineering education. *European Journal of Engineering Education*, 49(1), 85–102.

Bonetti, A., Johnson, P., & Clarke, M. (2025). A systematic mapping study of experiential learning in software engineering education. *Journal of Systems and Software*, 204, 111784.

Connor, A., McCallum, A., & Dick, M. (2014). Peer review and self-assessment in software engineering projects: An experiential learning approach. *Computer Science Education*, 24(3–4), 215–232.

Crawley, E. F., Malmqvist, J., Östlund, S., Brodeur, D. R., & Edström, K. (2014). Rethinking engineering education: The CDIO approach (2nd ed.). *Cham, Switzerland: Springer*.

Devi, P., & Thendral, R. (2025). Applying Kolb's experiential learning cycle to concept-heavy engineering courses: An empirical study. *International Journal of Engineering Pedagogy*, 15(1), 45–58.

Ebadi, Y., Fernandes, M., & Mendes, A. J. (2020). Bridging the gap between academia and industry: A case study in teaching requirements engineering. *IEEE Transactions on Education,* 63(4), 285–293.

Garcia, L., Martínez, J., & Paredes, H. (2023). Mapping collaborative learning approaches in software engineering education. *Education and Information Technologies*, 28(9), 10345–10369.

Jatit Engagement Team. (2025). Improving requirement elicitation through peer learning and assessment in engineering education. *Journal of Theoretical and Applied Information Technology*, 103(5), 912–924.

Konak, A., Clark, T. K., &Nasereddin, M. (2014). Using Kolb's experiential learning cycle to improve student learning in virtual computer laboratories. *Computers & Education*, 72, 11–22.

Silva, R., Costa, C., & Pinto, A. (2021). A four-stage project-based learning model for software engineering education. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 16(3), 192–200.

WIETE Transformative Learning Team. (2023). The role of reflection and iteration in engineering education: A transformative learning perspective. *Global Journal of Engineering Education*, 25(2), 123–131.

Zowghi, D. (2011). Teaching requirements engineering through role playing: Lessons learnt. *Requirements Engineering*, 16(2), 101–116.