

A Hybrid Software Engineering Teaching Model (HSETM): Integrating Traditional and Contemporary Pedagogical Approaches

Ji Myong Song^{1,*}, Hwang Myong Hyok², Kim KwangHwi³, Choe Yong Jin⁴

Abstract

Software engineering education faces a persistent dichotomy between traditional lecture-based instruction, which effectively delivers theoretical frameworks but inadequately develops practical competencies, and contemporary pedagogical approaches such as problem-based learning (PBL) and flipped learning, which enhance learner engagement but risk weakening theoretical foundations. This study proposes and empirically validates the Hybrid Software Engineering Teaching Model (HSETM), a systematic integration of traditional and contemporary pedagogical methods designed to overcome this dichotomy. Grounded in the Outcome-Based Education (OBE) framework, HSETM sequentially integrates four instructional strategies: (1) traditional lectures for conceptual foundation, (2) flipped learning for preparatory engagement, (3) project-based learning for applied problem-solving, and (4) experiential learning (CDIO-Kolb integrated) for cyclical knowledge construction. A mixed-methods research design was employed with 147 undergraduate software engineering students at a four-year university in South Korea. Participants were assigned to an experimental group ($n=74$) receiving HSETM intervention and a control group ($n=73$) receiving traditional lecture-based instruction over 15 weeks. Quantitative analysis using ANCOVA revealed that the experimental group demonstrated significantly higher academic achievement (adjusted mean: 86.12 vs. 73.51, $F=47.23$, $p<.001$, $\eta^2=.32$), learning engagement (4.21/5.00 vs. 3.18/5.00, $d=1.71$), and core competencies including requirements analysis, teamwork, and communication skills compared to the control group. Two-way ANOVA identified a significant interaction effect between instructional method and prior knowledge level ($F=12.34$, $p<.01$), indicating that HSETM was particularly effective for learners with low prior knowledge. Qualitative analysis of interviews and reflective journals revealed four emergent themes: (1) recursive connection between theory and practice, (2) psychological safety through staged difficulty progression, (3) activation of peer learning, and (4) challenges including time burden and team conflict. This study contributes to software engineering education by presenting an empirically validated hybrid model that transcends the traditional-innovative dichotomy, offering both theoretical insights into pedagogical integration principles and practical guidelines for instructional design.

*Author for Correspondence

Ji Myong Song
E-mail: jmskys2009@gmail.com

¹School of Information Science and Technology, Kim Chaek University of Technology, Democratic People's Republic of Korea

²Faculty of Information Science and Technology, O Jung Hup Chongjin University of Education, Democratic People's Republic of Korea

³Faculty of Information Science and Technology, Kim Hyong Jik University of Education, Democratic People's Republic of Korea

⁴Faculty of Education Science, Kim Hong Jik University of Education, Democratic People's Republic of Korea

Received Date: March 24, 2026

Accepted Date: May 19, 2026

Published Date: June 20, 2026

Citation: Ji Myong Song, Hwang Myong Hyok, Kim KwangHwi, Choe Yong Jin. A Hybrid Software Engineering Teaching Model (HSETM): Integrating Traditional and Contemporary Pedagogical Approaches. *Current Trends in Information Technology*. 2026; 16(2): 33–43p.

progression, (3) activation of peer learning, and (4) challenges including time burden and team conflict. This study contributes to software engineering education by presenting an empirically validated hybrid model that transcends the traditional-innovative dichotomy, offering both theoretical insights into pedagogical integration principles and practical guidelines for instructional design.

Keywords: Software engineering education, hybrid teaching model, HSETM, flipped learning, project-based learning, experiential learning, mixed-methods research

INTRODUCTION

Background and Problem Statement

The advent of the Fourth Industrial Revolution and emergence of software-centric societies have

substantially elevated the importance of software engineering education. Software engineering professionals are expected to possess technical programming skills and demonstrate competencies in requirements analysis, system design, quality assurance, project management, and effective team collaboration [1, 2]. However, a significant gap persists between industry expectations and graduate competencies, with software project failures frequently attributed to deficiencies in requirements engineering and communication skills that are inadequately addressed in traditional curricula [2].

