Research and Practice on Hierarchical Teaching of the "Object-Oriented Programming" Course for College-Junior College Upgrading Students

Shiyuan Liu*, Kaile Xiao, Kun Liu, Cui Zhao
College of Applied Science and Technology, Beijing Union University, Beijing, China

Abstract: Aiming at the problems in the teaching of the "Object-Oriented Programming" course in colleges for college-junior college upgrading students, such as large differences in students' educational backgrounds, insufficient adaptability of the traditional "one-size-fits-all" teaching model, and disconnection between students' practical abilities and industry needs, this study takes the college-junior college upgrading students majoring in Computer Science and Technology at the School of Applied Technology of Beijing Union University as the practical objects. It constructs a two-dimensional teaching system of "background - goal" classification and "ability - knowledge" stratification. Students are stratified through multi-dimensional data collection and analysis; teaching resource construction is completed through knowledge reconstruction and school-enterprise collaboration; precise resource push and dynamic adjustment are realized with the help of an intelligent platform; and finally, the teaching model is optimized through pilot verification. The practical results show that this model has effectively improved the course pass rate, competition award rate, and employment contract signing rate of students. Among them, the employment contract signing rate of the 2024 graduating class has increased by 24.6 percentage points compared with that of the 2023 class, and national-level competition awards have achieved a breakthrough from scratch. This provides a replicable practical path for the teaching reform of computer-related courses in college-junior college upgrading education.

Keywords: College-Junior College Upgrading Education; Object-Oriented Programming; Hierarchical Teaching; Two-Dimensional Standards; Teaching Resource Planning

1. Introduction

As an important link connecting higher vocational colleges and ordinary undergraduate universities, college-junior college upgrading education has become a key path for cultivating applied talents under the background of the popularization of higher education since its nationwide implementation in 1999 [1,2]. With the deepening of the popularization of higher education, the scale of college-junior college upgrading students has continued to expand, and the number of applicants for college-junior college upgrading nationwide exceeded 2 million in 2023 [3]. At the same time, the student source structure has shown a diversified characteristic: the proportion of retired soldiers has been increasing year by year. Taking the computer major of Beijing Union University as an example, retired soldiers accounted for 53.3% of the college-junior college upgrading students of the 2023 class; there are significant differences in students' previous majors, with only 42.1% of students coming from computer-related majors, and the rest distributed in multiple fields such as mechanical and electrical engineering, economics and management, and art.

The industrial transformation and upgrading have put forward higher requirements for the practical abilities of computer talents [4]. Enterprises not only require students to have a solid programming foundation but also expect them to have object-oriented system design capabilities, teamwork skills, and the ability to quickly adapt to new technologies. However, the traditional teaching model of the "Object-Oriented Programming" course, which features "unified progress and unified content", is difficult to adapt to the personalized needs of college-junior college upgrading students. The survey found that there are three prominent problems in the teaching of this course:

(1) The textbook content is highly theoretical, with insufficient code case analysis. 62% of

students reported that they "understand the theory but cannot program". When facing actual project development, they cannot transform abstract concepts such as classes, objects, and inheritance into runnable code.

(2) The teaching method is single, mainly based on teacher lectures, resulting in the lack of students' subjective initiative. Students with a weak foundation, such as some retired soldiers and students with non-computer majors, are prone to develop a sense of frustration during the learning process and have low classroom participation; while students with a good foundation feel that "they are not challenged enough", which affects their learning enthusiasm.

(3) The evaluation system is rigid, with grades only determined by the final exam. It ignores the procedural improvement of abilities and cannot fully reflect students' engineering practice literacy, teamwork skills, etc., leading to a disconnection between teaching and employment needs.

