Journal of Education and Educational Research
ISSN: 2957-9465 | Vol. 12, No. 1, 2025

Curriculum Design based on Outcome-Based Education Concept: A Case Study of the C Programming Course

Dongmei Bao [1, 3], Lkhagva Ariunjargal [2, *]

[1] School of Educational Studies, Mongolian National University of Education, Ulaanbaatar, Mongolia
[2] Department of Educational studies and methodology, School of Educational Studies, Mongolian National University of Education, Ulaanbaatar, Mongolia
[3] College of Artificial Intelligence and Big Data, Hulunbeir University, Inner Mongolia 021008, China

*** Corresponding author:** Lkhagva Ariunjargal (Email: mubisariunjargal@msue.edu.mn)

Abstract: This study, grounded in the principles of Outcome-Based Education (OBE), presents the design and implementation of a practical course syllabus for the C Programming Language within an industry-university collaboration framework. The syllabus development emphasizes the core tenets of "student-centered learning, outcome orientation, and continuous improvement." By aligning curriculum design with market demands, adopting effective teaching methodologies, and establishing a robust evaluation system, the syllabus addresses students' practical needs and professional aspirations, ensuring the comprehensive achievement of course objectives. The study incorporates process evaluation via the PTA platform and summative evaluation through system development projects, employing Rubric scoring criteria to holistically assess students' programming knowledge, skills, and attitudes. The findings reveal that the syllabus design and pedagogical approach significantly enhance students' competencies in programming, providing valuable insights and a reference model for talent cultivation in software engineering education.

Keywords: OBE Concept; Curriculum Design; Industry-University Collaboration; C Programming Language.

1. Introduction

With the advancement of modern engineering education, the cultivation of practical and innovative abilities has become a central objective in the academic sector. As a flagship initiative of industry-university collaboration, the talent development program for software engineering is designed in alignment with engineering accreditation standards, explicitly defining its training objectives and expected outcomes. This approach aims to equip students with the ability to seamlessly integrate theoretical knowledge with practical applications, thereby addressing the industry's demand for highly skilled, application-oriented professionals.

Characteristics of the OBE Concept and Current Research

The Outcome-Based Education (OBE) concept emphasizes a student-centered approach, focusing on achieving learning outcomes while continuously enhancing teaching quality. This concept aligns closely with the goals of engineering accreditation. Dewey's "learning by doing" theory provides a practical foundation for OBE, advocating for the resolution of real-world problems as a means to improve learning outcomes.

Research on OBE has deepened, particularly in teaching design and evaluation methodologies. In 2003, Acharya conducted an in-depth analysis of OBE's theoretical framework, proposing four fundamental principles for its implementation. These include defining learning outcomes, realizing their achievement, assessing the outcomes, and applying them effectively. Acharya also developed a four-step teaching model comprising *Defining* (learning outcomes), *Realizing* (achieving outcomes), *Assessing* (evaluating outcomes), and *Using* (applying outcomes).

Aziz and Hashem further contributed to OBE research by introducing a student performance evaluation technique based on a Fuzzy Logic System (FYS). This approach addresses both Course Learning Outcomes (CLO) and Program Learning Outcomes (PLO), providing timely feedback to students. It helps them identify areas of weakness, bridge learning gaps, and enhance the attainment of desired outcomes.

In China's talent cultivation model, school-enterprise joint training has become a significant trend (BAO DONGMEI & Ариунжаргал, 2024). The curriculum design in this study, based on the OBE educational philosophy, is collaboratively developed by enterprises and universities. It aims to leverage the strengths of both parties to provide students with a more systematic and practice-oriented learning pathway.

The C Programming Language Course and Curriculum Design under the OBE Concept

As a core course within the software engineering program, the *C Programming Language* course is designed to develop students' foundational programming knowledge and practical application skills. To achieve these objectives, the practical teaching component is led by industry professionals, integrating real-world engineering projects to enhance students' practical competencies.

To refine the curriculum further, scholars have optimized the course design by applying the OBE concept. This involves clearly defining course objectives, creating outcome-oriented teaching activities, and establishing a robust evaluation framework. These efforts ensure comprehensive improvements in teaching quality and student learning outcomes.

This OBE-based curriculum design not only fulfills the requirements of engineering accreditation but also serves as a practical guide and innovative model for talent cultivation in the field of software engineering.