Traditionally, software engineering courses have been delivered through lecture-based instruction, wherein instructors transmit theoretical knowledge and students passively receive the information [3]. While this approach efficiently delivers comprehensive theoretical frameworks, it exhibits fundamental limitations: (1) a disconnect between theoretical knowledge and practical application, (2) passive learner engagement that promotes surface rather than deep learning, and (3) insufficient development of teamwork and communication competencies essential for professional practice [4, 3].

In response to these limitations, various contemporary pedagogical approaches have been introduced in software engineering education. Flipped learning, wherein students engage with lecture content before class and participate in active learning activities during class, has been shown to have positive effects on learner engagement and achievement [5]. Project-based learning (PBL) engages students in authentic problem-solving experiences that develop integrated competencies [6]. Experiential learning models, particularly those integrating the CDIO (Conceive-Design-Implement-Operate) framework with Kolb's experiential learning cycle, facilitate cyclical knowledge construction through concrete experience, reflective observation, abstract conceptualization, and active experimentation [1, 7].

However, the adoption of contemporary approaches has not been successful. Flipped learning and PBL may impose cognitive overload on learners with insufficient prior knowledge, potentially resulting in disengagement rather than enhanced learning [8]. Moreover, project-based activities conducted without adequate theoretical grounding may devolve into trial-and-error exercises that fail to achieve the intended learning objectives [8, 3].

The Need for Integrated Approaches

Recent scholarship has recognized that the dichotomy between traditional and contemporary pedagogy represents a false choice. Traditional lectures excel in structured knowledge transmission, whereas contemporary approaches foster active engagement and skill development [3]. Rather than selecting one approach over another, an integrated model that leverages the strengths of both approaches while mitigating their respective weaknesses may offer optimal learning outcomes.

Emerging research has explored various integrations, such as flipped learning combined with Havruta methods [5], OBE-PBL frameworks [6], and CDIO-Kolb integrated models [1]. However, existing integrated models exhibit several limitations: (1) most integrate only two pedagogical approaches, failing to comprehensively leverage the full spectrum of instructional strategies; (2) limited empirical validation using rigorous experimental designs and mixed-methods approaches [9]; and (3) insufficient attention to individual learner differences, particularly prior knowledge levels, in examining differential model effectiveness.

Research Purpose and Questions

This study addresses these limitations by proposing and empirically validating the Hybrid Software Engineering Teaching Model (HSETM), a systematic integration of traditional lectures, flipped learning, project-based learning, and experiential learning within an Outcome-Based Education framework. The model is designed to provide structured theoretical foundations while progressively engaging learners in increasingly complex, applied, and experiential activities.

The specific research questions guiding this study are:

- *RQ1*: Does the Hybrid Software Engineering Teaching Model (HSETM) produce significantly higher academic achievement in software engineering than traditional lecture-based instruction?
- *RQ2*: What effects does HSETM have on learner engagement, satisfaction, and core competencies (requirements analysis, teamwork, communication)?
- *RQ3*: Does the effectiveness of HSETM vary according to learners' prior knowledge levels (high vs. low)?

THEORETICAL FRAMEWORK

Traditional Approaches to Software Engineering Education

Traditional software engineering education has predominantly employed instructor-centered teaching methods. This approach is characterized by a systematic presentation of theoretical content covering software development lifecycles, requirements engineering methodologies, design patterns, testing strategies, and project management frameworks [3]. From Bloom's taxonomy perspective, lecture-based instruction effectively achieves knowledge and comprehension-level learning objectives by efficiently delivering standardized content to large cohorts.

However, traditional approaches demonstrate significant limitations when evaluated against the competencies required by contemporary software development practices. First, a theory-practice gap emerges as students struggle to apply classroom-acquired concepts to authentic problem situations [1]. Second, passive reception limits cognitive engagement and may encourage surface rather than deep learning approaches to learning. Third, the collaborative nature of professional software development, emphasized in agile methodologies and contemporary practice, receives insufficient attention in traditional lecture formats [4].

Contemporary Pedagogical Approaches

Flipped Learning

Flipped learning inverts traditional instructional models by delivering direct instruction through pre-class video lectures and utilizing class time for active learning [5]. This approach enables learners to engage with foundational content at their own pace, while dedicating synchronous class time to problem-solving, discussion, and application activities. Research has demonstrated that flipped learning enhances learner interest, engagement, and achievement when implemented effectively [5]. However, challenges include ensuring learner preparation and managing the increased cognitive load of students with limited prior knowledge.