Foreign research on hierarchical teaching started relatively early [5]. The "Individualized Hierarchical Teaching Theory" proposed by American scholars emphasizes the use of technical means to design personalized learning paths for students, track students' learning behaviors through intelligent systems, and dynamically adjust teaching content and progress. The German "Ferdinand Model" focuses on the differences in learning content and goals, constructs a stepped training system, and divides the teaching content into three levels: basic, extended, and research, to meet the needs of students with different abilities [6]. The hierarchical class grouping in Japanese high schools and the personalized course selection model in South Korea also provide diverse references for the practice of hierarchical teaching, focusing on curriculum combination according to students' interests and abilities [7]. These practices generally pay attention to the combination of technical tools and formative evaluation, such as using online platforms to track students' learning trajectories and dynamically adjust teaching strategies.

In terms of domestic research, scholars such as Xiong and He and Tian and Zhang have theoretically demonstrated the adaptability of hierarchical teaching to the current situation of student sources in vocational colleges, pointing out that hierarchical teaching can effectively solve the problem of large differences in students' foundations and improve the pertinence of teaching [8,9]. Li et al. and Gu have carried out hierarchical teaching practices in college English education and mathematics courses, verifying the role of this model in improving teaching quality [10,11]. However, the existing research has the following limitations:

The research objects mostly focus on students in ordinary colleges or higher vocational colleges, and there are few studies targeting college-junior college upgrading students. College-junior college upgrading students have both the vocational education background of the junior college stage and the academic improvement needs of the undergraduate stage. Their learning characteristics are significantly different from those of students in ordinary colleges, so the existing research results are difficult to be directly applied.

The practical scope is concentrated on basic courses such as English and mathematics, and there is insufficient exploration of hierarchical teaching in core computer professional courses. As a core basic course for computer-related majors, "Object-Oriented Programming" is crucial for students' subsequent professional course learning and career development, and it is urgent to carry out targeted hierarchical teaching research.

The basis for stratification mostly relies on academic performance, and it does not fully combine students' career planning and industry needs, resulting in a disconnection between teaching and employment. A considerable proportion of college-junior college upgrading students are employment-oriented, so teaching should be more closely designed around the ability requirements of industry positions.

At the theoretical level, the "two-dimensional hierarchical system" and "hierarchical teaching resource planning" constructed in this study enrich the theoretical framework of hierarchical teaching for computer-related courses in college-junior college upgrading education. By stratifying students based on their background and goals, as well as their abilities and knowledge, it provides practical support for defining the type attribute of vocational undergraduate education and clarifies how vocational undergraduate education can balance the personalized needs of students and the cultivation of vocational abilities in the process of talent training.

At the practical level, through differentiated

teaching content and a dynamic evaluation mechanism, it can effectively solve the problem that "students with a weak foundation cannot keep up, and students with outstanding abilities are not challenged enough" among college-junior college upgrading students, and improve students' programming practical abilities and industry adaptability. At the same time, the research results can provide references for the teaching reform of computer-related courses in similar colleges, help college-junior college upgrading education achieve the training goal of "connection between vocational and general education and integration of production and education", and transport more high-quality applied computer talents that meet the needs of the industry to the society.

2. Construction of the Two-Dimensional Standards of "Background-Goal" Classification and "Ability-Knowledge" Stratification

In view of the diverse backgrounds and significant differences in programming fundamentals among students pursuing the college-to-university upgrade program, and by integrating industry demands with educational principles, a scientific "classification standard" (grouping based on students' backgrounds and goals) and "stratification standard" (leveling based on students' competence and knowledge) have been established. As shown in Figure 1, through the dual-dimensional standards of "background-goal" classification and "competence-knowledge" stratification, combined with data analysis and industry needs, student groups are scientifically divided and targeted teaching paths are designed.

