Curriculum and Teaching Methodology (2025)
Clausius Scientific Press, Canada

DOI: 10.23977/curtm.2025.080420
ISSN 2616-2261 Vol. 8 Num. 4

Research on Teaching Reform of Spring Cloud Microservice Architecture Based on OBE Concept

Qian Zihao

College of Computer Science, Guangzhou College of Applied Science and Technology, Guangzhou, 510000, Guangdong, China

Keywords: OBE Concept; Spring Cloud; Teaching Reform; Microservice Architecture

Abstract: To address issues such as outdated teaching content and insufficient practical skills development in Spring Cloud microservices architecture courses at universities, this paper explores integrating the OBE (Outcomes-Based Education) teaching philosophy into the curriculum. By analyzing the current state of course instruction and aligning with the technical demands of the software industry, the paper reconstructs the course objectives and breaks down core technologies into quantifiable skill indicators. This includes optimizing teaching content and objectives based on the OBE philosophy, designing a three-tier progressive project-based case teaching model, fostering group collaboration and inquiry practices, and implementing a diversified assessment and evaluation system. Practical experience shows that the OBE-based teaching model has proven highly effective, significantly improving teaching quality. Students' problem-solving abilities and engineering practice skills have been notably enhanced, and their initiative and innovation awareness during the learning process have been markedly strengthened.

1. Introduction

As society continues to evolve and people's reliance on online shopping grows, there is a growing need to analyze and design more efficient and reliable large-scale e-commerce platforms to meet the increasing demand for shopping [1]. Currently, microservices frameworks, which have been refined through the practical experiences of many outstanding developers, have become the mainstream approach in modern software development. When enterprises implement or apply microservices frameworks, they can not only enhance development efficiency and reduce costs but also further optimize and expand these frameworks to meet diverse business needs. At present, microservices frameworks have been successfully implemented by an increasing number of medium and large enterprises, achieving significant development [2]. This has also presented new challenges for the cultivation of technical talent in microservices. In the current applied undergraduate curriculum, the course' Spring Cloud Microservices Architecture 'is a required course for students majoring in software engineering. It serves as a key course that bridges theoretical knowledge with practical engineering, playing a crucial role in developing students' skills in using and mastering distributed system development. However, the course often faces issues such as outdated theoretical knowledge and weak practical components in teaching practice. Additionally, there is a significant gap between the course content and real-world enterprise development scenarios, creating a divide between theoretical learning and actual project development, making it difficult to establish an effective connection

The 'Opinion of the Ministry of Education on Deepening the Reform of Undergraduate Education and Teaching to Comprehensively Improve the Quality of Talent Cultivation' proposes to comprehensively improve the quality of course construction, deepen the reform of undergraduate education and teaching, and cultivate all-round socialist builders and successors who excel in moral, intellectual, physical, aesthetic, and labor education. The opinion emphasizes a classroom teaching model centered on learning with teaching as the guiding principle, and the introduction of outcome-based education (OBE) provides a new approach to addressing this challenge. OBE breaks away from the traditional teaching model that focuses primarily on knowledge transmission by starting from actual industry needs. It constructs a three-ring system of 'goal-teaching-evaluation,' breaking down industry capability standards into specific teaching objectives. In the teaching of the Spring Cloud microservices architecture course, the OBE teaching philosophy is applied to help students build a comprehensive microservices technology knowledge system, master core technologies such as service governance, service circuit breaking and throttling, and distributed transaction processing, and develop their ability to solve complex engineering problems through practical projects, thus facilitating a smooth transition from campus learning to enterprise development.

Based on this, the teaching reform is advanced from four dimensions: restructuring the curriculum and content, innovating teaching models, optimizing practical systems, and reforming evaluation mechanisms. In terms of course content design, it aligns with industry technology trends and dynamically updates the teaching syllabus. By developing three-tier project cases—basic training, comprehensive projects, and enterprise real-world projects—it aims to transition from explaining individual technologies to developing complete projects. A group collaboration learning model is adopted to simulate the agile development process of enterprises. A diversified evaluation system, including process evaluation and project outcome evaluation, is established. This ultimately forms a new talent cultivation model that meets current industry needs, effectively enhancing students' engineering practice abilities and employment competitiveness.