Project-Based Learning (PBL)

Project-based learning engages students in authentic and complex problems that require sustained inquiry and culminate in tangible products or presentations. In software engineering education, PBL typically involves team-based software development projects that address real-world requirements [6]. The OBE-PBL integrated model has demonstrated effectiveness in achieving learning outcomes (achievement rates 0.78-0.79) through library management system development cases [6]. PBL develops integrated competencies, including requirements analysis, design, implementation, testing, and project management, while fostering teamwork and communication skills.

Experiential Learning and CDIO Framework

Experiential learning, grounded in Kolb's experiential learning cycle (concrete experience → reflective observation → abstract conceptualization → active experimentation), emphasizes learning through experience and reflection [7]. The CDIO (Conceive-Design-Implement-Operate) framework provides a structured approach to engineering education that integrates technical knowledge with professional skills through full-cycle product development experiences. The CKIL (CDIO-Kolb Integrated Learning) model has demonstrated positive effects on learner engagement and teamwork competency among undergraduate engineering students [1].

Theoretical Synthesis: The Need for Integration

The complementary strengths and weaknesses of traditional and contemporary approaches suggest that integration, rather than selection, optimizes learning outcomes. Traditional lectures provide structured theoretical foundations that are essential for subsequent applied activities. Flipped learning prepares learners for active engagement while maintaining flexibility. PBL develops applied competencies through authentic problem solving. Experiential learning facilitates deep learning through cyclical experiences and reflections [10].

This theoretical synthesis aligns with broader educational frameworks that emphasize learner-centered approaches, active learning, and competency development while acknowledging the continued value of structured knowledge transmission. The proposed Hybrid Software Engineering Teaching Model (HSETM) operationalizes this synthesis through the sequential integration of four instructional strategies within an OBE framework, as elaborated in the following section.

METHODOLOGY**Research Design**

This study employed a mixed-methods research design that combined quantitative experimental methods with qualitative inquiry. The quantitative component utilized a quasi-experimental, nonequivalent control group pretest-posttest design. The qualitative component employed grounded theory methodology to explore learners' experiences and perceptions of the instructional intervention.

The Hybrid Software Engineering Teaching Model (HSETM)

The Hybrid Software Engineering Teaching Model (HSETM) was the central contribution of this study. The HSETM integrates four instructional strategies within an Outcome-Based Education framework, organizing them sequentially to support progressive knowledge construction and skill development.

Phase 1: Traditional Lecture (20% of Instructional Time)

The first phase establishes theoretical foundations through structured lectures delivered by the instructor. This phase focuses on core concepts, including software development lifecycles, requirements engineering principles, system design methodologies, quality assurance strategies, and project management frameworks, as shown in Figure 1.

Lectures incorporate case studies and examples to contextualize the theoretical content. From Bloom's taxonomy perspective, this phase targets knowledge and comprehension levels.

Phase 2: Flipped Learning (25% of Instructional Time)

The second phase implements flipped learning to transition learners from passive reception to active engagement. Before class, learners engage with short video lectures (10–15 min) covering preparatory content. Class time is dedicated to quizzes that assess preparation, structured discussions that apply concepts to scenarios, and guided problem-solving activities. This phase targets comprehension and application levels while introducing collaborative learning.

Phase 3: Project-Based Learning (35% of Instructional Time)

The third phase engages learners in authentic software development projects completed in teams of four to five members. Projects are designed to address realistic requirements based on industry cases. The activities include requirements elicitation and analysis, system design and architecture, peer evaluation of design artifacts, and iterative refinement based on feedback. This phase targets application and analysis levels while developing teamwork and communication skills.

Phase 4: Experiential Learning (20% of Instructional Time)

The fourth phase operationalizes Kolb's experiential learning cycle within the CDIO framework. Learners engage in concrete experiences through prototype implementation, followed by reflective observations guided by structured reflective journals. Abstract conceptualization occurs through the

synthesis of implementation experiences with theoretical principles. Active experimentation involves iterative improvements based on reflective insights. This phase targets the synthesis and evaluation levels, completing the learning cycle.