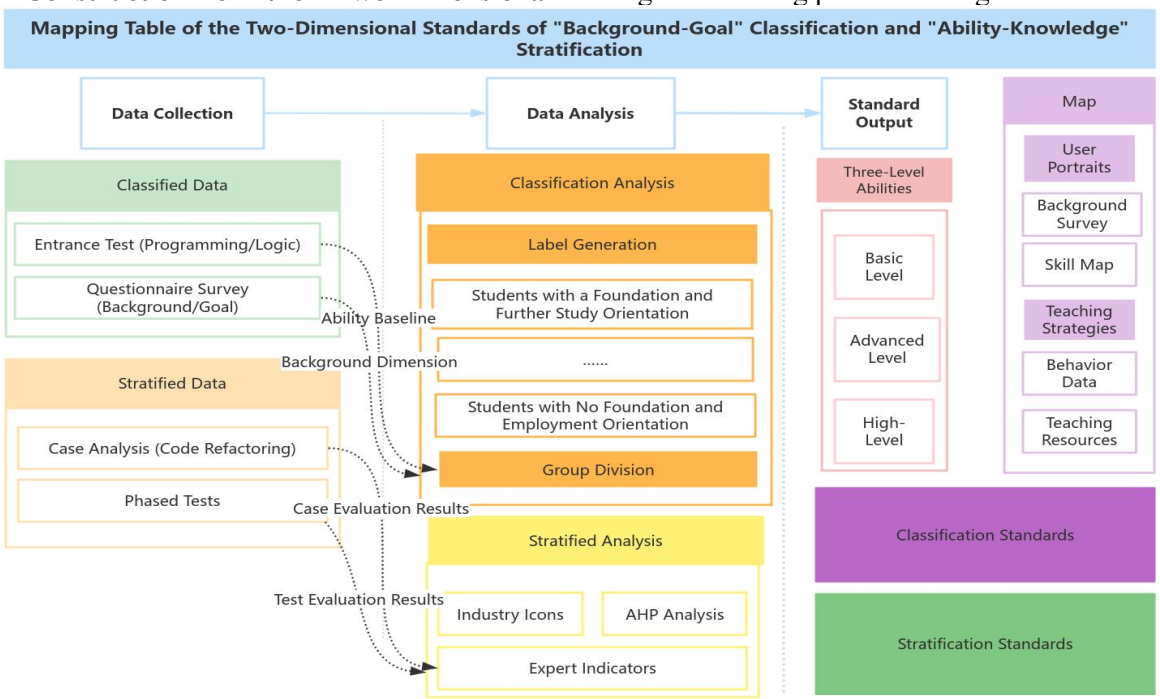

Figure 1. Construction of the Two-Dimensional Standards of "Background-Goal" Classification and "Ability-Knowledge" Stratification

2.1 Data Collection

The collection of classified data is the basis for the construction of the two-dimensional standards. The entrance test covers programming and logical knowledge, including basic syntax knowledge and simple programming logic. Through this test, the baseline ability of students can be obtained, and their initial level in programming can be understood. The questionnaire survey is used to collect students' background information, such as their previous majors (computer-related, mechanical and electrical engineering, economics and management, art, etc.), whether they are retired soldiers, and their goals (further study or employment-oriented). This information can reflect students' learning foundations and future development expectations, providing a basis for classification.

The collection of stratified data mainly relies on case analysis and phased tests. The case analysis adopts code refactoring tasks,

requiring students to optimize, expand, or refactor the given program code, so as to evaluate students' understanding and application ability of object-oriented thinking and obtain case evaluation results. The phased tests are conducted at different stages of the teaching process to examine students' mastery of phased knowledge and obtain test evaluation results, providing data support for stratified analysis.

2.2 Data Analysis

In the classification analysis link, labels are generated based on the collected classified data. For example, according to the entrance test scores and the background of previous majors, students are divided into categories such as "students with a foundation and further study orientation" (those with previous majors in computer-related fields, excellent entrance test scores, and plans for further study in the future) and "students with no foundation and employment orientation" (those with previous majors not in computer-related fields, poor entrance test scores, and employment as the main goal), and group division is carried out. In this way, the characteristics and needs of students in different categories can be clearly understood. In the stratified analysis, combined with industry icons (reflecting the ability requirements of the industry for talents at different levels), AHP (Analytic Hierarchy Process) analysis (analyzing the weights of various factors affecting students' abilities), and expert indicators (inviting experts in the computer field to evaluate students' abilities), an in-depth analysis of students' ability and knowledge levels is conducted. For example, experts will comprehensively evaluate students' object-oriented design ability and code implementation ability based on their performance in case analysis and tests, combined with the actual needs of the industry, providing professional basis for stratification.