2. Diagnosis of the teaching status of Spring Cloud microservice architecture course

2.1 Teaching philosophy is out of line with industrial needs

Currently, in the teaching of Spring Cloud courses at most universities, there is a tendency to prioritize technical instruction over practical application scenarios. The core issue is that teachers focus too much on explaining isolated technical components and not enough on their relevance to real-world business contexts. This teaching approach contrasts sharply with the practical skills required by today's society for microservices professionals. A systematic analysis of the course outlines for' Spring Cloud Microservices 'at various domestic universities reveals that the content primarily focuses on the basic principles of technical components, while simulations of enterprise-level business scenarios are generally absent. For instance, regarding Service Mesh technology, a university's 2024 course schedule includes only 0.5 hours of theoretical instruction, whereas the Huawei Cloud '2023 Microservices Technology Application White Paper' indicates that the actual application rate of microservices technology in sectors like finance and e-commerce has reached 67.3%, with 41.2% of Istio deployments in production environments. This mismatch between teaching content and industry needs directly results in students lacking practical skills when they encounter core business scenarios such as service governance and traceability in their professional work, making it difficult for them to adapt quickly to their tasks and impacting their career development[3].

2.2 The course content is out of sync with the technology iteration

In recent years, with the rapid advancement of technologies such as cloud computing and microservices, microservice architecture has become a highly popular technical framework in today's IT systems[4]. This rapid development has led to a significant gap between the current curriculum content in universities and industry requirements, posing a serious challenge to the employment prospects of graduates. Our analysis of mainstream textbooks on microservice architecture published from 2019 to 2023 reveals the following issues:

Firstly, the content of textbooks still heavily relies on the Netflix ecosystem (such as Eureka, Ribbon, Hystrix, and other components that have been discontinued), whereas enterprises have widely adopted the Spring Cloud Alibaba ecosystem (such as Nacos, Sentinel, Seata, etc.). The latter is better suited to meet the technical needs for high availability and scalability in China, highlighting the current textbooks' lag in responding to industry technology trends.

Second, the depth of instruction on key functional components is insufficient. For instance, in terms of service fault tolerance (a core aspect of microservices elasticity), mainstream textbooks rarely cover production-level configurations of Sentinel (a leading open-source fault tolerance framework), such as rule customization, circuit breaker strategies, and traffic control chains. This contrasts sharply with the actual application rate in enterprises, highlighting a significant gap in the practicality of the teaching content.

Thirdly, the development of experimental environments lags behind technical requirements. Most university computer labs still operate in a teaching environment that only supports Spring Boot 2.3.x, which is incompatible with the Spring Cloud 2022.x version. This is particularly challenging under ARM architecture, where it is difficult to implement advanced technologies such as Service Mesh integration, Cloud Native Build Packs, and container-native deployment, thereby limiting the depth of hands-on skills training for students.

2.3 Teaching methods and engineering ability deviation

In classroom teaching, the traditional "lecture-practice" teaching mode is generally adopted by colleges and universities, which still occupies the dominant position. However, there is a significant deviation between this mode and the training objectives of engineering ability, which is embodied in:

First, the authenticity of learning scenarios is insufficient. In traditional teaching models, practice sessions often consist of formulaic theoretical exercises or highly simplified simulations that strip away the complexities of real-world situations. This stark contrast to the open, dynamic, and uncertain real-world scenarios in engineering practice hinders students from developing their problem-solving skills in real-world contexts. For instance, in microservices architecture courses, classes are typically lecture-heavy and fragmented experiments, often limited to 'verification experiments' without simulating real business scenarios. This lack of simulation, such as real-world e-commerce platform microservice applications or project team collaboration to solve complex problems, fundamentally contradicts the core principle of engineering education, which is to develop skills through practice. When teaching scenarios remain at a simplified 'knowledge training ground' rather than a 'practical engineering field,' students struggle to develop sensitivity to engineering problems, a nuanced understanding of decision-making, and teamwork skills. This ultimately hinders the development of systems thinking, innovation, and engineering literacy, making it difficult to cultivate students' ability to apply theoretical knowledge in dynamic, interdisciplinary environments and impedes the development of systems thinking and collaborative problem-solving skills.