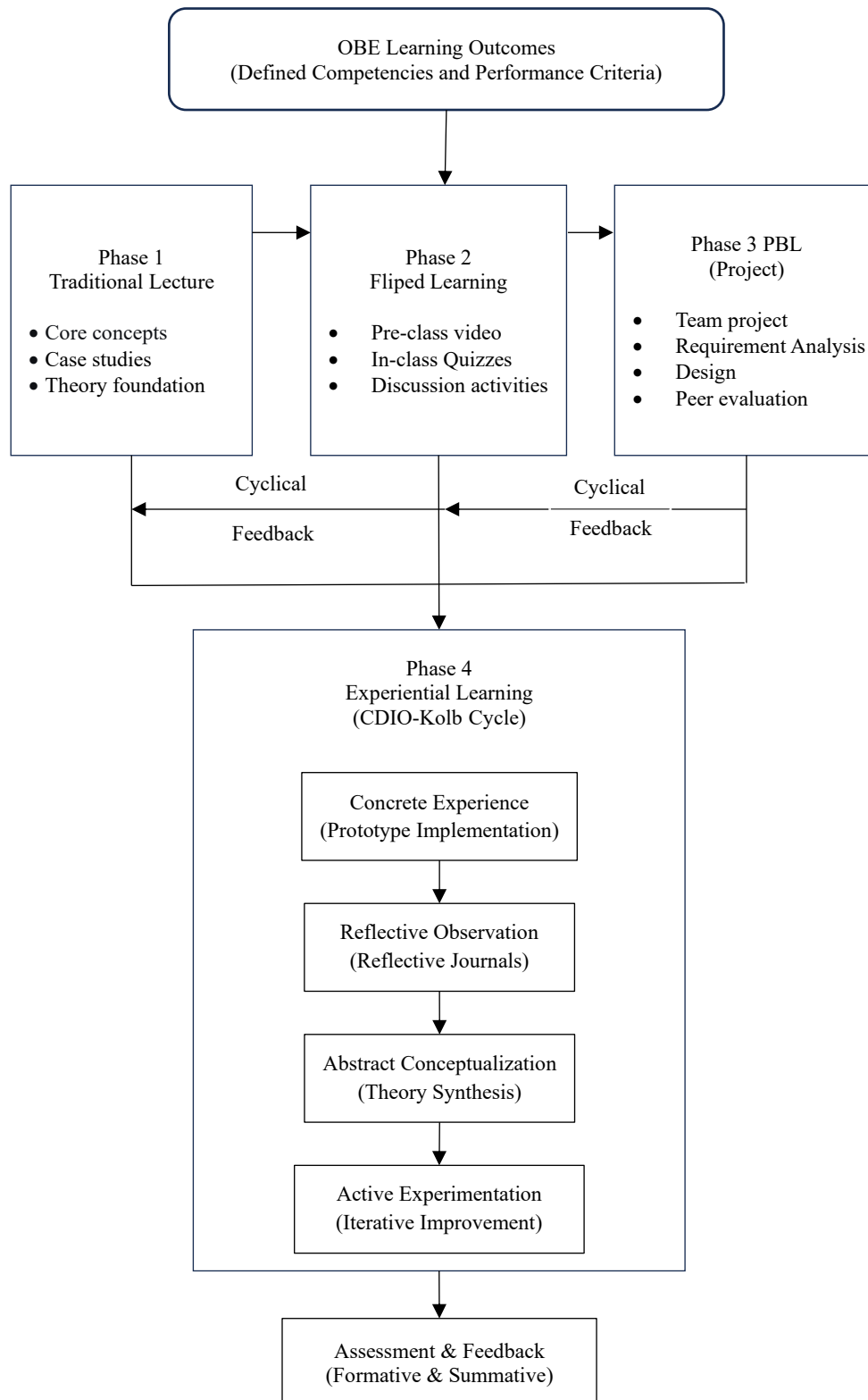


Figure 1. HSETM conceptual framework.

Table 1. HSETM phase characteristics.

Phase	Instructional strategy	Weight	Primary activities	Learning objectives (bloom)	Theoretical basis
1	Traditional lecture	20%	Core concepts, case studies, theory foundation	Remember, understand	Cognitivism
2	Flipped learning	25%	Pre-class video, in-class quizzes, discussion	Understand, apply	Social constructivism
3	Project-based learning	35%	Team project, requirements analysis, design, peer evaluation	Apply, analyze	Social constructivism
4	Experiential learning	20%	Prototype implementation, reflective journals, iterative improvement	Synthesize, evaluate	Experiential learning (Kolb, CDIO)

Cyclical Feedback Mechanisms

The HSETM incorporates cyclical feedback mechanisms that enable recursive learning. Learners encountering difficulties during PBL implementation may return to Phase 1 lecture materials for theoretical clarifications. Insights gained during experiential learning may inform revisions to the requirements analysis conducted during Phase 3. These feedback loops prevent linear, compartmentalized learning and promote integrated knowledge construction (Table 1).