With the development of modern engineering education, the cultivation of practical and innovative abilities has become a key objective in the education sector. As a flagship

program of industry-university collaboration, the talent development plan for software engineering is designed based on engineering accreditation standards, clearly defining its training objectives and expected outcomes. Through this approach, the software engineering program aims to equip students with the ability to closely integrate theoretical knowledge with practical applications, meeting the industry's demand for highly skilled, application-oriented professionals.

2. Theoretical Framework

Curriculum Design Based on the OBE Concept

The C Programming Language course, a foundational component of the software engineering program, is designed to enhance students' programming knowledge, practical skills, and professional attitudes through hands-on learning experiences.

This study, situated within the framework of industry-university collaboration, develops a curriculum design guided by the principles of Outcome-Based Education (OBE). It examines the effectiveness of the proposed framework in real-world teaching scenarios, evaluating its impact on student learning outcomes and identifying opportunities for refinement and optimization.

Table 1. Course Information

Course Title	Practical Course on *C Programming Language Design*				
Credits/Hours	1/36	**Course Category**	Professional Education Module	**Course Nature**	Specialized Practical Course
Offered by	School of Artificial Intelligence and Big Data		**Program**		Software Engineering
Prerequisite Requirements	Students are expected to have foundational theoretical knowledge and skills in *C Programming Language*.				
Support for Subsequent Courses	This course provides a foundational professional knowledge base for subsequent courses, including *Data Structures*, *Python Programming*, *Java Programming*, *Java Course Design*, and *Graduation Project*.				
Course Description	*C Programming Language Design* is a procedural, abstract, and general-purpose programming language widely used in low-level development. It allows for simple compilation, low-level memory handling, and the development of high-efficiency programs without requiring additional runtime support while generating minimal machine code. Despite offering many low-level processing features, C retains its cross-platform characteristics. Programs written in standard C can be compiled and executed on a wide variety of computing platforms, including embedded processors and supercomputers. This makes it an essential language for developers working on diverse systems.				

To ensure the alignment of course objectives with graduation requirements, a detailed mapping between the competencies targeted in the C Programming Language course and the graduation requirement indicators has been established. Table 2 illustrates how the course objectives support the development of specific professional skills and knowledge areas outlined in the graduation criteria. This mapping highlights the direct contributions of the course to achieving foundational, practical, and innovative outcomes, reinforcing the broader goals of the software engineering program.

Application of Teaching Methods

This study employs a variety of innovative teaching methods aimed at enhancing students' learning outcomes and overall abilities. The specific approaches include:

1. **Group Discussions**: Encouraging students to share their perspectives through group discussions to stimulate critical thinking and problem-solving skills.
2. **Observation in Real-World Environments**: Organizing field research and on-site observations to help students deepen their understanding of learned concepts in real-world contexts, while fostering practical application abilities.
3. **Learning from Others' Work**: Guiding students to analyze and appreciate real project cases, enabling them to draw inspiration and improve their design and creative skills.
4. **Collaborative Projects**: Emphasizing teamwork by assigning group tasks, helping students develop communication skills, collaboration awareness, and team spirit.

The integration of these teaching methods has not only significantly enhanced students' engagement in learning but also promoted their comprehensive development in knowledge, skills, and attitudes.

Course Assessment and Evaluation

To accurately evaluate students' learning outcomes in programming knowledge, skills, and attitudes, this course adopts a two-stage evaluation approach combining formative and summative assessments:

- **Formative Assessment:** Conducted through the PTA platform, it focuses on students' learning performance and teaching quality, aiming to dynamically optimize course content and improve teaching methods.
  - **Knowledge Evaluation**: Assesses students' understanding of fundamental programming concepts through tests.
  - **Skills Evaluation**: Focuses on students' ability to apply knowledge, including code accuracy, functionality implementation, and problem-solving skills.
  - **Attitude Evaluation**: Examines students' professional ethics, teamwork abilities, and sense of responsibility through their efforts to correct errors in code.
- **Summative Assessment:** Based on the quality of systems developed by students, it emphasizes their practical application abilities.
  - **Knowledge Evaluation**: Evaluates students' ability to address knowledge gaps during the development process, as assessed by engineers and instructors.
  - **Skills Evaluation**: Tests students' practical application and comprehensive utilization of knowledge in system development.