2.3 Standard Output

Finally, a three-level ability map is output, clarifying the ability requirements for the basic level, advanced level, and high-level. Students at the basic level should master the basic concepts of object-oriented programming, be able to carry out simple class design and object creation, and complete the implementation of basic program functions; students at the advanced level need to have good object-oriented system design capabilities, be able to use mechanisms such as inheritance and polymorphism for medium-scale program development; students at the high-level should be able to carry out complex system architecture design and have the ability to develop large-scale projects in teamwork.

At the same time, classification standards and stratification standards are formed, as well as a mapping table including user portraits (students' basic information and learning characteristics), background surveys (students' previous majors, learning experience, etc.), skill maps (skills that students at different levels should master), teaching strategies (teaching methods and means for students at different levels), behavior data (students' learning behavior performance), and teaching resources (teaching materials, cases, etc. suitable for students at different levels). This provides an accurate basis for hierarchical teaching and ensures that teaching can be targeted.

3. Planning of Hierarchical Teaching Resources

The planning of hierarchical teaching resources should be centered on "competence progression" and "industry alignment". Through modular design, school-enterprise collaborative development, and empowerment by technological tools, a resource system integrating "learning-practice-application" should be constructed, as shown in Figure 2.

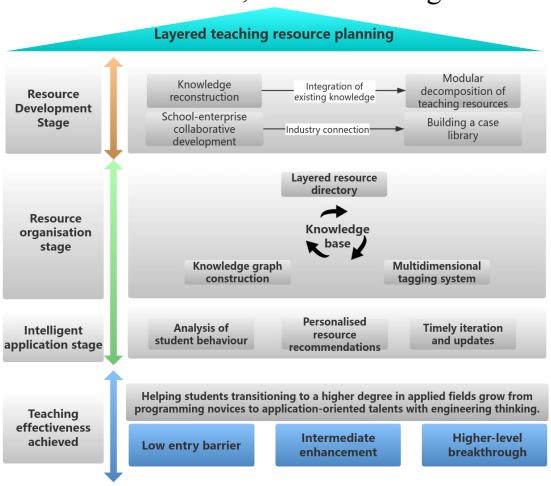

Figure 2. Path of Hierarchical Teaching Resource Planning

3.1 Resource Construction Stage

Knowledge reconstruction is an important part of resource construction. The original knowledge is integrated, the core knowledge points of the "Object-Oriented Programming" course are sorted out, and they are reorganized and presented in accordance with the object-oriented thinking to realize the modular decomposition of teaching resources. For example, knowledge points such as class definition, object creation, inheritance, and polymorphism are taken as independent modules, and each module includes theoretical explanations, case analyses, practical exercises, etc., which is convenient for students at different levels to learn according to their own needs.

School-enterprise collaborative development is carried out to strengthen the connection with the industry. Technical experts from enterprises are invited to participate in the construction of teaching resources, and typical cases from actual enterprise projects are introduced into teaching. For example, in cooperation with software development companies, real projects such as the "e-commerce order management system" and "enterprise employee management system" are simplified and adapted to be suitable as teaching cases, and a rich case library is built. These cases can enable students to be exposed to problems in actual work and improve their engineering practice capabilities.

3.2 Resource Organization Stage

A knowledge map is built to visually display the knowledge system of the "Object-Oriented Programming" course and the connections between various knowledge points. Combined with the hierarchical resource directory, teaching resources are classified and organized according to the ability requirements of students at different levels, and a knowledge base with a multi-dimensional label system is constructed. For example, each teaching resource is labeled with the suitable level (basic level, advanced level, high-level), knowledge point type (theoretical, practical, case, etc.), difficulty coefficient, etc., so that the teaching resources can be organized and managed in an orderly manner according to the needs of students at different levels, facilitating teachers and students to quickly find and use them.