Participants

The participants were 147 undergraduate students enrolled in the Software Engineering course in the Department of Computer Engineering at a four-year university in North Hamgyong Province of the DPRK. Participants were assigned to experimental ($n=74$) and control ($n=73$) groups based on their student ID numbers (odd/even). Both groups were taught by the same instructor using identical textbooks and learning objectives for 15 weeks.

Pretest equivalence was confirmed through independent t-tests, which showed no significant differences in prior software engineering knowledge ($t=.87$, $p=.39$), grade distribution ($\chi^2=2.31$, $p=.51$), or gender distribution ($\chi^2=.43$, $p=.67$).

Instruments

- *Academic achievement test*: A 40-item instrument assessing knowledge across four software engineering domains: requirements engineering, design, testing, and project management (ten items per domain). Items were developed based on prior research [2, 9] and reviewed by three software engineering faculty members and two industry practitioners for content validity (CVI=.89). The pretest reliability (Cronbach's α) was .82, and the posttest reliability was .87.
- *Learning engagement scale*: A 12-item instrument assessing the cognitive, affective, and behavioral dimensions of engagement ($\alpha=.91$).
- *Satisfaction survey*: A 6-item instrument assessing overall course satisfaction ($\alpha=.88$).
- *Core competency assessment*: Self-report measures for requirements analysis, teamwork, and communication competencies (five items each), supplemented by instructor evaluation of project deliverables.
- *Qualitative data*: Semi-structured interviews (approximately 40 min each) with 20 experimental group participants and 15 reflective journals.

Data Analysis

Quantitative data were analyzed using SPSS version 28.0. Pretest equivalence was assessed using independent t-tests. Hypothesis testing employed analysis of covariance (ANCOVA) with pre-test scores as covariates and two-way ANOVA to examine interaction effects. Effect sizes were reported as partial eta-squared (η^2) and Cohen's d .

Qualitative data were analyzed using the grounded theory methodology, which involves open, axial, and selective coding. Trustworthiness was established through peer debriefing and member checking.

RESULTS

Academic Achievement (RQ1)

ANCOVA results revealed significant differences in academic achievement between the experimental and control groups after controlling for pretest scores. The experimental group demonstrated a mean post-test score of 86.34 (SD=7.82) compared to 73.28 (SD=10.14) for the control group. The adjusted mean difference was significant ($F=47.23$, $p<.001$, $\eta^2=.32$), indicating a large effect size (Table 2).

Domain-specific analysis revealed consistent superiority of the experimental group across all assessed domains: requirements engineering (88.45 vs. 71.23), design (85.67 vs. 74.12), testing (84.23 vs. 72.45), and project management (86.12 vs. 75.34). The largest difference emerged in requirements engineering, suggesting that PBL experiences involving authentic user requirements analysis contributed substantially to this area.

Learning Engagement, Satisfaction, and Core Competencies (RQ2)

The experimental group participants reported significantly higher learning engagement, satisfaction, and core competency scores than the control group participants (Table 3).

The very large effect size for learning engagement ($d=1.71$) suggests that the integration of varied instructional strategies effectively captured and sustained learner attention. The substantial improvement in teamwork competency ($d=1.37$) indicates that team-based activities in Phases 3 and 4 contributed significantly to the development of collaborative skills.

Differential Effects by Prior Knowledge Level (RQ3)

A two-way ANOVA was used to examine whether HSETM effectiveness varied according to learners' prior knowledge level (high vs. low, defined as the top and bottom 30% of pretest scores) (Table 4).

Significant main effects were observed for group ($F=45.23$, $p<.001$) and prior knowledge ($F=38.91$, $p<.001$). Critically, a significant interaction effect was identified ($F=12.34$, $p<.01$), indicating that HSETM was differentially effective according to the prior knowledge. The model produced substantially larger gains for low prior knowledge learners (+17.07 points) than for high prior knowledge learners (+8.67 points). This suggests that the HSETM may help reduce achievement gaps by providing structured scaffolding for learners with weaker foundational knowledge.