- **Attitude Evaluation**: Assesses students' learning attitudes, participation, and enthusiasm through system presentations and defenses.

Table 2. Table of the Relationship Between Course Objectives and Graduation Requirement Indicators

Course Objectives	Supported Graduation Requirement Indicators	Graduation Requirements	Support Strength
Understand modular and structured programming concepts. Master the basic elements of C language, including data types, statement formats, and function structures. Familiarize with the C language environment and common debugging methods.	**1.2**: Apply natural sciences, engineering fundamentals, and professional knowledge to the deduction, analysis, and comparison and synthesis of solutions for software engineering problems.	Apply mathematics, natural sciences, fundamental software engineering, and professional knowledge to solve complex engineering problems.	L
	4.3: Design research pathways for specific software engineering problems, develop experimental plans, build experimental systems, record experimental results, analyze and interpret these results, and synthesize information to draw reasonable and effective conclusions.	Conduct research on complex software engineering problems based on scientific principles and methods, including designing experiments, analyzing and interpreting data, and synthesizing information to draw reasonable and effective conclusions.	L
Ability to perform structured program design using C language. Ability to solve one-dimensional data storage problems using C language. Ability to utilize library functions and implement custom function calls in C language.	**3.1**: Master the basic methods and techniques of software product design, development, quality assurance, and testing, while understanding the various factors that influence software product design objectives and technical solutions.	Design solutions for complex software engineering problems, including the design of software systems, modules, or processes that meet specific requirements. Demonstrate innovation in the design process while considering factors such as society, health, safety, laws, culture, and the environment.	M
	3.2: Develop software algorithm processes, unit modules (components, parts) for specific requirements, and complete their design, development, and testing. Demonstrate innovative ideas and adopt novel approaches during the design, development, and testing stages.		M
Through this course, students will master the fundamental C language programming skills, including software functionality analysis, interface design, coding implementation, program debugging, and analysis. These skills will serve as a solid foundation for learning other programming languages in the future.	**5.1**: Understand the principles and methodologies for using major methods, platforms, and tools in the software engineering field, including their differences and applicable domains.	Develop, select, and use appropriate technologies, resources, modern engineering tools, and information technology tools for addressing complex software engineering problems, including prediction and simulation, while understanding their limitations.	M
	5.2: Select and use appropriate technologies, resources, modern software engineering tools, and information technology tools to predict and simulate complex software engineering problems, while analyzing their limitations.		M

Table 3. The Relationship Between Course Content and Course Objectives

№	Course Content	Expected Learning Outcomes	Hours	Teaching Methods	Support for Course Objectives
1	Selected System User Requirements Analysis	Master the basic concepts of software user requirements and understand the methods for preparing requirement documents	6	Lecture, Discussion, Case Analysis	Through this course, students will master the fundamental C language programming skills, including software functionality analysis, interface design, coding implementation, program debugging, and analysis. These skills will serve as a solid foundation for learning other programming languages in the future.
2	Classification of User Requirements: Functional and Non-Functional	Ability to categorize user requirements into functional and non-functional requirements and conduct analysis	6	Interactive Lecture, Task-Driven Approach	
3	Transforming User Requirements into Use Case Diagrams and Data Flow Diagrams	Learn to transform requirements into use case diagrams and data flow diagrams, enhancing software modeling skills	6	Classroom Exercises, Group Collaboration	
4	Implementation Based on Use Case Diagrams and Modular Programming	Master the concepts of modular and structured programming, and write functional code based on use case diagrams	6	Practical Teaching, Code Writing Guidance	Understand modular and structured programming concepts. Master the basic elements of C language, including data types, statement formats, and function structures. Familiarize with the C language environment and common debugging methods.
5	Program Function Testing and Debugging	Understand the basic methods of software testing, and identify and fix errors in programs	6	Site Visits to Real Projects, Project-Driven Teaching	
6	System Performance Optimization and User Experience Enhancement	Apply programming, testing, and optimization techniques to improve software performance and user experience	6	Experiments, Discussions, Summarization	Ability to perform structured program design using C language. Ability to solve one-dimensional data storage problems using C language. Ability to utilize library functions and implement custom function calls in C language.