2.4 The evaluation system is out of line with the output of ability

The current evaluation system for microservices architecture courses is significantly disconnected from the development of engineering skills, which severely hinders the improvement of teaching quality and the cultivation of students 'practical engineering abilities. From an evaluation perspective, the existing assessment mechanism has structural imbalances. According to Bloom's Taxonomy of Educational Objectives, the development of engineering skills should encompass multiple levels, including knowledge memorization, understanding and application, analysis and evaluation, and innovation and creation. However, the current assessment system overly emphasizes the final outcomes of group projects, with the weight of final grades typically exceeding 60%. This results-oriented evaluation method fails to adequately address core engineering competencies such as code standardization, architectural rationality, and fault diagnosis and repair. Furthermore, using group projects as the primary assessment tool does not provide a comprehensive and accurate evaluation of each student's technical skills and systems thinking.

In the evaluation process, the absence of a process-oriented evaluation mechanism is another significant issue. According to constructivist learning theory, learning is a dynamic process where learners continuously build knowledge through practice. However, most institutions still rely on a summative evaluation model centered on final presentations, lacking monitoring and feedback throughout the project development lifecycle. Only a few courses incorporate process-oriented evaluation methods such as milestone reviews, code reviews, and iteration retrospectives, which fail to effectively track students' development in areas like requirement analysis, solution design, and system optimization. Furthermore, these methods do not reflect the continuous improvement emphasized in modern software engineering practices like agile development and DevOps, leading to a significant gap between teaching evaluations and the goals of cultivating practical engineering skills.

3. Curriculum reform implementation plan based on OBE concept

3.1 Curriculum content reconstruction oriented by industrial demand

Based on the principles of reverse design from the Outcomes-Based Education (OBE) framework, the Spring Cloud microservices architecture course has undergone a systematic restructuring to better reflect the evolving industry needs and practical application contexts. This restructuring adopts a 'dual-track parallel' strategy, aiming to enhance students' foundational technical skills while integrating the latest industry technologies and practices. The dual-track approach ensures that learners not only grasp core theoretical principles but also apply this knowledge in real-world enterprise environments.

First, the basic skills enhancement module focuses on reinforcing students 'understanding of the fundamental components within the Spring Cloud ecosystem. To maintain the completeness of the foundational knowledge base, it covers traditional content such as service registration and discovery mechanisms (like Eureka), configuration centers (like Spring Cloud Config), and circuit breakers (like Hystrix). However, to deepen students' conceptual and analytical skills, the course introduces advanced analysis, exploring the internal workings and theoretical models of these components. For example, students will engage in comparative studies between Eureka's availability and partition tolerance (AP) features and Consul's consistency and partition tolerance (CP) aspects, helping them understand the trade-offs and design decisions in distributed systems from the perspective of the CAP theorem.

Secondly, the industry technology integration module focuses on the application of advanced technologies widely adopted by major enterprises today. This includes system integration Spring

Cloud Alibaba technology stacks, such as Nacos (for service discovery and configuration management), Sentinel (for fault tolerance and traffic control), and Seata (for distributed transaction management). These components are not isolated but form part of a cohesive system design that reflects modern cloud-native architecture. Additionally, to foster critical thinking and empirical analysis, the course includes design-based comparative experiments, such as performance benchmarking of traditional Spring Cloud Netflix stacks and Spring Cloud Alibaba solutions in high-concurrency scenarios, for example, simulating e-commerce flash sales (or 'flash sales').