3.3 Intelligent Application Stage

With the help of student behavior analysis, the intelligent platform tracks students' learning trajectories, such as online learning duration, homework completion status, and test scores. Based on these behavior data, students' learning characteristics and needs are analyzed to realize personalized resource recommendation. For example, for students with high learning enthusiasm but a weak foundation, more teaching resources and exercises at the basic level are recommended; for students with strong abilities and sufficient learning potential, extended resources at the advanced or high-level are recommended.

At the same time, the resources are updated iteratively in a timely manner. With the continuous development of computer technology and changes in industry needs, the teaching resources are reviewed and updated regularly. For example, new programming technologies, frameworks, and tools are introduced into the teaching resources to ensure the timeliness and applicability of the resources, enabling students to be exposed to the most cutting-edge knowledge and technologies.

3.4 Achievement of Teaching Effects

Through the planning of hierarchical teaching resources, college-junior college upgrading students are helped to grow from programming beginners to applied talents with engineering thinking. For students at the basic level, low-threshold entry resources are provided to help them master the basic concepts and programming methods of object-oriented programming and be able to carry out simple program development; for students at the advanced level, intermediate-level enhanced resources are provided to improve their system design capabilities and programming skills, enabling them to independently complete medium-scale project development; for students at the high-level, high-level breakthrough resources are provided to cultivate their complex system architecture design capabilities and teamwork skills, enabling them to participate in the development of large-scale projects and achieve the goal of hierarchical teaching.

4. Construction and Optimization of the Hierarchical Teaching Evaluation System

As shown in Figure 3. Construction of a Scientific Hierarchical Teaching Evaluation System through the Five-Step Method of

"Demand Analysis → Indicator Design → Tool Development → Dynamic Adjustment → Pilot Verification".

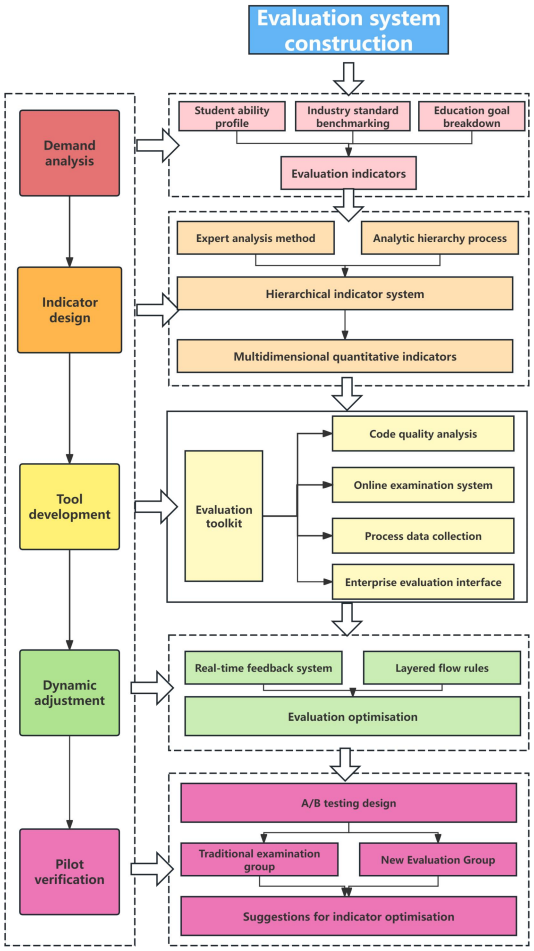

Figure 3. Five-Step Evaluation System

4.1 Demand Analysis and Evaluation Goal Setting

Starting from three aspects: students' ability portraits (students' ability levels and characteristics), industry standard alignment (ability requirements of enterprises for talents), and educational goal decomposition (training goals of undergraduate education for students), the evaluation indicators are determined. The specific requirements for students at different levels in knowledge mastery, skill application, engineering practice, etc. are clarified. For example, students at the basic level should be able to correctly write simple object-oriented programs; students at the advanced level should be able to design and implement medium-scale object-oriented systems; students at the high-level should be able to participate in the development and optimization of complex systems.

