ISSN:3007-9632 DOI: https://doi.org/10.70114/aimedr.2025.2.1.P140

Research on Outcome-Based Education (OBE) in Software Engineering Curriculum Reform

Wenshan Li [1, a] *, Ao Liu [2, b], Chenyi Huang [2, c]

[1] School of Cybersecurity, Chengdu University of Information Technology, Chengdu, China;

[2] School of Cyber Science and Engineering, Sichuan University, Chengdu, China.

[a] helenali@cuit.edu.cn, [b] aliu@scu.edu.cn, [c] huangchenyi@stu.scu.edu.cn

Abstract. To meet engineering education accreditation requirements and enhance software talent training quality, this study introduces the Outcome-Based Education (OBE) concept into software engineering curriculum reform. Addressing issues such as ambiguous course objectives, insufficient practical components, and monolithic assessment mechanisms, we systematically reconstruct ability-oriented teaching objectives, optimize content and pedagogy, establish a diversified process-based evaluation system, and validate the reform's effectiveness through a practical case study. Results demonstrate that OBE-driven reform significantly improves students' engineering practice capabilities and professional competencies.

Keywords: Outcome-based education (OBE); Software engineering; Course design; Curriculum reform; Engineering education accreditation.

1. Introduction

Accreditation of engineering education has taken off worldwide with its three interconnected principles of: student-centeredness, outcome-based orientation, and continuous improvement. Outcome-Based Education (OBE), the driving force behind accreditation, focuses on designing and implementing educational activities from the vantage point of students' expected competencies at graduation (i.e., students' learning outcomes) [1]. China's Engineering Education Accreditation Standards mandate that programs establish outcome-oriented internal evaluation mechanisms [2]. Current software engineering courses, however, mostly follow a traditional course design, where theoretical knowledge is prioritized over practices. To be more specific, the evaluation of a student's course grade is mainly based on a written final exam. This strategy leads to severe gaps between what students are actually learning from school and the skills the industry really needs, thus failing to develop students' engineering abilities [3]. In this paper, we present a systematic OBE approach to re-design course objectives, course content and course assessments, aiming at to improve students' overall engineering capabilities.

2. Current Challenges and Necessity for OBE

Software engineering programs often have poorly-defined learning goals, consisting of general statements (e.g., "mastery of basic theories") without defining measurable goals, such as "solving complex engineering problems" or "ability to work as part of a team". Most of teaching time is spent on theoretical lectures. The practical modules are often not relevant to industry workflows (e.g., agile development, DevOps, secure coding). This disconnect means students can leave programs with outdated skills and competencies not being effectively developed [4]. The traditional course assessment is largely centered on final exams, making it unable to keep track of skill development over time (e.g., requirements analysis, coding standards, team contribution), affecting validity outcomes. Surveys of professional practice show that over 60% of employers will place high value on graduates' hands-on experience of modern toolchains. Much of what is typically done in the traditional curriculum may fail to provide students with content or knowledge to do this work [5].

Adopting OBE is imperative. Its backward design principle [1] requires:

ISSN:3007-9632 DOI: https://doi.org/10.70114/aimedr.2025.2.1.P140

(1) Defining graduation competencies (based on program goals, industry needs);
(2) Aligning content, pedagogy, and assessments with these competencies;
(3) Ensuring all activities drive outcome attainment.

This aligns with the Engineering Education Accreditation Standards' core logic: "graduation requirements support program objectives; course objectives support graduation requirements". Moreover, it is increasingly supported by evidence of its efficacy in technical disciplines [6].

3. OBE-Driven Reform Strategies

3.1 Objective Restructuring: Outcome-Centered Design

Objectives are derived from program graduation requirements using OBE's backward design [1], which can be further divided into four objectives. In a software engineering course, these objectives include solving complex problems, design/development, teamwork and project management. The reformed outcome-centered course design is illustrated in Figure 1.

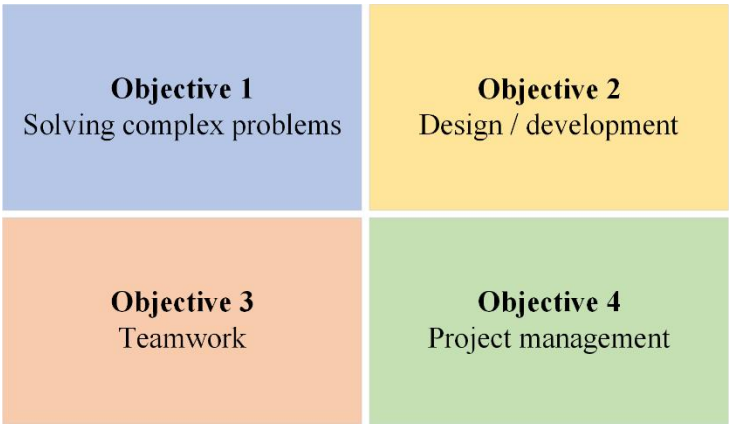


Fig. 1 Software engineering course design based on the OBE concept.

Objective 1 (Solving complex problems): Independently conduct requirements analysis, modeling, and specification drafting for medium-scale systems.

Objective 2 (Design/development): Select architectures/design patterns; implement and unit-test systems.

Objective 3 (Teamwork): Fulfill team roles; collaborate effectively; deliver team artifacts.

Objective 4 (Project management): Use tools (e.g., JIRA, Git) for task tracking, version control, and risk management.

3.2 Content Optimization: Integrating Practice and Innovation

Content is modularized to target objectives:

(1) Theory-practice integration: Embed core knowledge (e.g., software processes, design patterns) into project workflows.

(2) Industry-aligned modules: Add "User Story Development", "Secure Code Auditing", "Microservices Architecture", "CI/CD Pipelines" and "Risk Management Simulation". These align with competency frameworks identified as critical for early-career software engineers [7].