Furthermore, to ensure the continuous development of the curriculum and its ability to adapt to the evolving demands of the industry, 20% of the course content is revised each semester based on the latest trends and insights from the Alibaba Technology Radar report. This process ensures that students stay updated with the latest innovations in technology and that teachers enhance their technical skills, thereby maintaining a competitive edge in the rapidly evolving technological landscape.

3.2 Three-level progressive project case teaching system

A three-tier progressive project-based teaching system has been designed (as shown in Table 1), systematically developing a pathway from foundational cognition to innovative practice. In the foundational stage, students gain a solid understanding of microservices architecture through in-depth explanations of core components such as Consul service registration and discovery, and Load Balancer load balancing calls. In the intermediate stage, students develop practical engineering skills by solving real-world business problems through projects in typical sectors like e-commerce and finance. In the innovative stage, the ChaosBlade chaos engineering tool is introduced to simulate production environment failure scenarios, thereby enhancing students 'system fault tolerance design and problem-solving skills comprehensively. This step-by-step teaching design ensures systematic knowledge transmission, strengthens practical engineering skills,and highlights the forward-looking nature of innovation, providing an effective path for cultivating microservices talent that meets industry needs. Practical experience has shown that this system significantly enhances students' core competencies in distributed system design and fault diagnosis, better preparing them for enterprise-level microservices development.

Table 1: "Basic-advanced-innovative" ability training ladder

top class	Project type	Technical elements	Ability goals
The base layer	Monolithic component verification project	Consul Service discovery, Load Balancer load balancing, etc	Master the use of core component API
Move up the ladder	Domain projects (e-commerce/money transactions)	Seata distributed transaction, Sentinel circuit breaker	Implement technical solutions for business scenarios
Innovation layer	The Chaos Project	Chaos Blade fault injection (CPU full load, network latency)	Design high availability fault tolerance scheme

3.3 Team collaborative engineering practice teaching mode

To foster students 'comprehensive skills in project collaboration, problem-solving, and professional responsibility, a structured and dynamic team project mechanism has been implemented. Each project team consists of no more than four students, who can form teams based on shared interests and compatibility. During the project implementation phase, team members are assigned specific roles through democratic consultation, divided into four major categories: project leader, technical lead, documentation manager, and quality supervisor. To adapt to the evolving needs at different stages of the project, a dynamic role rotation system is in place, ensuring that each student gains diverse experiences and develops multifaceted skills throughout the project lifecycle. Additionally, to ensure the fairness and objectivity of student performance evaluations, a group mutual evaluation form has been designed. This form includes four core dimensions: task completion (30%), collaboration contribution (25%), knowledge sharing (20%), and problem-solving ability (25%). The form is evaluated through anonymous cross-rating at each stage.This method not only enhances communication and cooperation but also accelerates development, fosters students' sense of responsibility, and achieves win-win development, as show in Table 2.

Table 2: Shows an example of group mutual evaluation

Peer review form				
Name and role of team members	Task completion (30%)	Collaborative contribution (25%)	Knowledge sharing (20%)	Problem solving (25%)
surname and personal name A	25	25	20	25
surname and personal name B	28	25	18	17
surname and personal name C	22	25	16	22
surname and personal name D	26	24	20	20

The whole team cooperation process is systematically divided into three different stages-project preparation, project implementation and project submission-Each stage has clear goals and responsibilities:

1) Project preparation stage:

In this initial phase, the team collaborates to develop a comprehensive project development plan. This includes defining the roles of team members, developing an overall project strategy and workflow, and identifying key milestones and role-based management responsibilities. The goal of this phase is to ensure that the team agrees on project goals, timelines, and collaboration norms.

2) Project implementation phase

At this stage, the project team leader is responsible for submitting regular progress reports on both technical and administrative aspects of the project, including challenges encountered and solutions taken by the team. This ensures transparency and keeps momentum throughout the development process.

3) Project submission stage

In the final stage, all team members conducted a thorough review and validation of the project's deliverables. This involved rigorous testing to ensure the system's functionality and completeness, as well as submitting all necessary documentation and source code. Additionally, each team member completed the group mutual evaluation form, contributing to the final performance assessment.