4.2 Design of Hierarchical Evaluation Indicator System

By using the expert analysis method and the analytic hierarchy process, experts in computer education and enterprise technical experts are invited to demonstrate and screen the evaluation indicators, and a hierarchical indicator system and multi-dimensional quantitative indicators are constructed. For example, the weights of indicators such as knowledge mastery, skill application, teamwork, and innovation ability are determined through the analytic hierarchy process, providing a scientific basis for evaluation and ensuring that the evaluation results can comprehensively and accurately reflect students' learning status and ability levels.

4.3 Development of Evaluation Tools

A set of evaluation tools including code quality analysis, online examination system, process data collection, and enterprise evaluation interface is created. The code quality analysis tool can analyze and evaluate the standardization, readability, and efficiency of the code written by students; the online examination system facilitates phased tests and final exams, realizing the automation and intelligence of examinations; the process data collection tool can record various behavior data of students in the learning process, such as online learning, homework submission, and discussion participation; the enterprise evaluation interface provides a channel for enterprises to participate in student evaluation, and enterprises can evaluate students' performance in internships or projects, collecting students' learning data through multiple channels and in an all-round way.

4.4 Dynamic Adjustment

Based on the real-time feedback system and hierarchical flow rules, the evaluation system is optimized. The real-time feedback system can promptly feed back students' evaluation results to teachers and students. Teachers can adjust teaching strategies according to the feedback, and students can understand their own strengths and weaknesses and learn in a targeted manner. The hierarchical flow rules allow students to move between different levels. For example, if students at the basic level perform well within a certain period and meet the ability requirements of the advanced level, they can be promoted to the advanced level for learning; if students at the advanced level perform poorly, they can also be

demoted to the basic level for consolidation learning. This ensures that the evaluation can dynamically reflect students' learning status and ability changes.

4.5 Pilot Verification

An A/B test is designed, dividing students into the traditional examination group and the new evaluation group. The traditional examination group adopts the traditional evaluation method mainly based on the final exam, while the new evaluation group adopts the hierarchical evaluation system constructed in this study. The differences in course scores, learning enthusiasm, practical abilities, etc. between the two groups of students are compared, feedback from students and teachers is collected, suggestions for indicator optimization are put forward, and the evaluation system is continuously improved to ensure the scientificity and effectiveness of the evaluation system.

5. Practical Effects and Analysis

5.1 Course Learning Effects

Taking 183 students as the practical objects, a comparison of the course scores before and after the implementation of hierarchical teaching shows that the average score of high-level students reaches 85 points, with 40% of them scoring above 90 points, an increase of 15 percentage points compared with traditional teaching. These students have performed well in project development and algorithm application, and can independently complete the design and implementation of complex object-oriented systems.

The average score of advanced-level students is 80 points, with 60% of them scoring between 70 and 85 points, and no failures. They can well master the core knowledge points of the course, have certain system design and programming capabilities, and can complete the development of medium-scale projects.

The average score of basic-level students is 75 points, with a pass rate of 95%, an increase of 20 percentage points compared with traditional teaching. These students are mainly retired soldiers and students with non-computer majors. Through hierarchical teaching, they can master the basic concepts and programming methods of object-oriented programming, complete the basic program development tasks, and their learning confidence and enthusiasm have been significantly improved.

In addition, 65% of students reported that "the difficulty of the hierarchical teaching content is suitable for their own level", and 74.5% of students "highly recognized" the hierarchical method, indicating that hierarchical teaching can meet the learning needs of students at different levels and has been widely recognized by students.

5.2 Competition and Innovation Achievements

After the implementation of hierarchical teaching, students' performance in discipline competitions has been significantly improved. In the competitions, students can apply the knowledge learned in the course to design and develop, showing strong engineering practice capabilities and innovation capabilities.