(3) Project-driven learning: Simulate full lifecycle projects (e.g., "Smart Campus Trading Platform," "IoT Device Manager") to enable learning by doing.

3.3 Assessment Enhancement: Diversified Process Evaluation

To improve the evaluation process in the software engineering course, a multi-stage evaluation system is established. For each student, the final grade is based on two parts: process-based evaluation and summative evaluation. Compared with the traditional course design where the evaluation is mainly based on a final exam, the reformed course design places a higher value on

ISSN:3007-9632 DOI: https://doi.org/10.70114/aimedr.2025.2.1.P140
process-based evaluation, and the written tests are even optional. The evaluation system is illustrated in Figure 2.

(1) Process-based (60-70%):

Milestone reviews (requirements/docs, code, tests);

Peer evaluations (Scrum-style);

Participation in technical reviews/code walkthroughs.

(2) Summative (30-40%):

Project defense with industry-expert panels;

Written tests on core theories (optional).

Rubrics are explicitly mapped to objectives to enable continuous improvement.

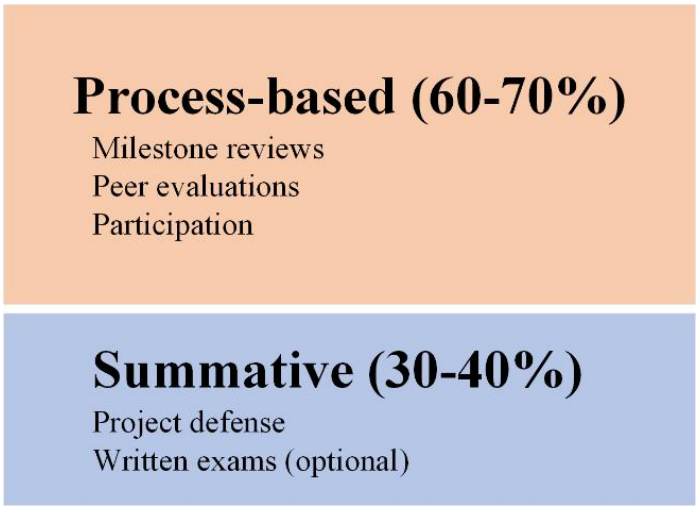


Fig. 2 The OBE-driven evaluation system design for a Software engineering course.

4. Case Study: Implementing OBE in Software Engineering

A semester-long pilot was conducted in a software engineering course.

4.1 Case Description

Student teams (4-5 members) developed self-selected projects under simulated industry conditions. One team built a "Network Security Log Analysis and Alert System."

Objective alignment: Project tasks explicitly targeted Objectives 1-4 (e.g., Git-based collaboration, risk management plans).

Content execution: Theoretical lessons (e.g., UML, design patterns) were synchronized with project phases. Added modules included "Log Data Compliance Analysis" and "Secure Alert Module Coding."

Evaluation: Four milestone reviews (requirements/design/Alpha/Beta) and peer evaluations culminated in a final defense judged by faculty and an industry engineer.

4.2 Outcomes

Student feedback: Project-driven learning boosted engagement and practical understanding. Skills in requirements analysis, system design, and tool usage notably improved. Real-world challenges (e.g., changing requirements, technical decisions) enhanced problem-solving abilities.

Industry evaluation: The engineer praised project relevance and students' professionalism (documentation quality, security awareness, tool proficiency).

Instructor insights: Process-based assessments identified weaknesses (e.g., secure coding gaps), guiding future content adjustments.

ISSN:3007-9632 DOI: https://doi.org/10.70114/aimedr.2025.2.1.P140

5. Conclusion and Future Work

OBE-based reform, through objective restructuring, industry-relevant content, and process-focused assessments, effectively enhanced the course's practicality, relevance, and outcome attainment. The case study confirmed improved student competencies in engineering practice, teamwork, and professionalism.

Future research would be conducted in the following directions:

(1) Enrich project repository: Develop diverse projects involving cloud/AI/big data.

(2) Strengthen industry collaboration: Engage engineers in curriculum design, teaching, and evaluation; establish internship bases.

(3) Refine continuous improvement: Use outcome analytics, feedback, and industry input to dynamically optimize the course.

(4) Leverage EdTech: Adopt cloud IDEs, DevOps tools, and automated testing to modernize project workflows.

The implementation of Outcome-Based Education in software engineering curricula constitutes an iterative and evolutionary journey. This transformation remains imperative to cultivate future-ready professionals. Only through unwavering commitment to student competency development and strategic alignment with dynamic industry demands can we effectively nurture high-caliber software engineering talent equipped to navigate evolving technological landscapes.

References

[1] Spady, W. G. (1994). Outcome-Based Education: Critical Issues and Answers. AASA.

[2] China Engineering Education Accreditation Association (CEEAA). Engineering Education Accreditation Standards [S]. Latest edition.

[3] Jiang, Z. L. (2017). Competency-oriented construction of teaching frameworks in computing disciplines. Computer Education, (1), 8–12.

[4] Li, Z. Y. (2015). Outcome-oriented instructional design. China University Teaching, (3), 32–39.

[5] Global Tech Talent Survey. (2022). Bridging the Academia-Industry Divide: Employer Perspectives on Graduate Readiness. TechSkills Foundation.

[6] Anderson, R., & Lee, S. (2023). Scaling Outcome-Based Education in Computing: Challenges and Best Practices. IEEE Transactions on Education, 66(1), 45-54.

[7] Johnson, M. K., & Rodriguez, P. (2021). Essential Competencies for Entry-Level Software Engineers: An Industry-Wide Analysis. Journal of Systems and Software, 183, 111087.