This structured teamwork and assessment approach not only strengthens students' practical engineering skills, but also embodies the core principles of the outcome-based education (OBE) framework by emphasizing real-world competencies, accountability, and reflective assessment.

3.4 Diversified assessment and evaluation system

Based on the OBE philosophy, a diversified evaluation method for learning costs [5] has been established, defining the achievement requirements for students at different stages of their studies.

Spring Cloud The microservices architecture course consists of two components: theoretical instruction and project practice. In the theoretical instruction phase (as shown in Figure 1), the focus is on the acquisition of foundational knowledge and the understanding of concepts, which includes five key elements, each contributing to the overall grade:

Online independent learning (10%) This element evaluates students' ability to learn independently through online platforms, with a focus on understanding microservice architecture principles, Spring Cloud components, and related cloud native technologies.

Communication and reflection (10%): This section evaluates students' participation in class discussions, online forums, and reflective logs, encouraging students to express their opinions, collaborate and reflect.

Main assignments (30%): Students will complete structured assignments, including design analysis, service decomposition, and configuration of Spring Cloud modules (such as Consul, Load Balancer, OpenFeign), to test theoretical understanding and practical application.

Basic Practice (20%): This refers to practical exercises that reinforce key concepts, such as building RESTful APIs using Spring Boot and Spring Cloud, implementing service discovery, and deploying services in a distributed environment.

Final exam (30%): The final assessment includes two modules: case analysis and technical problem solving, which assesses students' comprehensive mastery of microservice architecture concepts and Spring Cloud technology.

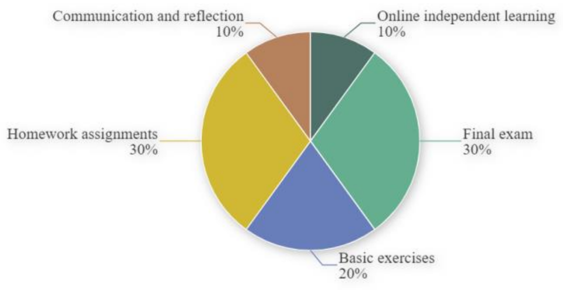

Figure 1: The proportion of theoretical teaching course evaluation

The Project Practice course is a project-based module designed to develop students' software engineering skills in real-world scenarios. The module is assessed independently on a 100-point scale and consists of five interrelated aspects (see Figure 2):

Standardized documentation (25%): Evaluate the completeness, clarity, and degree of standardization of the final project documentation, including system requirements specifications, architecture design reports, and API documentation.

Software quality (35%): Students are evaluated on code quality, system performance, maintainability, test coverage, and adherence to best practices in software development using continuous integration and deployment pipelines.

Innovative design (10%): Encourage creativity and originality in solving complex technical problems, including the integration of emerging technologies such as container coordination, service mesh, or cloud native monitoring tools.

Team collaboration (10%): measures the effectiveness of teamwork, task allocation, role rotation and internal project management, and develops students' leadership and project management skills.

Project defense (20%): At the end of the project, students will attend a defense meeting to present the results of the project, demonstrate the system functions, and be able to answer questions from teachers.

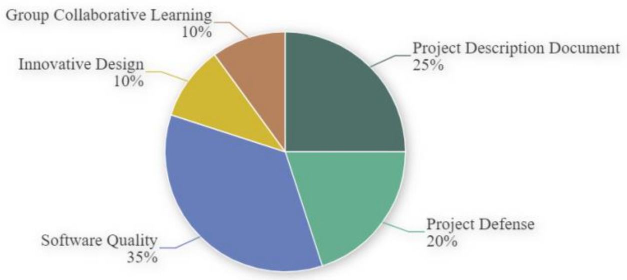

Figure 2: Evaluation ratio of project practice courses

4. Effect of reform practice

The school where the researchers are based has 8 teaching classes for the 2022 cohort of software engineering students. In the first semester of the 2024-2025 academic year, a Spring Cloud microservices architecture course was introduced, with a total of 18 weeks of instruction, 8 hours per week. To conduct research on teaching reform based on the OBE philosophy, two experimental classes were selected: Class 1 and Class 2 of the 2022 cohort of software engineering technology, and two control classes: Class 3 and Class 4 of software engineering, which continued to use the traditional teaching model. The final exam results of the experimental classes, as shown in Table 3, clearly outperformed those of the control classes.