This evaluation framework ensures a comprehensive assessment of students' knowledge, skills, and attitudes, promoting their overall development and aligning with the course objectives.

Specific Evaluation Criteria

The evaluation of this course is based on the OBE (Outcome-Based Education) philosophy, utilizing the Rubric scoring method to assess students' learning outcomes more scientifically and accurately. This evaluation framework ensures consistency between formative and summative assessment goals, providing a solid foundation for improving teaching quality.

Table 4. Rubric Scoring Method Table

	Criteria	Score
Knowledge	Understanding of software concepts	6
	Understanding user needs and requirements	6
	Understanding modular and structured programming concepts	6
	Knowledge of modeling	6
	Software testing and verification	6
Skills	Defining user requirements	10
	Analyzing user requirements	10
	Designing program models	10
	Dividing programs into components and writing modular code	10
	Testing, debugging, and improving program functionality	10
Attitude	Commitment to improving program performance	10
	Enthusiasm for mastering modern programming languages and new technologies	10
	Total	100

3. Method

Research Sample

The sample of this study consisted of 160 university students majoring in software engineering.

Research Procedure

This study first reviewed the current research on Outcome-Based Education (OBE) in curriculum design, clarifying the research background and theoretical foundation. It also analyzed the normative guidance provided by engineering education accreditation standards on course objectives and teaching content. Based on this, the study explored the foundational, practical, and logical characteristics of the C Programming Language course as a core component of the software engineering program, providing theoretical support for curriculum design.

Guided by the OBE concept, the study developed a curriculum framework for the C Programming Language course, defining course objectives, designing teaching activities, and specifying expected learning outcomes. A scientific evaluation system was constructed, utilizing rubric scoring criteria aligned with course objectives to ensure comprehensive and fair assessment. The new curriculum framework was then implemented in teaching practice, with its impact on students' learning outcomes recorded.

By comparing the control group and the experimental group, the study analyzed differences in learning outcomes, skill enhancement, and learning attitudes between the two groups, thereby validating the effectiveness of the new curriculum framework.

Data analysis

This study utilized SPSS 26.0 software to conduct statistical analysis of the surveyed students' final assessment results and related data, providing data support for the research.

4. Results

Pre-Test Results of Learning Levels in Control and Experimental Groups

To verify the accuracy of student performance in the control and experimental groups, this study conducted a statistical test and comparison of the two groups' scores before implementing the *C Programming Language* practical course. A detailed analysis of the evaluation results was also performed.

Table 5. Comparison of Average Scores Between the Control and Experimental Groups

Group	Sample Size	Mean	Standard Deviation	F-Value	P-Value
Control	80	65.76	12.245	3.455	0.065 > 0.05
Experimental	80	64.11	14.952		

According to the results of an independent sample variance analysis, the average score of students in the control group was 65.76, while that of the experimental group was 64.11, with an F-value of 3.455 ($P = 0.065 > 0.05$). This indicates that there is no statistically significant difference in scores between the two groups, suggesting that the learning levels of the control and experimental groups were generally consistent before the implementation of the practical course. This provides a comparable foundation for the subsequent experimental study.

Post-Experiment Analysis of Learning Outcomes in Control and Experimental Groups

This study aims to compare the effectiveness of the traditional curriculum and the OBE (Outcome-Based Education) optimized curriculum in promoting the development of students' knowledge, skills, and attitudes through a teaching experiment. By evaluating changes in students' learning outcomes, the study seeks to validate the effectiveness of the optimized curriculum.

The improvement in students' learning levels is primarily reflected through performance evaluations. These evaluations are based on the achievement of knowledge, skills, and attitude objectives set for the course, assessing the extent to which the course objectives are met:

- **Knowledge Outcomes**: Assessed through the breadth of theoretical knowledge reflected in students' work.
- **Skills Outcomes**: Measured by the completion quality of students' work and the professional abilities applied during the creative process.
- **Attitude Outcomes**: Evaluated through students' explanations of their work, plans for future improvements, and reflections on their projects.

The entire evaluation process is conducted collaboratively by instructors and industry mentors to ensure comprehensive and objective results. This thorough evaluation serves as a strong foundation for determining the effectiveness of the optimized curriculum and its potential for wider implementation.