Table 2. Academic achievement descriptive statistics and ANCOVA results.

Group	Pretest mean (SD)	Posttest mean (SD)	Adjusted mean	F	p	η^2
Experimental (n=74)	54.27 (9.83)	86.34 (7.82)	86.12	47.23	<0.001	0.32
Control (n=73)	53.94 (10.21)	73.28 (10.14)	73.51			

Table 3. Learning engagement, satisfaction, and core competencies.

Variable	Experimental (n=74)	Control (n=73)	t	p	Cohen's d
Learning Engagement (5pt)	4.21 (0.52)	3.18 (0.67)	10.34	<0.001	1.71
Course Satisfaction (5pt)	4.38 (0.47)	3.45 (0.71)	9.27	<0.001	1.53
Requirements Analysis (5pt)	4.12 (0.58)	3.21 (0.82)	7.68	<0.001	1.28
Teamwork Competency (5pt)	4.31 (0.51)	3.42 (0.76)	8.23	<0.001	1.37
Communication Competency (5pt)	4.08 (0.62)	3.51 (0.69)	5.31	<0.001	0.87

Table 4. Posttest achievement by group and prior knowledge level.

Group	High prior knowledge (n=44)	Low prior knowledge (n=44)	Overall
Experimental	91.23 (5.21)	81.45 (6.34)	86.34
Control	82.56 (7.12)	64.38 (8.21)	73.28
Difference	+8.67	+17.07	+13.06

Qualitative Findings

Grounded theory analysis of interview transcripts and reflective journals revealed four major themes characterizing learners' experiences with HSETM (Appendix B).

Theme 1: Recursive Theory-Practice Connection

Learners consistently reported that the cyclical structure of the HSETM enabled meaningful connections between theoretical concepts and practical applications. Rather than experiencing theory and practice as separate domains, the participants described a recursive process in which theoretical understanding informed practice and practical experience deepened theoretical comprehension.

"When I first heard the lectures, I wondered why we needed to document requirements so meticulously. However, when communication problems arose during the team project, I realized viscerally why the lecture content was important. I then went back to review the lecture slides." (Participant 7, Male, Year 3).

Theme 2: Psychological Safety Through Staged Difficulty

Participants appreciated the graduated progression from lectures to flipped learning, PBL, and experiential activities. The staged approach reduced cognitive load and anxiety by establishing foundational knowledge before introducing complex, applied tasks.

"Last year in another course, we started with team projects immediately and I had no idea what to do—it was really stressful. However, this course taught concepts first, followed by discussions, and only then did we start the project. I felt much more prepared." (Participant 12, Female, Year 4).

Theme 3: Activation of Peer Learning

The flipped learning discussions and PBL team activities activated peer learning processes, wherein learners supported each other's understanding. Participants valued the opportunity to learn from peers with complementary knowledge and skills to their own.

"When I had questions, it was not always easy to ask the instructor immediately. However, I can discuss with team members and solve problems together. This naturally improved our teamwork and created an atmosphere in which we helped each other learn." (Participant 5, Male, Year 3)

Theme 4: Implementation Challenges

The participants also identified the challenges associated with HSETM. The time burden was frequently mentioned, particularly during periods when multiple activities coincided. Team conflict, although less common, was reported as a source of stress by some participants.

"There were times when it felt like too much—watching videos, completing quizzes, working on the project all at once. During the midterm period, it was particularly overwhelming." (Participant 18, Female, Year 3)

DISCUSSION

Summary of Findings

This study proposed and empirically validated the Hybrid Software Engineering Teaching Model (HSETM), which integrates traditional lectures, flipped learning, project-based learning, and experiential learning within an Outcome-Based Education framework. Three major findings emerged from the results.

First, HSETM produced significantly higher academic achievement than traditional instruction, with a large effect size ($\eta^2=.32$). This finding supports the theoretical proposition that integrating multiple pedagogical approaches yields superior learning outcomes compared with single-method instruction. The particularly strong performance in requirements engineering suggests that authentic project experiences combined with theoretical instruction effectively develop critical competency.