5.3 Improvement of Employment Quality

By comparing the employment data of the 2023 class (without the implementation of hierarchical teaching) and the 2024 class (with the implementation of hierarchical teaching), it is found that the employment rate of college-junior college upgrading students majoring in computer science in the 2023 class is 93.05%, and the contract signing rate is 70.05%; the employment rate of the 2024 class has increased to 94.65%, and the contract signing rate has reached 94.65%, which is the same as the employment rate. From the perspective of employment positions, 35% of high-level students have entered Internet companies to engage in algorithm development or system design work; 60% of advanced-level students have joined software development positions; 80% of basic-level students are engaged in software testing or technical support work. The matching degree between positions and abilities has been significantly improved.

6. Existing Problems and Improvement Directions

6.1 Existing Problems

6.1.1 Insufficient accuracy of the hierarchical dynamic adjustment mechanism

The existing hierarchical adjustment mainly relies on academic performance and does not fully incorporate changes in students' career planning and interests. For example, some advanced-level students show a strong interest in algorithm design, but they cannot be promoted

to the high-level in a timely manner because they have not participated in relevant evaluations; when the retired soldier group transitions from the basic level to the advanced level, there is insufficient knowledge connection, which is prone to "knowledge gaps".

6.1.2 Lagging connection between course content and industry frontiers

In the high-level teaching content, some projects only involve basic logic and do not introduce technologies required in actual enterprise scenarios such as high-concurrency architecture and big data analysis; the basic-level exercises still focus on verification tasks, lacking problem-solving training in real workplace scenarios, resulting in a long adaptation period for students after employment.

6.1.3 Insufficient coverage of soft skills in the evaluation system

The assessment focuses on code correctness and theoretical mastery, while ignoring the evaluation of soft skills such as communication and expression, and teamwork. For example, in the high-level team project assessment, only the function implementation is evaluated, and "efficiency of demand communication" and "rationality of team division of labor" are not included in the scoring standards, which is inconsistent with the comprehensive ability requirements of enterprises for engineers.

6.2 Improvement Directions

6.2.1 Optimizing the hierarchical evaluation dimensions

"Career planning interviews" (once a semester) and "interest assessments" (Holland Vocational Test) are added to construct a three-dimensional evaluation model of "ability - interest - planning". For the retired soldier group, a "theoretical enhancement module" is designed, and small-class weekend courses are used to supplement basic theories such as data structures and computer networks, shortening the cross-level adaptation period.

6.2.2 Deepening the content of school-enterprise integration

Teaching resources are updated in collaboration with enterprises: the high-level adds a "distributed system foundation" module, introduces technologies such as Redis cache and SpringCloud microservices, and combines case teaching of "distributed game server design"; the basic level adds a "workplace scenario simulation" module, such as fault diagnosis training based on enterprise operation logs, to improve students' ability to solve practical problems. It is planned to include AI programming tools (such as GitHub Copilot) in the high-level elective content in 2025 to keep up with the development of industry technologies.

6.2.3 Improving the comprehensive evaluation system

In the high-level team project assessment, indicators of "communication effectiveness" (10%) and "rationality of task allocation" (10%) are added, and quantitative evaluation is conducted through project meeting records and team mutual evaluation forms; the advanced level and basic level add the "technical document writing" assessment item, requiring the use of Markdown to write program design specifications; a "code readability scoring table" is introduced for all levels, and the high-level additionally evaluates code optimization capabilities, requiring students to propose at least 3 performance improvement plans for the project, so as to comprehensively improve students' engineering literacy.