Table 6. Progress in Knowledge, Skills, and Attitudes of Students in the Experimental Course

№	Indicator	Results and Evaluation of the Previous Program		Results and Evaluation After Conducting the Experimental Lesson		Improvement and Outcomes	
		Average Score	%	Average Score	%	Average Score	%
1	Understanding of software concepts	4.9	81.7%	5.8	95.8%	0.85	14.1%
2	Understanding user needs and requirements	4.6	76.7%	5.6	93.5%	1.0	16.8%
3	Understanding modular and structured programming concepts	4.4	72.7%	5.3	87.9%	0.9	15.2%
4	Knowledge of modeling	4.2	69.2%	5.0	83.1%	1.2	13.9%
5	Software testing and verification	4.4	74%	5.4	89.8%	1.0	15.8%
6	Defining user requirements	7.0	70.1%	8.7	87%	1.7	16.9%
7	Analyzing user requirements	6.9	69.1%	8.6	85.5%	1.7	16.4%
8	Designing program models	6.7	67.1%	8.3	83.1%	1.6	16%
9	Dividing programs into components and writing modular code	6.6	65.6%	8.1	80.8%	1.5	15.1%
10	Testing, debugging, and improving program functionality	6.8	67.6%	8.4	84.1%	1.6	16.5%
11	Commitment to improving program performance	7.1	70.6%	8.5	84.9%	1.4	14.3%
12	Enthusiasm for mastering modern programming languages and new technologies	6.9	68.8%	8.3	83%	1.4	14.2%

5. Discussion

The results of this study demonstrate that the OBE-based curriculum design effectively enhances students' learning outcomes in the *C Programming Language* course. By aligning the curriculum with market demands and incorporating innovative teaching methods, such as group discussions and collaborative projects, the new framework significantly improves students' theoretical understanding, technical skills, and professional attitudes.

Moreover, the combination of formative and summative assessments ensures a comprehensive evaluation of students' learning progress. This approach not only bridges knowledge gaps but also strengthens students' practical application abilities. Collaboration between educators and industry professionals plays a crucial role in narrowing the gap between academia and industry, equipping students with the skills needed to tackle real-world challenges.

6. Conclusion

The findings of this study highlight the importance of Outcome-Based Education (OBE) in designing practical courses within the framework of industry-university collaboration. The OBE-optimized curriculum significantly enhances students' comprehensive competencies, making it a valuable model for talent cultivation in software engineering. Future research could further explore the long-term impact of such curriculum designs on students' career development and

employability.

References

[1] Gong Jianmin & Xiao Beilei. (2020). Who is the Audience for Curriculum Syllabus Development? — On the Implementation of the Student-Centered Concept. Research in Higher Engineering Education, 143–150.

[2] Gong Jianmin & Xiao Beilei. (2020). Who Should Develop the Curriculum Syllabus? — A Discussion on the Design of OBE Talent Training Programs. Research in Higher Engineering Education, 180–197.

[3] BAODONGMEI, Л.Ариунжаргал, “On Development of School- Enterprises Cooperative Education in China: A Literature Review (1998-2023)”. INTERNATIONAL JOURNAL OF ENDUCATION AND HUMANITIES, Vol.14 No.2 P 50-56. 2024. https://doi.org/10.54097/zxvw9j34 ISSN : 2770-6702.

[4] Wu Yingjie, Liu Wenzhong, Wan Daqian, & Gao Mingxing. (2023). Curriculum Syllabus Design Based on the Concept of Engineering Education Professional Accreditation.

[5] Jiang Li. (2023). Teaching Design and Practice Research on the "Simulated Tour Guide" Course Based on the OBE Teaching Model. Master's Thesis, Zhejiang Normal University.

[6] Han Guixiang. (2023). Implementation and Optimization of Tennis Specialization Syllabus for Physical Education Majors under the OBE Concept — A Case Study of Wuhan Institute of Physical Education. Master's Thesis, Wuhan Institute of Physical Education.

[7] Acharya C. (2003). Outcome-Based education (OBE): A new paradigm for learning.Triannual Newsletter,78-82.

[8] Aziz A, Hashem M M A.(2022).Fuzzy Logic-Based Assessment of Students Learning Outcome in Implementing Outcome-Based Education. Proceeding sof the International Conference on Big Data,IoT,and Machine Learning , 745–759.