Second, HSETM positively influenced the students' learning engagement, satisfaction, and core competencies. The very large effect on learning engagement ($d=1.71$) indicates that varied instructional strategies effectively sustained learner interest and involvement. Qualitative findings illuminate this result, with learners describing recursive theory-practice connections and psychological safety through staged progression as mechanisms that support engagement.

Third, HSETM demonstrated differential effectiveness according to prior knowledge, with low-prior-knowledge learners benefiting substantially more than high-prior-knowledge learners. This interaction effect suggests that the structured scaffolding provided by the sequential integration of instructional strategies—particularly the theoretical foundation established in Phase 1—may compensate for weaker foundational knowledge, potentially reducing achievement gaps.

Theoretical Contributions

This study makes several theoretical contributions to software engineering education research.

First, it extends the literature by demonstrating that the traditional-innovative dichotomy in pedagogical approaches represents a false choice. Rather than replacing traditional lectures with contemporary methods, systematic integration leverages the strengths of each method while mitigating their weaknesses. This finding aligns with recent studies recognizing the continued value of structured knowledge transmission within learner-centered frameworks [8].

Second, this study provides empirical validation for integrated pedagogical design principles. The sequential structure—foundation (lectures) → preparation (flipped) → application (PBL) → synthesis (experiential)—operationalizes the principles of cognitive load theory by gradually increasing task complexity while maintaining supportive structures. The significant interaction effect of prior knowledge supports the scaffolding function of this sequential design.

Third, this study advances our understanding of how experiential learning can be systematically integrated with other instructional approaches. The CDIO-Kolb cycle implemented in Phase 4 did not operate in isolation but was supported by preceding phases that provided theoretical foundations and applied experiences. This integration addresses the criticism that experiential learning may lack systematic knowledge development.

Practical Implications

The findings offer practical implications for software engineering educators and instructional designers.

First, the HSETM provides a concrete and implementable framework for course design. Educators need not choose between traditional and contemporary approaches but can integrate multiple strategies sequentially. The phase weights (20%-25%-35%-20%) offer a starting point that can be adapted to specific course contexts and student needs.

Second, the differential effectiveness for low-prior-knowledge learners suggests that HSETM may serve as an inclusive instructional approach that supports diverse learner populations. Institutions concerned about achievement gaps may consider hybrid models that provide structured scaffolding while engaging all learners in authentic, project-based experiences.

Third, the qualitative findings regarding time burden highlight the importance of careful workload management. While integrating multiple approaches offers pedagogical benefits, instructors must ensure that the total workload remains manageable. Strategies may include proportionally reducing lecture time as active learning components increase and aligning assessment timing to avoid multiple deadlines.

Limitations and Future Research

This study has several limitations. First, the study was conducted at a single university with 147 participants, which limits its generalizability. Future research should examine HSETM implementation across diverse institutional contexts and cultural settings.

Second, the study assessed outcomes immediately after the 15-week intervention. Longitudinal research is needed to examine whether learning gains persist and transfer to subsequent courses or professional practices.

Third, while this study controlled for prior knowledge, other individual difference variables (learning styles, motivation, and self-regulation) were not examined. Future research should investigate whether HSETM effectiveness varies according to additional learner characteristics.

Fourth, this study did not decompose the relative contributions of each phase to the overall outcomes. Component analysis research could identify which elements of HSETM are most critical for specific learning outcomes.

Fifth, the study employed a single instructor for both the experimental and control conditions. While this controlled for instructor effects, it also means that the results may be influenced by instructor expertise in implementing the model. Replication studies with multiple instructors would strengthen the confidence in the findings.

CONCLUSION

This study proposed and empirically validated the Hybrid Software Engineering Teaching Model (HSETM), demonstrating that the systematic integration of traditional lectures, flipped learning, project-based learning, and experiential learning within an Outcome-Based Education framework produces superior learning outcomes compared to traditional instruction. The model proved particularly beneficial for learners with low prior knowledge, suggesting the potential for reducing achievement gaps. Qualitative findings illuminated the mechanisms underlying these effects, including recursive theory-practice connections and psychological safety through staged progression.