7. Conclusion

This study aims at the pain points in the teaching of the "Object-Oriented Programming" course for college-junior college upgrading students, and constructs a three-in-one teaching system of "two-dimensional stratification, hierarchical teaching resource planning, and dynamic evaluation". Precise stratification is realized through multi-dimensional data collection and analysis; teaching resource construction is completed through knowledge reconstruction and school-enterprise collaboration; precise resource push and dynamic adjustment are realized with the help of an intelligent platform; and finally, the teaching model is optimized through pilot verification. Practice has proved that hierarchical teaching can not only significantly improve the course learning effect but also provide a promotable model for the reform of computer-related professional courses in college-junior college upgrading education. In the future, it is necessary to further optimize the hierarchical mechanism and the depth of school-enterprise integration, and promote the continuous improvement of teaching quality in the direction of "precision, personalization, and professionalism".

Acknowledgments

This paper was supported by Beijing Union University Teaching Reform Project: Research and Practice on Layered Teaching of the "Object-Oriented Programming" Course for Upgrading from College to University (JJ2025Q007).

References

[1] Zhang Lili. A Study on the Teaching Reform Path of Economic Mathematics Course for Economics and Management Majors. Western China Quality Education, 2025, 11(18): 168-172. DOI: 10.16681/j.cnki.wcqe.202518037.

[2] Xu Ying. Exploration of Stratified Teaching Mode for College Mathematics under the OBE Orientation — A Case Study of Xi'an Automotive Vocational University. Learning Weekly, 2025, (28): 50-53. DOI: 10.16657/j.cnki.issn1673-9132.2025.28.013.

[3] Sun Lei. A Study on the Stratified Teaching Mode of Automotive Professional Courses under the Background of Vocational Education College Entrance Examination — Practical Exploration Targeting Students with Weak Academic Foundation in Cultural Courses. Automotive Maintenance and Repair, 2025, (18): 29-30. DOI: 10.16613/j.cnki.1006-6489.2025.18.016.

[4] Fan Bowen, Qin Tao, Wang Jinyi, et al. Exploration on the Development of Higher Vocational Professional Courses Based on Stratified Teaching — A Case Study of Forging Technology and Equipment. Forging & Stamping Equipment & Manufacturing Technology, 2025, 60(04): 175-179. DOI: 10.16316/j.issn.1672-0121.2025.04.036.

[5] Bi Wenli. A Study on Enhancing the Internal Motivation of Higher Vocational College Students in English Learning. Journal of Hubei Open Vocational College, 2025, 38(16): 169-171+175. DOI: CNKI: SUN: HBHS.0.2025-16-058.

[6] Bi Wenli. A Study on Enhancing the Internal Motivation of Higher Vocational College Students in English Learning. Journal of Hubei Open Vocational College, 2025, 38(16): 169-171+175. DOI: CNKI: SUN: HBHS.0.2025-16-058.

[7] Ye Fan. A Study on the Cultivation Mechanism of Digital Literacy for Higher Vocational Scenarios. Journal of Academic Library and Information Science, 2025, 43(05): 54-60. DOI: CNKI: SUN: DXTQ.0.2025-05-008.

[8] Xiong Rongsheng, He Weixiang. A Study on the Necessity and Feasibility of Implementing Stratified Teaching in Higher Vocational Colleges (II). Journal of Zhejiang Business Technology Institute, 2004, (04): 67-70. DOI: CNKI: SUN: ZJGS.0.2004-04-023.

[9] Tian Mingwu, Zhang Lei. A Study on the Necessity of Implementing Classified and Stratified Teaching in Higher Vocational Colleges. Journal of Higher Education, 2017, (11): 179-181. DOI: 10.19980/j.cn23-1593/g4.2017.11.086.

[10] Li Chen, Zhu Jing, Xue Yuan. A Study on the Problems and Countermeasures of Hierarchical Teaching in Higher Vocational English. Journal of Jiamusi Vocational Institute, 2025, 41(09): 148-150. DOI: CNKI: SUN: JMSJ.0.2025-09-050.

[11] Gu Yuanyuan. A Study on Hierarchical Teaching of Higher Vocational Mathematics Supported by Information Technology. Life and Partner, 2025, (31): 63-65. DOI: CNKI: SUN: RSBX.0.2025-31-024.