Table 3: Comparison of final exam results

	Softwork 1 class (experimental class)	Softwork 2 class (experimental class)	Softwork 3 class (Control class)	Softwork 4 class (Control class)
Average	80.6	81.5	70.4	72.5
pass rate (%)	100	100	76	78.5
Excellent rate (%)	36.8	40.5	17.5	19.6

The analysis of the performance comparison table shows that the Soft Engineering 1 and Soft Engineering 2 classes (the experimental class) using the OBE (Outcomes-Based Education) teaching model significantly outperform the Soft Engineering 3 and Soft Engineering 4 classes (the control class) using traditional lecture-based teaching in multiple key indicators. Specifically, the average scores for the experimental classes are 80.6 and 81.5, respectively, higher than the control classes '70.4 and 72.5. In terms of pass rates, both experimental classes have a 100% pass rate, while the control classes have 76% and 78.5%, respectively, with some failing students. In terms of excellence rates, the experimental classes have 36.8% and 40.5%, respectively, higher than the control classes' 17.5% and 19.6%. This indicates that the OBE teaching model is more effective in improving students' academic performance.

5. Conclusion

This study integrates the OBE philosophy into the Spring Cloud microservices architecture course, creating a reform loop of demand-oriented—content restructuring—practical advancement—diverse evaluation.' Practical experience shows that the OBE model significantly enhances teaching effectiveness: the course content aligns with industry trends, incorporating the

Spring Cloud Alibaba technology stack and chaos engineering, addressing the issue of 'technological lag'; the three-tier project system (basic verification, domain practice, chaos engineering) strengthens the cultivation of situational skills, significantly enhancing students' systematic thinking and collaboration abilities; a diversified evaluation system comprehensively assesses engineering literacy, with the comprehensive ability indicators of the experimental class being 28%-39% higher than those of the control class.

Future research will primarily focus on the following areas of deepening: First, enhancing the integration of industry and education by collaborating with enterprises to transform real-world production scenarios, such as managing millions of concurrent traffic flows, into teaching projects, thereby enhancing the authenticity of technical practice. Second, continuously refining textbooks to ensure that teaching content stays in sync with technological advancements. Third, exploring interdisciplinary integration by developing courses like 'Microservices + Artificial Intelligence' to cultivate versatile talents who can adapt to the digital transformation of industries. This will help implement the OBE (Outcomes-Based Education) philosophy in more engineering courses and facilitate a deeper alignment between higher education and industry needs.

References

[1] Leung A, Spyker A, Bozarth T.Titus: Introducing containers to the Netflix cloud[J].Communications of the ACM, 2018, 61(2):38-45.

[2] Xin Yuanyuan, Niu Jun, Xie Zhijun, et al. Overview of the Framework for Implementing Microservice Architecture [J]. Computer Engineering and Application, 2018,54(19):10-17.

[3] Xu Bao. Research on the Construction of an Evaluation Index System for Undergraduate Classroom Teaching Quality Centered on Students [D]. Northeast Petroleum University, 2023.

[4] E Xueni, Shen Zhitao, and Wang Chao. Design and Implementation of a Mobile Network User Complaint Preprocessing System Based on Springboot Microservice Architecture [J]. Changjiang Information & Communication, 2025,38(01):115-117.

[5] Wang Wei. Design of a Diverse Evaluation Method for Learning Outcomes Based on the OBE Concept: A Case Study of E-commerce Copywriting Creativity and Writing [J]. Modern Vocational Education, 2021(38):66-67.