This study contributes to software engineering education by transcending the traditional-innovative dichotomy, offering both theoretically grounded design principles and empirically validated practical guidelines. As software engineering education continues to evolve in response to industry demands and technological advances, hybrid models that integrate diverse pedagogical approaches offer promising pathways for developing the comprehensive competencies required of contemporary software professionals.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Professor Kim Gwang Chol, for his invaluable guidance, encouragement, and insightful feedback throughout all the stages of this research. I am also sincerely grateful to Professor Nam Il Ju for his careful review and constructive comments on the earlier versions of this manuscript.

I appreciate the faculty members of the Faculty of Information Science and Technology at Kim Chaek University of Technology for their assistance in designing the instructional model and implementing the course. I am also thankful to Kwak Gyong Jin for his dedication and support in the course operations.

Above all, I owe my deepest gratitude to the 147 students who enrolled in the Software Engineering course and participated in this study with remarkable commitment and sincerity. Over the 15 weeks of instruction, they generously shared their experiences, thoughts and reflections. This research would not have been possible without their enthusiastic participation and honest feedback.

Finally, I would like to express my heartfelt appreciation to my family for their unwavering support and encouragement during this journey.

REFERENCES

1. Pudumalar S, Parkavi R. An integrated CDIO-Kolb framework for experiential learning in software engineering. *J Eng Educ Transform*. 2026;288-93.
2. Garousi V, Giray G, Tüzün E, Catal C, Felderer M. Aligning software engineering education with industrial needs: a meta-analysis. *J Syst Softw*. 2019;156:65-83. doi:10.1016/j.jss.2019.06.044.
3. González-Guerra LH, Pérez-Murueta P. Evolution of educational paradigms: from knowledge transfer to competency-based learning in computer science. In: *IEEE Global Engineering Education Conference (EDUCON)*; 2025. p. 1-5. doi:10.1109/EDUCON62633.2025.11016528.
4. Kulenavar ND, C S, F M, B I, Hanchinamani GS, Jayalaxmi GN. Students unlock the power of real systems: an experiential learning in system software course. *J Eng Educ Transform*. 2024;37:858-64. doi:10.16920/jeet/2024/v37is2/24132.
5. Li J, Zhang X, Hu Z. The design and application of flip classroom teaching based on computer technology. *Int J Emerg Technol Learn*. 2018;13:95. doi:10.3991/ijet.v13i10.9453.
6. Zhang J, Chen G, Zhang R, Chen Y, Yang H. Exploration of software engineering teaching cases. In: *Proceedings of the 2025 10th International Conference on Modern Management, Education and Social Sciences (MMET 2025)*. Springer Nature; 2025. p. 284.
7. Kolb DA. *Experiential Learning: Experience as the Source of Learning and Development*. Upper Saddle River (NJ): FT Press; 2014.
8. Pashchenko O, Koroviaka Y, Mamaikin O, Kozhushkina T, Rastsvietaiev V. Effectiveness of blended learning in the informatics course: analysis of online and offline formats. *Молодь і Ринок*. 2025;7-8:50-6. doi:10.24919/2308-4634.2025.334056.
9. Gordillo A, López-Fernández D. Are educational escape rooms more effective than traditional lectures for teaching software engineering? A randomized controlled trial. *IEEE Trans Educ*. 2024;67:660-8. doi:10.1109/TE.2024.3403913.
10. Garrison DR, Vaughan ND. *Blended Learning in Higher Education: Framework, Principles, and Guidelines*. San Francisco (CA): John Wiley & Sons; 2008.

APPENDICES

Appendix A. HSETM implementation timeline (15-week schedule).

Week	Phase	Activities	Assessment
1	Pretest	Baseline knowledge assessment	Pretest
2–4	Phase 1	Traditional lectures: SDLC, requirements, design	Formative quizzes
5–7	Phase 2	Flipped learning: video lectures, in-class discussions	Weekly quizzes, participation
8–11	Phase 3	PBL: Team project initiation, requirements analysis, design	Project milestones, peer evaluation
12–14	Phase 4	Experiential learning: Implementation, reflection, iteration	Reflective journals, prototype
15	Posttest	Final assessment	Posttest, project submission

Appendix B. Sample reflective journal prompts.

1. Please describe a specific challenge you encountered during the project implementation. How did you approach the solution?
2. How does your implementation experience connect to theoretical concepts covered in lectures?
3. What would you do differently if you were to approach this task again?
4. Please describe a contribution you made to your team's work this week. What did you learn from this experience?
5. What questions remain for you about requirements engineering or system design?